Introducción

El número de sistemas informáticos en las organizaciones medianas y grandes se encuentra en continuo crecimiento, por lo que la gestión de usuarios se ha convertido en uno de los problemas más relevantes en este tipo de organizaciones.

Dentro de una organización, los mismos usuarios, con diferentes roles acceden a diferentes aplicaciones o servicios, por lo que la demanda de los usuarios por contar con un único usuario/clave para acceder a las aplicaciones va en un sostenido aumento y es una necesidad que debe atenderse.
Además, no es deseado que los desarrolladores de aplicaciones dentro de una organización deban entender, modelar e implementar el manejo de identidades en sus aplicaciones.
Adicionalmente, se cuenta con plataformas de desarrollo, bases de datos y sistemas operativos heterogéneos, de acuerdo a las necesidades del negocio.

Por esto, las organizaciones modernas necesitan independizar la gestión de usuarios de los sistemas, aún cuando los sistemas están desarrollados en plataformas diferentes y se ejecutan en sistemas operativos heterogéneos.

Los usuarios no pueden dar o delegar permisos a otros usuarios según su criterio, por lo que se requiere de un control central de derechos de acceso. Muchas organizaciones pretenden independizar la gestión de acceso a las aplicaciones del área de informática. El área de administración de la seguridad es la responsable de la aplicación de las políticas adecuadas para la organización.
Por otra parte, el personal de la seguridad informática pretende administrar los derechos de acceso de los usuarios de manera amigable, sin la necesidad de tener conocimientos técnicos específicos sobre la implementación, el software, y el esquema utilizado para lograr el objetivo.

Estas razones nos motivaron al desarrollo de una arquitectura de autenticación que cubra esas expectativas, utilizando herramientas estándares y desarrollo propio.

Índice

Objetivo

1. Autenticación única

Single Sign-On es un proceso de autenticación sesión/usuario que permite a un usuario proveer sus credenciales una sola vez para acceder a múltiples aplicaciones. El SSO autentica al usuario para acceder a todas las aplicaciones a las cuales tiene permitido acceder. Elimina los requerimientos a autenticaciones subsiguientes cuando el usuario cambia de aplicación en una sesión en particular. Los usuarios no autenticados son derivados al servicio de autenticación y retornados a la aplicación solo después de una autenticación exitosa.
Los sistemas de Single Sign-On en organizaciones medianas y grandes pueden convertirse en un punto de falla crítico. Si falla el servicio de Single Sign-On el usuario no podrá acceder a ningún recurso protegido por más que estos se mantengan en funcionamiento. Por lo tanto, es esencial que el sistema de Single Sign-On sea un sistema completamente probado.

Los beneficios del Single Sign-On son:

i. Proveer una política de autenticación/autorización uniforme para toda la organización.
ii. Elimina la necesidad de que los desarrolladores de aplicaciones tengan que implementar el módulo de autenticación/autorización en sus aplicaciones.
iii. Reducción drástica de los pedidos de recuperación de passwords.

2. Control de acceso basado en roles:

Aspectos de las políticas de seguridad: La integridad, disponibilidad, y confidencialidad de las partes de un sistema, base de datos, y redes de datos, son cosas a tener en cuenta por todos los sectores. La corrupción, divulgación no autorizada, o el robo de recursos corporativos puede perturbar las operaciones de una organización y tienen impacto inmediato en la parte financiera, legal, privacidad personal y confianza pública.

En muchas organizaciones, el usuario final no es el dueño de la información a la cual se le permite acceder. El dueño de dicha información es la organización misma, por lo que las decisiones de control de acceso son determinadas por los roles que los usuarios toman como parte de dicha organización. Los roles incluyen las funciones, responsabilidades, y las tareas para las que está calificado el usuario.

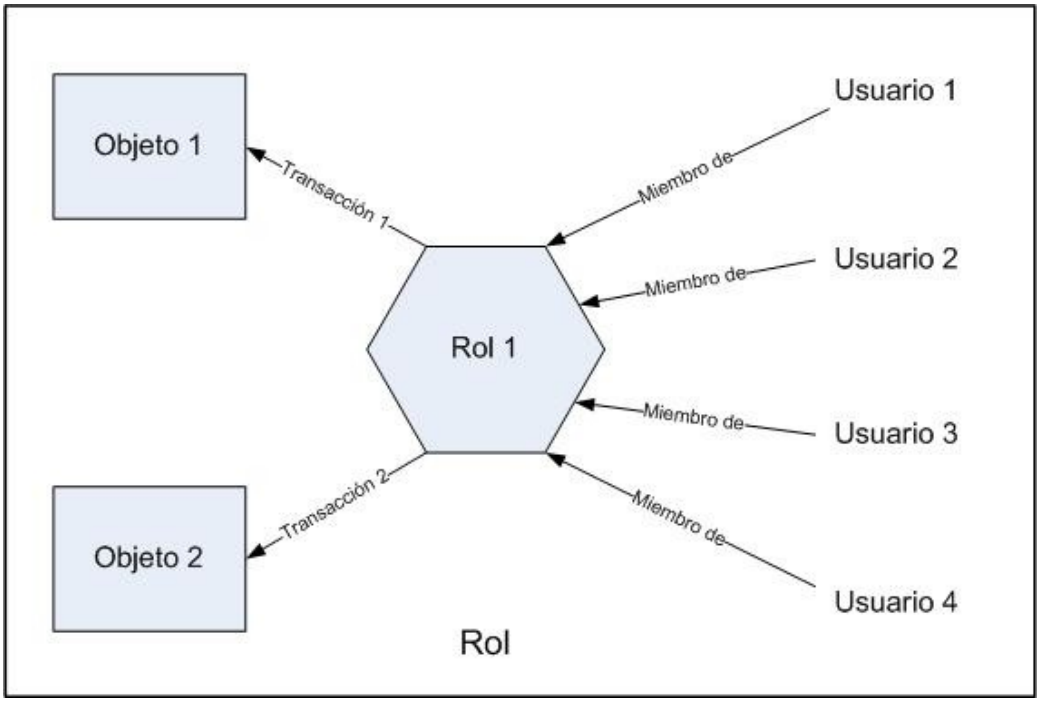
Una política de control de acceso basada en roles (RBAC – Role based access control) basa las decisiones de control de acceso en las funciones que un usuario tiene permitidas realizar dentro de la organización. Los usuarios no pueden dar o delegar permisos a otros usuarios según su criterio, por lo que se requiere de un control central de derechos de acceso. El área de administración de la seguridad es la responsable de la aplicación de las políticas adecuadas para la organización.

La determinación de los miembros y la asignación de transacciones a un rol son generalmente en conformidad a las líneas de protección que especifica la organización. Estas líneas se derivan de leyes existentes, regulaciones, cuestiones éticas, o prácticas aceptadas.
El término transacción es usado para referirse al vínculo entre los procesos de transformación y el acceso al almacenamiento de datos, no se le da el uso convencional.

El control de acceso basado en roles, en muchas aplicaciones tiene que ver más con el acceso a las funciones e información que estrictamente al acceso a la información.
Dentro de un sistema basado en roles, la principal preocupación es proteger la integridad de la información: “quién puede realizar que acción sobre que información”

Un rol puede ser descripto como un conjunto de transacciones que un usuario o conjunto de usuarios puede realizar dentro del contexto de una organización. Las transacciones son asignadas a los roles por un administrador del sistema. Los miembros de un rol también son agregados y eliminados por el administrador del sistema.

Los roles son orientados a grupos. Por cada rol se mantiene un conjunto de transacciones asignadas y tiene un conjunto de miembros asociados. Es decir, el control de acceso basado en roles, da un significado a los nombres de los roles describiendo una relación muchos-a-muchos entre los usuarios y sus derechos de acceso.



Principio de los mínimos privilegios: El principio de los mínimos privilegios requiere que a un usuario no se le otorguen más privilegios que los necesarios para desarrollar su

trabajo. Asegurar los mínimos privilegios requiere identificar cual es el trabajo del usuario, determinar cual es el conjunto mínimo de privilegios que necesita para desarrollar su tarea, y restringir al usuario únicamente a un dominio con los privilegios.

Separación de funciones: El mecanismo de control de acceso basado en roles puede ser utilizado por el administrador del sistema para la aplicación de políticas de separación de funciones. La separación de funciones es considerada valiosa para evitar fraudes en los que existen varias tareas asociadas a un trabajo, ya que no es un solo individuo al que se permite ejecutar todas las transacciones de un conjunto.

3. Repositorio de usuarios:

Consolidación de la información de los usuarios de toda la organización, centralizando (en un repositorio) el dar de alta, baja o cambiar las cuentas y contraseñas de usuarios.

La información acerca de los usuarios se encuentra almacenada frecuentemente en una base de datos especial llamada directorio. El directorio por lo general contiene información más descriptiva y más basada en atributos que la base de datos. La información contenida en un directorio normalmente se lee mucho más de lo que se escribe, por lo que están optimizados para proporcionar una respuesta rápida a operaciones de búsqueda o consulta en lugar de estar preparados para ser eficientes en actualizaciones complejas de grandes volúmenes de datos.
Los directorios pueden aceptar replicación de información con el fin de aumentar la disponibilidad y fiabilidad, y también para reducir el tiempo de respuesta. Se aceptan inconsistencias temporales que luego son sincronizadas.

4. Front-end de administración de usuarios/roles:

El personal de la seguridad informática pretende administrar los derechos de acceso de los usuarios de manera amigable, sin la necesidad de tener conocimientos técnicos específicos sobre la implementación, el software, y el esquema utilizado para lograr el objetivo.
Para esto se desarrollará una aplicación web que permita la administración de usuarios y roles a través de una interfaz sencilla e intuitiva que permita:

I. Administrar aplicaciones
II. Administrar roles
III. Administrar usuarios
IV. Administrar la asignación de roles a usuarios.

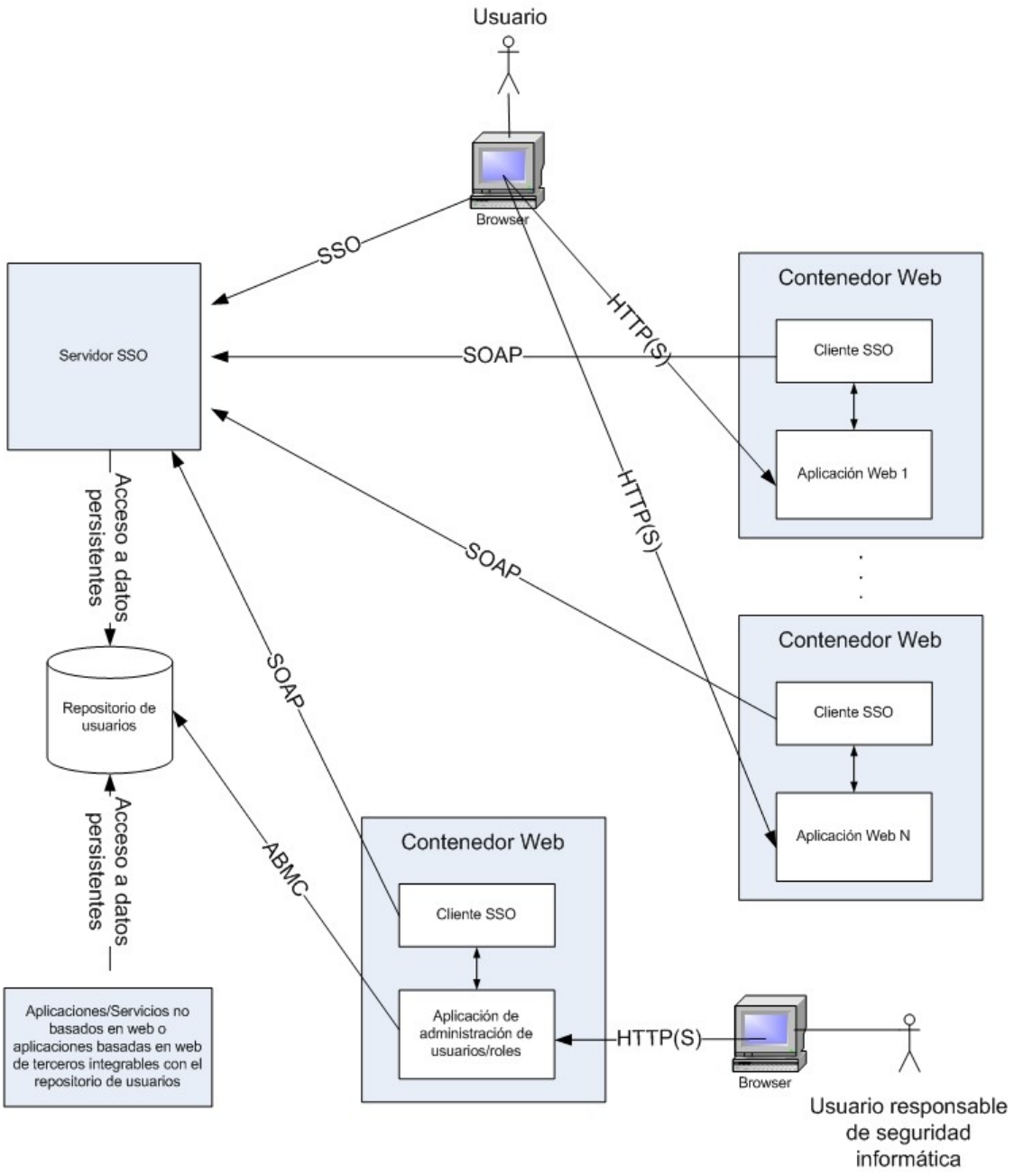
Resumen

A lo largo del desarrollo de esta Tesina de Grado veremos los pasos que hemos seguido para lograr un ambiente de Single sign-on en una organización. En la Parte I propondremos una Arquitectura abstracta donde se describirán los diferentes componentes y actores involucrados en ambiente. En la Parte II analizaremos las alternativas disponibles para cada componente dentro de la arquitectura las cuales evaluaremos para luego tomar la decisión de cuales serán utilizados para este trabajo. En la Parte III instanciaremos la arquitectura propuesta en la Parte I con los componentes seleccionados en la Parte II y describiremos la funcionalidad y la interacción de cada componente para lograr un ambiente de SSO en una organización. En la Parte IV describiremos las instalaciones y configuraciones necesarias para lograr el objetivo. Por último, en la Parte V de esta Tesina de Grado describiremos la funcionalidad de un Portal de Aplicaciones y de una Aplicación para aprovisionamiento de usuarios/roles de desarrollo propio.

Parte I: Arquitectura

En esta sección definiremos de manera abstracta los componentes que estarán involucrados en el ambiente de Single Sign-On basándonos en los objetivos propuestos, los que en posteriormente serán seleccionados e instanciados para llevar a cabo el desarrollo del ambiente.

Arquitectura propuesta:



Componentes

Repositorio de usuarios: Consolidación de la información de los usuarios de toda la organización, centralizando (en un repositorio) el dar de alta, baja o cambiar las cuentas y contraseñas de usuarios.

Servidor de SSO: El servidor de SSO valida la información de autenticación provista por un cliente contra el repositorio de identidades, y luego de una autenticación exitosa, activa una sesión de usuario que es válida sobre todas las aplicaciones web que participan del ambiente de SSO.

Cliente de SSO: Consume servicios del servidor de SSO y permite la integración de las aplicaciones asociadas a SSO al ambiente de ejecución.

Aplicaciones asociadas a SSO: Aplicaciones Web capacitadas para SSO. Estas aplicaciones se basan en los los componentes SSO del Cliente de SSO y del Servidor de SSO para proveer al usuario la experiencia de autenticación única.

Aplicación de administración de usuarios/roles: Aplicación web que permite la administración de usuarios y roles a través de una interfaz sencilla e intuitiva.

Aplicaciones y servicios no basados en web o aplicaciones web de terceros integradas con el repositorio de usuarios: Otras aplicaciones y servicios no basados en web que aprovechan el repositorio de usuarios para mantener nombres de usuario y contraseñas únicas para los usuarios de la organización. También entran en esta categoría aplicaciones web desarrolladas por terceros que permiten la integración con un repositorio de usuarios determinado para mantener la unicidad de nombres de usuario y contraseñas.

Flujo

1. El **Usuario** requiere un recurso protegido de una **Aplicación asociada a SSO.**
2. El **Cliente SSO** integrado a la **Aplicación asociada** intercepta el requerimiento y, al ser un **Usuario** no autenticado, redirecciona al **Usuario** a autenticarse sobre el **Servidor de SSO**.
3. El **Usuario** ingresa sus credenciales.
4. El **Servidor de SSO** procesa las credenciales de **Usuario** y las valida utilizando el **Repositorio de usuarios**.
5. Si las credenciales del **Usuario** son válidas, el **Usuario** es autenticado y se genera una sesión de SSO para el usuario. El **Usuario** es redirigido a la **Aplicación asociada a SSO**.
6. El **Cliente de SSO** que protege la **Aplicación asociada** intercepta el requerimiento, y, utilizando los servicios del **Servidor de SSO**, chequea la validez de la sesión y obtiene el **Usuario** autenticado desde el **Servidor SSO** utilizando el protocolo SOAP.
7. El **Cliente de SSO** introduce el **Usuario** autenticado retornado por el servicio web del **Servidor de SSO** en el requerimiento **HTTP** y lo cede a la aplicación asociada.

Parte II: Selección de alternativas

En esta sección determinaremos cada componente de la arquitectura propuesta. Específicamente seleccionaremos el repositorio que será utilizado para almacenar la información de los usuarios de la organización y el framework de Single Sign-On que permitirá a los usuarios tener un inicio de sesión único para todas las aplicaciones web de la organización.

Capitulo I: Selección del repositorio de usuarios

Las dos alternativas que evaluaremos como repositorio de usuarios son una base de datos relacional y un servicio de directorio. Para llevar a cabo esta tarea describiremos un servicio de directorio y expondremos ventajas y desventajas con respecto a las bien conocidas bases de datos relacionales.

Servicio de directorio

Protocolo LDAP

LDAP ("Lightweight Directory Acces Protocol", en español Protocolo Ligero de Acceso a Directorios) es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio.

Descripción de directorio

Un directorio es una base de datos, pero en general contiene información más descriptiva y más basada en atributos.

La información contenida en un directorio normalmente se lee mucho más de lo que se escribe. Como consecuencia los directorios no implementan normalmente los complicados esquemas para transacciones o esquemas de reducción que las bases de datos utilizan para llevar a cabo actualizaciones complejas de grandes volúmenes de datos. Las actualizaciones en un directorio son usualmente cambios sencillos de todo o nada, si es que permiten algo.

Los directorios están para proporcionar una respuesta rápida a operaciones de búsqueda o consulta.

Los directorios pueden tener capacidad de replicar información de forma amplia, con el fin de aumentar la disponibilidad y fiabilidad, y a la vez reducir tiempo de respuesta. Cuando se duplica la información de un directorio, pueden aceptarse inconsistencias temporales entre la información que hay en las réplicas, siempre que finalmente exista una sincronización.

Hay muchas formas de proporcionar un servicio de directorio. Los diferentes métodos permiten almacenar en el directorio diferentes tipos de información, establecer requisitos diferentes para hacer referencias a la información, consultarla y actualizarla, la forma en que protege al directorio de accesos no autorizados. Algunos servicios de directorios son locales, proporcionando servicios a un contexto restringido. Otros servicios son globales, proporcionando servicio en un contexto mucho más amplio.

Funcionamiento de LDAP

El servicio de directorio LDAP se basa en un modelo cliente-servidor.

Uno o más servidores LDAP contienen los datos que conforman el árbol de directorio LDAP o base de datos troncal, el cliente LDAP se conecta con el servidor LDAP y le hace una consulta. El servidor contesta con la respuesta correspondiente, o bien con una indicación de donde puede el cliente hallar más información. No importa con que servidor LDAP se conecte el cliente ya que siempre observará la misma vista del directorio; el nombre que se le presenta a un servidor LDAP hace referencia a la misma entrada a la que haría referencia en otro servidor LDAP.

Diferencias con una base de datos relacional

El sistema gestor de una base de datos (Database Management System o DBMS) es usado para procesar peticiones (queries) o actualizaciones a una base de datos relacional. Estas bases de datos pueden recibir cientos o miles de órdenes de inserción, modificación o borrado por segundo.

Un servidor LDAP es usado para procesar peticiones (queries) a un directorio LDAP. Pero LDAP procesa las órdenes de borrado y actualización de un modo muy lento. En otras palabras, LDAP es un tipo de base de datos, pero no es una base de datos relacional. No está diseñada para procesar cientos o miles de cambios por minuto como los sistemas relacionales, sino para realizar lecturas de datos de forma muy eficiente.

Las características de una base de datos relacional (RDBMS o Relation Database Management Systems) son:

1. Realizan operaciones de escritura intensivas: las bases de datos relacionales estás preparadas para hacer un uso constante de operaciones orientadas a transacciones, que implican la modificación o borrado constante de los datos almacenados.
2. Esquema específico para cada aplicación: las bases de datos relacionales son creadas para cada aplicación específica, siendo complicado adaptar los esquemas a nuevas aplicaciones.
3. Modelo de datos complejo: permiten manejar complejos modelos de datos que requieren muchas tablas, foreign keys, operaciones de unión (join) complejas,etc.
4. Integridad de datos: todos sus componentes están desarrollados para mantener la consistencia de la información en todo momento. Esto incluye operaciones de rollback, integridad referencial y operaciones orientadas a transacciones.
5. Además las transacciones se efectúan siempre aisladas de otras transacciones. De tal forma que si dos transacciones están ejecutándose de forma concurrente los efectos de la transacción A son invisibles a la

transacción B y viceversa, hasta que ambas transacciones han sido completadas.

6. Disponen de operaciones de roll-back (vuelta atrás). Hasta el final de la transacción ninguna de las acciones llevadas a cabo pasa a un estado final. Si el sistema falla antes de finalizar una transacción todos los cambios realizados son eliminados (roll-back)

Las características de un servidor LDAP son:

1. Operaciones de lectura muy rápidas. Debido a la naturaleza de los datos almacenados en los directorios las lecturas son más comunes que las escrituras.
2. Datos relativamente estáticos. Los datos almacenados en los directorios no suelen actualizarse con mucha frecuencia.
3. Entorno distribuido, fácil replicación.
4. Estructura jerárquica. Los directorios almacenan la información de forma jerárquica de forma nativa.
5. Orientadas a objetos. El directorio representa a elementos y a objetos. Los objetos son creados como entradas, que representan a una colección de atributos.
6. Esquema estándar. Los directorios utilizan un sistema estándar que pueden usar fácilmente diversas aplicaciones.
7. Atributos multi-valor. Los atributos pueden almacenar un valor único o varios.
8. Replicación multi-master. Muchos de los servidores LDAP permiten que se realicen escrituras o actualizaciones en múltiples servidores.

Ventajas en el uso de LDAP

Un directorio LDAP destaca sobre los demás tipos de bases de datos por las siguientes características:

1. Es muy rápido en la lectura de registros.
2. Permite replicar el servidor de forma muy sencilla y económica.
3. Muchas aplicaciones de todo tipo tienen interfaces de conexión a LDAP y se pueden integrar fácilmente.
4. Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.
5. Usa un sistema jerárquico de almacenamiento de información.
6. Permite múltiples directorios independientes.
7. La mayoría de aplicaciones disponen de soporte para LDAP.
8. La mayoría de servidores LDAP son fáciles de instalar, mantener y optimizar.

Usos prácticos de LDAP

Dadas las características de LDAP sus usos más comunes son:

1. Directorios de información**.** Por ejemplo bases de datos de empleados organizados por departamentos (siguiendo la estructura organizativa de la empresa) o cualquier tipo de páginas amarillas.
2. **Sistemas de autenticación/autorización centralizada. Grandes sistemas donde se guarda gran cantidad de registros y se requiere un uso constante de los mismos.**
3. Sistemas de correo electrónico. Grandes sistemas formados por más de un servidor que accedan a un repositorio de datos común.
4. Sistemas de alojamiento de páginas web y FTP, con el repositorio de datos de usuario compartido.
5. Grandes sistemas de autenticación basados en RADIUS, para el control de accesos de los usuarios a una red de conexión o ISP.
6. Servidores de certificados públicos y llaves de seguridad**.**
7. **Autenticación única o "Single Sign-On" para la personalización de aplicaciones.**
8. Libretas de direcciones compartidas.

Servidores LDAP disponibles

Los servidores LDAP más utilizados son:

1. OpenLDAP.
2. Sun SunONE.
3. Siemens DirX Server.
4. Syntegra Intrastore Server.
5. Computer Associates eTrust Directory.
6. Novell NDS Corporate Edition.
7. Microsoft ADS
8. Red Hat Directory Server.
9. Apache Directory Server.
10. Open DS.

Conclusión

Dada las características de un servicio de directorio y las ventajas expuestas sobre una base de datos relacional resultó natural inclinarse por un servicio de directorio para implementar el repositorio de usuarios de la organización.

Un punto importante que fue tenido en cuenta es que muchas aplicaciones y servicios del mercado proveen integración con LDAP para proveer la autenticación de usuarios.

Una vez seleccionado un servicio de directorios nos decidimos por la instalación de un servidor OpenLDAP por ser una implementación libre y de código de abierto, además de ser muy simple su instalación y configuración.

Capitulo II: Selección del framework de autenticación/autorización

Las alternativas tenidas en cuenta para la selección de un framework de autenticación/autorización fueron: CAS, Sun OpenSSO Enterprise, Microsoft Active Directory Federation Services, y JOSSO.
Para determinar el framework de autenticación/autorización a utilizar, primero ser expondrán las funcionalidades que proveen cada uno de los cuatro frameworks nombrados, luego se presentaran las características que hemos tomado en cuenta para la selección, y, por último, analizaremos con cuales de estas características cumple cada framework y con cuales no, para de esta forma determinar cual se utilizará en el desarrollo propuesto.

Presentación de la funcionalidad provista por cada framework de autenticación/autorización

CAS - JA-SIG Central Authentication Service

CAS es una solución Java Enterprise para autenticación de aplicaciones web que provee los beneficios del Single Sign-On (SSO). Técnicamente, SSO puede ser logrado porque la autenticación es quitada de la aplicación web y manejada centralmente, logrando que la autenticación se lleve a cabo una sola vez y "recordada" a lo largo de la sesión.

JA-SIG Central Authentication Service es open-source y está compuesta por un componente servidor Java y varias librerías clientes escritas en varios lenguajes, entre los cuales se encuentran PHP, PL/SQL, Java, ASP, y Perl, entre otros.

CAS logra el Single Sign-On a través de cookies. Las cookies son destruidas una vez que el usuario realiza el logout o una vez que el usuario cierre el browser. Las cookies generadas por CAS son llamadas TGT Cookies (Ticket GranTing Cookie) las cuales contienen un identificador único y un tiempo de expiración, el que normalmente es de 8 horas.

CAS provee diferentes manejadores de autenticación para autenticar credenciales. Los desarrolladores también pueden usar su propio manejador de autenticación. CAS autentica credenciales del tipo nombre de usuario/contraseña, certificados X509, etc. Para autenticar diferentes tipos de credenciales, diferentes tipos de manejadores de autenticación son utilizados.

CAS también provee un componente "Recordarme". Los desarrolladores pueden configurar este componente en los diferentes archivos de configuración y cuando el usuario cliquea sobre el checkbox "Remember me" del formulario de login, recordando sus credenciales por el período de tiempo configurado (3 meses por defecto) redireccionandolo a la respectiva URL sin desplegar el formulario de autenticación aún cuando el usuario abre un nuevo explorador.

Además CAS está basado en los estándares, XML, HTTP, SOAP, SAML, JAAS.

Sun OpenSSO Enterprise

Sun OpenSSO Enterprise integra los servicios de autenticación y autorización, Single Sign-On, y protocolos de federación abiertos basados en estándares, para proveer una solución global para proteger los recursos de red previniendo accesos no autorizados a servicios web, aplicaciones y contenido web.

OpenSSO Enterprise provee administración de acceso permitiendo la implementación de autenticación, autorización basada en políticas de seguridad ("policies"), federación, SSO, y seguridad de servicios web desde un framework unificado. El núcleo de la aplicación es un archivo WAR que puede ser fácilmente desplegado un contenedor web.

Cuando un usuario o una aplicación externa requiere acceso al contenido almacenado en un servidor, un agente de la política de seguridad intercepta el requerimiento y lo dirige a OpenSSO, el cual, a su vez, requiere las credenciales (usuario/password) para la autenticación. Si las credenciales coinciden con las almacenadas en repositorio de usuarios, OpenSSO Enterprise determina que es un usuario autenticado. Luego de la autenticación, el acceso al contenido requerido es determinado por el agente de la política de seguridad que evalúa las políticas asociadas a la identidad. Las políticas son creadas usando OpenSSO Enterprise e identifican que identidades pueden acceder a un recurso en particular, especificando las condiciones bajo las cuales la autorización es válida. Basándose en los resultados de la evaluación de la política de seguridad, el agente otorga o deniega el acceso del usuario.

Funciones de OpenSSO Enterprise

Control de acceso

OpenSSO Enterprise administra los accesos autorizados a los servicios y los recursos de la red. Mediante la implementación de autenticación y autorización, OpenSSO Enterprise asegura que el acceso a recursos protegidos está restringido a usuarios autorizados. En resumen, un agente de la política de seguridad intercepta un requerimiento de acceso a un recurso y se comunica con OpenSSO para autenticar el solicitante. Si el usuario es autenticado exitosamente, el agente de la política de seguridad evalúa la política asociada con el recurso requerido y el usuario para determinar si el usuario autenticado está autorizado a acceder al recurso. Si el usuario está autorizado, el agente de la política de seguridad permite el acceso al recurso, proveyendo también datos acerca de la identidad para personalizar la interacción.

Administración de federación

Con la introducción de los protocolos de federación en el proceso de administración de acceso, la información de identidad y derechos puede ser comunicada a través de dominios de seguridad, abarcando muchos socios de confianza. Mediante la configuración de un *circulo de confianza* y la definición de las aplicaciones y servicios como *proveedores* en el círculo (ya sea como proveedores de identidad o proveedores de servicio), los usuarios pueden optar por asociar, conectar o ligar las identidades que tienen configuradas localmente a estos proveedores. Las identidades locales ligadas son federadas y le permiten al usuario iniciar sesión en un sitio que provee identidad y acceder a través de el a un sitio que provee servicios sin la necesidad de re-autenticarse; Single Sign-On (SSO).

Seguridad en servicios web

Un servicio web es un servicio o aplicación que expone algún tipo de funcionalidad de negocio o infraestructura a través de una interfaz de red independiente del lenguaje y la plataforma; las organizaciones pueden usar estos servicios web para construir grandes arquitecturas orientadas a servicios. En particular, el servicio define su interfaz usando el lenguaje de descripción de servicios web (WSDL), y se comunica usando SOAP y mensajes XML. El cliente del servicio web (WSC) se comunica con el proveedor del servicio web (WSP) a través de un intermediario – normalmente un firewall o balanceador de carga.

Aunque los servicios web están disponibles de forma abierta, esto ocasiona riesgos de seguridad. Sin las protecciones de seguridad apropiadas, un servicio web puede exponer vulnerabilidades que pueden traer consecuencias. Por lo tanto, asegurar la integridad, confidencialidad y seguridad de servicios web a través de la aplicación de un modelo de seguridad es crítico tanto para las organizaciones como para los consumidores. Un modelo de seguridad exitoso asocia los datos de identidad con los servicios web y crea interacciones seguras servicio-a-servicio. El modelo de seguridad adoptado por OpenSSO Enterprise identifica al usuario y preserva su identidad a lo largo de varias interacciones, manteniendo la privacidad y la integridad de datos, usando tecnologías existentes, y registrando las interacciones.

Identidad de servicios web

OpenSSO expone ciertas funciones como servicios web de identidad que permiten a los desarrolladores invocarlos fácilmente cuando desarrollan sus aplicaciones usando uno de los ambientes de desarrollo compatibles (IDE).

Además OpenSSO provee:

- Portabilidad
- Basado en estándares abiertos (SAML, SOAP, XML, HTTP)
- Configuración del repositorio de datos
- Independencia del repositorio de usuarios
- Recursos basados en web y no basados en web
- Arquitectura distribuida
- Internacionalización

Microsoft Active Directory Federation Services

ADFS es un componente de Microsoft® Windows Server™ 2003 R2 que proporciona tecnologías de inicio de sesión único (SSO) Web para autenticar un usuario en varias aplicaciones Web durante una única sesión en línea. ADFS cumple este objetivo mediante el uso compartido seguro de identidades digitales y derechos de uso, o "Notificaciones", a través de límites de seguridad y empresariales.

Funcionalidades de ADFS:

1. Federación y SSO Web

Cuando una organización utiliza el servicio de directorio Active Directory™, actualmente disfruta de la ventaja de la funcionalidad SSO mediante la autenticación integrada en Windows dentro de los límites de empresa o de seguridad de la organización. ADFS amplía esta funcionalidad a las aplicaciones para Internet, lo que permite a los clientes, asociados y proveedores tener una experiencia de usuario SSO Web similar y más eficaz cuando tienen acceso a las aplicaciones basadas en Web de la organización. Además, los servidores de federación se pueden implementar en varias organizaciones para facilitar transacciones federadas business-to-business (B2B) entre organizaciones asociadas.

2. Interoperabilidad de Servicios Web (WS)-*

   ADFS proporciona una solución de administración de identidades federadas que interopera con otros productos de seguridad que admiten la arquitectura de servicios Web WS-*. ADFS lo hace empleando la especificación de federación de WS-*, denominada WS-Federation. La especificación Federación WS lo hace posible para entornos que no utilizan el modelo de identidad de Windows para federar con entornos de Windows. Para obtener información acerca de las especificaciones de WS-*.

3. Arquitectura extensible

   ADFS proporciona una arquitectura extensible que admite el tipo de token del lenguaje de marcado de aserción de seguridad (SAML) y la autenticación Kerberos (en el escenario SSO Web federado con confianza de bosque). ADFS también puede realizar la asignación de notificaciones, por ejemplo, modificando notificaciones utilizando lógica empresarial personalizada como variable en una solicitud de acceso. Las organizaciones pueden utilizar esta extensibilidad para modificar ADFS para que coexista con su infraestructura de seguridad y directivas de negocio actuales.

4. Además, los servicios de federación pueden ser desplegados en varias organizaciones facilitando las transacciones federadas Empresa a Empresa (Business-to-Busines, B2B) entre socios de la organización.

Si utilizan ADFS, las organizaciones pueden ampliar sus infraestructuras de Active Directory existentes para proporcionar acceso a recursos ofrecidos por asociados de confianza a través de Internet. Estos asociados de confianza pueden incluir otras partes externas u otros departamentos o subsidiarias de la misma organización.

ADFS admite la autorización y la autenticación distribuida por Internet. ADFS se puede integrar en la solución de administración de acceso existente de una organización o departamento para traducir las condiciones que se utilizan en la organización a notificaciones acordadas como parte de una federación. ADFS puede crear, proteger y comprobar las notificaciones que se mueven entre organizaciones. También puede auditar y controlar la actividad entre organizaciones y departamentos para ayudar a proteger transacciones seguras.

Agente Web de ADFS

El Agente Web de ADFS es un componente de ADFS. Se utiliza para consumir tokens de seguridad y también para permitir o rechazar acceso a un usuario a una aplicación Web. Para lograr esto, el servidor Web requiere una relación con un Servicio de federación de recursos para poder dirigir al usuario al Servicio de federación conforme sea necesario.

El Agente Web de ADFS se puede utilizar para dos diferentes tipos de aplicaciones:

- Aplicaciones conscientes de demandas: Una aplicación ASP.NET que se escribe para objetos ADFS publicados que permiten la consulta de demandas del token de seguridad ADFS. La aplicación toma decisiones de autorización con base en estas demandas. Las fallas en el token de seguridad para este tipo de aplicación resultan en que el cliente puede ver un mensaje de "Acceso denegado" y los eventos en el registro de sucesos del Servicio de federación.

- Aplicaciones basadas en el token de Windows NT: Una aplicación que utiliza mecanismos de autorización basados en Windows. El Agente Web de ADFS soporta la conversión de un token de seguridad ADFS a un token de acceso de Windows NT a nivel de personificación.

JOSSO – Java Open Single Sign-On

JOSSO, o Java Open Single Sign-On, es una infraestructura de SSO, J2EE open source y basada en Spring que tiente como objetivo proveer una solución para autenticación y autorización de usuarios centralizada y neutral a la plataforma. JOSSO utiliza servicios web para asegurar la identidad del usuario, permitiendo la integración del servicio Single Sign-On a aplicaciones no Java (por ejemplo: PHP, Microsoft ASP) utilizando SOAP sobre el protocolo HTTP.

Componentes principales:

- Single Sign-On cross domain/cross organization transparente a Spring y J2EE.
- Framework que permite la implementación de componentes de identidad a medida usando Spring o un contenedor de IoC (Inversión de control) incorporado.
- Puede ejecutarse en Apache Tomcat y en el servidor de aplicaciones JBoss, entre otros.
- Soporta Apache Http 2.x lo que permite SSO de forma transparente con aplicaciones Ruby, PHP, Python, Perl, etc.
- Está integrado con seguridad Spring para permitir autorización de grano fino.
- Provee información de identidad a aplicaciones Web y EJBs a través de Servlets y la API de seguridad EJB respectivamente.
- Soporta Autenticación robusta usando certificados de cliente X.509.
- Autenticación Windows.

- Admite LDAP para almacenar la información de usuarios y credenciales.
- Admite bases de datos para almacenar la información de usuarios y credenciales.
- Permite la funcionalidad “Recordarme”
- Permite reseteo de contraseñas.
- Provee una API de cliente para PHP, lo que permite SSO para aplicaciones PHP.
- Provee una API de cliente para Microsoft ASP, lo que permite SSO para aplicaciones ASP.
- Está basado en estándares: JAAS, Servicios Web/SOAP, EJB, Struts, Servlets/JSPs, J2EE
- Es 100% Java

Características tomadas en cuenta para la elección

1. Soporte de SSO.
2. Soporte de usuario/roles (autorización).
3. Soporte para lenguajes de programación web más utilizados (JAVA, PHP, ASP).
4. Multiplataforma.
5. Basado en estándares.
6. Open Source.
7. Soporte LDAP.
8. Facilidad de implementación.

Características requeridas con las que cumple cada framework

De las cuatro alternativas propuestas tenemos que:

CAS cumple con las siguientes características:

1. Soporte de SSO
2. Basado en estándares: XML, HTTP, SOAP, SAML.
3. Multiplataforma.
4. Soporte para los lenguajes de programación web más utilizados (Java, PHP, ASP).
5. Open source.
6. Soporte para LDAP.

Características requeridas con las que no cumple CAS:

1. CAS es una solución para autenticación de aplicaciones web, no es una solución para autorización, por lo tanto no cumple con el requisito de soportar control de acceso basado en roles.

ADFS cumple con las siguientes características:

1. Soporte de SSO
2. Soporte de usuario/roles (autorización).
3. Basado en estándares: WS-Federation, X.509, XML, HTTP, SOAP, SAML.
4. Soporte para LDAP a través de ADAM (Active Directory Application Mode).

Características requeridas con las que no cumple ADFS:

1. ADFS es la implementación de Microsoft del protocolo WS-Federation, por lo que no cumple con el requisito de ser open source.
2. ADFS no es multiplataforma, funciona bajo Windows Server 2003 R2 Enterprise Edition.
3. Solo puede ser integrado con aplicaciones desarrolladas con el framework .NET.

OpenSSO cumple con las siguientes características:

1. Soporte de SSO.
2. Soporte de usuario/roles (autorización).
3. Basado en estándares: XML, HTTP, SOAP, SAML.
4. Multiplataforma.
5. Soporte para los lenguajes de programación web más utilizados (Java, PHP, ASP).
6. Open source.
7. Soporte para LDAP.

Características requeridas con las que no cumple OpenSSO:

- No existen características requeridas que no cumpla OpenSSO.

JOSSO cumple con las siguientes características:

1. Soporte de SSO.
2. Soporte de usuario/roles (autorización).
3. Basado en estándares: XML, HTTP, SOAP, SAML.
4. Multiplataforma.
5. Soporte para los lenguajes de programación web más utilizados (Java, PHP, ASP).
6. Open source.
7. Soporte para LDAP.

Características requeridas con las que no cumple JOSSO:

- No existen características requeridas que no cumpla JOSSO.

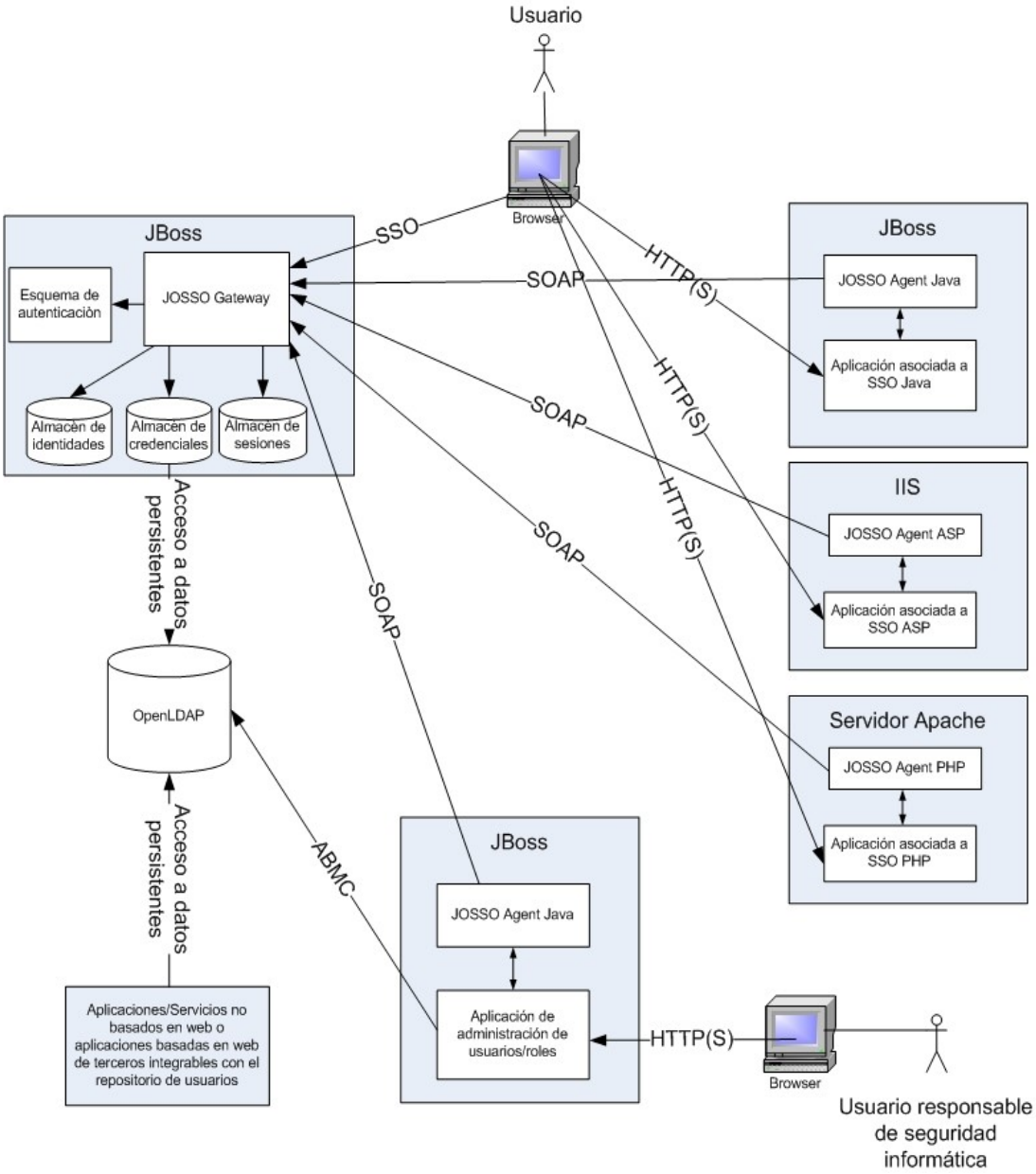
Conclusión

Como se puede observar, las dos alternativas que cumplen con los requisitos propuestos son JOSSO y OpenSSO.
Para la realización del desarrollo hemos optado por la utilización de JOSSO debido a que es considerablemente más simple de instalar y configurar, además de contar con una documentación más orientada a la implementación y configuración del framework.
Cabe destacar que la funcionalidad de protocolos de federación provista por OpenSSO y por ADFS no ha sido tomada en cuenta para la elección del framework de autenticación/autorización.

Parte III: Instanciación de la arquitectura

En esta sección definiremos una arquitectura concreta definiendo los componentes que estarán involucrados en el ambiente de Single Sign-Oon, basándonos en la arquitectura definida en la Parte I y en la selección de los componentes en la Parte II.

Arquitectura instanciada



Componentes

Una vez seleccionadas las alternativas, los componentes de la arquitectura quedan instanciados de la siguiente manera:

Repositorio de usuarios: Servicio de directorio (OpenLDAP)

Servidor de SSO: JOSSO Gateway.

Cliente de SSO: Diferentes JOSSO Agents (JOSSO Agent Java, JOSSO Agent PHP, JOSSO Agent ASP).

Aplicaciones asociadas a SSO: Aplicaciones Web desarrolladas en diferentes lenguajes (JAVA, PHP, ASP).

Aplicación de administración de usuarios/roles: Aplicación web que permite la administración de usuarios y roles a través de una interfaz sencilla e intuitiva.

Aplicaciones y servicios no basados en web o aplicaciones web de terceros integradas con el repositorio de usuarios.

Flujo

1. El **Usuario** requiere un recurso protegido de una **Aplicación asociada a JOSSO.**
2. El **JOSSO Agent** correspondiente a la **Aplicación asociada** intercepta el requerimiento y, al ser un **Usuario** no autenticado, redirecciona al **Usuario** a autenticarse sobre el **JOSSO Gateway**.
3. El **Usuario** ingresa sus credenciales, dependiendo del esquema de autenticación seleccionado, puede requerir un par usuario/contraseña o un certificado de cliente X.509, en este caso se requerirá un par usuario/contraseña.
4. El **JOSSO Gateway** procesa las credenciales de **Usuario** y las valida utilizando el servidor **OpenLDAP**.
5. Si las credenciales del **Usuario** son válidas, el **Usuario** es autenticado y se genera una sesión de **JOSSO** para el usuario, que es guardada en el almacén de sesiones de **JOSSO**. El **Usuario** es redirigido a la **Aplicación asociada a JOSSO**.
6. El **JOSSO Agent** que protege la **Aplicación asociada** intercepta el requerimiento, y, utilizando los servicios del **JOSSO Gateway**, chequea la validez de la sesión y obtiene el **Usuario** autenticado desde el **JOSSO Gateway** utilizando el protocolo **SOAP**.
7. EL **JOSSO Gateway** toma el identificador de sesión provisto desde el almacén de sesiones de JOSSO y obtiene la información del usuario asociado desde el almacén de identidades.
8. El **JOSSO Agent** introduce el **Usuario** autenticado retornado por el servicio web del **JOSSO Gateway** en el requerimiento **HTTP** y lo cede a la **Aplicación web asociada.**

9. La **Aplicación web** procesa el requerimiento **HTTP** y eventualmente usa la información provista en el requerimiento **HTTP** para obtener datos del usuario autenticado.

Componentes JOSSO

JOSSO provee una infraestructura orientada a componentes que permite la creación y combinación de varios esquemas de autenticación, credenciales, y almacenes de sesión con otros componentes.

WebConfiguration: Representa propiedades de configuración comunes relacionadas al Web SSO.

Authenticator: Componente responsable de realizar un esquema de autenticación concreto.

IdentityManager: Componente responsable de la autenticación del usuario usando el Authenticator configurado y de administrar las sesiones de usuarios autenticados.

SessionManager: Componente responsable de administrar el ciclo de vida de una sesión de usuario SSO.

AuditManager: Componente responsable de propagar los eventos SSO que escuchan los Audit Handlers.

EventManager: Componente responsable de propagar eventos SSO genéricos, tales como la información del ciclo de vida de la sesión y los requerimientos de autenticación.

AuthenticationScheme: Representa cualquier esquema de autenticación usado por un Authenticator, tales como el par usuario/contraseña, certificados X.509, etc. Un AuthenticationScheme define el mecanismo usado para autenticar un usuario. Las implementaciones provistas son:

- UserNamePasswordAuthScheme: Esquema de autenticación básico, soporta credenciales del tipo usuario/contraseña.
- X509CertificateAuthScheme: Esquema de autenticación basado en certificados.
- BindUsernamePasswordAuthScheme: Esquema de autenticación básico, soporta credenciales del tipo usuario/contraseña validadas a través de una fuente externa como un servidor LDAP, usando una operación bind.

CredentialStore: Recurso que provee la funcionalidad de almacenamiento de credenciales. Es usado para desacoplar el esquema de autenticación de el mecanismo de persistencia de credenciales.

BindableCredentialStore: Representa un almacén de credenciales que puede ligarse al mecanismo de persistencia usando las credenciales provistas. Usado en combinación con un BindUsernamePasswordAuthScheme permite a JOSSO delegar el proceso de

autenticación al mecanismo de persistencia subyacente, por ejemplo una base de datos relacional o un servidor LDAP tal como OpenLDAP.

IdentityStore: Representa un recurso para almacenar la información de usuario. Las implementaciones definen el mecanismo de persistencia específico para almacenar la información.

- DataSourceIdentityStore: Implementación basada en DataSource de un IdentityStore de Base de Datos y un CredentialStore.
- JDBCIdentityStore: Implementación JDBC de un IdentityStore de Base de Datos y un CredentialStore.
- LDAPBindIdentityStore: Implementación de un IdentityStore y un CredentialStore que obtiene credenciales e información de usuarios y roles desde un servidor LDAP usando JNDI, basándose en las propiedades de la configuración.
- LDAPIdentityStore: Implementación de un IdentityStore y un CredentialStore que obtiene credenciales e información de usuarios y roles desde un servidor LDAP usando JNDI, basándose en las propiedades de la configuración.
- MemoryIdentityStore: Implementación basada en memoria de un IdentityStore y un CredentialStore que lee información desde archivos XML.

SessionStore: Representa un recurso para almacenar sesiones. Las implementaciones definen el mecanismo de persistencia específico para almacenar sesiones.

- DataSourceSessionStore: Un SessionStore de Base de Datos que usa un DataSource para obtener conexiones al servidor de Base de Datos. Permite reconstruir el estado de la sesión luego de apagar el sistema.
- JdbcSessionStore: Este SessionStore de Base de Datos obtiene las conexiones directamente desde el driver JDBC. Permite reconstruir el estado de la sesión luego de apagar el sistema.
- MemorySessionStore: SessionStore basado en memoria que utiliza un Map. Esta implementación es Thread Safe.
- SerializedSessionStore: Implementación de un SessionStore que utiliza la serialización Java para persistir una sesión SSO de usuario. Permite reconstruir el estado de la sesión luego de apagar el sistema.

AuditTrailHandler: Este componente procesa pistas de auditoría generados por el AuditManager. Un AuditHandler puede parar el proceso de SSO como autenticación si es necesario, como parte de la política de seguridad.

- LoggerAuditTrailHandler: Un AuditTrailHandler que solo loguea todos los eventos de una categoría específica.

EventListener: Componente que recibe eventos SSO propagados por el EventManager tales como la autenticación de un usuario, expiración de sesión, etc. El AuditManager es una implementación de EventListener.

Parte IV: Instalaciones y configuraciones

En esta sección describiremos los pasos a seguir para lograr un ambiente de Single Sign-On en una organización, aprovechándonos de la arquitectura definida en la Parte III.III. Más específicamente, en el Capitulo I dictaremos los pasos genéricos que deben llevase a cabo para configurar el ambiente de SSO (Servidor y Cliente de SSO) utilizando el framework JOSSO, en los capítulos subsiguientes mostraremos como instalar y configurar JOSSO en las diferentes arquitecturas. En el Capitulo II mostraremos como instalar y configurar el JOSSO Gateway en un contenedor web JBoss, en el Capitulo III mostraremos como instalar y configurar un JOSSO Agent para un contenedor web JBoss, en el Capitulo IV mostraremos como instalar y configurar un JOSSO Agent para un servidor Internet Information Server y, en el Capitulo V, describiremos los pasos para instalar y configurar un JOSSO Agent para un servidor de aplicaciones Apache con PHP.
Para finalizar la Parte IV se describirá la instalación y configuración de un servidor de directorio OpenLDAP.

Capitulo I: Pasos genéricos para la configuración del ambiente de SSO

Básicamente la configuración del ambiente de SSO utilizando el framework JOSSO consta de dos pasos:

1. Instalación y configuración del JOSSO Gateway.
2. Instalación y configuración de uno más JOSSO Agents.

Para configurar el JOSSO Gateway, además de las configuraciones propias del servidor de aplicaciones en el que se ejecutará, deberemos incluir dos archivos de configuración:

1. josso-config.xml
2. josso-gateway-config.xml

En el archivo de configuración general josso-config.xml simplemente se indica cuales son los archivos de configuración que deben ser tomados al iniciar el servidor. En el caso de que el servidor de aplicaciones actúe solamente como servidor de SSO se deberá incluir en el archivo josso-config.xml únicamente la referencia al archivo de configuración josso-gateway-config.xml, en el caso de que el servidor de aplicaciones actúe solamente como cliente de SSO se deberá incluir en el archivo josso-config.xml únicamente la referencia al archivo de configuración josso-agent-config.xml, y, en el caso de que el servidor de aplicaciones actúe como cliente de SSO tanto como servidor de SSO, se deberán incluir referencias a ambos archivos en el archivo josso-config.xml.

El archivo de configuración josso-gateway-config.xml:

El contenido de este archivo esta fuertemente ligado a los componentes JOSSO que hemos visto en la Parte anterior, ya que en este archivo indicaremos que

implementaciones utilizaremos y los parámetros que estas implementaciones utilizarán para llevar a cabo el SSO.
Específicamente en el archivo josso-gateway-config.xml deberemos indicar:

1. El **Authenticator**.
   El Authenticator es el responsable de realizar el esquema de autenticación. Para eso deberemos indicar: el AuthenticationScheme que utilizaremos (por ejemplo UsernamePasswordAuthSchema), lo que indicará al servidor de SSO el esquema de autenticación que será utilizado, y el CredentialStore (por ejemplo LDAPIdentityStore), que indicará al servidor de SSO como será implementado el almacén de credenciales. En el caso de utilizar un servidor de directorio como almacén de credenciales se indicarán parámetros como el host y puerto donde se encuentra el servidor LDAP, el usuario administrador de LDAP, la raíz del directorio donde estarán almacenados los usuarios, la raíz del directorio donde estarán almacenados los roles, y otras propiedades almacenadas en el directorio.
2. El **IdentityManager.**
   El IdentityManager es el responsable de la autenticación del usuario utilizando el Authenticator y de administrar las sesiones de los usuarios autenticados. Para eso deberemos indicar: la implementación de IdentityStore (por ejemplo LDAPIdentityStore). En el caso de utilizar un servidor de directorio como almacén de credenciales se indicarán parámetros como el host y puerto donde se encuentra el servidor LDAP, el usuario administrador de LDAP, la raíz del directorio donde estarán almacenados los usuarios, la raíz del directorio donde estarán almacenados los roles, y otras propiedades almacenadas en el directorio.
3. El **SessionManager**
   El SessionManager es el responsable de administrar el ciclo de vida de una sesión de usuario SSO. Para eso deberemos indicar el tiempo de expiración de la sesión SSO, y la Implementación de SessionStore utilizada (por ejemplo MemorySessionStore)
4. El **AuditManager**
   El componente AuditManager es el responsable de propagar eventos SSO que escuchan los AuditHandler. Para que el componente logre su objetivo deberemos indicar cual es la implementación de AuditTrailHandler utilizada (por ejemplo LoggerAuditTrailHandler)
5. El **EventManager**
   El componente EventManager es el responsable de propagar eventos SSO genéricos. Para que el componente logre su objetivo deberemos indicar la implementación de EventManager a utilizar (por ejemplo JMXSSOEventManagerImpl).

Para configurar un JOSSO Agent, además de las configuraciones propias del servidor de aplicaciones en el que se ejecutará, deberemos incluir dos archivos de configuración:

1. josso-config.xml
2. josso-agent-config.xml

El archivo de configuración josso-agent-config.xml:

El contenido de este archivo consta de los parámetros que tomará el cliente de SSO para realizar la comunicación y consumir los servicios del servidor de SSO. Básicamente deberemos definir el punto de entrada al servicio web (endpoint) indicando el host y el puerto, las URLs de Login y Logout, y las aplicaciones asociadas a SSO que se encuentran dentro del servidor.

Capitulo II: Instalación y configuración del JOSSO Gateway

Prerequisitos

1. Se asume JBoss 4.0.5.GA.
2. Suponemos seteadas las variables JAVA_HOME y JBOSS_HOME.
3. JDK 1.5
4. JOSSO 1.5.

Instalación

1. Descomprimir el archivo descargado correspondiente a **JOSSO**
2. Editar el archivo build.properties dejando las siguientes opciones en true:

```
exclude.config=true
exclude.samples=true
```

3. Correr los siguientes comandos:

```
./build.sh war
./build.sh install-jboss4
```

4. Para cambiar las pantallas de login, editar los archivos:

```
src/webapp/josso/resources/Aplicaciones.css
src/webapp/josso/signon/error.jsp
src/webapp/josso/signon/info.jsp
src/webapp/josso/signon/layout.jsp
src/webapp/josso/signon/loginResult.jsp
src/webapp/josso/signon/usernamePasswordLogin.jsp
```

5. Finalmente instalar corriendo el siguiente comando:

```
./build.sh deploy-jboss4
```

Configuración

1. Agregar en el archivo $JBOSS_HOME/server/default/conf/login-config.xml dentro del tag policy

```
<application-policy name = "josso">
 <authentication>
  <login-module code = "org.josso.jb4.agent.JBossSSOGatewayLoginModule"
        flag = "required">
    <module-option name="debug">true</module-option>
  </login-module>
 </authentication>
</application-policy>
```

2) Editar el archivo $JBOSS_HOME/server/default/deploy/jbossweb-tomcat55.sar/server.xml

   i. Agregar dentro del tag nombrado a continuación, como primer elemento:

```
<Engine name="jboss.web" defaultHost="localhost">
```

El siguiente contenido:

```
<Realm className="org.josso.jb4.agent.JBossCatalinaRealm"
                appName="josso"
                userClassNames="org.josso.gateway.identity.service.BaseUserImpl"
                roleClassNames="org.josso.gateway.identity.service.BaseRoleImpl"
                debug="1" />
```

   ii. Comentar el tag Realm que existía:

```
<!--
<Realm className="org.jboss.web.tomcat.security.JBossSecurityMgrRealm"
                  certificatePrincipal="org.jboss.security.auth.certs.SubjectDNMapping"
                  allRolesMode="authOnly"/>
 -->
```

   iii. Agregar dentro del tag nombrado a continuación (en cualquier posición):

```
<Host name="localhost"
                autoDeploy="false" deployOnStartup="false" deployXML="false"
                configClass="org.jboss.web.tomcat.security.config.JBossContextConfig"
>
```

El siguiente contenido:

```
<Valve className="org.josso.tc55.agent.SSOAgentValve" debug="1"/>
```

3. Agregar los archivos de configuración siguientes:

```
$JBOSS_HOME/server/default/conf/josso-agent-config.xml
$JBOSS_HOME/server/default/conf/josso-config.xml
$JBOSS_HOME/server/default/conf/josso-gateway-config.xml
```

i. **Archivo josso-agent-config.xml**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<agent>
  <!-- JOSSO Agent classes -->
!-- JOSSO Agent classes -->
    <class>org.josso.jb4.agent.JBossCatalinaSSOAgent</class>
    <!--class>org.josso.tc50.agent.CatalinaSSOAgent</class>-->
    <!--<class>org.josso.tc55.agent.CatalinaSSOAgent</class>-->
    <!--class>org.josso.jb32.agent.JBossCatalinaSSOAgent</class>-->

  <!-- Login/Logout URLs -->
 <gatewayLoginUrl>
          https://192.168.238.3:8443/josso/signon/login.do
</gatewayLoginUrl>
  <gatewayLogoutUrl>
          https://192.168.238.3:8443/josso/signon/logout.do
</gatewayLogoutUrl>
  <!--
  <gatewayLoginErrorUrl>
    http://192.168.238.3:8080/josso/signon/login.do
  </gatewayLoginErrorUrl>
  -->

<!--
    Usefull when working in N-Tier modes behind a reverse proxy or load balancer
    Here you should place the reverse proxy or load balancer base URL.

    Note : When using this options, the gatewayLoginURL and gatewayLogoutURL
should also point to this host.

    <singlePointOfAccess>http://reverse-proxy-host:8080</singlePointOfAccess>

    <gatewayLoginUrl>
```

```
     http://reverse-proxy-host:8080/josso/signon/login.do</gatewayLoginUrl>
    <gatewayLogoutUrl>
     http://reverse-proxy-host:8080/josso/signon/logout.do
    </gatewayLogoutUrl>
-->
   <!-- Mininum interval between sso session access , in milliseconds -->
    <sessionAccessMinInterval>1000</sessionAccessMinInterval>

   <!-- JOSSO Agent service locator configuration -->
  <service-locator>
    <class>org.josso.gateway.WebserviceGatewayServiceLocator</class>
    <endpoint>192.168.238.3:8080</endpoint>
<!-- Associate an identity to SOAP messages
       <username>wsclient</username>
       <password>wsclientpwd</password>
-->
<!-- Enabled SSL on the SOAP circuit.
       <transportSecurity>confidential</transportSecurity>
-->
  </service-locator>
<!--
    JOSSO Parnter application definitions :

    Configure all web applications that should be a josso partner application within this
server.
    For each partner application you have to define the propper web-context.
-->
  <partner-apps>

     <partner-app>
        <context>/Usuarios</context>
<!-- This is an optional feature :
          You can reference any web resource collection that should not be subject to
SSO protection.
          The SSO agent will not provide identity nor demand authentication to requests
matching the security constraint associated to this web resource collections.
          In order to work, the security constraint must not contain auth-constraints
declarations.
          See sample web.xml file from josso partnerapp.
          <security-constraint>
             <ignore-web-resource-collection>public-resources</ignore-web-resource-
collection>
          </security-constraint>
 -->
     </partner-app>
  </partner-apps>
</agent>
```

ii. **Archivo josso-config.xml**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<configuration>
 <hierarchicalXml fileName="josso-gateway-config.xml"/>
<!-- SSO Agent-only entries -->
 <hierarchicalXml fileName="josso-agent-config.xml"/>
 <!-- <hierarchicalXml fileName="josso-reverseproxy-config.xml"/> -->
</configuration>
```

iii. **Archivo josso-gateway-config.xml (considerando soporte de LDAP)**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <domain>
  <name>SampleDomain</name>
  <type>web</type>
  <authenticator>
    <class>org.josso.auth.AuthenticatorImpl</class>

    <authentication-schemes>

       <!-- Basic Authentication Scheme -->
       <authentication-scheme>
         <name>basic-authentication</name>
         <class>org.josso.auth.scheme.UsernamePasswordAuthScheme</class>

         <!-- ================================================ -->
         <!-- LDAP Credential Store                            -->
         <!-- ================================================ -->
         <credential-store>
           <class>org.josso.gateway.identity.service.store.ldap.LDAPIdentityStore</class>
           <initialContextFactory>com.sun.jndi.ldap.LdapCtxFactory</initialContextFactory
>
           <providerUrl>ldap://192.168.238.5:389</providerUrl>
           <securityPrincipal>cn=tesis,dc=tesis</securityPrincipal>
           <securityCredential>tesis1</securityCredential>
           <securityAuthentication>simple</securityAuthentication>
           <!-- Valid values are : SUBTREE, ONELEVEL -->
           <ldapSearchScope>SUBTREE</ldapSearchScope>
           <usersCtxDN>ou=usuarios,dc=tesis</usersCtxDN>
           <principalUidAttributeID>uid</principalUidAttributeID>
           <rolesCtxDN>ou=grupos,dc=tesis</rolesCtxDN>
           <uidAttributeID>uniqueMember</uidAttributeID>
           <roleAttributeID>cn</roleAttributeID>
```

```
        <credentialQueryString>
            uid=username,userPassword=password
        </credentialQueryString>
        <userPropertiesQueryString>
            sn=apellido,givenName=nombre
        </userPropertiesQueryString>

      </credential-store>
      <credential-store-key-adapter>
        <class>
          org.josso.gateway.identity.service.store.SimpleIdentityStoreKeyAdapter
        </class>
      </credential-store-key-adapter>
    </authentication-scheme>
  </authentication-schemes>
</authenticator>

<sso-identity-manager>
  <class>org.josso.gateway.identity.service.SSOIdentityManagerImpl</class>
  <!-- ========================================================= -->
  <!-- LDAP Identity Store                                       -->
  <!-- ========================================================= -->
  <sso-identity-store>
    <class>org.josso.gateway.identity.service.store.ldap.LDAPIdentityStore</class>
    <initialContextFactory>com.sun.jndi.ldap.LdapCtxFactory</initialContextFactory>
    <providerUrl>ldap://192.168.238.5:389</providerUrl>
    <securityPrincipal>cn=tesis,dc=tesis</securityPrincipal>
    <securityCredential>tesis1</securityCredential>
    <securityAuthentication>simple</securityAuthentication>
    <!-- Valid values are : SUBTREE, ONELEVEL -->
    <ldapSearchScope>ONELEVEL</ldapSearchScope>
    <usersCtxDN>ou=usuarios,dc=tesis</usersCtxDN>
    <principalUidAttributeID>uid</principalUidAttributeID>
    <rolesCtxDN>ou=grupos,dc=tesis</rolesCtxDN>
    <uidAttributeID>uniqueMember</uidAttributeID>
    <roleAttributeID>cn</roleAttributeID>
    <credentialQueryString>
       uid=username,userPassword=password
    </credentialQueryString>
    <userPropertiesQueryString>
       sn=apellido,givenName=nombre
    </userPropertiesQueryString>
    <ldapSearchScope>SUBTREE</ldapSearchScope>
  </sso-identity-store>

  <sso-identity-store-key-adapter>
    <class>
    org.josso.gateway.identity.service.store.SimpleIdentityStoreKeyAdapter
    </class>
  </sso-identity-store-key-adapter>
```

```
</sso-identity-manager>

<sso-session-manager>
  <class>org.josso.gateway.session.service.SSOSessionManagerImpl</class>

  <!--
  Set the maximum time interval, in minutes, between client requests
  before the SSO Service will invalidate the session. A negative time
  indicates that the session should never time out.
  -->
  <maxInactiveInterval>1</maxInactiveInterval>

  <sso-session-store>
    <class>
    org.josso.gateway.session.service.store.MemorySessionStore
    </class>
  </sso-session-store>

  <sso-session-id-generator>
    <class>
     org.josso.gateway.session.service.SessionIdGeneratorImpl
    </class>
    <!--
    The message digest algorithm to be used when generating session
    identifiers.  This must be an algorithm supported by the
    java.security.MessageDigest class on your platform.

    In J2SE 1.4.2 you can check :
    Java Cryptography Architecture API Specification & Reference -
    Apendix A : Standard Names
    Values are : MD2, MD5, SHA-1, SHA-256, SHA-384, SHA-512
    -->
    <algorithm>MD5</algorithm>
  </sso-session-id-generator>
</sso-session-manager>

<sso-audit-manager>
    <class>
        org.josso.gateway.audit.service.SSOAuditManagerImpl
    </class>
    <handlers>
        <handler>
            <class>
                    org.josso.gateway.audit.service.handler.LoggerAuditTrailHandler
            </class>
            <name>LoggerAuditTrailHandler</name>
            <category>org.josso.gateway.audit.SSO_AUDIT</category>
        </handler>
    </handlers>
</sso-audit-manager>
```

```
 <sso-event-manager>
    <class>
        org.josso.gateway.event.security.JMXSSOEventManagerImpl
    </class>
    <oname>josso:type=SSOEventManager</oname>
 </sso-event-manager>

</domain>
```

Capitulo III: Instalación y configuración del JOSSO Agent en un contenedor web JBoss

Prerequisitos

1. Se asume JBoss 4.0.5.GA.
2. JDK 1.5
3. JOSSO 1.5.

Instalación

1. Descomprimir el archivo descargado correspondiente a **JOSSO**
2. Editar el archivo build.properties dejando las siguientes opciones en true:

```
exclude.config=true
exclude.samples=true
```

3. Correr los siguientes comandos:

```
./build.sh war
./build.sh install-jboss4
```

4. Finalmente instalar corriendo el siguiente comando:

```
./build.sh deploy-jboss4
```

Configuración

1. Agregar en el archivo $JBOSS_HOME/server/default/conf/login-config.xml dentro del tag policy

```
<application-policy name = "josso">
 <authentication>
   <login-module code = "org.josso.jb4.agent.JBossSSOGatewayLoginModule"
         flag = "required">
     <module-option name="debug">true</module-option>
   </login-module>
 </authentication>
</application-policy>
```

2. Editar el archivo $JBOSS_HOME/server/default/deploy/jbossweb-tomcat55.sar/server.xml
    i. Agregar dentro del tag nombrado a continuación, como primer elemento:

```
<Engine name="jboss.web" defaultHost="localhost">
```

El siguiente contenido:

```
<Realm className="org.josso.jb4.agent.JBossCatalinaRealm"
                appName="josso"
                userClassNames="org.josso.gateway.identity.service.BaseUserImpl"
                roleClassNames="org.josso.gateway.identity.service.BaseRoleImpl"
                debug="1" />
```

ii. Comentar el tag Realm que existía:

```
<!--
<Realm className="org.jboss.web.tomcat.security.JBossSecurityMgrRealm"
                    certificatePrincipal="org.jboss.security.auth.certs.SubjectDNMapping"
                    allRolesMode="authOnly"/>
 -->
```

iii. Agregar dentro del tag nombrado a continuación (en cualquier posición):

```
<Host name="localhost"
                autoDeploy="false" deployOnStartup="false" deployXML="false"
                configClass="org.jboss.web.tomcat.security.config.JBossContextConfig"
>
```

El siguiente contenido:

```
<Valve className="org.josso.tc55.agent.SSOAgentValve" debug="1"/>
```

3. Agregar los archivos de configuración siguientes:

```
$JBOSS_HOME/server/default/conf/josso-agent-config.xml
$JBOSS_HOME/server/default/conf/josso-config.xml
```

- **Archivo josso-config.xml**

```
<configuration>
  <hierarchicalXml fileName="josso-agent-config.xml"/>
</configuration>
```

- **Archivo josso-agent-config.xml**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<agent>
  <class>org.josso.jb4.agent.JBossCatalinaSSOAgent</class>
   <gatewayLoginUrl>https://192.168.238.3:8443/josso/signon/login.do</gatewayLoginU
rl>
    <gatewayLogoutUrl>https://192.168.238.3:8443/josso/signon/logout.do</gatewayLog
outUrl>
  <service-locator>
    <class>org.josso.gateway.WebserviceGatewayServiceLocator</class>
    <endpoint>192.168.238.3:8080</endpoint>
  </service-locator>
  <partner-apps>
      <partner-app>
         <context>/AgenteJava</context>
      </partner-app>
      <partner-app>
         <context>/Usuarios</context>
      </partner-app>
  </partner-apps>
</agent>
```

Usando este archivo se puede configurar:

- La URL de Login del Gateway, que representa la URL donde el usuario será redirigido en el acceso a un recurso protegido.
- La URL de Logout del Gateway, que representa la URL donde el usuario será redirigido en un requerimiento de logout.
- El Service Locator que será usado para invocar los servicios de Single Sign-On del Gateway.
- Las aplicaciones protegidas por JOSSO

Utilización

Una aplicación web segura requiere que el usuario inicie sesión para acceder al recurso protegido. Las credenciales del usuario son verificadas contra un realm de

seguridad, una vez autenticado, el acceso sera otorgado solo a los recursos especificados dentro de la aplicación web.

La seguridad en una aplicación web son configurados usando tres elementos:

1. El elemento <login-config> especifica como el usuario es redirigido a iniciar sesión y la ubicación del realm de seguridad. Si el elemento está presente, el usuario debe estar autenticado para acceder a cualquier recurso que esté restringido por un elemento <security-constraint> definido en la aplicación web.
2. Un elemento <security-constraint> es usado para definir los privilegios de acceso a una colección de recursos a través de su URL.
3. Un elemento <security-role> representa un grupo en el realm. Este nombre de rol de seguridad es utilizado en el elemento <security-constraint> y puede ser ligado a un nombre de rol alternativo en el código del servlet a través del elemento <security-role-ref>

- **Archivo web.xml**

```
<welcome-file-list>
        <welcome-file>/index.jsp</welcome-file>
    </welcome-file-list>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>
                protected-resources
            </web-resource-name>
            <url-pattern>/faces/pages/*</url-pattern>
            <http-method>HEAD</http-method>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
            <http-method>PUT</http-method>
            <http-method>DELETE</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>SISTEMA_AGENTE_JAVA</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
    </security-constraint>

    <login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
            <form-login-page>/faces/login-redirect.jsp</form-login-page>
            <form-error-page>/faces/login-redirect.jsp</form-error-page>
        </form-login-config>
    </login-config>
```

```
<security-role>
        <description>Rol Agente Java</description>
        <role-name>SISTEMA_AGENTE_JAVA</role-name>
</security-role>
```

En particular este archivo web.xml especifica que solo los usuarios asociados con el rol SISTEMA_AGENTE_JAVA pueden acceder a la aplicación.

Cuando un usuario no autenticado requiere acceso a la aplicación, será redirigido a la página /faces/login-redirect.jsp, la que redirigirá, a su vez, al formulario de autenticación de JOSSO.

2. **login-redirect.jsp**

```
<%@page contentType="text/html; charset=iso-8859-1" language="java"
session="true" %>
<% response.sendRedirect(request.getContextPath() + "/josso_login/"); %>
```

Una vez autenticado el usuario puede accederse a la aplicación. Desde la aplicación es posible acceder a la información del usuario a través de la API estándar de seguridad de Servlet:

```
request.getRemoteUser()
```

El método anterior retorna el nombre de usuario del usuario que ingresó a la aplicación.

También se puede acceder a las propiedades definidas en el elemento <userPropertiesQueryString> definidas en el archivo de configuración josso-gateway-config.xml.

```
<userPropertiesQueryString>
        sn=apellido,givenName=nombre
</userPropertiesQueryString>
```

Estas propiedades también estarán disponibles a través de la API estándar de seguridad de Servlet, permitiendo al usuario acceder a propiedades adicionales sin tener que realizar una consulta al recurso que contiene la información adicional.

Para poder acceder a las propiedades debe castearse el Usuario Principal asociado al HttpServletRequest de la siguiente manera:

```
org.josso.gateway.identity.SSOUser ssoUser =
            (org.josso.gateway.identity.SSOUser)request .getUserPrincipal();
```

Estas propiedades se encuentran almacenadas en las instancias de la clase org.josso.gateway.identity.SSOUser como un arreglo de objetos de la clase org.josso.gateway.SSONameValuePair.

```
for (int i = 0 ; i < ssoUser.getProperties().length ; i++){
        ssoUser.getProperties()[i].getName();
        ssoUser.getProperties()[i].getValue();
}
```

Por último, para chequear si un usuario tiene autorización para una acción determinada o un recurso utilizamos:

```
request.isUserInRole("SISTEMA_ADMINISTRADOR")
```

Capitulo IV: Instalación y configuración del JOSSO Agent en un servidor Internet Information Server

Prerequisitos

1. Java Runtime Environment (JRE) 5.0
2. JOSSO 1.5
3. JOSSO Gateway setup
4. Internet Information Server (ISS)
5. Variable de entorno JRE_HOME seteada.

Instalación

La integración con ASP es lograda a través de un Java Bean wrappeado en un componente ActiveX.

1. Descomprimir el archivo josso-1.5.tar.gz
2. Setear la variable JOSSO_HOME apuntando al directorio donde se descomprimió el archivo josso-1.5.tar.gz. Suponemos que dicho directorio es c:\josso

```
set JOSSO_HOME=c:\josso
```

3. Crear la ubicación donde se alojará el cliente ActiveX de JOSSO:

```
mkdir %JRE_HOME%\axbridge\
mkdir %JRE_HOME%\axbridge\bin
```

4. Instalar y registrar el Agente ASP ActiveX de JOSSO que se encuentra en %JOSSO_HOME%\src\plugins\microsoft\dist

```
copy %JOSSO_HOME%src\plugins\microsoft\dist\JOSSOActiveX.dll %JRE_HOME
%\axbridge\bin
cd %JRE_HOME%\axbridge\bin
regsvr32 JOSSOActiveX.dll
```

Si la operación fue exitosa el ActiveX será identificado como:

clsid={A9D9756E-DCC3-4566-AD99-4153C2C06BD8}
progid=JOSSOActiveX.Bean.1

5. Crear el directorio donde se ubicará la aplicación ASP que será protegida por JOSSO:
   Suponemos que la aplicación se denominará AgenteAsp

```
mkdir c:\inetpub\AgenteAsp
```

6. Ejecutar la herramienta administrativa “Internet Services Manager” y agregar un directorio virtual asociado al directorio creado
7. Crear un directorio en el directorio virtual para contener los archivos que darán soporte JOSSO a la aplicación.

```
mkdir c:\inetpub\AgenteAsp\josso-asp
```

8. Copiar los archivos ubicados en %JOSSO_HOME %\src\plugins\microsoft\asp\josso\:
   - josso-debug.asp
   - josso-login.asp
   - josso-logout.asp
   - josso-security-check.asp
   - josso.asp

```
copy %JOSSO_HOME%\src\plugins\microsoft\asp\josso\*.*
c:\inetpub\AgenteAsp\josso-asp
```

Configuración

La configuración del cliente ASP debe realizarse en el archivo global.asa.

- **Archivo global.asa**

```
<object runat="server" scope="application" id="josso"
progid="JOSSOActiveX.Bean.1">
</object>

<!--METADATA TYPE="TypeLib"
        uuid="{FA564C45-6AE6-4610-9C50-C5B2D37AD9BB}"
-->

<script language="vbscript" runat="server">
 sub Application_OnStart

' JOSSO Gateway SOAP end point
   josso.setProperty "gwy.endpoint", "192.168.238.3:8080"
```

```
' JOSSO Gateway Login/Logout urls
    josso.setProperty "gwy.login", "https://192.168.238.3:8443/josso/signon/login.do"
    josso.setProperty "gwy.logout", "https://192.168.238.3:8443/josso/signon/logout.do"

' JOSSO ASP Agent base code i.e. /josso-asp for http://my-host.com/josso-
asp/josso.asp, etc
    josso.setProperty "agent.basecode", "/AgenteAsp/josso-asp"

' JOSSO Log4J configuration properties (Opcional!)
' josso.setLog4jProperties("c:\josso-wa\log4j.properties")

' Initialize josso object
    josso.init()
  end sub
</script>
```

Usando este archivo se puede configurar:

- La URL de Login del Gateway, que representa la URL donde el usuario será redirigido en el acceso a un recurso protegido.
- La URL de Logout del Gateway, que representa la URL donde el usuario será redirigido en un requerimiento de logout.
- El endpoint del Gateway, que representa la ubicación donde los servicios web de josso están escuchando.
- La base del código del Agente ASP: por ejemplo, http://localhost/AgenteAsp/josso-asp

Utilización

Para proteger las páginas de una aplicación ASP deberá incluirse el script josso.asp en todas las páginas de la aplicación.

```
<!--#include file='josso-asp/josso.asp'-->
```

El usuario y la información de la sesión pueden ser obtenidos de la siguiente manera:

```
set jossoUser = jossoCurrentUser()
dim jossoSession
jossoSession = getJOSSOToken()
```

Para chequear si el usuario está autenticado:

```
if jossoUser is nothing then
   jossoRequestLogin()
end if
```

Desde la aplicación es posible acceder a la información del usuario a través del método:

```
jossoUser.getName()
```

El método anterior retorna el nombre de usuario del usuario que ingresó a la aplicación.

También se puede acceder a las propiedades definidas en el elemento <userPropertiesQueryString> definidas en el archivo de configuración josso-gateway-config.xml.

```
<userPropertiesQueryString>
        sn=apellido,givenName=nombre
</userPropertiesQueryString>
```

Estas propiedades también estarán disponibles a través del método josso.getUserProperties() , permitiendo al usuario acceder a propiedades adicionales sin tener que realizar una consulta al recurso que contiene la información adicional.

```
dim jossoProperties
set jossoProperties = josso.getUserProperties()
dim i, jossoProperty

For i = 0 to ( jossoProperties.count() - 1  )
          jossoProperties.getName(i)
          jossoProperties.getValue(i)
Next
```

También puede accederse a la colección de roles de un usuario a través del método findRolesByUsername($user->getName()):

```
dim jossoRoles
set jossoRoles = josso.getUserRoles(jossoUser.getName())

dim j, jossoRole
For j = 0 to ( jossoRoles.count() - 1  )
set jossoRole = jossoRoles.getRole(j)
```

```
        jossoRole.getName()
Next
```

Por último, para chequear si un usuario tiene autorización para una acción determinada o un recurso utilizamos:

```
if josso.isUserInRole(jossoSession, "SISTEMA_AGENTE_ASP_ROL1") then
```

Capitulo V: Instalación y configuración del JOSSO Agent en un servidor de aplicaciones Apache con PHP

Prerequisitos

1. PHP 4 o 5
2. JOSSO 1.5
3. JOSSO Gateway instalado

Instalación

1. Descomprimir el archivo josso-1.5.tar.gz

2. Instalar las páginas de JOSSO PHP en la aplicación, copiar los archivos ubicados en $JOSSO_DIR/josso-1.5/src/plugins/php/php en la carpeta que contiene la aplicación.

Los archivos son los siguientes:

- class.jossoagent.php
- class.jossorole.php
- class.jossouser.php
- josso.php
- josso-login.php
- josso-logout.php
- josso-security-check.php
- y la carpeta nusoap.

Configuración

1. **Archivo josso-cfg.inc**

```
<?php

// Josso agent configuration
$josso_gatewayLoginUrl = 'https://192.168.238.3:8443/josso/signon/login.do';
$josso_gatewayLogoutUrl = 'https://192.168.238.3:8443/josso/signon/logout.do';

// WS client configuration :
$josso_endpoint = 'http://192.168.238.3:8080';

// This could be also /, it points to the path where JOSSO code is found, for example the josso-security-check.php page.
$josso_agentBasecode = "/AgentePhp/josso";

$josso_proxyhost = '';
$josso_proxyport = '';
```

```
$josso_proxyusername = '';
$josso_proxypassoword = '';

// Min. session access interval for each user, in seconds.
$josso_sessionAccessMinInterval=1;

?>
```

Usando este archivo se puede configurar:

- La URL de Login del Gateway, que representa la URL donde el usuario será redirigido en el acceso a un recurso protegido.
- La URL de Logout del Gateway, que representa la URL donde el usuario será redirigido en un requerimiento de logout.
- El endpoint del Gateway, que representa la ubicación donde los servicios web de josso están escuchando.
- La base del código del Agente PHP: por ejemplo, http://localhost/AgentePhp/josso

Utilización

Para proteger las páginas de una aplicación php deberá incluirse el script josso.php en todas las páginas de la aplicación.

```
include "./josso/josso.php";
```

El usuario y la información de la sesión pueden ser obtenidos de la siguiente manera:

```
$user = $josso_agent->getUserInSession();
$sessionId = $josso_agent->getSessionId();
```

Para chequear si el usuario está autenticado:

```
if (isset($user))
```

Desde la aplicación es posible acceder a la información del usuario a través del método:

```
$user->getName()
```

El método anterior retorna el nombre de usuario del usuario que ingresó a la aplicación.

También se puede acceder a las propiedades definidas en el elemento <userPropertiesQueryString> definidas en el archivo de configuración josso-gateway-config.xml.

```
<userPropertiesQueryString>
        sn=apellido,givenName=nombre
    </userPropertiesQueryString>
```

Estas propiedades también estarán disponibles a través del método $user->getProperties(), permitiendo al usuario acceder a propiedades adicionales sin tener que realizar una consulta al recurso que contiene la información adicional.

```
$properties = $user->getProperties();
if (is_array($properties)) {
          foreach ($properties as $property) {
          $nombrePropiedad=$property['name'];
                    $valorPropiedad=$property['value'];
    }
}
```

También puede accederse a la colección de roles de un usuario a través del método findRolesByUsername($user->getName()):

```
$roles = $josso_agent->findRolesByUsername($user->getName());
foreach ($roles as $role) {
          $nombreDelRol=$role->getName()
}
```

Por último, para chequear si un usuario tiene autorización para una acción determinada o un recurso utilizamos:

```
  if ($josso_agent->isUserInRole('SISTEMA_AGENTE_PHP'))
```

Capitulo VI: Instalación y configuración de un servidor de directorios OpenLDAP

Prerequisitos

1. Se supone la versión de OpenLDAP 2.4.11
2. Se supone descargado el archivo openldap-2.4.11.tgz (puede conseguirse en ftp://ftp.openldap.org/pub/OpenLDAP/openldap-release/)

Instalación

1. Descomprimir el archivo openldap-2.4.11.tgz

```
tar xfz openldap-2.4.11.tgz
```

2. Posicionarse en el directorio donde se descomprimió el archivo: (suponemos que el directorio es /instalacion/openldap/

```
cd /instalación/openldap
```

3. Ejecutar los siguientes comandos:

```
./configure
```

```
make depend
```

```
make
```

```
make install
```

Configuración

1. Editar el archivo /etc/ldap/ldap.conf

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE  dc=tesis
URI     ldap://192.168.238.5:389

#SIZELIMIT    12
#TIMELIMIT    15
#DEREF                never
```

2. Editar el archivo /etc/ldap/sldap.conf

```
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema

# Un esquema describe uno o más objetos que pueden existir en un directorio LDAP.
#Los esquemas core.schema y cosine.schema incluyen descripciones de todos los
#objetos de bajo nivel que son necesarios para el esqueleto de un directorio LDAP. El
#esquema inetorgperson.schema describe los objetos inetOrgPerson que se integran
#a la mayoría de los directorios.

pidfile         /var/run/slapd/slapd.pid
argsfile        /var/run/slapd/slapd.args

# OpenLDAP almacena su ID de proceso en un archivo nombrado por la variable de
#configuración pidfile. Los scripts para iniciar y detener el servidor OpenLDAP usan el
#contenido de este archivo.

#La variable argsfile especifica el archivo que contiene los argumentos para la linea de
#comandos. El servidor OpenLDAP usará estos argumentos automáticamente cada
# vez que se inicie.

modulepath    /usr/lib/ldap
moduleload    back_bdb

loglevel        none
```

```
# loglevel none indica que no existe logging.

database bdb

# La variable database especifica la base de datos de back-end usada por OpenLDAP
#para almacenar los datos. El módulo bdb es usado para soportar Berkeley DB.

suffix "dc=tesis"

# La variable de configuración suffix especifica el nombre de la entrada base en el
#directorio. Todas las demás entradas del directorio serán descendientes de este
#objeto. Esta variable estará basada en el nombre del dominio.

rootdn "cn=tesis,dc=tesis"
rootpw tesis1

# Cada directorio LDAP tiene un nombre distinguido raíz (DN), el cual a grandes
#rasgos es análogo al usuario root del sistema UNIX. El usuario especificado por la
#variable rootdn puede leer, escribir y buscar en cualquier parte del directorio.

# La variable de configuración rootpw especifica el password de la cuenta root DN

directory       "/var/lib/ldap"

La variable directory especifica la ubicación en el sistema de archivos donde los
archivos de la base de datos que contiene los datos del directorio están almacenados.

index           objectClass eq

La variable index especifica los atributos de objeto para los cuales OpenLDAP
mantendrá indices. Un índice puede reducir considerablemente la cantidad de tiempo
necesario para encontrar un objeto basado en un atributo en particular.
```

3. Agregar las entradas iniciales en el directorio
    i. Crear un archivo LDIF (entradas_iniciales.ldif) con el siguiente contenido:

```
# tesis
dn: dc=tesis
dc: tesis
objectClass: dcObject
objectClass: organization
o: tesis

# usuarios, tesis
dn: ou=usuarios,dc=tesis
description: Usuarios
```

```
objectClass: organizationalUnit
ou: usuarios

# grupos, tesis
dn: ou=grupos,dc=tesis
description: Grupos
objectClass: organizationalUnit
ou: grupos
```

ii. Ejecutar ldapadd para insertar las entradas en el directorio:

```
ldapadd -D "cn=tesis, dc=tesis" -W < entradas_iniciales.ldif
```

Parte V: Aplicación de aprovisionamiento de usuarios y roles de desarrollo propio

Muchas organizaciones pretenden independizar la gestión de acceso a las aplicaciones del área de informática. El área de administración de la seguridad es la responsable de la aplicación de las políticas adecuadas para la organización.

Por otra parte, el personal de la seguridad informática pretende administrar los derechos de acceso de los usuarios de manera amigable, sin la necesidad de tener conocimientos técnicos específicos sobre la implementación, el software, y el esquema utilizado para lograr el objetivo.

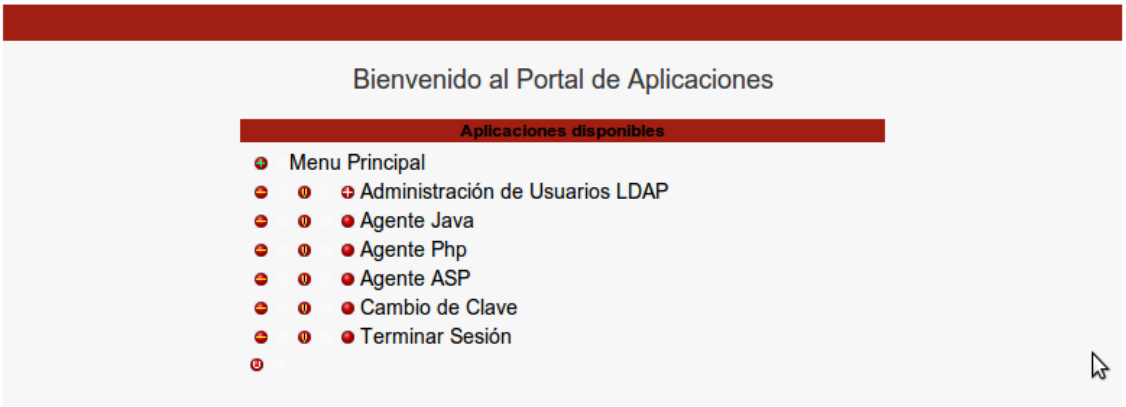
Para garantizar que se concedan o revoquen los derechos correctos de acuerdo con las reglas de la organización, simplificando colocar a los usuarios en los grupos adecuados durante el proceso de aprovisionamiento y cambiar las funciones de los usuarios durante su desempeño en la organización hemos desarrollado una aplicación de aprovisionamiento de usuarios y roles con una interfaz de administración totalmente intuitiva y ágil que permita realizar estas actividades de forma correcta y de manera oportuna.

Como complemento se desarrolló un portal de aplicaciones que permite a los usuarios de la organización centralizar el acceso a las aplicaciones que tiene disponible, y una interfaz de administración del portal.

Portal de aplicaciones y administración de aplicaciones

A través del portal de aplicaciones los usuarios tendrán una vista homogeneizada de todas las aplicaciones a las que puede acceder y, a su vez, los usuarios responsables de la seguridad podrán ingresar a la administración de roles y usuarios simplemente como una aplicación más a la que ellos tienen acceso.
En la captura siguiente se puede observar la página inicial de portal de aplicaciones, donde se ven listadas las aplicaciones a las que el usuario que inició la sesión puede acceder.

En el caso particular de ser un usuario con permisos para acceder a administrar usuarios, roles y aplicaciones, puede visualizarse el icono, además de tener listada entre sus aplicaciones la aplicación “Administración de Usuarios LDAP”.
Al hacer click sobre el icono se desplegarán las opciones para eliminar () una aplicación del portal y para modificar () una aplicación perteneciente al portal.
Además, junto a la leyenda “Menú principal” se encontrará el icono que permitirá agregar una nueva aplicación al portal de aplicaciones.



Modificación de una aplicación

Al presionar sobre el icono se desplegaran las propiedades de la aplicación a modificar, permitiendo realizar los cambios necesarios.

Eliminación de una aplicación

De la misma forma al presionar sobre el icono se podrán visualizar las propiedades de la aplicación seleccionada y un botón para realizar la acción.

Agregar una aplicación

Por último, al cliquear sobre el icono, se desplegará el formulario para ingresar las propiedades de la nueva aplicación. Puntualmente estas propiedades son:

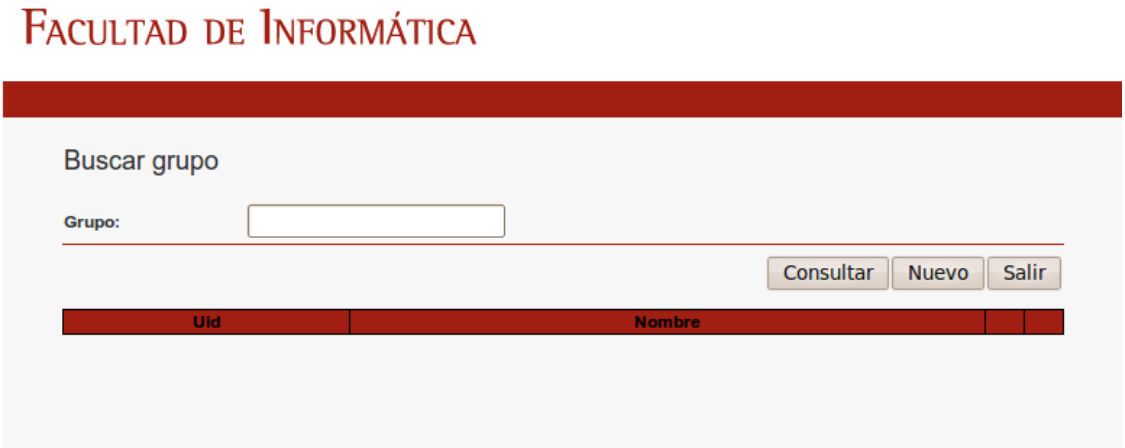
1) Nombre de la aplicación.
2) Url de la aplicación.
3) Orden en el que la aplicación se ubicará al desplegar el menú en la página de bienvenida del portal de aplicaciones.
4) Target: determina si la aplicación se abrirá en la ventana actual del explorador o está se abrirá en una nueva ventana.
5) Rol para el cual estará disponible la aplicación en el portal. Como prerequisito tenemos que todas las aplicaciones tendrán un rol general al que se le permitirá el ingreso. Por lo tanto un usuario tendrá generalmente dos roles, el rol general que le permitirá el ingreso a la aplicación y uno o más roles específicos que determinan las acciones que este podrá efectuar una vez dentro de la aplicación.

Administración de usuarios/roles

Como se mencionó anteriormente los usuarios con permisos para acceder a administrar usuarios, roles y aplicaciones tienen listadas entre sus aplicaciones la aplicación “Administración de Usuarios LDAP”. Al ingresar a esta aplicación se puede observar el menú que permite seleccionar entre administrar roles o usuarios.

Administración de roles

La pantalla principal de administración de roles nos permite consultar y crear un nuevo rol para ser utilizado por las aplicaciones dentro de la organización.



Consultar grupos

La consulta puede realizarse ingresando un filtro para recuperar solo los grupos que coincidan con la cadena de caracteres ingresada.
Una vez consultados los grupos podrán visualizarse su identificador y un nombre descriptivo. Conjuntamente con cada grupo aparecerán los iconos que permitirán modificar y eliminar un grupo, este último solo estará disponible en el caso de que el grupo no contenga usuarios.

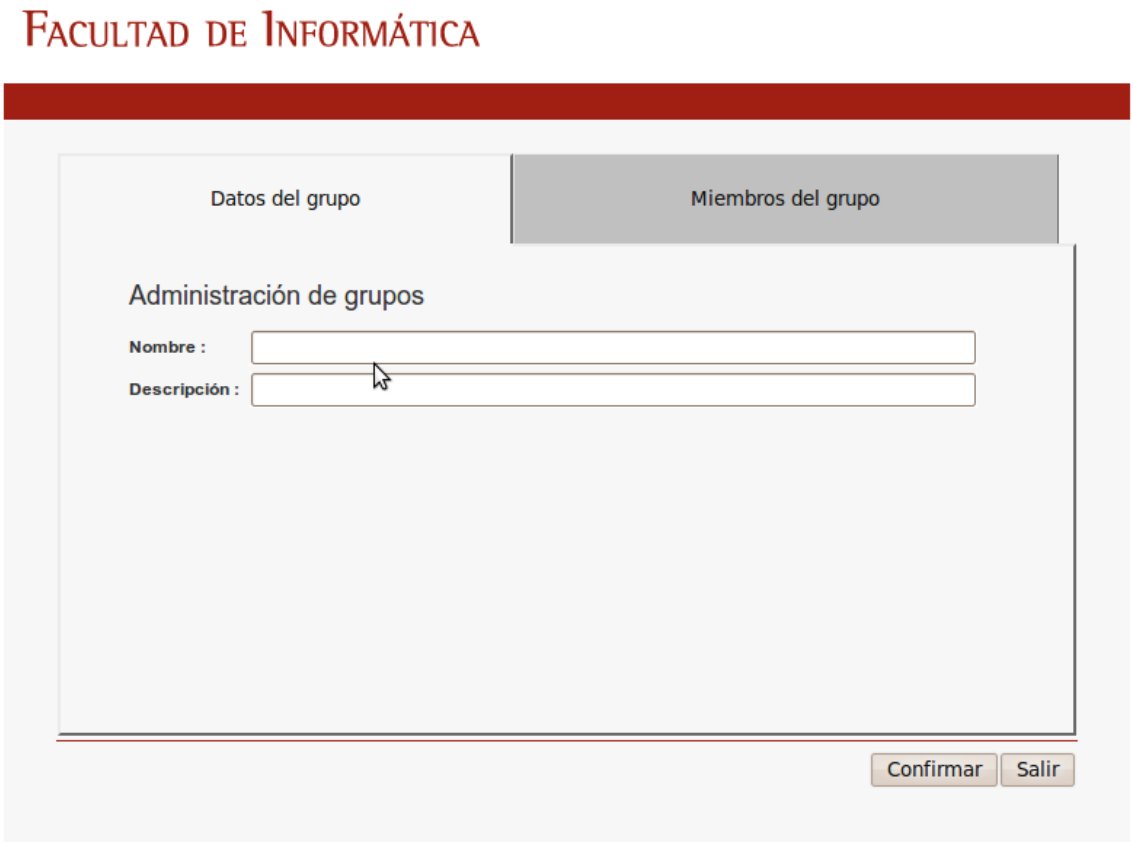
Modificar grupo

La modificación de un grupo simplemente permite cambiar la descripción del grupo seleccionado, además de tener la posibilidad de visualizar los miembros del grupo.

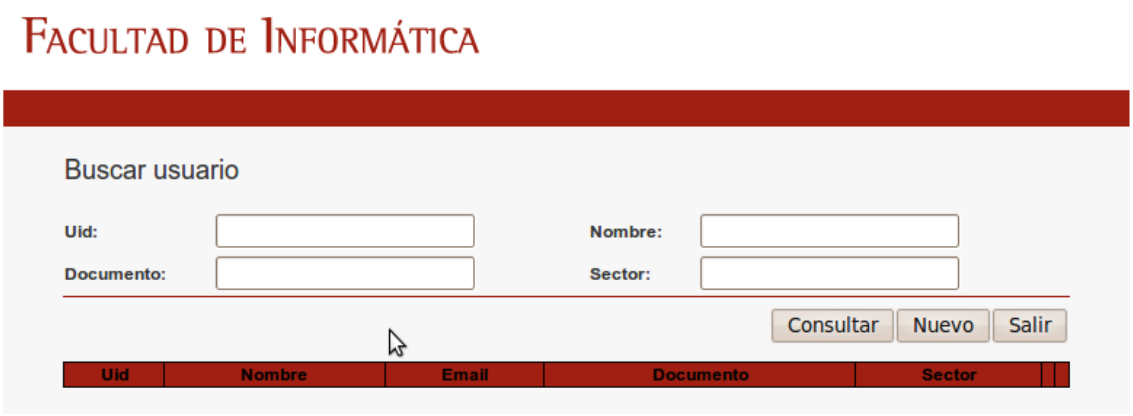
Agregar grupo

Al presionar el botón “Nuevo” se ingresará al formulario de creación de un grupo. Las propiedades que deben ingresarse son:

1) El identificador del grupo.
2) El nombre descriptivo del grupo.



Administración de usuarios



La pantalla principal de administración de usuarios nos permite consultar y crear un nuevo usuario perteneciente a la organización.

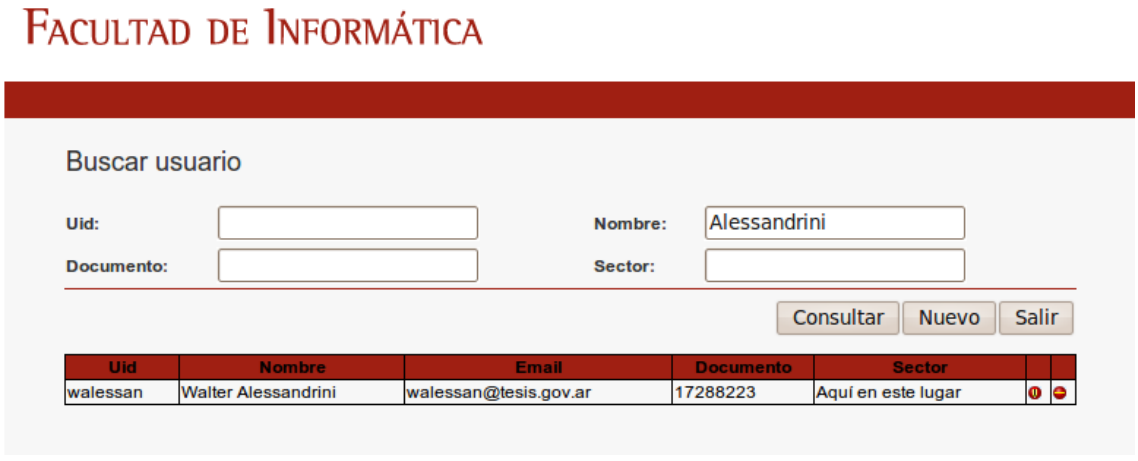
Consultar usuarios

La consulta puede realizarse ingresando filtros para recuperar solo los usuarios que coincidan con las cadenas de caracteres ingresadas.
Los filtros que pueden aplicarse son:

1) Nombre de usuario.
2) Nombre y apellido del usuario.
3) Numero de documento
4) Sector o departamento.

De ingresarse más de un filtro solo se recuperarán los usuarios que coincidan con todos los filtros aplicados.

Una vez consultados los usuarios podrán visualizarse su nombre de usuario (identificador), su nombre y apellido, su e-mail, su número de documento, y el sector al que pertenece. Conjuntamente con cada usuario aparecerán los iconos que permitirán modificar y eliminar un usuario.



Modificar usuario

El formulario de modificación de un usuario permite modificar cualquiera de las propiedades del mismo. Entre las propiedades que pueden modificarse se encuentran: el nombre y apellido, el e-mail, el número de documento, el sector o departamento de la organización al que pertenece, el número de teléfono, el fax, y la contraseña del usuario.

Agregar usuario

Al presionar el botón “Nuevo” en la pantalla de administración de usuarios se ingresará al formulario de creación de un usuario. Las propiedades que deben ingresarse son:

1) El nombre de usuario
2) El nombre de pila del usuario
3) El apellido del usuario
4) La dirección de e-mail del usuario.
5) El número de documento del usuario
6) La contraseña que el usuario utilizará para ingresar a cualquiera de los sistemas de la organización para los que tenga permisos de acceso.

De forma opcional además pueden ingresarse:

1) El sector o departamento al que el usuario pertenece dentro de la organización.
2) El número de teléfono.
3) El número de fax.

Asignación de grupos a un usuario

Dentro del marco de la modificación de un usuario existe la posibilidad de administrar los grupos que un usuario tiene asignado, lo que determinará las aplicaciones a las cuales un usuario puede acceder así como los derechos de acceso a recursos que posee el usuario.

Para administrar los grupos que tiene asignado un usuario simplemente debe ingresarse a la solapa “Grupos asignados” y marcar los grupos a los que se desea que el usuario pertenezca de la lista de grupos disponibles para luego agregarlos a través del botón “>”, o seleccionar los grupos que se desea que el usuario deje de pertenecer de la lista de grupos asignados y quitarlos utilizando el botón “<”.

Una vez hecho esto, solo resta confirmar la operación.

Adicionalmente existe la posibilidad de filtrar los grupos disponibles ingresando la cadena de caracteres que se desee que coincida con el o los grupos que asignaremos al usuario en cuestión.

Conclusión

Esta tesina de grado define una arquitectura para que las aplicaciones web de una organización desarrolladas en diferentes plataformas tengan la capacidad de SSO en el marco de una organización con un manejo dinámico de usuarios, y se desarrolló en dicho marco una aplicación de aprovisionamiento que permite definir roles y usuarios asociados para cada aplicación con capacidad de SSO.
Esta aplicación permite independizar la seguridad informática del área de desarrollo, dejando la responsabilidad de administrar el control de acceso basado en usuarios y roles a la oficina de seguridad informática mediante una interfaz sencilla e intuitiva sin la necesidad de contar con conocimientos técnicos específicos ni conocer la forma en que se encuentra implementado el repositorio de usuarios.

El hecho de soportar la autenticación en un servicio de directorio permite integrar un gran rango de aplicaciones externas, ya que la mayoría de las mismas brindan una interfaz de integración con servicios de directorios.
Como extensión y siempre considerando el marco de una organización, se puede extender la solución para integrarla con la base de datos de RRHH. Esta transparencia se puede lograr directamente sin necesidad de adaptación utilizando un directorio virtual que permita ver cualquier base de datos como parte de un servicio de directorio.

Bibliografía

Role-Based Access Control, David Ferraiolo y Rchard Kuhn, National Institute of Standards and Technology - http://www.csrc.nist.gov/groups/SNS/rbac/.

CAS User Manual, Scott Battaglia – http://www.ja-sig.org.

CAS Overview – JusForTechies – http://www.jusfortechies.com/cas/overview.html.

JOSSO - Java Open Single Sign-On Project Home, Sebastian Gonzalez Oyuela, Gianluca Brigandi - http://www.josso.org

Sun OpenSSO Enterprise 8.0, Technical overview, Sun Microsystems, Inc. - http://docs.sun.com/app/docs/coll/1767.1

Microsoft Technet, Federation Service - http://technet.microsoft.com/en-us/library/cc784321%28WS.10%29.aspx

LDAP-es – http://www.ldap-es.org

TIOBE Software, The coding standards company, TIOBE Programming Community Index - http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

AuthenticationWorld.com, The business of authentication, SSO and LDAP Authentication - http://www.authenticationworld.com/Single-Sign-On-Authentication/SSOandLDAP.html

Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), Qusay H. Mahmoud - http://java.sun.com/developer/technicalArticles/WebServices/soa/

OASIS Standards - http://www.oasis-open.org/specs/