Universidad de la Ciencias Informáticas

Facultad 8

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

Diseño e Implementación de la seguridad del Sistema de Investigación e Información Policial

Autor: Yunior Alayo Rondon.

Tutor: Ing. Enrique José Altuna Castillo.

DECLARACION DE AUTORIA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de Ciencias Informáticas (UCI) que haga el uso que estime pertinente con este trabajo

Para que así conste formo la presente a los ___ días del mes de Mayo de 2009

Autor: Yunior Alayo Rondon

_____

Tutor: Ing. Enrique José Altuna Castillo

_____

Agradecimientos

Dedicatoria

Resumen

El presente trabajo tiene como objetivo diseñar e implementar un componente arquitectónico encargado de la seguridad del SIPOL (Sistema de Información e Investigación Policial) que se esta desarrollando como parte de la modernización del CICPC (Centro de Investigaciones Científicas Penales y Criminalísticas) de Venezuela. La investigación dio como resultado el componente anteriormente mencionado usando durante el proceso de desarrollo la metodología ágil XP (Extreme Programming) y es una extensión del framework de seguridad Acegi Sucurity System.

Abstract

This work has as target to design and implement an architectonic component for the security of SIIPOL (Information and Investigation Police System by his acronym in Spanish) that is being developed as the modernization process of the CICPC (Penal and Criminology Investigation Center by his acronym in Spanish) of Venezuela. This investigation had as result the architectonic component using during the developing process the agile methodology XP (Extreme Programming) and is an extension of Acegi Sucurity System framework.

Índice

Introducción

Debido a la naturaleza de las aplicaciones web es necesario que estas sean accesibles a través de Internet. Esa característica además de las ventajas trae consigo que se deba realizar un esfuerzo adicional con el objetivo de garantizar la seguridad en general del sistema informático. El Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC) de Venezuela como parte de su modernización decidió implantar un nuevo sistema para la gestión de sus procesos. En el proceso de evaluación del estado actual de la organización se encontró que el sistema anterior presentaba varias brechas de seguridad entre las cuales se encontraban que los datos del sistema viajaban sin encriptación, cualquier persona con conocimiento de las credenciales de un usuario del sistema podía acceder a funcionalidades importantes, el sistema no brindaba protección contra ataques como la suplantación de usuarios, no existía una agrupación de los usuario según su nivel de acceso, las contraseñas se almacenaban en la base de datos en texto plano lo cual implica que cualquier usuario con acceso a esta podría adquirir una identidad del sistema, el sistema era vulnerable a ataques de negación de servicio dando la posibilidad de que cualquier herramienta automatizada pudiera dejara fuera de servicio a una o varias sedes en todo el país, la complejidad generada por la falta de granularidad en la permisología permitía que los errores de administración fueran muy frecuentes y difíciles de encontrar. Teniendo como base todo lo anteriormente planteado se decidió que el nuevo sistema debía cumplir con varios requisitos no funcionales referentes a la seguridad debido a la sensibilidad de la información gestionada por este.

Esta investigación presenta como **objeto de estudio** la Seguridad en Aplicaciones Web y como **campo de acción** la Seguridad del Sistema de Investigación e Información Policial (SIIPOL). Durante la fase de elaboración surge la interrogante ¿Cómo garantizar la seguridad del SIIPOL? lo cual nos conduce hacia nuestro objetivo principal: Diseñar e implementar la seguridad del SIIPOL, desglosado en varios objetivos específicos explicados a continuación:

Analizar los posibles ataques a sistemas informáticos: Es necesario tener un conocimiento claro sobre los ataques más frecuentes contra los cuales el sistema debe estar protegido.

Estudiar las implementaciones de seguridad de sistemas existentes: Nos permite reutilizar código común usado con éxito en sistemas con necesidades semejantes.

Estudiar los frameworks de seguridad existentes: Realizar un análisis de los frameworks para usar el que más funcionalidades nos brinde y se adapte con facilidad a las necesidades anteriormente planteadas.

Por tanto el uso de un framework flexible y extensible posibilitará la implementación de la seguridad del SIIPOL.

Con el propósito de llevar a cabo la investigación se planificó las siguientes **tareas investigativas**:

- Estudiar exhaustivamente los posibles ataques a aplicaciones web.
- Analizar las características del medio en que se aplicará la política de seguridad.
- Recopilar datos sobre los frameworks de seguridad existentes.
- Analizar las características de los frameworks de seguridad.
- Adaptar las funcionalidades brindadas por el framework para su uso posterior.
- Diseñar pruebas a la seguridad.

Capítulo 1. Fundamentación Teórica

1.1 Estado del Arte

1.1.1 Plataforma

Java™ 2 Platform, Enterprise Edition (J2EE™) define un estándar para el desarrollo de aplicaciones empresariales basándolas en componentes modulares estandarizados ofreciéndoles una serie de servicios y manejando varios detalles del funcionamiento de la aplicación automáticamente. Tiene como principal ventaja que es posible escribir el código una vez y ejecutarlo en todas partes (portabilidad) además presenta un Application Programming Interface (API) para Java Database Connectivity (JDBC). Tiene un modelo de seguridad que protege los datos aun si es una aplicación de Internet. Presenta soporte para Java Servlets API, Java Server Pages (JSP) y Extensible Markup Language (XML). Permite interoperabilidad de Servicios WEB. Por tanto podemos resumir que es una plataforma altamente soportada, portable, escalable y segura.

1.1.2 Entorno de Desarrollo Integrado (IDE)

Netbeans.

NetBeans es un proyecto de código abierto fundado y patrocinado por Sun MicroSystems es un IDE que permite escribir, compilar, corregir errores y ejecutar programas. Permite crear aplicaciones de escritorio, web y para dispositivos móviles. Soporta además lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. Permite desarrollar aplicaciones usando la plataforma J2EE. Puede ser usado en varias plataformas entre las que se encuentran Windows, Linux, Mac OS X y Solaris.

Eclipse

Eclipse IDE comenzó como un proyecto de International Business Machines (IBM) Canadá. Fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge.

Como característica principal tiene el empleo de módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente, al permitirle a Eclipse extenderse

usando otros lenguajes de programación como son C/C++, Python permite trabajar con lenguajes para procesado de texto como LaTeX. La arquitectura *plug-in* permite escribir cualquier extensión deseada en el ambiente.

El Software Development Kit (SDK) de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

La última versión estable es la 3.3 y fue liberada el 25 de junio de 2007. Dentro de la rama 3.3, su versión actual más avanzada es la 3.3.1.1, liberada el 23 de octubre de 2007.

Características

- Editor de Texto
- Resaltado de Sintaxis
- Compilación en tiempo real
- Pruebas unitarias con Junit
- Control de Versiones
- Asistentes (Wizards), para la creación de proyectos, clases, tests, etc.
- Refactorización

Selección del IDE:

Se decidió hacer un uso parcial de abos IDEs. El Eclipse se utilizará para el desarrollo de las funcionalidades más vinculadas a la aplicación web debido a las facilidades que brinda de navegación a través del código fuente, rendimiento, estabilidad entre otros factores que lo favorecen respecto al NetBeans IDE, pero además este último se utilizará para desarrollar otros componentes de la seguridad como Applets para los cuales existe un mayor soporte.

1.1.3 Metodologías

RUP

La metodología de desarrollo RUP forma parte de las llamadas metodologías pesadas. Entre sus principales características figuran que cuenta con una Forma disciplinada de asignar tareas y responsabilidades.

RUP tiene la facilidad de adaptarse a cualquier proyecto, y no tan solo de software aplicando varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Se enfoca en modelos, en lugar de la producción de una gran cantidad de documentación. Fomenta el uso de buenas prácticas como las siguientes:

- Desarrollo Iterativo.
- Modelamiento visual.
- Arquitectura de componentes.
- Verificación de la calidad.

XP

La metodología programación extrema (XP por sus siglas en inglés) es realmente un acercamiento al desarrollo de software deliberado y disciplinado que ha sido usada en varias compañías de diferentes tamaños y con diferentes características en toda la industria. Esta diseñada para entregar el software que el cliente necesita en el momento que lo necesita. Permite a los desarrolladores responder a los cambios del cliente aun al final de ciclo de vida. Además enfatiza en el trabajo en equipo incluyendo al cliente directamente en el proceso de desarrollo XP se basa en la simplicidad, la comunicación y el reciclado continuo de código. Promueve los valores de comunicación directa entre las personas; el intento de mantener el software lo más simple posible.

Debido a la naturaleza del **campo de acción** de esta investigación se usará la metodología XP. Esta decisión se tomó porque se deben dar soluciones rápidas y no embellecidas y la solución la implementará un desarrollador.

1.1.4 Frameworks.

El concepto framework se emplea en el desarrollo de software. En general, con el término framework, nos estamos refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En términos informales, un framework se puede considerar como funcionalidades generales a las cuales se les puede agregar las específicas para resolver un problema determinado y obtener una solución informática.

Spring

Spring Framework (también conocido simplemente como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores. Por su diseño ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar.
Está diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además intenta mantener un mínimo acoplamiento entre la aplicación y el propio framework de forma que podría ser desvinculada de él sin demasiada dificultad.

Los principales módulos de este framework son:

Core: Como su nombre indica, es el núcleo de Spring. Permite técnicas de Inversión del Control (IoC) como la inyección de dependencias.

Context: Proporciona herramientas para acceder a los beans y da soporte a propagación de eventos, resource bundles, carga de recursos y creación transparente de contextos por parte de los contenedores.

DAO (Data Acces Object): Proporciona una capa de abstracción

JDBC (Java Database Connectivity) y una forma de administrar transacciones.

ORM (Object Relational Mapping): Provee capas de integración para APIs (Application Programming Interface) de mapeo objeto-relacional.

AOP (Aspect Oriented Programming): Proporciona una implementación de programación orientada a aspectos, permitiendo definir puntos de corte e interceptores.

Web: Provee de características de integración orientadas a la web, como inicialización de contextos mediante servlet listeners y un contexto de aplicación orientada a la web.También permite integrar de forma sencilla otros frameworks como Struts, Java Server Faces (JSF) o WebWork.

Spring MVC: Provee una implementación Modelo-Vista-Controlador que permite el uso del resto de funcionalidades del Spring Framework.

Entre las principales características de este framework se encuentran:
Se centra en la capa intermedia y proporciona enganches para manejar la solución elegida para la capa de presentación y de integración. Permite Internacionalización y localización. Spring viene integrado con un framework de seguridad, Acegi Security, que gestiona todo el mecanismo de login, autentificación y autorización.

Spring proporciona un estupendo marco para el testing. Al estar basado en POJOs (Plain Old Java Object), los test de unidad son triviales con su adaptación de JUnit. Para los test de integración provee herramientas basadas en su capacidad de IoC y de transaccionalidad.

En cuanto al mapeo objeto-relacional proporciona integración con varias implementaciones ORM. Existen dos formas de integración, a través de plantillas predefinidas del módulo SpringDAO o codificando DAOs directamente contra al API del ORM elegido. Cualquiera de las dos aproximaciones ofrece los beneficios de Spring, como ser configurados a través de IoC (Inversión del Control), transaccionalidad, wrapping común para excepciones de acceso a datos y manejo de la configuración independiente de la implementación.

Desde el punto de vista de la programación orientada a aspectos Spring permite el uso de aspectos en tiempo de ejecución mediante proxys dinámicos y permite la integración de AspectJ que ofrece funcionalidad completa. La Inyección de dependencias (DI) es una de las bases de Spring sobre la que se cimienta el resto de la arquitectura. La DI se encuentra en el corazón de Spring.

La configuración de Spring está basada en XML. Al no tener anotaciones no le hace dependiente de Java EE 5. Aunque en la versión 2.5 se introduce el uso de las anotaciones.
El diseño de Spring está pensado para ofrecer un modelo de cómo debe trabajar la aplicación y cómo se comunican sus partes. Está expresamente concebido para que deba ser extensible y acoplable con otros frameworks, ya que no ofrece una solución completa que abarque desde la presentación al modelo.

Acegi

Acegi Security System es un framework íntimamente ligado al proyecto Spring, que facilita la tarea de adoptar medidas de seguridad en aplicaciones Java, sean aplicaciones de escritorio o aplicaciones web. Un atributo importante que debemos tener en cuenta es que es open source y con el respaldo de un enorme y creciente grupo de usuarios que lo están utilizando, además de un manual de referencia con más de 100 páginas que no tiene nada que envidiar a la documentación de un producto comercial. La arquitectura de Acegi está fuertemente basada en interfaces y en patrones de diseño, proporcionando las implementaciones más comúnmente utilizadas y numerosos puntos de extensión donde nuevas funcionalidades pueden ser añadidas.

Las principales características de Acegi se pueden resumir en los siguientes puntos:

- Estable y maduro: Acegi Security 1.0.0 fue liberado en Mayo de 2006 después de más de dos años y medio de uso en la producción de largos proyectos de software habiendo sido descargado mas de 70000 veces y contando con cientos de contribuciones de la comunidad.
- No modifica el código existente permitiendo que el programador se centre en codificar sus reglas de negocio, lo cual nos indica que se podría construir la aplicación sin seguridad e *inyectar* estos criterios con posterioridad.
- Permite realizar la autentificación contra varios repositorios, lo cual permite desarrollar la aplicación contra un repositorio local en texto plano y usar en producción una base de datos, únicamente cambiando una línea del archivo de configuración de Spring.
- Hace posible el uso de seguridad declarativa. En un servidor de aplicaciones es posible definir los criterios de seguridad de un EJB o de la aplicación web en un archivo independiente del código, lo mismo se puede conseguir con Acegi para las aplicaciones que se desarrollan en Spring.
- Permite proteger los objetos antes y después de ser invocados. Si el resultado de una de las llamadas a un objeto de negocio produjera un comportamiento no deseado, su valor puede ser modificado antes de que el objeto apropiado reciba el resultado, proporcionando de esta forma *otro* tipo de seguridad a la aplicación.
- Cache de identificaciones. Acegi usa caches para ahorrar accesos a los distintos repositorios que se tenga configurado. Viene con un desarrollo por defecto que usa ehcache para tales labores.

- Asegura peticiones http y llamadas a métodos. En las aplicaciones web permite la identificación mediante Basic Html o formulario, así como, en algunos casos, identificarse contra los repositorios de los servidores de aplicaciones.
- Seguridad del Canal: Acegi puede re direccionar automáticamente las peticiones a través de un canal de trasporte apropiado.
- Soporte para LDAP: Soporta autenticación mediante un directorio LDAP.
- Usa el mecanismo de eventos de Spring. Pudiendo de esta manera ser notificado de cualquier suceso que acontezca en la aplicación, como por ejemplo las peticiones a las que se deniega el acceso, quien y cuando lo intentó.

Capítulo 2. Propuesta de Solución

2.1 Introducción

Como propuesta de solución de la presente investigación tenemos un componente de la arquitectura encargado de la seguridad de la aplicación apoyándonos en las herramientas anteriormente explicadas. Este componente tiene como misión principal dar soporte a las necesidades comunes de todos los sistemas de software y a las específicas del SIIPOL.

Este componente está dividido en las siguientes partes:

- Integración Acegi-JSF
- Integración Acegi-Ajax
- Integración Acegi-Applet
- Integración Acegi-Spring

2.2 Propuesta

2.2.1 Integración Acegi-JSF

Este sub-componente consiste en un conjunto de clases que extienden las funcionalidades del framework Acegi Security System con el objetivo de asegurar las peticiones a nivel de página. Debido al modo en que JSF procesa las peticiones se hace necesario interceptarlas en un momento determinado diferente al modo convencional.

2.2.2 Integración Acegi-Ajax

Las peticiones ajax en las aplicaciones web tienen características especiales lo cual influye en nuestra solución debido a que no se tiene el conocimiento de la url que es accedida por lo que este sub-componente permite realizar un enlace con otros componentes del sistema que nos permite acceder a la información necesaria.

2.2.3 Integración Acegi-Applet

La autenticación en el sistema comprende además de usuario y contraseña el procesamiento de huellas dactilares así como reconocimiento humano lo cual nos lleva a la necesidad de extender el framework base Acegi Security System para que soporte estas funcionalidades.

2.2.4 Integración Acegi-Spring

Como fue visto en el capítulo anterior Acegi Security System es un framework muy ligado a Spring y nuestra solución hará uso de esas características. Los sub-componentes anteriormente mencionados estarán vinculados al sistema a través de cuatro ficheros de configuración:

- **applicationContext-acegi-security.xml:** Comprende las configuraciones generales de la seguridad en este caso usando el sub-componente que da soporte a JSF.
- **applicationContext-acegi-security-ajax.xml:** Configuraciones para el procesamiento de las peticiones ajax en la aplicación.
- **applicationContext-applet.xml:** Configuraciones necesarias para la autenticación usando applet así como la creación de usuarios debido a que en este sistema es necesario procesar huellas dactilares.
- **definicionesSeguridad-Context.xml:** Configuración referente a la seguridad a nivel de pagina web, específicamente, la permisología necesaria para acceder a una página web.

2.3 Exploración

La metodología de desarrollo XP comienza con su fase de exploración. Durante esta fase se realiza el proceso de identificación de Historias de Usuario (HU) así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del sistema.

2.3.1 Historias de Usuario

Las historias de usuarios son la forma en que se especifican en XP los requisitos del sistema. Éstas se redactan desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido que ellas abarcan debe ser concreto y sencillo. Durante este proceso se identificaron 7 historias de usuarios las cuales se detallan a continuación.

Plantilla de Historia de Usuario

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: Número de la Historia de Usuario (HU), incremental en el tiempo.</td><td colspan="2">Usuario: Nombre de la HU, sirve para identificarla fácilmente entre desarrolladores y clientes.</td></tr>
<tr><td colspan="3">Nombre historia: El usuario del sistema que utiliza o protagoniza la HU</td></tr>
<tr><td colspan="2">Prioridad en negocio: Grado de importancia para el cliente.</td><td>Riesgo en desarrollo: Grado de dificultad para el desarrollador.</td></tr>
<tr><td colspan="2">Puntos estimados:</td><td>Iteración asignada: Iteración a la que corresponde.</td></tr>
<tr><td colspan="3">Programador responsable: Nombre del programador al cual se le asigna.</td></tr>
<tr><td colspan="3">Descripción: La descripción de la HU detallando las operaciones del usuario y opcionalmente las respuestas del sistema.</td></tr>
<tr><td colspan="3">Observaciones: Algunas observaciones de interés como glosario, información sobre usuarios, etc.</td></tr>
</table>

Historias de Usuario

Historia de Usuario	
Número: 1	**Usuario:** Desarrollador de CICPC
Nombre historia: Procesar petición Http	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Yunior Alayo	
Descripción: El usuario genera una petición http al servidor para llevar a cabo una tarea determinada la cual es procesada para determinar si el remitente tiene acceso al recurso solicitado.	
Observaciones:	

Historia de Usuario	
Número: 2	**Usuario:** Desarrollador de CICPC
Nombre historia: Procesar petición de salida de la aplicación	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Yunior Alayo	
Descripción: El usuario genera una petición http con el objetivo de registrar el fin la sección de trabajo (Conexión con el servidor para acceder a recursos)	
Observaciones: A partir del momento en que esta petición haya sido atendida el remitente no tendrá acceso a los recursos.	

Historia de Usuario	
Número: 3	**Usuario:** Desarrollador de CICPC
Nombre historia: Procesar petición por canal seguro (Https)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Yunior Alayo	
Descripción: El usuario genera una petición sobre un canal seguro de comunicaciones con el objetivo de acceder a recursos sensibles del sistema.	
Observaciones:	

Historia de Usuario	
Número: 4	**Usuario:** Desarrollador de CICPC
Nombre historia: Procesar petición Http Ajax	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Yunior Alayo	
Descripción: El usuario genera peticiones asincrónicas al servidor lo cual trae la necesidad de que el sistema soporte el control sobre estas.	
Observaciones:	

Historia de Usuario	
Número: 5	**Usuario:** Desarrollador de CICPC
Nombre historia: Procesar petición de autenticación con reconocimiento humano	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Yunior Alayo	
Descripción: El usuario genera una petición con el objetivo de reconocer si el remitente es quien dice ser y si no es una herramienta automatizada destinada a ataques a la seguridad del sistema.	
Observaciones:	

Historia de Usuario	
Número: 6	**Usuario:** Desarrollador de CICPC
Nombre historia: Procesar petición de autenticación con huella	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Yunior Alayo	
Descripción: El usuario genera una petición con el objetivo de reconocer si el remitente es quien dice ser y con reconocimiento de huellas dactilares.	
Observaciones:	

Historia de Usuario	
Número: 7	**Usuario:** Desarrollador de CICPC
Nombre historia: Obtener permisos del usuario autenticado	
Prioridad en negocio:	**Riesgo en desarrollo:**
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Yunior Alayo	
Descripción: El usuario genera una petición con el objetivo de conocer los permisos del usuario autenticado en el sistema.	
Observaciones:	

2.4 Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Esto se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo, donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario, por ejemplo, las pruebas unitarias, la integración y refactorización del código, y la preparación y ejecución de las pruebas de aceptación.

2.4.1 Estimación de esfuerzos por Historias de Usuario

Historia de Usuario	Puntos de Estimación
Procesar petición Http	1
Procesar petición de salida de la aplicación	1
Procesar petición por canal seguro (Https)	2
Procesar petición Http Ajax	3
Procesar petición de autenticación con reconocimiento humano	3
Procesar petición de autenticación con huella	3
Obtener permisos del usuario autenticado	1

2.4.2 Plan de Iteración

Después de ser descritas e identificadas las historias de usuario y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a la planificación de la etapa de implementación del sistema. Este plan muestra exactamente cuáles historias de usuario serán implementadas para cada iteración del sistema y las posibles fechas para estas liberaciones. En base a lo antes mencionado se decide realizar el sistema en dos iteraciones, las cuales se detallan a continuación:

Iteración 1

Esta iteración tiene como objetivo la implementación de las historias de usuario de mayor prioridad siendo estas necesarias para la implementación de las demás HU. Al finalizar se contará con las funcionalidades descritas en las historias de usuario 1, 3 y 4 las cuales permiten que al sistema se le pueda aplicar la seguridad. Además se tendrá la primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación.

Iteración 2

El objetivo de esta iteración es llevar a cabo la implementación de las restantes historias de usuario teniendo como base las ya implementadas.

2.4.3 Plan de duración de las Iteraciones

Iteraciones	Orden de las HU a implementar	Duración Total
Iteración 1	1. Procesar petición Http 2. Procesar petición por canal seguro (Https) 3. Procesar petición Http Ajax	2 Semanas
Iteración 2	1. Procesar petición de salida de la aplicación 2. Obtener permisos del usuario autenticado 3. Procesar petición de autenticación con reconocimiento humano 4. Procesar petición de autenticación con huella	3 Semanas

2.4.4 Plan de entregas

Con el objetivo de tener una mejor visión de plan de entregas se agruparon las historias de usuario en cuatros sub-componentes abordados anteriormente en la propuesta de solución quedando la distribución de esta manera:

Sub-Componente	Historias de Usuario
Integración Acegi-JSF	1. Procesar petición Http 2. Procesar petición por canal seguro (Https) 3. Procesar petición de salida de la aplicación
Integración Acegi-Ajax	1. Procesar petición Http Ajax
Integración Acegi-Applet	1. Procesar petición de autenticación con reconocimiento humano 2. Procesar petición de autenticación con huella
Integración Acegi-Spring	1. Obtener permisos del usuario autenticado

A continuación tenemos las entregas con sus respectivas fechas:

Sub-Componente	Iteración 1	Iteración 2
Integración Acegi-JSF	v 1.0	v 2.0
Integración Acegi-Ajax	v 1.0	v 2.0
Integración Acegi-Applet		v 1.0
Integración Acegi-Spring		v 1.0

2.5 Conclusiones

En este capítulo hemos abordado lo concerniente a las fases de Exploración y Planificación describiendo detalladamente los artefactos generados. Además se asume una implementación dividida en dos partes determinando los entregables y las fechas de entrega.

Capítulo 3. Implementación de la propuesta.

En el capítulo anterior obtuvimos varios artefactos como las Historias de Usuario, Plan de Iteración, Plan de Entregas entre otros los cuales nos permitirán llevar a cabo la implementación del componente. El proceso seguido será expuesto en el presente capítulo presentando una serie de artefactos obtenidos correspondiente a la metodología utilizada.

3.1 Tareas de Implementación.

Para llevar a cabo la implementación de las Historias de Usuario se hizo necesario dividirlas en tareas de implementación, valga la redundancia, estas serán desarrolladas por parejas de programadores. En el Anexo 1 se encuentran las tareas separadas por la Iteración a que pertenecen.

3.2 Diseño.

Cada uno de los sub-componentes que se implementarán presentan una estructura de paquetes agrupando funcionalidades comunes permitiendo un alto acoplamiento y una baja cohesión.

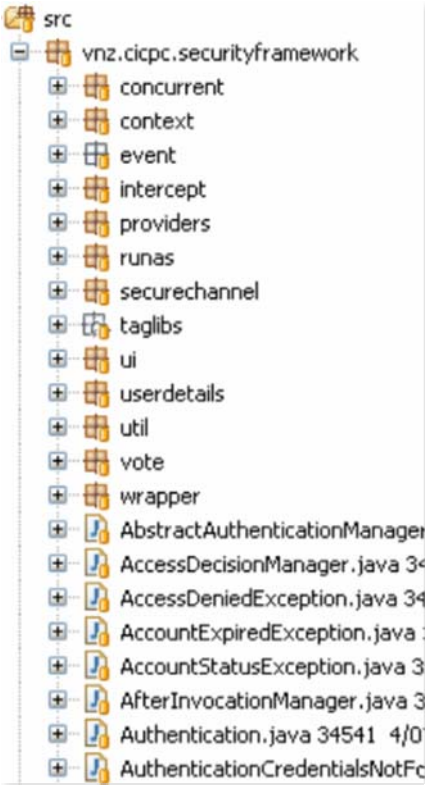

Figura 3.1: Distribución de paquetes del sub-componente Acegi-JSF.

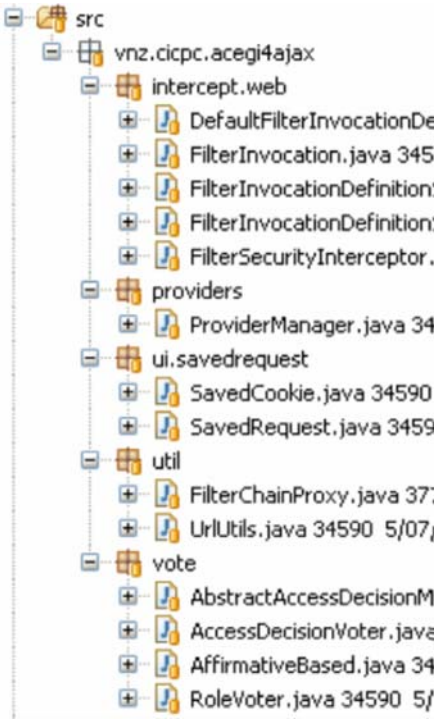

Figura 3.2: Distribución de paquetes de sub-componente Acegi-Ajax.

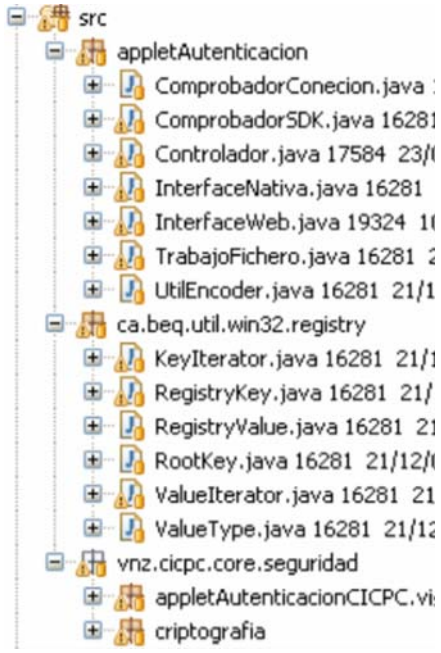

Figura 3.3: Distribución de paquetes del sub-componente Acegi-Applet.

3.3 Implementación.

A continuación explicaremos las clases esenciales de la totalidad de las implementadas con el objetivo de especificar las funcionalidades principales del componente a las que estas dan su aporte.

Acegi-JSF

Para la implementación del sub-componente Acegi-JSF se redefinieron las clases del framework con el objetivo de procesar las peticiones a pesar de las especificidades del framework de presentación JSF.

Las principales clases a redefinir se exponen a continuación:

FilterChainProxy: Esta clase tiene como principal funcionalidad obtener los filtros que deben procesar una petición a una determinada url para luego ejecutarlos en cadena como bien indica su nombre. Esta clase permite que las peticiones sean procesadas por los filtros que se encargan de garantizar la seguridad en la aplicación.

ChannelProcessingFilter: Este filtro se encarga de procesar las peticiones a los recursos de la aplicación para luego colaborando con otra clase decidir si esta petición necesita un canal seguro de comunicación o no.

SecureChannelProcessor: Esta clase tiene la funcionalidad de que una vez que se decide que la petición necesita un canal seguro, colaborando con la clase RetryWithHttpsEntryPoint realiza las operaciones necesarias para que la petición sea procesada a través de un canal seguro valga la redundancia.

Acegi-Ajax

Al igual que para el sub-componente anteriormente explicado para llevar a cabo la implementación del componente en cuestión se redefinieron las clases del framework de seguridad Acegi, en este caso para dar soporte a la tecnología Ajax. A continuación se explican algunas de ellas.

FilterChainProxy: Al igual que anteriormente se implemento esta clase con el mismo nombre pero en sub-componentes distintos además esta clase tiene que tratar con la particularidad de las peticiones ajax que al ser asincrónicas siempre se registra la misma url por tanto esta clase a través de un vinculo con el núcleo del sistema es capaz de saber que url es la que esta siendo solicitada y a partir de ese momento continuar con el proceso estándar de las peticiones.

La clase anteriormente explicada es en resumen la más importante a explicar pero a la vez colabora con otras clases del sub-componente para realizar sus tareas.

3.4 Pruebas Unitarias.

Las pruebas unitarias son algo fundamental en la metodología XP mediante la aplicación del Desarrollo Guiado por Pruebas (TDD por sus siglas en ingles). TDD es una práctica de programación que incluye otras dos prácticas, escribir las pruebas antes de las funcionalidades sobre las cuales estas se aplican y refactorización. Durante el proceso de desarrollo del componente se utilizo el framework JUnit en el desarrollo guiado por pruebas.

3.6 Conclusiones.

En el presente capitulo se expusieron aspectos referentes al proceso de diseño e implementación de la solución generando artefactos específicos de la metodología utilizada y podemos concluir que se tiene un componente listo para ser probado por el cliente en pos de determinar si cumple con las expectativas y explorar la posibilidad de incorporar nuevas funcionalidades a este.

Capítulo 4. Validación de la solución.

En este capítulo se verifica de forma integra que la solución obtenida funcione como es de esperar. Esto se lleva a cabo en conjunto con el cliente.

Además de lo anteriormente planteado se realiza un estudio haciendo énfasis en los aportes de la presente investigación.

4.1 Pruebas de Aceptación.

En la metodología XP las pruebas de aceptación se crean a partir de las Historias de Usuario. El cliente define los escenarios de prueba para verificar si la HU ha sido correctamente implementada. Una HU puede tener una o varias pruebas de aceptación. El cliente es responsable de verificar que el sistema pase las pruebas de aceptación y priorizar la corrección de las pruebas fallidas.

Para la realización de las pruebas de aceptación de la solución obtenida se usaron como artefacto de entrada los Casos de Prueba. Ver Anexo 2.

4.2 Resultados obtenidos.

Durante el proceso de prueba fue necesario realizar tres Iteraciones de pruebas para verificar que la solución cumplía con las necesidades del cliente en alcance, funcionalidad y calidad esperada.

4.3 Conclusiones.

Podemos concluir que al finalizar este capítulo contamos con un componente arquitectónico que es capaz de brindar seguridad al SIIPOL ya que se garantizó que todas las funcionalidades necesarias para ello funcionan de forma correcta a través de las pruebas de aceptación realizadas por el usuario donde todos los casos de prueba tuvieron un resultado satisfactorio.

Conclusiones

Para llevar a cabo esta investigación se siguió la guía brindada por la metodología ágil XP lo cual nos dejo además de la solución obtenida la una posible variante para el desarrollo de componente de esta índole. Este trabajo nos permitió analizar y aprovechar las funcionalidades brindadas por el framework de seguridad Acegi Security System así como descubrir posibles nuevas funcionalidades y áreas a cubrir con respecto a la seguridad como la tecnología ajax que es una de las tendencias actuales en el desarrollo de aplicaciones web.

Recomendaciones

Una vez cumplidos el objetivo general así como los objetivos específicos de la presente investigación permitiendo que el SIIPOL cuente con un componente arquitectónico que garantiza su seguridad se plantean las recomendaciones siguientes:

- Refactorizar de forma tal que pueda ser reutilizado el componente a pesar de que se liberen nuevas versiones del framework de seguridad Acegi.

- Refactorizar para eliminar la dependencia del componente con el sistema actual en que se esta usando.

- Refactorizar el sub-componente Acegi-Applet para que permita la autenticación de usuario multiplataforma.

Referencias Bibliográficas

1. Sun Developer Network. [En línea] Sun Microsystems. [Citado el: 13 de marzo de 2009.] http://developers.sun.com/.

2. **Juan Medín Piñeiro, Antonio García Figueras.** Hacia una arquitectura con JavaServer Faces, Spring, Hibernate y otros frameworks. [En línea] junio de 2006. [Citado el: 13 de marzo de 2009.] http://www.csi.map.es/csi/tecnimap/tecnimap_2006/01T_PDF/hacia%20una%20arquitectura.pdf.

3. JavaHispano. *Versrión 0.7 de Acegi Security System for Spring.* [En línea] 24 de enero de 2005. [Citado el: 13 de marzo de 2009.]

http://javahispano.org/contenidos/es/versrion_0_7_de_acegi_security_system_for_spring/?menuId=tag&onlypath=true.

4. **Berzal, Fernando.** Pruebas de Unidad con JUnit. *Técnicas útiles en el desarrollo del software.* [En línea] [Citado el: 13 de marzo de 2009.] http://elvex.ugr.es/decsai/java/pdf/8C-JUnit.pdf.

5. **Amaro Calderón, Sarah Dámaris, Valverde Rebaza, Jorge Carlos.** *Metodologías Ágiles.* Trujillo-Perú : Universidad Nacional de Trujillo, Facultad de Ciencias Físicas y Matemáticas, Escuela de Informática, 2007.

6. Programación Extrema . *Objetivos de la programación extrema.* [En línea] http://extre.blogspot.com/2006/11/objetivos-de-la-programacin-extrema.html.

7. **Beck, Kent.** *Extreme Programming Explained.* s.l. : Addison Wesley, 1999. ISBN: 0201616416.

8. **Beck, Kent y Fowler, Martin.** *Planning Extreme Programming.* s.l. : Addison Wesley, 2000. ISBN: 0-201-71091-9.

9. **González, Carlos Sánchez.** *ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras.* s.l. : UNIVERSIDADE DA CORUÑA, 2004.

Bibliografía

Acegi, Sitio Oficial de Framework. 2008. http://www.acegisecurity.org/ . [En línea] 2008.

Glosario de Términos

Anexos

Anexo 1: Tareas de Implementación.

Iteración 1

Tarea	
Número de tarea: 1	**Número de HU:** 1
Nombre de la tarea: Analizar implementación existente	
Tipo de tarea: Análisis	**Puntos estimados:** 0.3
Fecha inicio: 16 de Marzo de 2009	**Fecha fin:** 17 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar el código del framework y diseñar la forma en que se va a extender la funcionalidad bridada.	

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Implementación y Pruebas de unidad	
Tipo de tarea: Implementación	**Puntos estimados:** 0.4
Fecha inicio: 17 de Marzo de 2009	**Fecha fin:** 19 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Primeramente implementar las pruebas de unidad correspondientes. Luego implementar las clases necesarias y probar hasta que el código obtenido pase todas las pruebas de unidad.	

Tarea	
Número de tarea: 3	**Número de HU:** 3
Nombre de la tarea: Analizar implementación existente	
Tipo de tarea: Análisis	**Puntos estimados:** 0.4
Fecha inicio: 19 de Marzo de 2009	**Fecha fin:** 20 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar el código del framework y diseñar la forma en que se va a extender la funcionalidad bridada.	

Tarea	
Número de tarea: 4	**Número de HU:** 3
Nombre de la tarea: Implementación y Pruebas de unidad	
Tipo de tarea: Implementación	**Puntos estimados:** 0.33
Fecha inicio: 20 de Marzo de 2009	**Fecha fin:** 23 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Implementar las pruebas de unidad correspondientes. Luego implementar las clases necesarias y probar hasta que el código obtenido pase todas las pruebas de unidad.	

Tarea	
Número de tarea: 5	**Número de HU:** 4
Nombre de la tarea: Analizar implementación existente	
Tipo de tarea: Análisis	**Puntos estimados:** 0.3
Fecha inicio: 23 de Marzo de 2009	**Fecha fin:** 25 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar el código del framework y diseñar la forma en que se va a extender la funcionalidad bridada.	

Tarea	
Número de tarea: 6	**Número de HU:** 4
Nombre de la tarea: Implementación y Pruebas de unidad	
Tipo de tarea: Implementación	**Puntos estimados:** 0.4
Fecha inicio: 25 de Marzo de 2009	**Fecha fin:** 28 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Implementar las pruebas de unidad correspondientes. Luego implementar las clases necesarias y probar hasta que el código obtenido pase todas las pruebas de unidad.	

Iteración 2

Tarea	
Número de tarea: 7	**Número de HU:** 2
Nombre de la tarea: Analizar implementación existente	
Tipo de tarea: Análisis	**Puntos estimados:** 0.2
Fecha inicio: 28 de Marzo de 2009	**Fecha fin:** 29 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar el código del framework y diseñar la forma en que se va a extender la funcionalidad bridada.	

Tarea	
Número de tarea: 8	**Número de HU:** 2
Nombre de la tarea: Implementación y Pruebas de unidad	
Tipo de tarea: Implementación	**Puntos estimados:** 0.3
Fecha inicio: 29 de Marzo de 2009	**Fecha fin:** 30 de Marzo de 2009
Programador responsable: Yunior Alayo	
Descripción: Implementar las pruebas de unidad correspondientes. Luego implementar las clases necesarias y probar hasta que el código obtenido pase todas las pruebas de unidad.	

Tarea	
Número de tarea: 9	**Número de HU:** 5
Nombre de la tarea: Analizar implementación existente	
Tipo de tarea: Análisis	**Puntos estimados:** 0.3
Fecha inicio: 30 de Marzo de 2009	**Fecha fin:** 1 de Abril de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar el código del framework y diseñar la forma en que se va a extender la funcionalidad bridada.	

Tarea	
Número de tarea: 10	**Número de HU:** 5
Nombre de la tarea: Implementación y Pruebas de unidad	
Tipo de tarea: Implementación	**Puntos estimados:** 0.4
Fecha inicio: 1 de Abril de 2009	**Fecha fin:** 4 de Abril de 2009
Programador responsable: Yunior Alayo	
Descripción: Implementar las pruebas de unidad correspondientes. Luego implementar las clases necesarias y probar hasta que el código obtenido pase todas las pruebas de unidad.	

Tarea	
Número de tarea: 11	**Número de HU:** 6
Nombre de la tarea: Analizar implementación existente	
Tipo de tarea: Análisis	**Puntos estimados:** 0.33
Fecha inicio: 4 de Abril de 2009	**Fecha fin:** 6 de Abril de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar el código del framework y diseñar la forma en que se va a extender la funcionalidad bridada.	

Tarea	
Número de tarea: 12	**Número de HU:** 6
Nombre de la tarea: Implementación y Pruebas de unidad	
Tipo de tarea: Implementación	**Puntos estimados:** 0.33
Fecha inicio: 6 de Abril de 2009	**Fecha fin:** 9 de Abril de 2009
Programador responsable: Yunior Alayo	
Descripción: Implementar las pruebas de unidad correspondientes. Luego implementar las clases necesarias y probar hasta que el código obtenido pase todas las pruebas de unidad.	

Tarea	
Número de tarea: 13	**Número de HU:** 7
Nombre de la tarea: Analizar en implementar	
Tipo de tarea: Análisis e Implementación	**Puntos estimados:** 0.3
Fecha inicio: 9 de Abril de 2009	**Fecha fin:** 11 de Abril de 2009
Programador responsable: Yunior Alayo	
Descripción: Analizar de que funcionalidades depende la funcionalidad a implementar para luego pasar al proceso de implementación y pruebas de unidad.	

Anexo 2: Pruebas de Aceptación.

Iteración 1

Caso de Prueba de Aceptación	
Código: CP1HU-1	**Número de HU:** 1
Nombre: Procesar petición Http	
Descripción: Garantiza que se procese una petición http correctamente.	
Condiciones de Ejecución: La aplicación web debe estar disponible.	
Entrada/Pasos de Ejecución: Desde un navegador web escribir una url válida de la aplicación web	
Resultado Esperado: Si el usuario está autenticado se muestra la pagina solicitada Si el usuario no esta autenticado se muestra una pagina de error	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: CP3HU-3	**Número de HU:** 3
Nombre: Procesar petición por canal seguro (Https)	
Descripción: Garantiza que se pueda acceder al sistema a través de un canal seguro.	
Condiciones de Ejecución: La aplicación web debe estar disponible	
Entrada/Pasos de Ejecución: Desde un navegador web escribir una url válida de la aplicación web especificando https como protocolo.	
Resultado Esperado: Si el usuario está autenticado se muestra la página solicitada. Si el usuario no esta autenticado se muestra una pagina de error.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: CP4HU-4	**Número de HU:** 4
Nombre: Procesar petición Http Ajax	
Descripción: Garantiza que se puedan procesar las peticiones ajax.	
Condiciones de Ejecución: La aplicación web debe estar disponible.	
Entrada/Pasos de Ejecución: Acceder a funcionalidades del sistema que usen la tecnología ajax.	
Resultado Esperado: Si el usuario está autenticado se muestra la página solicitada. Si el usuario no esta autenticado se muestra una pagina de error.	
Evaluación de la Prueba: Prueba satisfactoria.	

Iteración 2

Caso de Prueba de Aceptación	
Código: CP2HU-2	**Número de HU:** 2
Nombre: Procesar petición de salida de la aplicación	
Descripción: Garantiza que se pueda salir de la aplicación y a partir de ese momento el usuario no acceda a las funcionalidades del sistema.	
Condiciones de Ejecución: Debe haber un usuario autenticado en el sistema.	
Entrada/Pasos de Ejecución: Acceder a la funcionalidad del sistema que permite salir de la aplicación. Tratar de acceder a una url luego de haber salido.	
Resultado Esperado: Se muestra la página de autenticación del sistema. Se muestra una página de error.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: CP5HU-5	**Número de HU:** 5
Nombre: Procesar petición de autenticación con reconocimiento humano	
Descripción: Garantiza que el sistema permita autenticar usuario con reconocimiento humano.	
Condiciones de Ejecución: La aplicación web debe estar disponible.	
Entrada/Pasos de Ejecución: Acceder a la pagina de autenticación del sistema a través de un navegador. Escribir usuario, contraseña y el texto mostrado en la imagen.	
Resultado Esperado: Si son correctos los datos el usuario entra al sistema. En caso contrario se muestra un mensaje de error.	
Evaluación de la Prueba: Prueba satisfactoria	

Caso de Prueba de Aceptación	
Código: CP6HU-6	**Número de HU:** 6
Nombre: Procesar petición de autenticación con huella	
Descripción: Garantiza que el sistema permita autenticar usuario a través de su huella dactilar.	
Condiciones de Ejecución: La aplicación web debe estar disponible. Un dispositivo capta huellas debe estar disponible en la PC.	
Entrada/Pasos de Ejecución: Acceder a la pagina de autenticación del sistema a través de un navegador. Escribir usuario, contraseña y poner el dedo en el capta huella.	
Resultado Esperado: Si son correctos los datos el usuario entra al sistema. En caso contrario se muestra un mensaje de error.	
Evaluación de la Prueba: Prueba satisfactoria	

Caso de Prueba de Aceptación	
Código: CP7HU-7	**Número de HU:** 7
Nombre: Obtener permisos del usuario autenticado	
Descripción: Garantiza que se conozcan los permisos del usuario autenticado en el sistema.	
Condiciones de Ejecución: Debe haber un usuario autenticado en el sistema. Se deben conocer los permisos que tiene asignado el usuario.	
Entrada/Pasos de Ejecución: Acceder a funcionalidades que tiene permiso en el sistema. Acceder a funcionalidades que no tiene permiso en el sistema.	
Resultado Esperado: Si accedió a una funcionalidad a la que tiene permiso en el sistema se muestra la página correspondiente. Si accedió a una funcionalidad a la que no tiene permiso en el sistema se muestra un mensaje de error.	
Evaluación de la Prueba: Prueba satisfactoria	