Facultad de Ingeniería
Universidad de la República

Desarrollo de sistema de Autenticación y Autorización

Identity and Access Manager (IAM)

Proyecto de Grado

Estudiantes:

Nicolas Eirea

Rudi Lausarot

Claudio Mundin

Tutores:

Ariel Sabiguero

Nicolas Guerra

Tribunal:

Cristina Mayr

Daniel Meerhoff

Leandro Scasso

4 de enero de 2016

Resumen

En muchas organizaciones la administración de los usuarios es llevada a cabo por los departamentos de informática, lo cual le quita mucho tiempo al personal informático la realización de tareas administrativas, que en muchas ocasiones son repetitivas y no son inherentes a la profesión. Este trabajo tiene como objetivo desarrollar un sistema informático de autenticación y autorización centralizado, que permita delegar las tareas de mantenimiento de usuarios.

El desarrollo de este sistema permite que las tareas administrativas de usuarios como el estado contractual de los funcionarios, sus cargos, altas o ceses, puedan llevarse a cabo por personal del departamento de recursos humanos. Para conseguirlo fue necesario hacer un relevamiento del estado del arte de los sistemas y las herramientas que permiten la administración de la autenticación y/o autorización en los sistemas de información. Se analizaron diferentes modelos de seguridad y distintos servicios de directorio para seleccionar el más adecuado que cumpliera con los grandes lineamientos establecidos por el cliente.

Se presentaron dos retos importantes durante el transcurso del proyecto. El principal desafío, fue la integración con sistemas heterogéneos y distintos recursos ya existentes. Por otro lado, surgió la necesidad de extender la implementación del modelo de seguridad elegido para poder contar con restricciones basadas en atributos arbitrarios, como lo es, la ubicación donde el usuario ingresa al sistema. Una vez elegido el modelo de seguridad, se llevó a cabo el relevamiento exhaustivo de los requerimientos planteados por el equipo informático y usuarios de la institución, así como también el de los sistemas informáticos y la arquitectura existente. Analizados dichos requerimientos, se procedió a realizar el diseño del sistema y de la arquitectura. Finalizado el desarrollo, se armaron los casos de usos necesarios para la validación de la administración, configuración y operación del sistema, los cuales fueron probados con éxito. De esta forma se consiguió disponer de un sistema de autorización y autenticación de usuarios y aplicaciones externas que puede ser ampliamente utilizado por distintas empresas.

Palabras clave: *IAM, LDAP, Servidores de AAA, Fortress, Autenticación, Autorización*

Índice

Índice de figuras

1. Introducción

El presente documento surge como resultado de la realización del proyecto de grado para la carrera de Ingeniería en Computación de la Universidad de la República (UdelaR). Dentro de este marco se propone desarrollar un sistema informático de autenticación y autorización para la Administración de los Servicios de Salud del Estado (ASSE) de la República Oriental del Uruguay.

Se toma una organización lo suficientemente grande y heterogénea de manera que existan muchas dependencias distribuidas en un amplio territorio geográfico, en donde cada una de las cuales pueden cumplir distintas funciones dentro de la institución. Además, se cuenta con una gran cantidad de funcionarios cumpliendo diversas actividades dentro de la organización, así como también miles de usuarios que utilizan los servicios brindados.

Este proyecto es realizado en forma conjunta con el departamento informático de ASSE con quienes se trabajó para poder cumplir con los objetivos planteados.

1.1. Motivación

La motivación principal del trabajo es que actualmente la administración de usuarios y funcionarios de la institución recae sobre el departamento de informática, lo cual quita mucho tiempo al personal técnico informático en la realización de tareas administrativas, las cuales suelen ser repetitivas en muchas ocasiones.

Otra motivación del trabajo surge debido a la necesidad de contar con un sistema propio, basado en herramientas de código abierto, capaz de consolidar todas las tareas relativas a la autorización de usuarios en los sistemas internos y la autenticación, no sólo vinculadas a los sistemas operativos sino también a las distintas herramientas informáticas existentes, por ejemplo correo electrónico, red inalámbrica, aplicaciones adquiridas o desarrolladas internamente, etc.

1.2. Objetivos

El objetivo general de este trabajo es centralizar el sistema informático de autorización y autenticación, y al mismo tiempo delegar la administración de usuarios al departamento de recursos humanos como corresponde.

Los objetivos específicos que se buscan alcanzar para poder cumplir con el trabajo son los siguientes:

- Estudio del estado del arte de los sistemas de autorización y autenticación existentes.
- Implementar un sistema de autorización y autenticación basado en software libre.
- Desarrollar una herramienta para la gestión de usuarios basado en el sistema implementado.

1.3. Resultados esperados

- Elección e implementación del sistema de autorización y autenticación.
- Desarrollo de una herramienta que permita administrar, configurar y operar el modelo incluyendo gestión de usuarios.
- Desarrollo de una interfaz para que otras aplicaciones de la organización puedan interactuar con el sistema.

1.4. Conocimientos previos

El presente documento está orientado a estudiantes o profesionales de carreras afines a Ingeniería en Computación, con conocimientos sobre redes de computadores y arquitectura de sistemas, así como para quienes trabajen como administradores de sistemas de información.

Los lectores con experiencia en el desarrollo de aplicaciones podrán entender más rápidamente ciertas secciones del documento, aunque el conocimiento en esta área no es requerido.

1.5. Organización del documento

El presente informe cuenta con nueve capítulos cuyo contenido se esboza a continuación.

El primer capítulo contiene una introducción inicial, la motivación y las características del problema a resolver así como los objetivos del proyecto y los resultados esperados.

A continuación, en el segundo capítulo se presenta un marco teórico básico necesario para comprender el trabajo. Se describe lo que son los conceptos de autorización y autenticación en sistemas de información, distintos modelos de seguridad y lo que es un servicio de directorios.

En el tercer capítulo se describe el estado del arte de las herramientas de autorización y autenticación existentes. También se describe la situación en que se encuentra la institución en un inicio, de manera de tener un panorama claro del escenario que se tiene como base en materia tecnológica.

El cuarto capítulo se realiza el análisis de los principales requerimientos del sistema, así como la descripción de los procesos transitados para lograr comprender los problemas planteados.

Luego, comienza el quinto capítulo en donde se plantea el diseño alcanzado para cumplir los requerimientos y se detallan las decisiones de diseño tomadas.

Siguiendo con el desarrollo del sistema, el sexto capítulo describe la implementación del sistema, el entorno y el proceso de desarrollo utilizado. También

aquí es donde se plantea la arquitectura del sistema y sus interfaces con sistemas y recursos externos.

El séptimo capítulo describe la forma en que se validó el trabajo así como los casos de uso generados para ello.

Finalmente, en el capítulo octavo se describen las conclusiones generales y resultados obtenidos en el proyecto y en el capítulo noveno se mencionan las posibles líneas de trabajo a futuro.

En forma adicional se entregan los siguientes anexos:

Anexo 1: Instalación de la aplicación

Anexo 2: Prototipo Java con JNDI para LDAP

Anexo 3: Configuraciones

2. Marco Teórico

En este capítulo se presentan los conceptos teóricos para que el lector pueda entender y analizar la solución que se expone más adelante.

En las siguientes secciones se describen los conceptos básicos de los distintos modelos, sistemas y herramientas utilizados para entender el funcionamiento del sistema desarrollado así como del software con el cual se interactúa.

Dentro de los modelos de seguridad presentados se enfoca el estudio en: Discretionary access control (DAC), Mandatory access control (MAC), Role-Based Acces Control (RBAC). Por otra parte se realiza la introducción y descripción al concepto de servicios de directorios, haciendo énfasis en la implementación realizada por el protocolo Lightweight Directory Access Protocol (LDAP).

2.1. Autorización y autenticación

Antes de presentar los diferentes modelos de seguridad y los servicios de directorios, es aconsejable tener claro los conceptos de autorización y autenticación. Es común confundirlos o pensar que uno abarque al otro al momento de nombrarlos, aunque son conceptos distintos.

Autenticación: consiste de un sistema o procedimiento para certificar que un usuario es quien dice ser. La forma habitual de realizar dicho chequeo es utilizando la combinación de usuario y contraseña.

Autorización: consiste en dar acceso a una serie de recursos, luego de que el usuario se ha autenticado de manera exitosa.

Se observa que el concepto de autenticación valida al usuario y el concepto de autorización indica qué puede hacer el usuario mencionado.

2.2. Modelos de seguridad

En la presente sección se evalúan algunas políticas de seguridad para realizar el control de acceso a usuarios. No se realizará un análisis exhaustivo de todos los modelos de seguridad debido a la realidad del problema. Se evaluarán Discretionary access control (DAC), Mandatory access control (MAC), Role-Based Access Control (RBAC).

2.2.1. Discretionary access control (DAC)

DAC [5] garantiza o restringe el acceso a un objeto mediante una política de acceso, determinada por el grupo dueño del objeto y/o los subobjetos. El mecanismo de control se define por las credenciales provistas por el usuario durante la autenticación. Dichas credenciales pueden ser por ejemplo; nombre usuario y password. El nombre de este modelo de seguridad hace referencia a discreción dado que una entidad puede otorgar permisos de acceso a otras entidades, es

decir que el propietario determina los privilegios de acceso a los objetos.

Dentro de las representaciones de DAC, se encuentra la matriz de control de accesos, donde las filas representan a los usuarios y las columnas a los objetos. Por lo tanto para cada lugar (i,j) de la matriz se representa el permiso del usuario i sobre el objeto j.

DAC maneja los siguientes conceptos para definir las restricciones de acceso:

- Propiedad: Dentro del sistema en el cual se desea aplicar el modelo de seguridad DAC cada objeto debe tener un dueño. La política de acceso es determinada por el dueño del recurso. Un objeto que no tenga un dueño definido no se encuentra protegido. Normalmente el dueño de un recurso es el usuario que lo crea.
- Permisos: Hace referencia a los derechos de acceso que el dueño de un recurso asigna a usuarios individuales o a grupos de usuarios. El par ordenado (S_i,O_j), significa que el sujeto S_i puede acceder al objeto O_j de acuerdo con los derechos (read, write, execute, etc.) otorgados.
- Sujetos: Cuando se nombra a un sujeto en realidad se hace referencia a usuarios o grupos de usuarios.
- Objetos: Los recursos del sistema se definen como objetos sobre los cuales un usuario va a tener acceso.

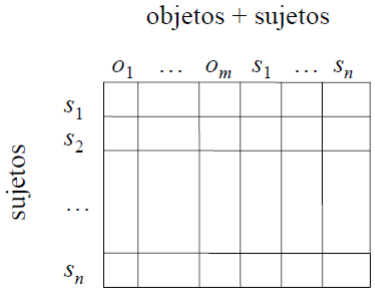

Figura 1: Matriz de control de acceso,extraída de [1]

Como ventaja encontramos que DAC es un modelo de seguridad flexible y adaptable a muchos sistemas y aplicaciones. Es amplimanente utilizado, especialmente en ambientes comerciales e industriales.

Dentro de las principales desventajas encontramos que los derechos son aplicados por los propios usuarios. Como indica el modelo los permisos son aplicados a discreción del usuario, con lo cual no se tiene control y se puede producir que se asignen permisos por error de parte de los usuarios. Esto hace que DAC sea muy vulnerable al ataque de troyanos, dado que un objeto puede copiar datos a

otro y puede dar acceso para manipular los datos a un usuario que en principio no tenía permiso sobre esta información.

2.2.2. Mandatory access control (MAC)

El modelo de seguridad MAC [5] define y asegura: un nivel de seguridad centralizado y parámetros de políticas de seguridad confidenciales. El manejo de las políticas de seguridad y su seteo se establece en una red segura y está limitado a los administradores del sistema. Controla el acceso comparando niveles de seguridad frente a las autorizaciones, este control es obligatorio dado que una entidad establece los niveles de seguridad y las autorizaciones de las entidades. Por lo que es responsabilidad del administrador del sistema dar acceso o alterar los controles de acceso. Se utiliza en sistemas multinivel que procesan información altamente sensible.

Cuando hacemos referencia a un **Sistema multinivel**, indica un sistema de computadoras que maneja múltiples niveles de clasificación entre sujetos y objetos, como por ejemplo la información gubernamental o militar.

Todo recurso del sistema y los usuarios, tienen asignados una etiqueta de seguridad. La etiqueta de seguridad sigue el modelo de clasificación de la información militar, en dónde la confidencialidad de la información es lo más relevante (política de seguridad multinivel). Cada nivel de seguridad es un elemento de un conjunto jerárquicamente ordenando, los niveles dentro de esa jerarquía son: altamente secreto (AS), secreto(S), confidencial(C) y sin clasificar (U). El orden que se aplica dentro de la jerarquía cumple el siguiente criterio: AS >S >C >U. En este tipo de sistemas todas las decisiones de seguridad las impone el sistema, comparando las etiquetas del usuario que desea acceder frente al recurso accedido, siguiendo un modelo matemático (Bell-LaPadula [6]).

La principal ventaja del modelo de seguridad MAC, es que el sistema es quien impone las restricciones y no los usuarios. La definición de niveles de seguridad impide que fluya información de un nivel superior a uno inferior, fortaleciendo las políticas de seguridad para garantizar la confidencialidad e integridad de los datos.

Como desventaja se observa que la administración se delega a un grupo reducido de personas, la misma se vuelve difícil de mantener, a su vez no otorga la flexibilidad necesaria para los ambientes web.

2.2.3. Role-Based Acces Control (RBAC)

El control de acceso en el modelo de seguridad RBAC, es No Discrecional al igual que el modelo MAC, sin embargo, RBAC está basado en roles. Se otorgan privilegios en función del rol que tiene el usuario dentro de la organización. Este método minimiza el trabajo de administración de permisos y roles de los usuarios por parte de los administradores del sistema. Las principales características de RBAC son:

- Asegurar que los usuarios autorizados tienen acceso a los recursos basándose en permisos asignados a los roles del usuario.
- Contar con una jerarquía de roles que posibilita de una manera natural organizarlos para reflejar la organización de las empresas, para esto se basa en la relación de responsabilidad de los usuarios. Esto facilita significativamente la administración del sistema.

En la definición del estándar RBAC [2] se describen las características que incluyen un modelo de referencia y especificaciones funcionales para las características definidas en el modelo de referencia.

RBAC define un conjunto de elementos básicos: usuarios, roles, permisos, operaciones y objetos, junto con relaciones como tipos y funciones. Brinda la funcionalidad de llevar a cabo dos propósitos. Primero el modelo de referencia define el alcance de las características que son incluidas en el estándar, esto identifica: un conjunto mínimo de características en todos los sistemas RBAC, aspectos de la jerarquía de roles, aspectos de las restricciones sobre relaciones estáticas y aspectos de las restricciones dinámicas de las relaciones. Como segundo propósito el modelo de referencia provee un lenguaje preciso y consistente en términos de conjuntos de elementos y funciones para ser usados en la definición de una especificación funcional.

Las características necesarias para especificar un sistema RBAC se pueden separar en tres grupos:

- **Operaciones administrativas.**
  Definen funciones en término de una interfaz administrativa y un conjunto de semánticas asociadas que permite crear, borrar y mantener los elementos y sus relaciones. Por ejemplo, crear y borrar un rol de usuario.
- **Revisiones administrativas.**
  Definen funciones en términos de una interfaz administrativa y un conjunto asociado de semánticas que provee la capacidad de realizar operaciones query en los elementos y las relaciones de RBAC.
- **Funcionalidades a nivel de sistema.**
  Define características para la creación de sesiones de usuario que incluyen activación/desactivación de roles, refuerzo de restricciones en la activación del rol, y para la determinación de una decisión de acceso.

En el estándar se definen los siguientes términos:

- Un componente refiere a uno de los bloques principales de las características de RBAC: core, jerarquía, relaciones SSD (static separation duty), y relaciones DSD (dinamic separation duty)
- Los objetos pueden ser recursos del sistema a acceder, por ejemplo, un archivo, una impresora, un registro de base de datos, etc.
- Las operaciones son programas ejecutables, las cuales al ser ejecutadas realizan una acción para el usuario.

- Los permisos son una aprobación para realizar una operación en uno o más objetos protegidos por RBAC
- Un rol es una función en el contexto de la organización con una semántica asociada respecto a la autoridad y a la responsabilidad conferida sobre el usuario asignado al rol.
- Un usuario es definido como un ser humano para simplificar su definición, ya que puede ser una máquina, una red, etc.

El modelo de referencia RBAC es definido en cuatro componentes, donde cada uno incluye al anterior [7]:

1. Core RBAC; define un conjunto mínimo de elementos y relaciones. Esto incluye relaciones usuario-rol, relaciones permiso-rol, activación del rol de usuario como parte de la sesión de usuario del sistema operativo.
2. Componente jerárquico; agrega relaciones para dar soporte a la jerarquía de roles. Matemáticamente, es un orden parcial definiendo una relación entre roles, esto es, conjuntos de roles de los usuarios y permisos autorizados.
3. Separación estática de relaciones de trabajo; agrega relaciones de exclusividad entre los roles con respecto a las asignaciones de los usuarios. Debido al potencial de las inconsistencias con respecto a la separación estática de relaciones de deber y las relaciones heredadas de la jerarquía de roles, el componente del modelo SSD define relaciones en ambos casos, en presencia y ausencia de jerarquía de roles.
4. Separación dinámica de relaciones de deber; define exclusividad de relaciones con respecto a los roles que están activos como parte de la sesión de usuario.

Lo medular de RBAC es el concepto de relaciones entre roles, sobre la cual un rol es una construcción semántica para una política dada. El CORE de RBAC incluye un conjunto de elementos, usuarios (USR), roles(ROLES), objetos(OBS), operaciones(OPS) y permisos(PRMS). La Figura 2 ilustra las relaciones de asignación de usuario (UA) y asignación de permisos (PA). En la Figura 2 también se ilustra la relaciones entre los componentes de RBAC, por ejemplo, un usuario puede ser asignado a uno o más roles y un rol puede ser asignado a uno o más usuarios.

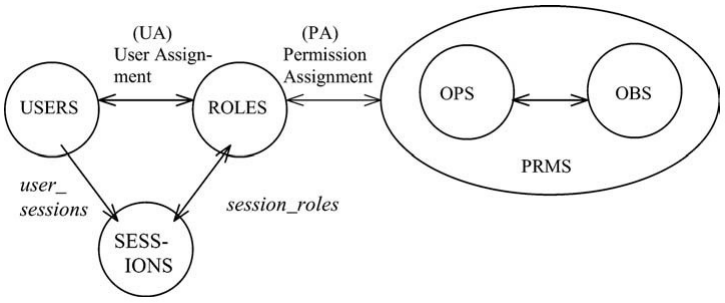

Figura 2: Core RBAC, extraída de [2]

Cada sesión está asociada con un usuario y cada usuario está asociado con una o más sesiones. Con la función sesión-roles se obtienen los roles activos por la sesión y con la función sesión-usuarios se obtiene el usuario que está asociado con la sesión. Los permisos disponibles para el usuario son los permisos asignados a los roles que están actualmente activos a través de todas las sesiones del usuario.

RBAC tiene un componente para el manejo de la jerarquía de roles (RH), como se describe en la Figura 3. La jerarquía de roles define una relación de herencia entre los ellos. Esta herencia es descripta en términos de permisos, por ejemplo, el rol r1 hereda el rol r2 si todos los privilegios de r2 son también privilegios de r1.

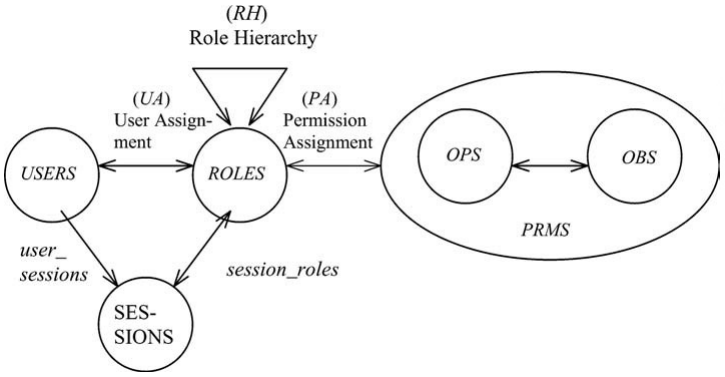

Figura 3: RBAC Jerárquico, extraída de [2]

Generalmente la jerarquía de roles soporta herencia múltiple, esto implica que: posee la habilidad de componer un rol con la herencia de varios roles y posibilita un tratamiento uniforme de las asignaciones de relaciones de herencia usuario-rol y rol-rol.

Los roles pueden tener capacidades solapadas, es decir que a los usuarios pertenecientes a diferentes roles se les pueden asignar permisos en común. Los roles en una jerarquía de rol limitada está restringida al descendiente inmediato y no soporta herencia múltiple. La notación usada $r \succeq r_1$ debe leerse r hereda de r_1

$\forall r, r_1, r_2 \in ROLES, r \succeq r_1$ y $r \succeq r_2 \Rightarrow r_1 = r_2$

Una jerarquía de rol general puede representarse por un diagrama de Hasse, los nodos en el grafo representan los roles de la jerarquía y hay una línea (flecha) de r_1 a r_2 cuando r_1 es un descendiente inmediato de r_2. En el grafo que se crea, $r_x \succeq r_y \Leftrightarrow \exists$ un camino de r_x a r_y. Adicionalmente no hay ciclos en el grafo debido a que la relación de orden es antisimétrica y transitiva. Usualmente el grafo se representa con arcos correspondientes a la relación de herencia orientados top-down, esto significa que el usuario hereda top-down y los permisos de rol son heredados bottom-up.

2.2.4. RBAC con restricciones

RBAC con restricciones agrega la separación de las relaciones de deber al modelo de RBAC. Esta separación de las relaciones, de deber se utiliza para hacer cumplir las políticas de conflicto de intereses que las organizaciones pueden emplear para impedir que los usuarios excedan un nivel razonable de autoridad para sus posiciones.

- Separación estática de relaciones de deber (SSD)

  Los conflictos de intereses pueden generarse en un sistema con relaciones basada en roles como resultado de la ganancia de privilegios asociada con roles en conflicto. La separación estática pude realizarse de diversas maneras, un ejemplo común de SSD es la que defina mutua exclusión de las asignaciones de los usuarios con respecto al conjunto de roles. Una política de SSD puede ser especificada e impuesta a determinados roles, generalmente se usan para poner restricciones a los elementos administrativos. Como se ilustra en la Figura 4, las relaciones SSD pueden coexistir con RBAC jerárquico. Cuando se aplica la relación SSD en presencia de una jerarquía de roles, se debe tener especial cuidado para asegurar que la herencia no interfiera con la política SSD, debido a que la jerarquía de roles ha sido definida para incluir la herencia de las restricciones SSD. Para manejar esta inconsistencia SSD es definido como una restricción en los roles de los usuarios autorizados que tengan una relación SSD.

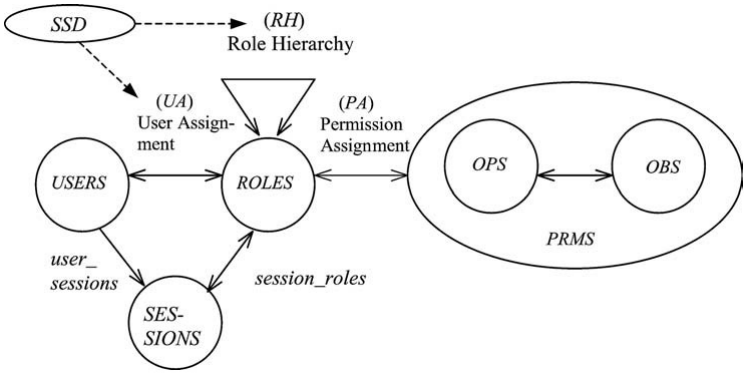

Figura 4: SSD con RBAC jerárquico, extraída de [2]

$SSD \subseteq (2^{ROLES}\text{x}N)$ *es una colección de pares* (rs, n) *en SSD, donde cada rs es un conjunto de roles, t es un subconjunto de roles en* rs*, y* $n \in N/n \geq 2$ *, con la propiedad de que ningún usuario es asignado a* n *o mas roles del conjunto* rs *en cada* $(rs, n) \in SSD$. Separación estática de deberes con jerarquía de roles En presencia de jerarquía de roles, SSD se redefine como sigue:

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} usuarios\, asignados(r) = \emptyset$$

En presencia de jerarquía de roles SSD se redefine basándose en usuarios autorizados, en lugar de en usuarios asignados.

$$\forall(rs,n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} usuarios\, autorizados(r) = \emptyset$$

- Separación dinámica de relaciones de deber (DSD)

  La separación dinámica de relaciones de deber reduce el número de potenciales permisos que pueden estar disponibles para un usuario adicionando restricciones en los usuarios que pueden ser asignados a un conjunto de roles. La separación dinámica de Roles, así como en SSD, tiene como intención limitar los permisos que están disponibles para un usuario. Sin embargo, DSD difiere de SSD en el contexto en el cual estas restricciones se aplican. SSD pone restricciones sobre los permisos totales del usuario, DSD limita la disponibilidad de permisos sobre un usuario definiendo restricciones sobre los roles que pueden ser activados durante una sesión de usuario. DSD provee soporte al principio de menor privilegio, en el que cada usuario tiene diferentes niveles de permiso en diferente momento, dependiendo del rol que está usando. DSD permite a un usuario ser autorizado para dos o más roles que no crean conflicto de intereses cuando actúan independientemente, pero producen problemas en las políticas cuando son activados simultáneamente. Por ejemplo, un usuario puede estar autorizado por ambos roles de Cajero y Supervisor de Cajero, donde el supervisor puede realizar correcciones a las ventas del cajero. Si el cajero quiere cambiar el rol a Cajero Supervisor, RBAC requerirá que el usuario deje el rol Cajero, y por lo tanto cierre la caja antes de asumir el rol Cajero Supervisor. Debido a que el usuario no tiene permitido asumir ambos roles, no se creará un conflicto de intereses. DSD define las restricciones sobre los roles que son activados en una sesión de usuario como se observa en la Figura 5.

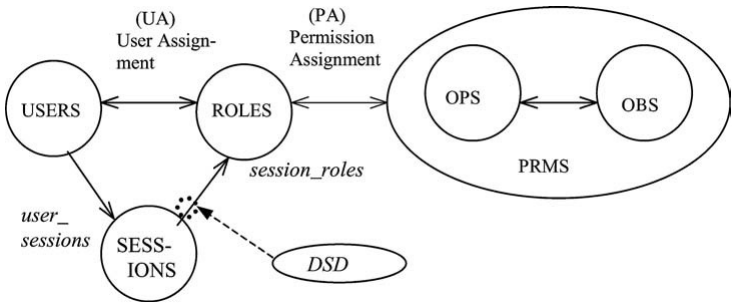

Figura 5: Separación dinámica de relaciones de deber, extraída de [2]

$DSD \subseteq (2^{ROLES}$x$N)$ es una colección de pares (rs, n) en DSD, donde cada rs es un conjunto de roles y $n \in N/n \geq 2$, con la propiedad de que ningún sujeto puede activar n o mas roles del conjunto rs en cada $dsd \in DSD$.

2.3. Servicio de directorio

En este capitulo se presentan la principales características de un Servicio de Directorio. Se trata de un conjunto de objetos con atributos organizados de una

manera lógica y jerárquica, los mismos trabajan de manera conjunta para brindar un servicio.

Hay muchos ejemplos sencillos de Directorios, casi todo el mundo ha utilizado alguno en alguna oportunidad, una guía de teléfonos es un ejemplo claro. Enfocándose mas en el mundo informático, un ejemplo sencillo de un servicio de directorio sería el de un servicio de nombres para corresponder los nombres de los dispositivos de red con sus respectivas direcciones de red. En los directorios se define que información será guardada y de que forma se organizará, permitiendo con esto ubicar la información pertinente.

El servicio de directorio puede ser visto como una base de datos que esta optimizado para realizar lecturas, proporcionando mecanismos de búsqueda para los diferentes atributos que se pueden asociar a un objeto [3]. Igualmente existen diferencias entre una base de datos y un servicio de directorio. A continuación se enumeran algunas características que las diferencian:

Lectura y Escritura de la información: el concepto que maneja el servicio de directorio es el de priorizar la lectura de datos sobre la escritura, esto se da porque en general la frecuencia en la que se modifican los datos es mucho menor con respecto a la lectura de los mismos, incluso se puede dar prioridad a realizar una lectura sobre una escritura. En una base de datos tanto la escritura como la lectura de información es tenida en cuenta a la hora de optimizar el rendimiento.

Generalización: las bases de datos manejan un conjunto de datos determinados, en el caso de los directorios los tipos de datos se manejan a través del uso de esquemas, dichos esquemas pueden ser creados y modificados siguiendo ciertas reglas, pero pueden ser adaptados a casi cualquier tipo de información.

Replica de los datos: las base de datos que son modificadas con gran frecuencia, por lo tanto las replicas de las mismas es acotada a un pequeño numero de servidores para tratar de mantener la consistencia de los datos. En cambio en un directorio, debido a la menor frecuencia de cambios se puede optar por una mayor cantidad de replicas de datos, esto lleva a que los usuarios puedan acceder a servidores que se encuentran en redes locales, mejorando el tiempo de respuesta en el uso de los mismos.

Performance: cuando se trata de servicios de directorios se espera que los mismos soporten miles de consultas en un periodo muy corto de tiempo, por segundo. En el caso de la base de datos se manejan transacciones, donde las base de datos deben soportar un numero relativamente menor de las mismas. Cabe señalar que cuando hablamos de una transacción nos referimos a un conjunto de operaciones que conforman una unidad de trabajo que se ejecuta de manera indivisible, esto quiere decir que nunca se ejecuta por la mitad, termina con éxito o no se ejecuta, por lo tanto lleva un nivel de complejidad mayor comparada una consulta en el servicio de directorios.

Hay muchas formas de proporcionar un servicio de directorio, diferentes métodos permiten diferentes tipos de información.

Los directorios electrónicos son dinámicos con respecto a otros directorios, por ejemplo en el caso de la guía telefónica cuando un en un registro de una persona se cambia su número, esto se vera reflejado en la próxima impresión de dicha guía.

2.3.1. Lightweight Directory Access Protocol LDAP

LDAP [8] es el acrónimo de Lightweight Directory Access Protocol. Se trata de un protocolo que permite administrar servicios de directorios, específicamente los servicios de directorios basados en X.500 [9]. X.500 es un conjunto de estándares de servicio de directorio presentados por ITU-T e ISO, dichos servicios se enmarcan dentro de los servicios de directorio digitales. LDAP fue desarrollado en 1993 en la Universidad de Michigan con el objetivo de reemplazar al protocolo Directory Access Protocol (DAP). LDAP es una versión mas simple que el protocolo DAP, de ahí deriva su nombre que hace referencia a ligero. Es mas simple porque en realidad utiliza un subconjunto de los estándares definidos en X.500.

Actualmente LDAP se encuentra en su tercera versión [10]. El modelo de información de LDAP está basado en entradas, donde la misma es una colección de atributos que tiene a nivel global un único nombre distintivo (Distinguished Name, DN). Cabe señalar que LDAP define como acceder a la información en el servidor a nivel del cliente, pero no define la manera en la cual se almacena dicha información. El DN es utilizado para referirse a la entrada de forma unívoca. Cada atributo de la entrada es de un tipo de dato y puede tener uno o más valores, por ejemplo *cn* es el atributo que se utiliza para representar el Common Name [8]. La manera en la cual se agrupan los atributos se denomina esquema. Los atributos se pueden caracterizar en dos grupos:

normales: son los que distinguen al objeto (nombre, ciudad, email, etc.).

operativos: son accedidos por el servidor para manipular los datos (fecha de modificación, etc.).

Las entradas del Directorio son organizadas en una estructura de árbol jerárquico. En general se manejan dos estructuras que se describen en los siguientes ejemplos:

1. En la cima del árbol se encuentran las entradas que representan a los países, debajo de éstas tenemos las entradas que representan a los estados o nombres de organizaciones y más abajo podríamos tener entradas que representen, por ejemplo: unidades de organizaciones, personas, impresoras, etc. En la Figura 6 se muestra un ejemplo de dicha estructura.

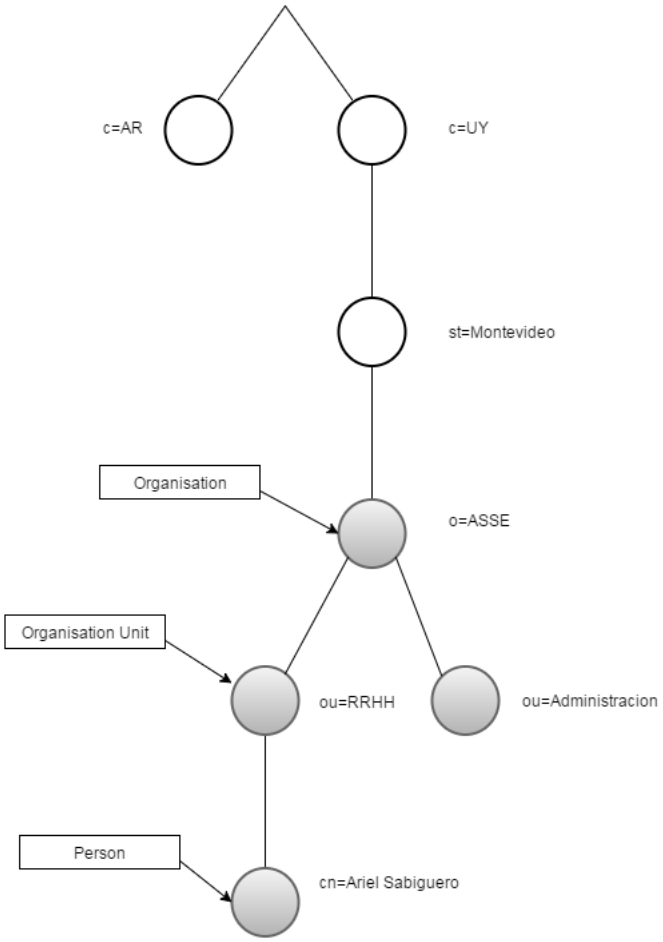

Figura 6: Árbol de directorio, extraída de [3]

2. En este caso, se dispone al árbol sobre la base de nombre de dominio de Internet. En la Figura 7 se muestra un ejemplo al respecto.

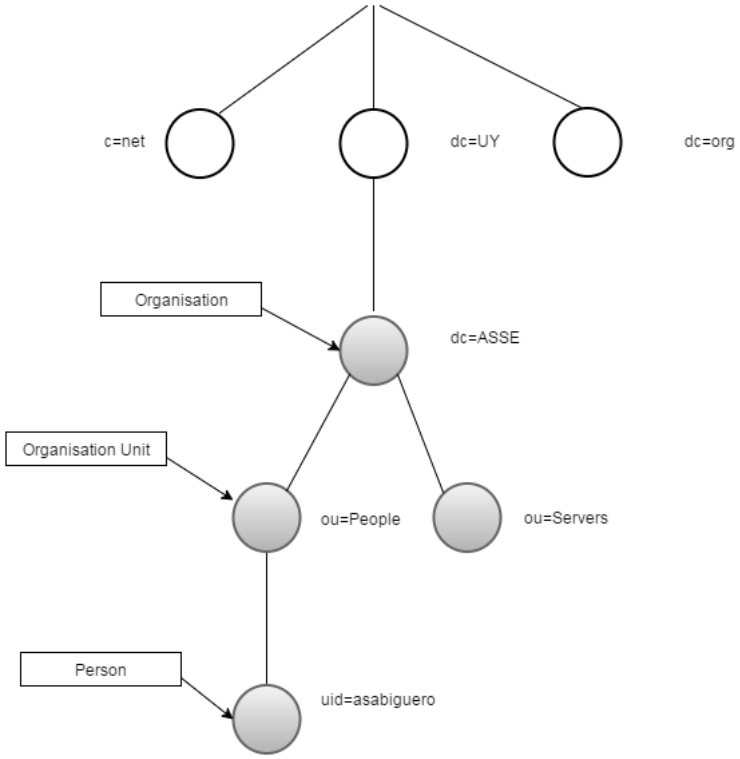

Figura 7: Árbol de directorio, basado en nombres de dominio, extraída de [3]

Con lo anterior podemos hacer referencia a una entrada utilizando su DN, por ejemplo para acceder a la entrada uid=asabiguero, haremos referencia a su dn: uid=asabiguero,ou=People,dc=asse,dc=uy.

Las entradas y objetos de un servicio de directorio LDAP se representan en formato LDIF (LDAP Data Interchange Format) [11]. Este formato de datos se utiliza para la importación y exportación de datos sin importar que servidor LDAP se utilice. Puede variar la manera en que los servidores LDAP almacenen físicamente sus datos, pero el intercambio de informacion se realiza en formato LDIF facilitando la intercomunicación entre diferentes servidores LDAP. El formato ldif es simplemente un archivo de texto ASCII con ciertas reglas, se puede resumir de la siguiente manera:

```
dn: nombre distintivo

atributo: valor

atributo: valor

atributo: valor

......
```

Un ejemplo de un archivo en formato ldif es el siguiente:

```
dn: uid=nicolas.guerra,ou=People,ou=Users,dc=asse
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
objectClass: sambaSamAccount
cn: nicolas.guerra
sn: nicolas.guerra
uid: nicolas.guerra
uidNumber: 1013
gidNumber: 1100
homeDirectory: /net/home04/nicolas.guerra
loginShell: /bin/bash
gecos: System User
givenName: nicolas.guerra
structuralObjectClass: inetOrgPerson
entryUUID: 6c49c082-1c71-1033-95e0-5955590334cf
creatorsName: cn=admin,dc=asse
createTimestamp: 20140128140836Z
sambaLogonTime: 0
sambaLogoffTime: 2147483647
sambaKickoffTime: 2147483647
sambaPwdCanChange: 0
displayName: nicolas.guerra
sambaSID: S-1-5-21-1544065822-457774965-1447756555-1005
sambaPrimaryGroupSID: S-1-5-32-1100
sambaLogonScript: logon.bat
sambaHomeDrive: H:
sambaLMPassword: 98B243DC240F6D21AAD3B435B51404EE
sambaAcctFlags: [U]
sambaNTPassword: AEFADF1FFA93F264F38AA8D4BF9F7F51
sambaPwdLastSet: 1391104912
sambaPwdMustChange: 1454176912
userPassword:: e1NTSEF9RTdDSlFZYVVxWHZ4WGl6VDEzT0FrMElkMjBka2FWRlI=
shadowLastChange: 16100
shadowMax: 730
entryCSN: 20140130180145.291177Z#000000#000#000000
modifiersName: cn=admin,dc=asse
modifyTimestamp: 20140130180145Z
```

2.4. Servicios de Directorio de menor uso

En esta sección se describen otros servicios de directorio que no son tan populares o utilizado como lo es LDAP.

2.4.1. NIS Network Information Service

NIS [12] es un servicio de directorio desarrollado por Sun Microsystem, en sus comienzo se lo llamaba Yellow Page, debido a un inconveniente en el registro del nombre el mismo tuvo que se cambiado a como se lo conoce. Es un utilizado básicamente en redes UNIX y permite el intercambio de informacion como ser usuarios, hosts, grupos. Sobre los años 90' Sun desarrollo una versión diferente denominada NIS+ o NIS plus, la misma agrega mas seguridad sobre la versión anterior, sin embargo esta nueva versión no es compatible con su antecesor. NIS basa su funcionamiento siguiendo el protocolo Remote Pocedure Call (RPC) especificado en el RFC 1831 [13]. En la misma red pueden coexistir varios servidores NIS, en este caso hay dos opciones que pueden estar combinadas:

1. Varios servidores maestros atienenden diferentes dominios
2. Un servidor NIS cumple la función de maestro sobre otros denominados esclavos, donde dichos servidores mantienen una copia de la base de datos recibida desde el servidor maestro, todas las tareas de actualización son realizadas en el servidor NIS maestro.

2.4.2. Netware Directory Service

Netware fue desarrollado por la empresa Novell [14], el soporte de este producto se extendió hasta el año 2015 pero actualmente esta casi en desuso. Las características que presenta son similares a la de los servicios de directorios ya presentados, donde encontramos que mantiene y distribuye recursos de red como ser: usuario, grupos, impresoras, etc. Tiene la posibilidad de contar con diferentes servidores replicados entre si, proporcionando seguridad a la hora de brindar el servicio ante la falla de alguno de los mismos. Al igual que en LDAP, Netware distribuye los objetos definidos en un árbol jerárquico. Los objetos pueden ser de tres tipos:

1. objetos físicos, como pueden ser usuarios o impresoras
2. objetos lógicos, haciendo referencia a grupos, colas de impresión.
3. otros objetos, aquí se agrupan los objetos que forman parte de las definiciones propias de Netware, como son unidades organizativas y todas las características que identifican a los objetos.

2.5. Resumen del capítulo

Se mostraron las ventajas y desventajas de cada uno de los modelos vistos en este capítulo, y siguiendo el enfoque que se plantea para el marco del proyecto se concluyo que el más adecuado es RBAC.

La premisa del proyecto es minimizar las tareas administrativas relativas al manejo de usuarios que lleva a cabo el personal de informática. DAC permite a los usuarios administrar los accesos en los objetos sobre los que tiene permisos, por lo tanto este modelo cumple con la premisa mencionada, sin embargo es un modelo que permite huecos de seguridad en lo que refiere al otorgamiento de permisos por parte de los usuarios. Por otro lado MAC se puede ver como el opuesto,

dado que el administrador del sistema es el encargado del manejo de las políticas de seguridad, esto no beneficia al personal informático ya que seguiría siendo quien ejecute esta tarea. El modelo no discrecional (RBAC) es el más apropiado, los usuarios tienen roles con permisos asignados para realizar tareas, como por ejemplo el manejo administrativo, pudiendo separar los roles (administrativo, informático, etc.) para varios usuarios de manera de cubrir lo planteado en este proyecto.

Además RBAC ofrece funcionalidades administrativas para el mantenimiento de las entidades. Esto sugiere la existencia de un usuario administrador que mantiene la estructura organizacional, junto con consultas administrativas sobre los elementos, posibilitando verificar la consistencia del modelo planteado de la organización. Por último brinda funcionalidades del sistema como lo son las de autenticación de usuarios, activación y desactivación de roles para fortalecer las decisiones de acceso al sistema.

3. Estado del arte

En este capítulo se presenta el estado actual en áreas de desarrollo tecnológico e investigación relacionadas a este proyecto.

En la primer sección se busca evaluar el estado del arte de sistemas o herramientas que permiten la administración de la autenticación y/o autorización en los sistemas de información.

En la segunda sección se describe la infraestructura actual de ASSE, dado que el proyecto toma como caso de estudio dicha institución, es de interés conocer la realidad presentada para luego demostrar como se aplica la solución que se busca.

3.1. Sistemas de autenticación y autorización

Los sistemas que realizan las tareas de autenticación y autorización, se agrupan en la categoría Identity Access Manager (IAM). Un sistema IAM proporciona políticas y procesos que permiten automatizar y facilitar el acceso a los sistemas de información [15]. El mismo puede ser utilizado para iniciar y manejar las identidades de los usuarios, así como también automatizar los permisos de accesos, todo esto acompañado de la auditoria correspondiente de la actividad del usuario. Las políticas que definen los controles de accesos para los usuarios, deben estar bien formadas para evitar accesos innecesarios. Es deseable que un sistema IAM cuente con las siguientes características:

- Proveer el manejo de identidad de los usuarios y automatizar los permisos de acceso.
- Incluir un servicio de directorio centralizado, para evitar el registro de información en diferentes lugares.
- Flexibilidad y facilidad en el manejo de permisos de acceso.
- Posibilidad para definir derechos de accesos que sean heredados por ciertas características, forzar o negar permisos.

A continuación se detallan un conjunto de herramientas que fueron evaluadas en el entorno de este proyecto.

3.1.1. IBM Security Identity and Access Manager

Este producto de IBM es una solución que gestiona las actividades de los usuarios en la empresa de manera automatizada y basada en políticas de acceso [16]. Dentro de las principales características con las que opera este producto tenemos:

- **Identidad del usuario:** Controla la información utilizada para describir al usuario. La identidad del usuario por si sola no provee los accesos a ningún recurso.

- **Administración de la Identidad:** Administra el ciclo de vida de la identidad de un usuario dentro de la organización. Crea o establece la identidad del usuario, las operaciones sobre la misma, y por ultimo determina la destrucción de la identidad del usuario en la organización
- **Cuenta de usuario:** Administra la información del usuario que hace posible el acceso a determinado recurso luego de un ingreso (login) satisfactorio.
- **Administración de la cuenta:** Manejo de las operaciones sobre la cuenta de un usuario: agregar, quitar, modificar, suspender.
- **Administración de permisos:** administra el acceso a los diferentes recursos en la organización.

3.1.2. Oracle Identity and Access Management Suite Plus

Este producto como su propio nombre lo indica es proporcionado por la empresa ORACLE. Se brindan dos enfoques para definir las características principales:

Solución integral, cubre una serie de puntos en el marco de las herramientas IAM: administración de la identidad, acceso a los sitios web, gestión de identidades, servicio de directorio.

Integración heterogénea, se integra con los principales servicios de directorios, servidores de aplicaciones, sistemas operativos y base de datos. Por otra parte trabaja con estándares como Liberty Alliance and OASIS y soporta SAML, SPML, WS-*, Kerberos, entro otros.

3.1.3. Apache Fortress

Apache Fortress es un kit de desarrollo de software de código abierto en Java basado en estándares, posibilita la gestión de acceso por identidad (IAM) en sistemas compatibles con LDAP v3. Fortress ofrece un sistema de gestión y aplicación totalmente compatible con el estándar ANSI RBAC basado en aplicaciones de código abierto existentes, como OpenLDAP y Apache Tomcat. Está diseñado para ser simple de implementar y rentable de mantener [17].

OpenLDAP se trata de una implementación de código abierto del protocolo LDAP [3].

Algunas de las principales características de Fortress son:

- Administración de RBAC, a través de: interfaces de programación, servicios y páginas Web.
- Administración de contraseñas, a través de : interfaces de programación, servicios y páginas Web.
- Auditoría centralizada de las actividades realizadas.
- Replicación del árbol de directorios.

3.2. Infraestructura ASSE

A continuación se describe la infraestructura existente en la institución ASSE previo a este trabajo, la cual se toma como base para integrar el nuevo sistema de autorización y autenticación. Esta tarea constituye uno de los puntos más importantes del trabajo ya que de no resultar exitosa peligra la viabilidad del proyecto.

3.2.1. Escenario ASSE

La arquitectura de red en ASSE cuenta con un servidor central de OpenLdap, este actuá como Maestro de la replica que se realiza a servidores esclavos ubicados en las Unidades Ejecutoras que son dependencias distribuidas geográficamente en todo el país. En los servidores esclavos se realiza la poda correspondiente para que quede solo la información que necesita la Unidad Reguladora. En la Figura 8 se puede ver gráficamente la distribución mencionada.

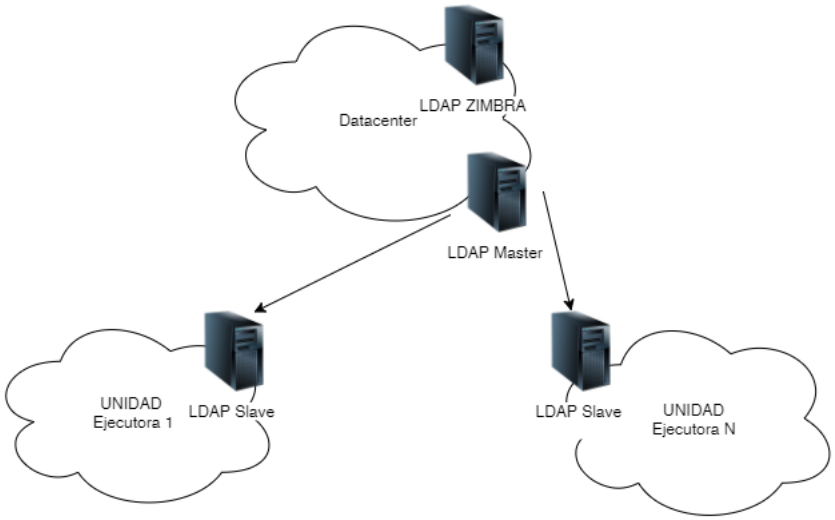

Figura 8: Diagrama de red actual

Progresivamente se está extendiendo a las diferentes unidades ejecutoras la tarea de autenticación de usuarios directamente al LDAP. La intención de la institución es llegar a que las aplicaciones desarrolladas por el equipo interno autentiquen y autoricen sobre el servidor LDAP.

3.2.1.1. Servicios de Directorio

3.2.1.1.1. Samba se trata de un conjunto de aplicaciones que implementan el protocolo de archivos compartidos de Microsoft Windows, el mismo se nombraba como *smb* actualmente se lo llama *cifs*. Básicamente samba permite que una maquina con Linux, Mac OS o Unix, puedan cumplir la función de servidor o cliente como si se tratara de un equipo Windows, con esto se logra conectar

a sistemas Windows con otros sistemas de forma casi transparente y así poder intercambiar información. Es posible también implementar un Controlador Principal de Dominio (PDC) para que los usuarios de la red puedan validarse [18].

3.2.1.1.2. NFS (Network File System) es un protocolo de nivel de aplicación en el Modelo OSI. Es utilizado para manejar archivos en sistemas remotos. Los sistemas clientes necesitan contar con un cliente de nfs para poder comunicarse con el servidor nfs.

3.2.1.2. Reglas de acceso

3.2.1.2.1. Samba Actualmente la institución cuenta con servidores samba que ofician como servidor: PDC, de impresión y de archivos. Samba cuenta con su propio repositorio de usuarios, pero en este caso esta configurado para que utilice OpenLDAP como repositorio. La administración de las altas y bajas de usuarios y máquinas se lleva adelante haciendo uso de las herramientas smbldap-tools [19], se trata de un conjunto de scripts hechos en perl diseñados para manejar grupos y usuarios guardados en un directorio LDAP. Estas herramientas pueden ser utilizadas tanto por administradores como por usuarios comunes, donde:

- Los administradores podrán realizar operaciones sobre todos los grupos y usuarios.
- Los usuarios podrán cambiar sus contraseñas y datos personales.

3.2.1.2.2. SUDO En los sistemas operativos tipo UNIX existe el programa sudo [20], el cual permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario (generalmente root).

Para llevar adelante dicha operativa se pueden definir reglas de accesos, las mismas deben ser editadas y guardadas en algún lugar, por defecto dicha tarea la cumple el archivo */etc/sudoers* . Una alternativa como backend de la definición de sudoers es *OpenLDAP*, se define una OU donde se administran dichas reglas de acceso, actualmente la institución cuenta con dicha implantación. La forma de administrar las entradas en OpenLDAP es de forma manual y de manera independiente.

3.2.1.3. Correo Electrónico Se tiene instalada la solución colaborativa Zimbra [21], que interactua con el software postfix [22] que ofrece la funcionalidad de servidor de correo. Zimbra cuenta con su propia interface de administración para crear los usuarios con sus respectivas cuentas de correo, así como también el manejo de alias. Zimbra tiene como backend de usuarios un servidor OpenLDAP dedicado, con lo cual nuevamente observamos que la administración es de forma independiente.

3.3. Resumen del capítulo

En este capítulo se realizó la investigación que releva las herramientas existentes, relacionadas al proyecto.

Las herramientas que se encuentran en la categoría Identity Access Manager (IAM), buscan realizar la integración de sistemas heterogéneos. El objetivo que persiguen es el mismo en todos los casos: proporcionar políticas y procesos que permiten automatizar y facilitar el acceso a los sistemas de información. Existe una amplia gama de herramientas, si bien todas buscan interactuar con la mayor cantidad de sistemas no todas lo hacen. En este capítulo se presentaron tres herramientas: IBM Security Identity and Access Manager, Oracle Identity and Access Management Suite Plus y Apache Fortress. Las mismas cumplen con el objetivo mencionado y permiten ser utilizadas por un gran numero de sistemas y herramientas. Al momento de analizar las herramientas la presentada por IBM parece ser la mas robusta y contar con un mejor soporte que el resto, la gran desventaja que se encuentra es el costo de licenciamiento. En el caso de la herramienta presentada por la empresa Oracle sucede algo similar. En contraposición la gran ventaja de Apache Fortress es que no tiene costo y cuenta con amplio respaldo del proyecto Apache.

En el relevamiento de la infraestructura de ASSE, se observa que la misma se esta orientando a un ámbito de centralización de la información de autenticación y autorización. Las configuraciones realizadas en las herramientas de Samba, sudo, sistemas operativos y correo electrónico hacen notar la necesidad de contar con un servidor de OpenLDAP que centralice toda la información de los usuarios. De todas maneras en lo que refiere a las tareas de administración aún son distribuidas, teniendo que realizarlas en cada herramienta en particular para realizar la administración de usuarios e información correspondiente.

4. Análisis de requerimientos del sistema

En este capítulo se analizan los requerimientos más importantes para el proyecto, además se describen los distintos procesos por los que se transitaron para llegar a comprender la realidad de los problemas planteados.

En las siguientes secciones se abordará el estudio de cada uno de los principales requerimientos.

4.1. Elección del modelo de seguridad

El sistema debe estar basado en un modelo de seguridad que permita el manejo de permisos para autenticar y autorizar usuarios en la organización. En el capítulo del estado del arte de este proyecto se han visto varios de los más conocidos y utilizados en la industria.

El modelo de seguridad, que más se aplica a la realidad del trabajo, es aquel que enfoca el control de acceso basándose en roles, ya que los recursos pertenecen a la organización y lo que se necesita es que existan permisos sobre esos recursos para que un usuario, de acuerdo al rol que cumple en la organización, pueda o no realizar distintas operaciones sobre el mismo. El modelo RBAC cumple con estas características, ademas es usado por la mayoría de las organizaciones que cuentan con más de 500 empleados [23].

4.2. Software libre

En caso de necesitar software externo para implementar la solucion, el mismo debe ser libre. Se debe considerar que la institución elegida como instancia de trabajo, actualmente cuenta con dos servidores OpenLDAP en funcionamiento.

El sistema de autorización y autenticación creado debe cumplir con las políticas establecidas de software libre y de código abierto, por lo que de las distintas herramientas y marcos de trabajo estudiados en el capítulo de estado del arte se ha decidido tomar como mejor alternativa la herramienta de código libre que implementa el modelo RBAC del proyecto Apache llamado Fortress, que además pertenece al mismo proyecto que OpenLDAP por lo que aplica al marco de trabajo del proyecto.

4.3. Marco de trabajo para RBAC

Dado el modelo elegido es necesario tener un sistema capaz de implementarlo, además lograr insertarse dentro de los sistemas existentes en la organización por lo que deberá poder interoperar con una heteregeneidad de ellos.

Como consecuencia de los anteriores requerimientos ya vistos el proyecto utilizará como servicio de directorio OpenLDAP, que es la implementación libre y de código abierto del protocolo LDAP, que ya se encuentra en funcionamiento en la

organización.

Como se ha visto en el capítulo de estado del arte existen varias herramientas posibles para implementar el modelo de seguridad RBAC. La mayoría de ellas son propietarias por lo que se han desestimado. Según el estudio de los requerimientos anteriores la herramienta más adecuada que puede utilizarse para el trabajo es Fortress.

4.4. Perfiles de usuario

De los requerimientos surge la necesidad de agrupar los roles según perfiles de usuarios, de esta manera la aplicación web gestionada por el departamento de recursos humanos de la institución al crear o modificar usuarios les puede asignar o desasignar perfiles.

Con esto se logra desacoplar a la gestión de usuarios todo el conocimiento de roles y permisos vinculados a RBAC, por lo que la creación de perfiles y asignación de roles a cada perfil, que es un trabajo no tan dinámico, es realizado por encargados de seguridad e informáticos.

4.5. Restricción de permisos por ubicación

El sistema debe considerar que un usuario puede tener ciertos permisos en una sucursal de la organización y otros permisos en otra sucursal. Los permisos deben depender de la ubicación en la que el usuario ingresa al sistema.

Si bien el estándar RBAC brinda toda la solución para gestionar los permisos, los requerimientos del trabajo exigen algo más que el control de acceso basado en roles, también se requiere que los roles estén sujetos a una determinada ubicación en la organización. Entonces, el problema a resolver surge de que si bien el estándar RBAC provee de separación dinámica de restricciones de servicio para que dinámicamente, en tiempo de ejecución, sea posible controlar la aplicación de roles, el tipo de restricciones vía atributos arbitrarios no se encuentra completamente provisto.

De esta manera, el problema identificado surge del requerimiento de la organización de poder activar o desactivar roles de acuerdo al atributo de ubicación del usuario en el momento del ingreso al sistema. Es decir, el usuario identificado en una ubicación del sistema tiene los permisos asociados a el o los roles que posee activados, si y sólo si, está definido en el sistema que ese usuario tiene ese o esos roles asociados a la ubicación en la que se encuentra en ese momento. Resumiendo el requerimiento sería: sólo activar el rol *X* si el sujeto está presente en el lugar *Y*.

4.6. Aplicación web

El sistema debe contar con una interfaz para su administración y otra interfaz que sólo provea los servicios necesarios para su utilización. Se debe implementar

una aplicación web para administrar el sistema y que pueda ser utilizada por personal administrativo para realizar operaciones sobre los usuarios. Permitiendo que el personal de recursos humanos de la organización pueda gestionar los usuarios, asignando o desasignando perfiles de manera que internamente se traducirán en permisos dentro del sistema.

Además, aplicaciones externas deben ser capaces de usar el sistema para poder brindar autenticación y autorizaciones de manera centralizada.

4.7. Módulos de extensión a otros sistemas

El sistema implementado debe tener la capacidad de interactuar con los sistemas centrales de la organización, como son el servicio de correo corporativo, el sistema de archivos, las reglas para comandos sudo y el sistema de permisos para la red inalámbrica. Debe ser modular, de manera de interrelacionarse con otros sistemas en un futuro.

4.8. Resumen del capítulo

Del análisis realizado en este capítulo algunos de los requerimientos ya quedan verificados, es el caso de la elección del modelo de seguridad en donde claramente la opción por el modelo RBAC es la más adecuada. También, la elección del marco de trabajo para el modelo elegido y el desarrollo con código abierto quedan cumplidos con la decisión tomada por la herramienta Fortress.

Los requerimientos de contar con perfiles para los usuarios y la restricción de permisos basada en la ubicación, requieren de un mayor estudio por lo que serán abordados en próximos capítulos de este documento.

Finalmente, se realiza un primer análisis para los requerimientos de contar con una aplicación web y con módulos para la interacción con otros sistemas, que permite comenzar con el diseño y posterior desarrollo de los mismos.

5. Diseño

En este capítulo se explican las principales decisiones de diseño para el presente trabajo, así como también los distintos caminos estudiados para poder tomarlas.

5.1. Proceso de selección del marco de trabajo para RBAC

En esta sección se describen los distintos escenarios planteados para la implementación del modelo de RBAC.

Se consideraron dos caminos probables para poder llevar a cabo la implementación de RBAC en el proyecto, uno utlizando la implementación de RBAC elegida y el otro crear una propia. Para esto se crearon dos prototipos distintos y se analizaron sus ventajas y desventajas para luego poder tomar la decisión:

1. Utilizar alguna interfaz de programación de aplicaciones (API) disponible en el mercado para interconectar una aplicación Java con un servidor LDAP externo.
2. Utilizar el sistema Fortress que provee el proyecto Apache para la gestión de acceso e identidad basada en roles.

A continuación se realiza una descripción de cada uno de los prototipos realizados, su comparación y análisis, para luego tomar una decisión.

5.1.1. Prototipo con interfaz de programación de aplicaciones

Se escogió JNDI (Java Naming and Directory Interface) que brinda Oracle. JNDI es una interfaz de programación de aplicaciones que provee funcionalidades de nombrado y directorio a aplicaciones programadas en Java. Esta interfaz busca objetos en la red y en particular incluye a LDAP como proveedor de servicio de manera que puede soportar las capacidades adicionales de este protocolo [24].

De esta manera es posible conectarse al OpenLDAP desde la aplicación Java y así poder administrar y manejar el árbol de directorio del servidor.

Se desarrolla con éxito una aplicación Java que inserta, edita, busca y borra un objeto dentro del servidor OpenLDAP. Ver detalles de implementacion en anexo.

5.1.2. Prototipo con Apache Fortress

Apache Fortress es un kit de desarrollo libre y de código abierto el cual se puede calificar dentro de la categoría de software Identity Access Manager. Fortress es totalmente compatible con RBAC y ARBAC02. Este sistema brinda servicios y API'S para llevar a cabo: autenticación, autorización, administración, auditoría y políticas de contraseñas [17].

Algunas de las principales características de Fortress son:

- Administración de RBAC, a través de: APIs, servicios y páginas Web.
- Administración de contraseñas, a través de : APIs, servicios y páginas Web.
- Auditoría centralizada de las actividades realizadas.
- Replicación del árbol de directorios.

Se procedió a desarrollar un prototipo para evaluar las diferentes prestaciones brindadas por Fortress.

En el ambiente de prueba se utilizaron los paquetes brindados por el sistema Fortress. La instalación se llevó a cabo en Windows 7 y los elementos instalados son:

- APIs, para integrar la funcionalidad de fortress con el prototipo.
- RBAC Commander, aplicación web que brinda acceso a las funcionalidades brindadas por fortress.
- Un LDAP de prueba que proporciona Fortress

Se logra desarrollar una aplicación Java integrada con las API de fortress y se testean las funcionalidades básicas relacionadas con el manejo de usuarios. También se evalúa la posibilidad de utilizar la aplicación RBAC Commander que brinda Fortress para adaptarla a los requerimientos del trabajo completando así las funcionalidades necesarias. Sin embargo, esto fue desestimado ya que esta aplicación está hecha con el framework Wicket y tiene fuertemente acopladas las funcionalidades con la capa de presentación, esto se contrapone con los objetivos de hacer un API Java común a varias aplicaciones.

5.1.3. Análisis de los prototipos realizados

Luego de realizados los dos prototipos propuestos se evaluaron las siguientes características:

Documentación e información JNDI tiene una extensa documentación oficial brindada por Oracle, además una gran cantidad de información se puede encontrar en la red acerca de JNDI específicamente para LDAP con datos, ejemplos, código, problemas y soluciones encontradas por desarrolladores en todo el mundo.

Fortress está respaldada por el proyecto Apache, cuenta con una comunidad importante y activa. La documentación oficial es buena, sin embargo, está enfocada para utilizar los aplicativos que se descargan y utilizan el Fortress, como el RBAC Commander y Enmass Policy Server. Por lo que no resultó sencillo entender cómo funciona Fortress y qué hacer para comenzar a utilizarlo rápidamente. De todas maneras Fortress SDK Java Docs es muy completo y cuenta con una gran cantidad de ejemplos. La información no oficial de la comunidad de Fortress es casi nula.

Usabilidad JNDI resultó sencilla de implantar, sin embargo, para hacer un manejo de RBAC como el que se pretende sería necesario diseñar cuidadosamente la forma en que se trabajará sobre el servidor LDAP. Fortress no fue tan sencillo de implantar, pero una vez puesto en marcha, el manejo de RBAC está ya resuelto por el framework, siempre y cuando se respete la estructura generada por Fortress en el servidor LDAP.

Compatibilidad con el OpenLDAP de ASSE JNDI puede trabajar sin problemas sobre el LDAP existente de ASSE, sin embargo, se tendrá que estudiar si esta estructura deberá tener algún cambio para adaptarse a la gestión con RBAC. Fortress genera una estructura en el OpenLDAP que debe ser respetada, se comparó esta estructura con la actual de ASSE y si bien tiene diferencias no son de gran envergadura, es posible exportar la base de OpenLDAP actual e importarla en la base de LDAP que generar la instalacion de Fortress. Por lo tanto ambas soluciones son adecuadas para convivir con las necesidades actuales de ASSE.

Calidad de la implementación Con JNDI habría que implementar muchas de las funcionalidades que Fortress tiene para la gestión con RBAC por lo que se considera que la utilización de un framework como Fortress nos permitiría tener una aplicación de mejor calidad.

5.1.4. Decisión de la herramienta

La decisión es utilizar la herramienta Fortress debido a que el sistema posee una API completa para RBAC en Java, lo que posibilita que una vez acoplado el sistema la mayoría de las funcionalidades necesarias para el desarrollo del proyecto estén disponibles. Además tiene un sólido respaldo del proyecto Apache con una comunidad activa e importante detrás, haciéndolo un sistema estable, escalable y que garantiza muy buen nivel de calidad.

5.2. Diseño para restricción basada en la ubicación

Este tipo de restricción se enmarca en lo que se llaman restricciones basadas en atributos arbitrarios. Los ejemplos más comunes de este tipo de restricciones son atributos de tiempo y atributos de ubicación. A pesar de que la restricción temporal esta fuera del estándar de RBAC que implementa Fortress, éste si la soporta en la activación y desactivación de roles, por lo que utilizando este sistema es posible definir que un rol sea activado o no dependiendo del atributo temporal (hora, día de la semana, día del mes, etc) al momento del acceso.

El ejemplo práctico que brinda la documentación de Fortress para la restricción temporal es: *Un cajero de un banco es asignado al rol "Cajero", pero sólo puede actuar dentro de ese rol durante el horario normal de negocios, es decir, de lunes a viernes de 9:00 a 17:00hs*. Continuando este mismo ejemplo se aplica la restricción basada en ubicación de la siguiente manera: *Es necesario que el cajero del banco asignado al rol "Cajero" solo pueda actuar dentro de ese rol en la sucursal "A" de ese banco, de manera que si ese mismo cajero del banco va a la sucursal "B" del mismo banco e intenta ingresar al sistema, no debería poder*

actuar dentro del rol "Cajero" [17].

Se realizó un contacto con los técnicos del proyecto Fortress en Apache enviando un mail a *fortress@directory.apache.org*, esto permitió luego intercambiar mails con Shawn McKinney, investigar como Fortress podría resolver este problema. Gracias a esta comunicación surgieron las siguientes alternativas que se describirán brevemente:

- **Modificar Fortress**. Tomando como implementación del modelo de referencia RBAC el sistema de código abierto Fortress, es posible modificarlo para implementar un nuevo validador de activación de roles, tomando como ejemplo los validadores ya existentes que se encuentran desarrollados para la restricción temporal. El nuevo validador debe implementar la interfaz Validator y se debe registrar en Fortress agregándolo en el archivo de configuración correspondiente y dejando los binarios disponibles.

  Complementando esta alternativa se podría agregar en Fortress un nuevo tipo de dato para la ubicación. Modificando el código para que los roles acepten este atributo, que deberá ser multivaluado para poder contener una lista de ubicaciones permitidas para un rol y para que los usuarios acepten un atributo multivaluado de ubicaciones en la asociación con cada rol. También agregar el atributo ubicación al usuario para que sea seteado en el momento del ingreso al sistema y así utilizar el nuevo validador para el chequeo.

  Agregar a Fortress otra entidad (ubicación) y otras relaciones ($Rol \longleftrightarrow Ubicacion$ y $Usuario \longleftrightarrow Rol \longleftrightarrow Ubicacion$) no es algo trivial y probablemente lo aleje de su cometido que es implementar ANSI RBAC (INCITS 359).

- **Definir la ubicación en el rol**. Crear un rol para cada ubicación, es decir si queremos el rol "A" en las ubicaciones x e y , entonces creamos el rol "Ax" y "Ay". Luego, simplemente asignamos el rol al usuario deseado.

  Esta alternativa es muy sencilla y no necesita cambios en los sistemas, sin embargo, en la mayoría de los casos de uso reales implicaría tener una cantidad importante de combinaciones de roles con ubicaciones lo cual complica en gran medida la administración, mantenimiento y claridad del sistema.

- **Desarrollo en una aplicación externa**. Desarrollar una nueva aplicación que implemente las funcionalidades de RBAC y en la que además, podamos tener los elementos necesarios para extenderlas cumpliendo así con los objetivos planteados.

  Como ya se debe desarrollar una aplicación externa a Fortress para este trabajo, esta alternativa es la que se decidió realizar, agregando este punto como un requerimiento para la aplicación a desarrollar.

A continuación se verá una formalización que nos permita entender las funciones existentes en el modelo RBAC de manera de tener una idea clara de cómo fueron extendidas para permitir la restricción de roles de acuerdo al atributo de ubicación del usuario.

5.2.1. Formalización de componentes RBAC

En esta sección se formaliza lo definido en el modelo RBAC. Para la notación nos basaremos en la propuesta realizada en el articulo `The NIST model for RBAC` [25]. Se llaman User, Role, Op, Ob y Session a los tipos de los usuarios, roles, operaciones, objetos y sesiones respectivamente, que son los elementos básicos del modelo RBAC [26].

Conceptualmente, un permiso describe la posibilidad de realizar una operación sobre un objeto determinado, y se lo describe de la siguiente manera:

$Perm \stackrel{\text{def}}{=} [[op_0 : Op; ob_0 : Ob]]$

donde el campo op0 es una operación sobre el objeto del campo ob0.

Los permisos pueden estar asociados con uno o más roles. Cada asociación de un permiso a un rol está definida de la siguiente manera:

$PA_{elem} \stackrel{\text{def}}{=} [[r_1 : Role; p_1 : Perm]]$

De manera análoga, los usuarios pueden estar asignados a uno o más roles. Cada asignación de un usuario a un rol se define de la siguiente manera:

$UA_{elem} \stackrel{\text{def}}{=} [[u_0 : User; r_0 : Role]]$

Cada usuario establece una sesión durante la cual activa algún subconjunto de los roles para los que está autorizado. Cada sesión es un mapeo entre un usuario y sus roles activos, un usuario puede tener más de una sesión establecida de manera simultánea. En términos formales:

$se : Session -> SE_{elem}$

donde SE_{elem} es:

$SE_{elem} \stackrel{\text{def}}{=} [[usr : User; rl : setRole]]$

Una de las funciones que más interesa para este trabajo es la que toma las decisiones de control de acceso a los recursos protegidos por el sistema.

$CheckAccess : Session \longrightarrow Op \longrightarrow Ob \longrightarrow Funcs$

Cuya semántica es especificada mediante pre y post condiciones. La precondición es un predicado sobre el estado y la postcondición es un predicado que relaciona el estado anterior y el posterior a la ejecución del mismo y la respuesta correspondiente.

$CheckAccess(sess\ op\ obj)$

$Pre\ s\ (CheckAccess\ sess\ op\ obj)$

$$\begin{aligned}
&\stackrel{\text{def}}{=} isOper\ s\ op \\
&\quad \wedge isObj\ s\ obj \\
&\quad \wedge isSession\ s\ sess \\
Pos\ s\ s'\ answ(CheckAccess\ sess\ op\ obj) &\\
&\stackrel{\text{def}}{=} (\exists\ role : Role; isRole\ s\ role \\
&\quad \wedge rol_activo_en_sesion\ s\ role\ sess \\
&\quad \wedge perm_asignado_al_rol\ s\ op\ obj\ role) \\
&\quad \wedge answ = ok.
\end{aligned}$$

5.2.2. Formalización de la extensión propuesta

A los componentes del Core RBAC básico debemos agregarle lo que corresponde a una ubicación definida.

Entonces, la fórmula definida anteriormente para asociar un usuario a uno o más roles deberá ser modificada para permitir la utilización del componente de ubicación. Formalmente esta nueva asociación sería:

$$UA_{elem} \stackrel{\text{def}}{=} [[u_0 : User; r_0 : Role; ub_0 : Ubi]]$$

De esta forma la función que realiza las decisiones de control de acceso a los recursos protegidos por el sistema devolverá una respuesta afirmativa si la sesión posee algún rol activo, en la ubicación definida, que tenga asignado el permiso (op, obj), en caso contrario se deniega la autorización.

$$CheckAccess(sess\ op\ obj\ ubi)$$

$$\begin{aligned}
&Pre\ s\ (CheckAccess\ sess\ op\ obj\ ubi) \\
&\quad \stackrel{\text{def}}{=} isOper\ s\ op \\
&\quad\quad \wedge isObj\ s\ obj \\
&\quad\quad \wedge isSession\ s\ sess \\
&\quad\quad \wedge isUbi\ s\ ubi \\
&Pos\ s\ s'\ answ(CheckAccess\ sess\ op\ obj\ ubi) \\
&\quad \stackrel{\text{def}}{=} (\exists\ role : Role; isRole\ s\ role \\
&\quad\quad \wedge rol_activo_en_sesion\ s\ role\ sess \\
&\quad\quad \wedge rol_asignado_ubicacion_usuario\ s\ role\ ubi\ sess \\
&\quad\quad \wedge perm_asignado_al_rol\ s\ op\ obj\ role) \\
&\quad\quad \wedge answ = ok.
\end{aligned}$$

5.3. Base de datos

Para manejar las entidades que no se encuentran ni en el modelo RBAC ni en Fortress pero son necesarias para cumplir con la extensión del modelo propuesto en la sección anterior se necesita una base de datos externa. Además, de ser necesario tener almacenadas las ubicaciones posibles para el sistema, también es un requerimiento que existan perfiles de usuarios, es decir, que un perfil es un

conjunto de roles que puede asociarse a uno o más usuarios.

Se crea un esquema de base de datos usando el motor MySQL. En este esquema se guardan los perfiles de usuario disponibles en el sistema, los perfiles de cada usuario, los roles pertenecientes a cada perfil, las ubicaciones disponibles y los botones asociados a la aplicación web. Para esto se crearon las siguientes tablas:

```
CREATE TABLE `funciones` (
`OBJETO` varchar(30) NOT NULL,
`OPERACION` varchar(30) NOT NULL,
`ACCION` varchar(30) NOT NULL,
`NOMBRE_BOTON` varchar(30) NOT NULL,
PRIMARY KEY (`OBJETO`,`OPERACION`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `perfiles` (
`NOMBRE` varchar(15) NOT NULL,
`DESCRIPCION` varchar(50) DEFAULT NULL,
`OU` varchar(15) DEFAULT NULL,
PRIMARY KEY (`NOMBRE`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `perfiles_roles` (
`PERFIL` varchar(15) NOT NULL,
`ROL` varchar(30) NOT NULL DEFAULT '',
`IS_ADMIN` tinyint(1) DEFAULT NULL,
PRIMARY KEY (`PERFIL`,`ROL`),
CONSTRAINT `perfiles_roles_ibfk_1` FOREIGN KEY (`PERFIL`)
REFERENCES `perfiles` (`NOMBRE`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `perfiles_usuarios` (
`PERFIL` varchar(15) NOT NULL,
`USUARIO` varchar(15) NOT NULL DEFAULT '',
`UBICACION` varchar(30) NOT NULL DEFAULT '',
PRIMARY KEY (`PERFIL`,`USUARIO`,`UBICACION`),
KEY `UBICACION` (`UBICACION`),
CONSTRAINT `perfiles_usuarios_ibfk_1` FOREIGN KEY (`PERFIL`)
REFERENCES `perfiles` (`NOMBRE`),
CONSTRAINT `perfiles_usuarios_ibfk_2` FOREIGN KEY (`UBICACION`)
REFERENCES `ubicaciones` (`NOMBRE`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `ubicaciones` (
`NOMBRE` varchar(30) NOT NULL,
`DESCRIPCION` varchar(30) DEFAULT NULL,
PRIMARY KEY (`NOMBRE`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

La tabla Funciones, contiene la relación entre el permiso (objeto, operación) con los botones disponibles en la aplicación web, de esta manera se puede modelar que si el usuario que ingresa al sistema se le habilita o no el botón correspondiente.

La tabla Perfiles, contiene los perfiles para cada unidad organizacional junto con una descripción.

La tabla Perfiles_Roles, contiene el mapeo para cada perfil, que roles tiene y si el rol es administrativo o no.

La tabla Perfiles_Usuario, contiene el mapeo para cada usuario del sistema que perfiles tiene para cada ubicacion.

La tabla Ubicaciones, contiene las ubicaciones con una descripción de la misma.

5.4. Interfaces del sistema

Dado los requerimientos el sistema, el mismo debe contar con dos tipos de interfaces: una para su completo funcionamiento y otra más restringida que será utilizado por las herramientas externas al sistema.

Asi es que el sistema tiene una interfaz de programación contenida en un jar en donde se implementan todas las funciones necesarias para su administración, operación y configuración. Esta interfaz es la que utiliza la apliación web para que usuarios dependiendo de su rol, puedan realizar todas estas operaciones sobre el sistema.

Para la interfaz restringida se ha decidido exponer, mediante servicios web, las distintas operaciones necesarias para la utilización del sistema por herramientas externas tal como verificar un usuario, administración de sus propios permisos y roles, y consultas necesarias para operar con ellos.

5.5. Resumen del capítulo

En este capítulo se confirma que la utilización de la herramienta Fortress para el desarrollo del sistema, es más adecuada para este proyecto que comenzar a implementar el modelo RBAC desde cero. Si bien los dos prototipos desarrollados cumplieron sus objetivos, se entiende que una vez que Fortress es instalada y configurada brinda al sistema la mayoría de las funcionalidades necesarias para el proyecto.

Uno de los requerimientos que no pueden ser implementados con Fortress es el de restringir los roles de un usuario según en qué ubicación este ingresando al sistema, en este capítulo se presenta una formalización del modelo RBAC sobre

el cual poder realizar un diseño de la solución, que luego es formalizada en el modelo RBAC extendido.

Se diseñaron dos interfaces para cumplir con los requerimientos del sistema. Una interfaz brinda el acceso a todas las funcionalidades del sistema y la otra interfaz más restringida, se ofrece mediante un servicio web en donde cualquier sistema o aplicación externa puede validar usuarios, chequear permisos de usuarios, etc.

6. Implementación

En este capítulo se describe el entorno de desarrollo que se utilizó para la implementación del sistema, la arquitectura propuesta así como el proceso de desarrollo realizado junto al plan de pruebas llevado a cabo para garantizar su funcionamiento. Se hace una breve descripción de cómo se realizaron los módulos correspondientes a sistemas ya existentes, por ejemplo: correo electrónico, sistemas de archivos, etc.

6.1. Proceso de desarrollo

Una metodología es una secuencia de pasos ordenados con la cual se obtiene un resultado, para el desarrollo de software existen varias metodologías. Para este proyecto se usó la metodología de desarrollo en cascada y de prototipado evolutivo.

El desarrollo en cascada combinado con el prototipado ayuda en el proceso para desarrollar de una forma ordenada, según lo marca la metodología. Usar el prototipado permite una retroalimentación temprana de los requerimientos implementados, así como también probar la tecnología a medida que evoluciona en el proceso de desarrollo y de esta manera poder tomar decisiones en forma más temprana.

Un requerimiento solicitado fue tener una capa de presentación web para validar el desarrollo con el cliente, debido a que el equipo no contaba con mucha experticia en el desarrollo web, inicialmente se prototiparon varias aproximaciones usando WebCommander, jsp, jsf y RichFaces. Se midió la facilidad de uso y adaptabilidad de las tecnologías con la capa de negocio implementada que interactua con el framework fortress prototipando casos en cada tecnología. [5].

6.1.1. Desarrollo en cascada

El desarrollo en cascada cuenta con una serie de etapas que deben completarse una a una antes de pasar a la siguente.

Análisis y definición de requerimientos: Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.

Diseño del sistema y del software: El procesos de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

Implementación y prueba unitaria: Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.

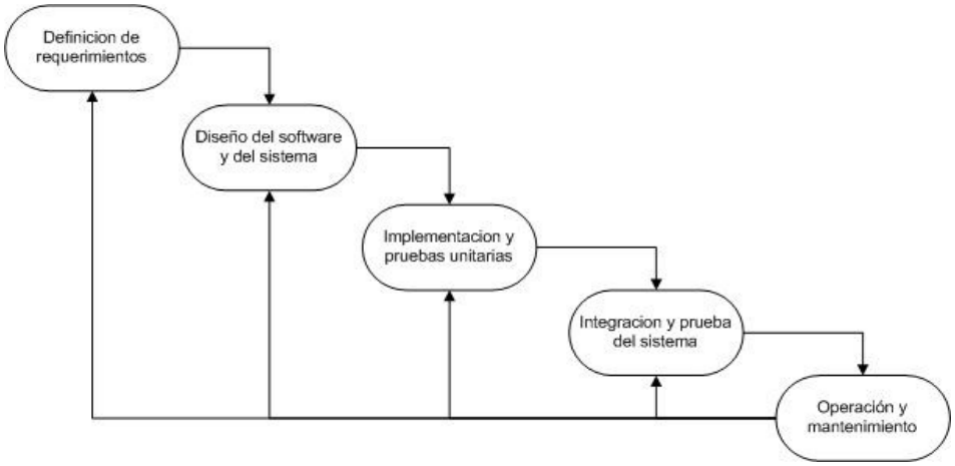

Figura 9: Modelo de desarrollo en cascada,extraída de [4]

Integración y prueba del sistema: Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.

Funcionamiento y mantenimiento: Por lo general (aunque no necesariamente), ésta es la fase mas larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

6.1.2. Prototipado y prototipado evolutivo o modelo incremental

El modelo de prototipos [5] permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se necesita determinar que el desarrollador, el usuario y el cliente estén de acuerdo en lo que se necesita, así como también la solución que se propone para dicha necesidad y de esta forma minimizar el riesgo y la incertidumbre en el desarrollo.

Este modelo se encarga del desarrollo de diseños para que estos sean analizados y prescindir de ellos a medida que se adhieran nuevas especificaciones, es ideal para medir el alcance del producto, pero no se asegura su uso real. Este modelo principalmente se lo aplica cuando un cliente define un conjunto de objetivos generales para el software a desarrollarse sin delimitar detalladamente los requisitos de entrada, procesamiento y salida, es decir cuando el responsable no está seguro de la eficacia de un algoritmo, de la adaptabilidad del sistema o de la forma en que interactúa. Este modelo se encarga principalmente de ayudar al ingeniero de sistemas y al cliente a entender de mejor manera cuál será el resul-

tado de la construcción cuando los requisitos estén satisfechos.

El prototipado evolutivo o modelo incremental es la unión de las mejores funcionalidades del modelo en cascada y del prototipado, permitiendo lograr mejoras incrementales a medida que se realiza la entrega de un prototipo, de esta manera se tienen las experiencias del proceso anterior insertas en cada entrega de prototipo. La diferencia con el modelo de prototipo es que el incremental busca ser operacional en cada entrega, y no ser solo un test de como sería el sistema final.

El paradigma de construcción de prototipos tiene tres pasos:

1. Escuchar al cliente. Recolección de requisitos. Se encuentran y definen los objetivos globales, se identifican los requisitos conocidos y las áreas donde es obligatorio más definición.
2. Construir y revisar el prototipo.
3. El cliente prueba el prototipo lo utiliza para refinar los requisitos del software.

6.1.3. Justificación de la metodología empleada

Dados los requerimientos de usar OpenLDAP junto a herramientas de software libre, se busca la existencia de un marco de trabajo (framework) de autenticación y autorización que estuviera enmarcado en los requerimientos del proyecto. Como se menciona en este documento, Fortress fue elegido. En las primeras pruebas de funcionamiento se presenta la necesidad de una aplicación web que validara el sistema a desarrollar, para esto se comienza a utilizar jsp.

En las primeras entregas del prototipo, el usuario ve la necesidad de que las pantallas tengan mayor complejidad, por lo que se debe comenzar a investigar herramientas que aceleraran el proceso de desarrollo de pantallas web y lo hicieran mas amigable, ya que el equipo de trabajo no cuenta con mucha experticia en el desarrollo web.

Se inician varias ramas de prototipos, con Wavemaker, con JSF y con RichFaces. Wavemaker fue descartado tempranamente porque no era de gran ayuda para este proyecto en concreto, por lo que la linea de JSF con RichFaces es el camino elegido. En el proceso de entrega se muestran al cliente dos prototipos en JSF, estos le resultan satisfactorios para lo que se pretendía tener como web para validar el proyecto, pero resulta complejo el desarrollo de determinadas funcionalidades para validar el sistema. Por otro lado, RichFaces brinda una suit amplia de componentes que resultan sencillos de incluir en la presentación, por lo que se sigue con esa línea de trabajo.

Se implementa la capa de presentación usando JSF 2.0 y RichFaces 3.4 para crear la web de validación del proyecto. Cada entrega de prototipo realizada sirve para plantear nuevos escenarios y así evolucionar los requerimientos y el software a lo que se realmente se necesita.

Uno de los requerimientos del proyecto es que el cambio de tecnología que se usa en la capa de presentación sea totalmente independiente, esto se cumple ya que la capa de negocio presenta una interfaz de programación completa. Este requisito se verifica en la evolución del desarrollo ya que distintos prototipos cambiaron de tecnología en la capa de presentación. Por otra parte, se puede prescindir de la web y usar el sistema invocando servicios web que ofrecen operaciones de consulta de permisos, validaciones, etc. Para el mantenimiento de los componentes del sistema es posible proceder de igual manera y crear nuevas funciones en el servicio web para invocar al sistema.

Las operaciones ofrecidas por el servicio web tienen como fin la interacción con aplicaciones externas para autorizar y autenticar usuarios de éstas. La capa web tiene como principal cometido servir de herramienta a los usuarios de RRHH y administradores en el mantenimiento de los componentes del sistema, tales como los roles, permisos, objetos, etc.

6.2. Entorno de desarrollo

La aplicación fue implementada utilizando Java JDK versión 1.7 para el servidor de aplicaciones JBOSS versión 6.1.0 [27]. En la capa de presentación se utilizó JSF 2.0 y RichFaces 3.4 [28]

El sistema utiliza como base las API de Fortress 1.0 RC36 [17] y OpenLDAP versión 2.4.39 [3]. La base de datos utilizada es MySQL version 5 [29].

Durante el desarrollo de la solución se utilizó un repositorio subversión instalado en los servidores del cliente para mantener el código fuente de manera centralizada y trabajar en forma paralela y organizada [30]. El cliente SVN utilizado fue Tortoise.

Para la generación de documentos se utilizó la herramienta TeXstudio 2.9.4 para crear documentos en LateX [31].

6.3. Arquitectura

La arquitectura propuesta para el sistema se basa en el modelo cliente-servidor. Para proveer las funcionalidades necesarias para cumplir con los requerimientos del proyecto en la figura 10 se puede ver el diagrama de componentes del sistema
.

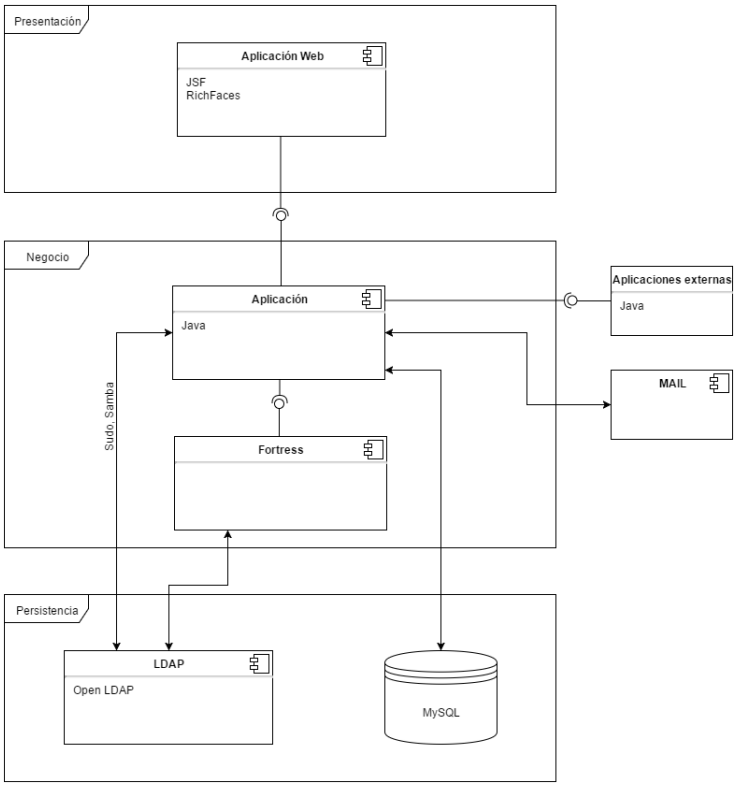

Figura 10: Diagrama de componentes del sistema

El sistema consta de una aplicación Java encargada de la lógica y que es la encargada de comunicarse con Fortress utilizando la interfaz Java que éste brinda. Además, la aplicación se comunica con la base de datos MySQL que contiene el modelo de datos para el manejo de usuarios y perfiles, mediante el desarrollo de módulos se permite la comunicación con sistemas externos como por ejemplo el servidor de correos o el propio OpenLDAP directamente para realizar operaciones que gestionan recursos para Samba o comandos Sudo.

La aplicación expone un API Java que permite realizar todas las funcionalidades para gestionar, configurar y administrar completamente el sistema implementado, incluyendo las extensiones al modelo RBAC para manejar perfiles y ubicaciones.

A su vez, se desarrolla una aplicación web utilizando el API Java generado anteriormente para el manejo administrativo del sistema, y poder validar con la misma todas las funcionalidades implementadas. En la figura 11 se muestra el diagrama de despliegue con los protocolos de comunicación entre cada uno de los elementos.

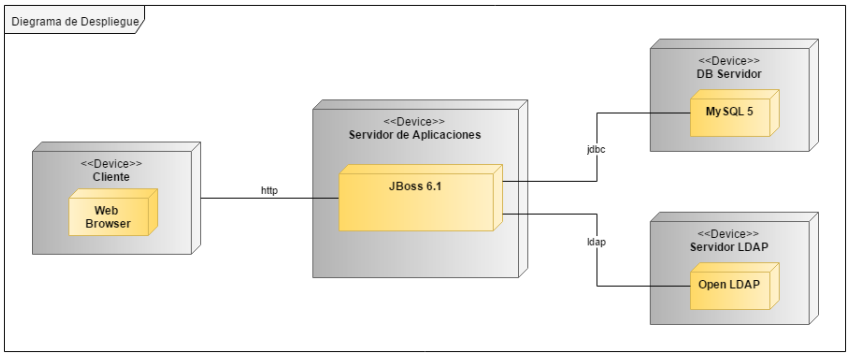

Figura 11: Diagrama de despliegue del sistema

Como se ha visto en el presente documento, para el modelo RBAC, del estándar implementado por Fortress, no existen perfiles ni ubicaciones, estas entidades se modelaron en una base de datos externa. El sistema mantiene las relaciones de perfiles con ubicaciones para los usuarios, así se logra implementar los requerimientos geográficos y de perfiles de usuarios, logrando así extender la implementación de Fortess para RBAC.

Para resumir, la arquitectura utilizada en el presente trabajo está compuesta de los siguientes componentes:

- Un servidor de aplicaciones JBoss que contiene:
  - Fortress.
  - Aplicación Java.
  - Aplicación Java Web.
- Un servidor OpenLDAP.
- Un servidor MySQL.

6.4. Implementación

En esta sección se describirán aspectos que tienen que ver con el desarrollo de la aplicación. Siguiendo el paradigma de programación por capas identificamos las siguientes:

- Presentación. Es la interfaz web para administrar el sistema.
- Negocio. Se encuentran las entidades necesarias para poder resolver la restricción basada en ubicación, para activar y desactivar roles dinámicamente en tiempo de ejecución. También se presenta una interfaz con el sistema Fortress asociado al OpenLDAP.
- Persistencia. Se accede a la base de datos MySQL donde se encuentran almacenadas las relaciones necesarias para resolver la restricción basada en ubicación.

La aplicación desarrollada utiliza el framework Fortress mediante llamadas a su API Java, y este último implementa todas las funcionalidades de configuración, administración y mantenimiento, por lo cual no es necesario implementarlas, pero si se requiere almacenar en la base de datos las asociaciones necesarias para cumplir con los requerimientos de perfiles de usuario y la restricción geográfica, dichas asociaciones son:

$Perfil \longleftrightarrow Rol$.
$Ubicaciones$.
$Usuario \longleftrightarrow Rol \longleftrightarrow Ubicacion$.

La asociación $Usuario \longleftrightarrow Rol$ ya existe en Fortress (núcleo RBAC), y éste la almacena en el LDAP, pero se necesita reproducirla en la aplicación desarrollada para permitir a todo el sistema contar con el atributo de ubicación en dicha asociación. Estas relaciones se almacenan en la base de datos MySQL.

Al momento del acceso de un usuario al sistema, se debe contar con el atributo de la ubicación, ya sea que el usuario lo ingresa, o que se determina automáticamente mediante la dirección ip desde la que accede, de esta forma el sistema crea la sesión de Fortress y consultando la asociación (usuario, ubicación) creada en la aplicación es posible activar y desactivar los roles según el atributo de ubicación.

6.5. Implementación de Módulos Específicos

6.5.1. SAMBA

Como se menciono en el capitulo 3.2 se cuenta con la herramienta smbldaptools para administrar usuarios y maquinas en samba. El script smbldap-useradd es parte de esta herramienta, es utilizado para dar de alta los usuarios o maquinas, el mismo se conecta al ldap configurado e inserta las entradas correspondientes. Se realizo el procedimiento de ingeniería inversa, donde se analizo el código de dicho script para obtener los atributos necesarios para realizar desde la aplicación el manejo de los usuarios y maquinas.

A continuación se detallan cuales son los atributos de openldap necesario para los usuarios:

```
objectClass = sambaSamAccount, para poder utilizar los atributos
samba.

sambaSID

sambaAcctFlags = [U], indica que se trata de un usuario

sambaHomeDrive

sambaKickoffTime

sambaLMPassword

sambaLogoffTime
```

```
sambaLogonScript

sambaLogonTime

sambaNTPassword

sambaPrimaryGroupSID

sambaPwdCanChange

sambaPwdLastSet

sambaPwdMustChange
```

La entrada en openldap que hace referencia a una maquina tiene leve variaciones:

```
objectClass = posixAccount

objectClass = account

objectClass = sambaSamAccount, para poder utilizar los atributos
samba.

cn = nombreDeLaMaquina$

gidNumber= GrupoDeMaquinas

sambaSID

sambaAcctFlags = [W], indica que se trata de una maquina

sambaDomainNam

sambaHomeDrive

sambaKickoffTime

sambaLMPassword

sambaLogoffTime

sambaLogonScript

sambaLogonTime

sambaNTPassword

sambaPrimaryGroupSID

sambaPwdCanChange=0

sambaPwdLastSet

sambaPwdMustChange
```

Se describen los atributos principales de lo anterior para que quede claro para que sirven y como se utilizan.

sambaLMPassword: guarda la contraseña LanMan de 16 bytes, la misma se representa como una cadena de caracteres en formato hexadecimal.

sambaNTPassword: guarda la contraseña NT de 16 bytes, la misma se representa como una cadena de caracteres en formato hexadecimal.

sambaSID: corresponde al identificador de seguridad para el usuario.

sambaAcctFlags: es un string de hasta once caracteres que representan las flags de la cuenta. Los valores mas comunes que encontramos son: U identifica que es un usuario, W identifica que se trata de una máquina, X significa que la contraseña no expira nunca, D indica que la cuenta esta desahabilitada.

sambaDomainNam: indica el nombre del dominio al cual pertenece el usuario.

sambaHomeDrive: es una letra que identifica el driver que es utilizado en conjunto con la variable sambaHomePath, para formar el camino UNC que indica el home del usuario.

sambaKickoffTime: este atributo indica el tiempo en el cual un usuario será bloqueado y por lo tanto no podrá conectarse mas. En caso de omitir este atributo la cuenta del usuario no expira nunca.

sambaLogoffTime: este atributo esta en desuso, pero igualmente se agrega en la información del usuario por estar incluido en el schema de samba que se utiliza en OpenLDAP.

sambaLogonScript: a través de este atributo se especifica el camino que identifica el script que se ejecuta cuando un usuario inicia sesión en el dominio. La extensión de dicho script puede ser .exe .bat .cmd.

sambaLogonTime: este atributo esta en desuso, pero igualmente se agrega en la información del usuario por estar incluido en el schema de samba que se utiliza en OpenLDAP.

sambaPrimaryGroupSID: indica el identificador de seguridad del grupo primario del usuario.

sambaPwdCanChange: especifica el tiempo que debe esperar un usuario para que pueda cambiar su contraseña. Si no es configurado, el usuario puede cambiar la contraseña cuando lo desee.

sambaPwdLastSet: este atributo indica en segundos el tiempo transcurrido desde la ultima vez que fueron modificados los atributos sambaLMPassword y sambaNTPassword. Hay que tener en cuenta que el tiempo en segundo es tomado desde el año 1970, no desde cuando precisamente se modificaron los atributos.

sambaPwdMustChange: se indica cuando el usuario debe cambiar su contraseña. Si el valor es cero, el usuario es forzado a cambiar la contraseña en su próximo inicio de sesión.

Las principales diferencias que se tienen en cuenta para describir que una entrada es un usuario o una maquina son las siguientes:

1. El valor que se indica en el atributo sambaAcctFlags, si el mismo es U se hace referencia a un usuario y si el valor es W se hace referencia a una maquina.

2. En caso de que se trate de un usuario el atributo uid puede contener cualquier cadena de caracteres, cuando se trata de una maquina se debe ingresar al final de la cadena el signo $.

6.5.2. Correo Electrónico

La implementación realizada maneja el alta y baja de mail. En el caso del alta de email se procesan las posibles colisiones que se puedan ocasionar, el requisito es que las direcciones de correo electrónico se definan utilizando los nombres y apellidos de los usuarios, por tal motivo se puede dar que dos personas tengan igual nombre y apellido y ahí se genera una colisión. Para mitigar dichas colisiones se definió un algoritmo que a la hora de crear la dirección de correo electrónico es ejecutado, el mismo se vale de los nombres y apellidos del usuario para asignar la dirección de correo.

1. chequear si la dirección formada por:
primer_nombre.primer_apellido esta disponible. Si esta disponible se opta por esta dirección, sino se continua en el siguiente punto.

2. chequear si la dirección formada por: primer_apellido.primer_nombre esta disponible. Si esta disponible se opta por esta dirección, sino se continua en el siguiente punto.

3. En caso de tener segundo nombre seguir desde aquí, sino ir directamente al punto 7.Chequear si la dirección formada por:
primer_nombreSegundo_nombreprimer_apellido esta disponible. Si esta disponible se opta por este dirección, sino se continua en el siguiente punto.

4. chequear si la dirección formada por el segundo_nombreprimer_apellido esta disponible. Si esta disponible se opta por esta dirección, sino se continua en el siguiente punto.

5. chequear si la dirección formada por:
el primer_nombreSegundo_nombreprimer_apellidoSegundo_apellido esta disponible. Si esta disponible se opta por esta dirección, sino se continua en el siguiente punto.

6. en caso de no encontrar aun una dirección disponible se le anexa un numero a la dirección formada por:
primer_nombreSegundo_nombreprimer_apellidoSegundo_apellido, para esto se hará uso de un contador que ira en aumento hasta encontrar la dirección disponible. Se encontrara en algún momento una dirección libre por tanto se termina la búsqueda

7. chequear si la dirección formada por:
   primer_nombre.primer_apellidoSegundo_apellido esta disponible. Si esta disponible se opta por esta dirección, sino se continua en el siguiente punto
8. chequear si la dirección formada por:
   primer_apellidoSegundo_apellido.primer_nombre esta disponible. Si esta disponible se opta por esta dirección, sino se continua en el siguiente punto
9. En caso de que aun no se haya encontrado una dirección libre se le anexa un numero a la dirección:
   primer_nombre.primer_apellidoSegundo_apellido, para esto se hará uso de un contador que ira en aumento hasta encontrar la dirección disponible. Se encontrara en algún momento una dirección libre por tanto se termina la búsqueda

6.6. Configuración de Samba + Openldap

En esta sección se describe la configuración necesaria para conectar Samba con Openldap.

6.6.1. Configurar Openldap

En primera instancia debemos configurar el servidor Openldap para que incluya el schema correspondiente al samba. Para esto debemos editar el archivo slapd.conf, la ubicación del mismo puede depender de la distribución linux en la que se este trabajando.
Generalmente el archivo se ubica en `/etc/openldap/slapd.conf`. Editamos el mismo y en la sección de inclusiones de esquemas agregar la siguiente linea: `include /etc/openldap/schema/samba.schema`. Algo recomendable es configurar los indices referentes a los atributos relevantes de samba, para esto en el archivo de configuración mencionado podremos agregar las siguientes lineas:

```
index sambaSID eq

index sambaPrimaryGroupSID eq

index sambaDomainName eq
```

6.6.2. Configuración Samba

Samba basa su configuración en el archivo `/etc/samba/smb.conf`, no se entrará en detalle de todos los parámetros que contiene. A continuación se detallan los atributos a tener en cuenta:

Usuario administrador del dominio

```
admin users = Administrator @"Domain Admins"
```

Configuración del servidor ldap donde nos conectamos

```
passdb backend = ldapsam:ldap://localhost
ldap suffix = dc=dominio,dc=com,dc=uy
```

```
ldap user suffix = ou=People
ldap group suffix = ou=Group
ldap machine suffix = ou=Computers
ldap admin dn = cn=Administrador,dc=dominio,dc=com,dc=uy
ldap passwd sync = yes
```

Para que las maquinas se agreguen automáticamente al dominio debemos configurar:

```
add machine script = /usr/sbin/smbldap-useradd -w %u
```

Ahora solo resta indicar cual es la contraseña del usuario especificado con el parámetro *ldap admin dn*, para esto ejecutamos el comando *smbpasswd -w password*.

6.6.2.1. Samba Tools Como se menciono anteriormente samba-tools se trata de un conjunto de scrpits desarrollados en perl, que permiten administrar las entradas de ldap. Se necesita realizar una configuración previa para poder utilizar esta herramienta.

En primera instancia especificamos el usuario con el cual nos conectaremos al servidor Openldap.
Luego para esto editamos el archivo: `/etc/smbldap-tools/smbldap_bind.conf` y modificamos los datos según nuestro ambiente, en el ejemplo que manejamos el usuario administrador sería *cn=Administrador,dc=dominio,dc=com,dc=uy*.

Para seguir adelante con la configuración debemos obtener el SID del dominio configurado, el SID es un valor único que sirve para identificar una entidad en sistemas Windows. El comando utilizado para obtener dicho valor es: *net getlocalsid*.

La configuración principal de samba-tools ser realiza en el archivo: `/etc/smbldap-tools/smbldap.conf`, teniendo en cuenta el valor obtenido en el paso anterior adaptar los valores de este archivo a nuestro entorno.

Por ultimo debemos poblar el árbol LDAP con las entradas de samba necesarias para comenzar a trabajar, dicha carga se realiza con el comando: `smbldap-populate -a Administrator`.

6.7. Configuración Sudo

Como se menciono en el capitulo 3.2 es posible utilizar openldap como backend del comando sudo. Es necesario realizar configuraciones a nivel del servidor y a nivel de los clientes. En el servidor que aloje el servicio de openldap se realiza la configuracion referente a la carga de datos, mientras que en los clientes es necesario realizar la configuración de la lectura de dichos datos.

6.7.1. Configuración del servidor

Como se menciono anteriormente es necesario realizar la configuración que permite realizar la carga de datos de las entradas correspondientes a sudo. Al igual que la configuración de samba Openldap debe contar con el schema para manejar las entradas de sudo se debe incluir el esquema correspondiente a los

objetos de sudo. Sera necesario entonces agregar la linea que indique la inclucion del esquema apropiado, `include /etc/openldap/schema/sudo.schema`

En el trabajo realizado se utilizo como base para el manejo de la información del sudo la definición de la unidad organizativa dentro de la cual se agregaran las entradas correspondientes. Para llevar adelante esto se hizo uso de la siguiente definición donde se especifica la que la ou sera SUDOers y que esta directamente bajo la entrada `dc=asse,dc=uy`

```
dn: ou=SUDOers,dc=dominio,dc=uy

objectClass: top

objectClass: organizationalUnit

ou: SUDOers
```

Luego de realizada la definición de la OU que contendrá las entradas correspondientes a los sudos, se registra una entrada por defecto que el comando sudo busca en primera instancia y cumple la misma función que las definiciones `Defaults` que se encuentra en el archivo `/etc/sudoers`. El parametro para observar de es el sudoOption, el mismo es multivaluado, en el ejemplo que se expone la variable SSH_AUTH_SOCK es mantenida para los usuarios que hagan uso del comando sudo.

```
dn: ou=Sudoers,dc=dominio,dc=uy

objectClass: organizationalUnit

ou: Sudoers

dn: cn=defaults,ou=Sudoers,dc=dominio,dc=uy

objectClass: top

objectClass: sudoRole

cn: defaults

description: Default sudoOption

sudoOption: env_keep+=SSH_AUTH_SOCK
```

En una entrada particular podemos contar con los siguientes atributos:

sudoHost: puede ser un nombre de host, una direccion ip, un sergmento de red. En caso de especificar el valor ALL, indica que puede ser cualqueir host.

sudoCommand: se trata de un comando linux, que opcionalmente puede incluir parametros

sudoRunAsUser: indica el usuario con el cual correra el comando

sudoUser: indica el usuario al cual se le aplica el sudo

6.7.2. Configuración del cliente

En este caso se le llama cliente a todo sistema linux que necesite leer las entradas de openldap. Como se menciono en el capitulo 3.2 el comando sudo por defecto obtiene las definiciones de sudo desde el archivo `/etc/sudoers`, en este caso se debe configurar el sistema para que el comando sudo pueda obtener dichas definiciones desde el servidor openldap.

En primera instancia debemos configurar el sistema para que opere como cliente de un servidor openldap, para esto es necesario tener instalados una serie de programas y configurar el cliente mediante el archivo `/etc/ldap.conf`. Cabe señalar que esta serie de pasos servirá para que otras herramientas, no solo el comando sudo, puedan utilizar el servicio de openldap de diferentes maneras, de todas formas en este capitulo si bien hay puntos que se aplican a otras herramientas, la configuración sera basada en uso del comando sudo. Los nombres de los programas puede variar dependiendo de la distribución de linux que se este utilizando, la idea no es entrar en detalle sobre esto y por tal motivo se toman en cuenta dos distribuciones: opensuse y ubuntu. En principio debemos contar con tres paquetes para que el sistema linux cumpla la función que necesitamos. El primer paquete instala las cosas necesarias para que el sistema actué como cliente de un servidor openldap. El segundo paquete permite interactuar al servicio Pluggable Authentication Modules (PAM) [32] con ldap, PAM es un conjunto de módulos que permite realizar la autenticacion de los usuarios de manera flexible y centralizada. El tercer paquete permite utilizar ldap en la configuración Name Service Switch (nsswitch) [33] del equipo. Básicamente la configuración del nsswitch) determina donde el sistema realiza la resolución de nombres y en que orden, por ejemplo ayuda a resolver donde buscar un nombre de usuario, o un nombre de grupo, o un nombre de host, etc., la configuración se lleva a cabo en el archivo `/etc/nsswitch.conf`. Para el caso de opensuse los paquetes mencionados son: libldap, pam_ldap y nss_ldap, en ubuntu los nombres varian un poco: libpam-ldap y libnss-ldap.

A continuación se muestra como quedan los principales parámetros que debemos tener en cuenta para configurar nuestro cliente, en el anexo C de este documento podremos encontrar los archivos completos.

En la configuración de nsswitch debemos agregar que ldap también sea backend en la busqueda de usuarios y grupos, por lo tanto en el archivo:
`/etc/nsswitch.conf` debemos contar con lo siguiente:

```
passwd: compat ldap
group:  compat ldap
```

Como se menciono anteriormente configurar un sistema para que funcione como cliente de ldap no solo nos servira para el comando sudo, pero aquí nos centraremos en las definiciones necesarias para que el comando sudo lea las entradas desde el servidor ldap.

> **BIND_TIMELIMIT:** especifica en segundo el tiempo limite que se espera para conectarse al servidor ldap, en caso de que se haya especificado mas de un servidor es el tiempo que se espera para intentar conectarse a otro servidor

BINDDN: en este parámetro indicamos el usuario ldap con el cual nos conectaremos al servidor, se debe indicar el usuario utilizando el formato de nombre distintivo (dn, Distinguished Name).

BINDPWD: indica la contrasena utilizada cuando se realice operaciones (lectura, escritura) sobre el ldap, siempre se utiliza en conjunto con el parametro BINDDN. La contrasena puede ser escrita en texto plana o codificada utilizando el algoritmo base64.

URI: indica uno a mas servidores ldap, se utiliza el formato uri para indicar el servidor y en caso de que se quiera especificar mas de uno se hace como una lista con espacios en blanco entre cada servidor. El formato par indicar un servidor es `ldap[s]://hostname[:port]`

HOST: en caso de que no se haya especificado ningun servidor en el parametro URI aqui se puede especifica el o los servidores ldap a urilizar, el formato difiere al que se utiliza en el parametro URI siendo de la sigueinte manera: host

$$: port$$

SUDOERS_BASE: indica la base en donde se buscaran las entradas de sudo en el ldap, en nuestro ejemplo se definio como base para las entradas sudo `ou=Sudoers,dc=dominio,dc=uy`

6.8. Configuración del servidor de Correo

El objetivo de esta sección es describir como se integra el servidor de correo con OpenLDAP. En principio la integración consiste en que el servidor de correo pueda realizar la autenticacion de sus usuarios utilizando OpenLDAP. Si bien el servidor de correo puede realizar otro tipo de tareas con OpenLDAP nos enfocaremos en lo mencionado. Volviendo al entorno de trabajo en el que se basa este proyecto, ASSE, recordemos que se cuenta con el software Zimbra instalado, si bien zimbra es una solucion colaborativa que puede conectarse con diferentes soluciones de servidores de correo como ser: sendmail, qmail, postfix; nos enfocaremos en postfix que es como se encuentra configurado.

6.9. Resumen del capítulo

El proceso de desarrollo transitado permite ir entendiendo los problemas del proyecto y encontrando nuevos requerimientos de manera de enriquecer el sistema, encontrar problemas y obstáculos en forma temprana en el desarrollo y poder tener una retroalimentación permanente con la contraparte del proyecto.

El entorno de desarrollo alcanzado es colaborativo y totalmente compatible con los sistemas existentes en la organización, de forma de minimizar los posibles problemas al momento de la implantación.

Para cumplir con todos los requerimientos de perfiles de usuarios y restricciones geográficas, se utiliza una base de datos externa donde poder almacenar las asociaciones adicionales que no incluye Fortress. Para esto se cuenta con una

base de datos MySQL que permite interactuar con la aplicación desarrollada.

Se define una arquitectura en capas en donde quedan claramente separadas la aplicación web con el sistema encargado de la comunicación con Fortress, OpenLDAP, MySQL y los diferentes módulos para sistemas externos como el correo electrónico, etc.

Se realizan las configuraciones necesarias para la integración con Samba, Sudoers y para el servidor de correo electrónico. Esta integración requiere de una serie de cambios en la configuración del OpenLDAP, sin embargo, se pueden realizar sobre el servidor que ya tenía en funcionamiento en la organización sin perder los datos existentes.

7. Testing

Para realizar la validación de esta aplicación se ha decidido utilizar el método de prototipado evolutivo. Este método se basa en la idea de desarrollar una implementación inicial exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta desarrollar un sistema que cumple con los requerimientos.

El prototipo consiste en una aplicación web para el sector de recursos humanos de una institución que cuenta con varias sucursales de distinto tipo, y distribuidas en una amplia área geográfica. La aplicación es usada por personal de recursos humanos de distinta jerarquía de manera que el personal encargado de dar de alta usuarios y asignar roles en una sucursal, sólo lo puede hacer dentro de los usuarios pertenecientes a esa sucursal. Los usuarios administrativos de la aplicación realizan el mantenimiento de la misma, manteniendo roles, permisos y políticas de seguridad.

En este proceso se fueron generando distintos casos de prueba a medida que el prototipo iba evolucionando. Los principales a mencionar se pueden dividir entre las funcionalidades necesarias para el personal de recursos humanos, las funcionalidades que necesitan los administradores de la aplicación y el sistema en si:

- Personal de recursos humanos. Estas funcionalidades requieren de permisos basados en la ubicación de logueo del usuario administrativo. El usuario al momento de loguearse fija la ubicación, luego contra esta ubicación se validan las acciones que puede realizar (permisos) dependiendo de los perfiles asignados al usuario y los roles asignados a los perfiles. De esta manera el usuario tendrá asignado un conjunto reducido de permisos basado en la ubicación dónde se logueó. El usuario eventualmente podría tener otros perfiles en otra ubicación, por lo que es necesario testear los ABM usando combinaciones de situaciones donde el usuario posea el permiso en una ubicación y en otra no, y probar ambos casos, logueandose en la ubicación con y sin permisos para el permiso a testear.
    - Alta, baja y modificación de usuarios en la sucursal.
    - Asignar usuarios a roles. Esta funcionalidad asigna un usuario de la sucursal a un rol del sistema pero sólo para dicha sucursal.
    - Desasignar usuarios a roles. Esta funcionalidad quita la asignación de un usuario de la sucursal a un rol del sistema que exista en esa sucursal.
    - Resetear contraseñas de usuarios pertenecientes a la sucursal.
- Administradores. Los administradores del sistema se encargan de crear y mantener la estructura de permisos, roles, politicas de contraseñas y perfiles del sistema. Estas funcionalidades no son dependientes de la ubicación dónde se logueó el usuario administrativo. Los perfiles no se mantienen dentro de la aplicación ya que exedía el proposito del proyecto, debido a este motivo las operaciones a testear no incluyen lo referente a perfiles.

- Alta, baja y modificación de roles.
- Alta, baja y modificación de permisos.
- Alta, baja y modificación de políticas de contraseñas.
- Asignar y desasignar permisos a roles.

- Sistema. Estas funcionalidades no son propias de usuarios, pero tienen un nivel de importancia medular en el sistema, estas funciones usan el núcleo del proyecto, que es la asignación de permisos. Fortress al momento de validar el usuario y la contraseña, crea una sesión y le asigna todos los permisos que posee el usuario, la validación consiste en validar que una vez logueado el usuario los datos de la sesión corresponden con los permisos que el usuario debe tener.
  - Validar usuario, contraseña (para una ubicación dada) y crear la sesión.
  - Chequear permisos en la sesión dependiente de la ubicación dada.

7.1. Casos de uso

7.1.1. Descripción General

La aplicación está implementada usando el framework Fortress, que implementa el modelo de seguridad ARBAC y RBAC. Fortress utiliza LDAP como almacenamiento de los datos. Para completar los requerimientos solicitados, se creó un esquema de base de datos usando el motor MySQL. En este esquema se guardan los perfiles de usuario disponibles en el sistema junto con los perfiles de cada usuario, para esto se crearon los scripts mostrados en 5.3

7.1.2. Especificación de Casos de Uso

7.1.2.1. Alta de Usuario

Este caso especifica el alta de un usuario en el sistema. Mediante esta funcionalidad se dan de alta usuarios en el LDAP, si al momento de dar de alta el usuario se le asigna un perfil, este se actualiza en la tabla Perfiles_Usuario. El usuario ingresa a la pantalla de Alta de Usuario si tiene los permisos asociados para esta acción. En la pantalla completa los campos solicitados, en caso de error se despliega un mensaje indicando cual fue el error para que sea corregido.

Pre Condiciones

- El usuario no debe existir en el sistema con el id que se ingresa el nuevo usuario.

Usuario	Sistema
1. El usuario con permiso de alta de usuario, ingresa al sistema y ejecuta la funcionalidad de alta de Usuario	
	2. El sistema muestra el formulario con los campos a ingresar y la lista de perfiles disponible para asignar al usuario
3. El usuario ingresa los datos requeridos en el formulario y presiona el botón “Alta de Usuario”	
	4. El sistema da de alta el usuario en LDAP y si se le asignaron perfiles los agrega en la tabla Perfiles_Usuarios. El sistema notifica al usuario mediante un pop up que el usuario fue ingresado con éxito.

Cuadro 1: Flujo Principal.

Usuario	Sistema
1.1.1 El usuario no tiene permiso para dar de alta usuarios	
	1.1.2 El sistema no muestra la opción dar de alta usuario

Cuadro 2: Flujo Alternativo, Usuario no tiene permiso para dar de Alta Usuarios.

Usuario	Sistema
3.1.1 El usuario no ingresa todos los campos requeridos o ingresa datos con formato incorrecto	
	3.1.2 El sistema muestra los mensajes de error correspondientes a las validaciones realizadas.

Cuadro 3: Flujo alternativo: Usuario ingresa datos no válidos para alta de usuarios.

Usuario	Sistema
	4.1.1 El sistema no puede dar de alta el usuario en LDAP o en la Base de Datos, muestra un mensaje diciendo que el Usuario no pudo ser dado de alta con éxito

Cuadro 4: Flujo alternativo: No se puede dar de alta en LDAP o Base de Datos

7.1.2.2. Modificación de Usuario

Se permite modificar los datos de un usuario: Nombre, apellido, dirección, email, teléfono, bloqueado, agregar y quitar perfiles(ver los que faltan) Con la restricción de que el usuario u1 que realiza la modificación tiene que tener el permiso para modificar un usuario u2 si el usuario u2 tiene un perfil en la ubicación que u1 está logueado. Si el usuario u1 tiene Rol admin, el usuario puede modificar todos los atributos del usuario u2, aunque u1 y u2 no se encuentren en la misma ubicación. Si se agregan/quitan perfiles a u2, serán todos en la ubicación en la que u1 se encuentra logueado. Solo se muestran los perfiles de u2 para la ubicación en la que u1 está logueado, sin importar si este es admin o no. Para cada perfil p del usuario u2, se determina si el perfil p se tiene que agregar/quitar y para este se busca la lista de roles del perfil p, y se agrega/quita cada rol r perteneciente al perfil p, luego se actualiza la lista de perfiles asociada al usuario u2.

Precondiciones

- El usuario tiene permiso para la funcionalidad modificar usuario.

Postcondiciones

- Se modificó el usuario en LDAP y en la base de datos.

Usuario	Sistema
1. El usuario con permiso de modificar usuario, ingresa al sistema y ejecuta la funcionalidad de modificación de usuario.	
	2. El sistema muestra la pantalla de modificación de usuario, con una búsqueda por usuario (si no se escribe nada y se da buscar, carga todos los usuarios del sistema)
3. El usuario elige un usuario a modificar	
	4. El sistema muestra los datos del usuario seleccionado.
5. El usuario modifica los datos del usuario y da Guardar cambios.	
	6. El sistema actualiza los datos del usuario en LDAP y la base de datos y muestra un mensaje de éxito.

Cuadro 5: Flujo Principal.

Usuario	Sistema
5.1.1 El usuario no ingresa datos válidos en los campos a modificar	
	5.1.2 El sistema muestra un mensaje de error por cada error introducido.

Cuadro 6: Flujo alternativo: Usuario ingresa datos no válidos.

7.1.2.3. Resetear contraseña a Usuario

Esta funcionalidad permite a un usuario u1, resetear la contraseña de un usuario u2. El usuario u1 tiene que tener permiso para ejecutar esta operación y estar logueado en la misma ubicación que u2.

Precondiciones

- El usuario tiene permiso para la funcionalidad Modificar Usuario y está logueado en la misma ubicación que el usuario al cual se le va a resetear la contraseña. La funcionalidad resetear contraseña está disponible en el menú de los datos a modificar del usuario.

Postcondiciones

- Se reseteó la contraseña del usuario u2.

Usuario	Sistema
1. El usuario con permiso de modificar usuario, ingresa al sistema y ejecuta la funcionalidad de resetear contraseña	
	2. El sistema muestra la pantalla de modificar usuario. Una tabla de usuarios y una búsqueda por identificador de usuario, si se presiona buscar, sin especificar nada en el campo texto, el sistema devuelve todos los usuarios.
3. El usuario elige un usuario y da click en resetear contraseña.	
	4. El sistema asigna una contraseña por defecto al usuario definida en un archivo de properties.

Cuadro 7: Flujo Principal.

Usuario	Sistema
	4.1.3 El sistema no pudo resetear la contraseña al usuario, se muestra un mensaje de error.

Cuadro 8: Flujo alternativo: Sistema no puede resetear la contraseña al usuario.

7.1.2.4. Asignar\Desasignar Rol a Usuario

Esta funcionalidad permite a un usuario u1, asignar un usuario u2 a un rol r1, en la ubicación que se encuentra u1, si u2 no se encuentra en la misma ubicación que u1, no se puede ejecutar la operación aunque u1 tenga permisos para ejecutar la operación “Asignar Rol a Usuario” en la ubicación u1. El sistema muestra la pantalla para asignar roles, donde en una tabla de usuarios se elige el usuario para asignar el rol, luego se muestra la lista de roles que posee el usuario y la lista de roles disponibles a ser asignados/desasignados, según se quiten de los que ya tiene o se agreguen nuevos.

Esta funcionalidad está disponible en el sistema aunque se desaconseja su utilización, para esto se recomienda crear un perfil y asignar el rol al perfil, luego usar la funcionalidad “Asignar perfil a usuario”. Así como también se recomienda fuertemente que la asignación de roles a perfiles sea disjunta. La asignación/desasignación de roles sin perfiles asociados genera el problema de inconsistencia cuando un usuario tiene un rol asignado manualmente, luego si se le quita al usuario un perfil que tiene ese rol, se quitará el rol asociado al usuario que fue agregado por fuera de los perfiles. Igualmente ocurre una situación indeseada cuando un usuario tiene un rol (r1) asignado usando un perfil y (luego se le asigna por fuera ese mismo rol o no) se quita el rol (r1), el usuario quedará inconsistente en los roles del perfil, ya que no va a tener todos los roles del perfil

que tiene asignado, no siendo esta la situación especificada en los requerimientos. Si en vez de esto, se agrega el rol a un perfil y se lo asocia al usuario, luego cuando se quite un perfil que contiene este se evita toda inconsistencia.

Precondiciones

- El usuario tiene permiso para la funcionalidad Asignar Rol a Usuario.

Postcondiciones

- Se asignó el rol r1 al usuario u2.

Usuario	Sistema
1. El usuario con permiso de asignar rol, ingresa al sistema y ejecuta la funcionalidad de asignación de rol.	
	2. El sistema muestra la pantalla de asignar/desasignar rol con la lista de usuarios disponibles.
3. El usuario elige un usuario	
	4. El sistema muestra para el usuario elegido la lista de roles asignados y los roles disponibles a ser asignados. El usuario elige un rol a asignar/desasignar al usuario y da guardar cambios.
5. El usuario elige un rol a asignar/desasignar al usuario y da guardar cambios.	
	6. El sistema asigna/desasigna el rol al usuario y muestra un mensaje de éxito. .

Cuadro 9: Flujo Principal.

Usuario	Sistema
	6.1.1 El sistema no pudo asignar/desasignar el rol al usuario, se muestra un mensaje de error.

Cuadro 10: Flujo alternativo: Sistema no puede asignar/desasignar el rol al usuario.

7.1.2.5. Alta de Rol

Este caso especifica el alta de un rol en el sistema. Mediante esta funcionalidad se dan de alta roles en el LDAP. El usuario ingresa a la pantalla administración de roles, en esta se muestra un panel para dar de alta Roles y la lista de roles

existente en el sistema, si se elige un rol de la lista de roles se muestra una ventana con la opción de eliminar el rol.

Precondiciones

- El nuevo rol (r) a ser dado de alta no debe existir en el sistema.

Postcondiciones

- Se da de alta el rol r en el sistema.

Usuario	Sistema
1. El usuario con permiso para dar de alta roles, ingresa al sistema y ejecuta la funcionalidad de alta de rol.	
	2. El sistema muestra el formulario con los campos a ingresar para dar de alta el rol.
3. El usuario ingresa los datos requeridos en el formulario y presiona el botón “agregar”	
	4. El sistema da de alta el rol en LDAP y muestra un mensaje de éxito, lo agrega a la tabla que muestra la lista de todos los roles del sistema.

Cuadro 11: Flujo Principal.

Usuario	Sistema
3.1.1 El usuario ingresa un rol ya existente.	
	3.1.2 El sistema muestra un mensaje de error.

Cuadro 12: Flujo alternativo: El rol ya existe en el sistema

7.1.2.6. Baja de Rol

Este caso especifica la baja de un rol en el sistema. Mediante esta funcionalidad se dan de baja roles en el LDAP. El usuario ingresa a la pantalla administración de roles, en esta se muestra un panel para dar de alta Roles y la lista de roles existente en el sistema, si se elige un rol de la lista de roles se muestra una ventana con la opción de eliminar el rol.

Precondiciones

- El nuevo rol (r) a ser dado de baja debe existir en el sistema.

Postcondiciones

- Se da de baja el rol r en el sistema.

Usuario	Sistema
1. El usuario con permiso para dar de baja roles, ingresa al sistema y ejecuta la funcionalidad de baja de rol.	
	2. El sistema muestra una tabla con la lista de todos los roles del sistema
3. El usuario elige un rol.	
	4. El sistema muestra en una panel el rol seleccionado con el botón eliminar.
5. El usuario da eliminar	
	6. Se elimina el rol del sistema y se muestra un mensaje de éxito.

Cuadro 13: Flujo Principal.

Usuario	Sistema
	6.1.1 El sistema muestra un mensaje de error.

Cuadro 14: Flujo alternativo: El rol no pudo ser eliminado

7.1.2.7. Asignar\Desasignar permiso a Rol.

Esta funcionalidad permite asignar uno o varios permisos (pi) a un rol.

Esta funcionalidad está disponible en el sistema para asignar/desasignar permisos a roles, primero se selecciona un rol y luego se despliega un panel con la lista de permisos existentes en el sistema, y en otro la lista de permisos que ya posee el rol seleccionado. Seleccionando uno o varios permisos del panel de la izquierda se agregan los permisos, y seleccionando los de la derecha se quitan. Al guardar los cambios se despliega un mensaje indicando si la operación tuvo éxito o hubo un error, en caso de éxito los permisos del rol serán los del panel de la derecha.

Precondiciones

- El usuario tiene permiso para la funcionalidad Asignar/Desasignar permiso a Rol.

Postcondiciones

- Se asignaron/desasignaron los permisos al rol.

Usuario	Sistema
1. El usuario con permiso de asignar\desasignar permiso a rol, ingresa al sistema y ejecuta la funcionalidad de asignación de permisos a rol.	
	2. El sistema muestra la pantalla de asignar\desasignar permisos a rol con la lista de roles del sistema.
3. El usuario elige un rol.	
	4. El sistema muestra para el rol elegido la lista de permisos asignados y los permisos disponibles a ser asignados.
5. El usuario elige una lista de permisos a asignar/desasignar al rol y da guardar cambios.	
	6. El sistema asigna/desasigna el/los permisos al rol y muestra un mensaje de éxito.

Cuadro 15: Flujo Principal.

2.7.5.1 El sistema no puede asignar\desasignar permisos al rol.

Usuario	Sistema
	6.1.1 El sistema no pudo asignar\desasignar el/los permisos al rol, se muestra un mensaje de error.

Cuadro 16: Flujo alternativo: El sistema no puede asignar\desasignar permisos al rol.

7.1.2.8. Alta de Permiso.

Este caso especifica el alta de un permiso en el sistema. Mediante esta funcionalidad se dan de alta permisos en el LDAP. El usuario ingresa a la pantalla administración de permisos, en esta se muestra un panel para dar de alta permisos y la lista de permisos existente en el sistema. Se ingresa el nombre de la operación (nom_op) y se elige un objeto permiso (per) a asociar, luego se da click en Agregar Permiso y se ingresa al sistema el permiso (nom_op, per).

Precondiciones

- El nuevo permiso a ser dado de alta no debe existir en el sistema.

Postcondiciones

- Se da de alta el permiso en el sistema.

Usuario	Sistema
1. El usuario con permiso para administrar permisos, ingresa al sistema y ejecuta la funcionalidad alta permiso	
	2. El sistema muestra el formulario con los campos a ingresar para dar de alta permisos.
3. El usuario ingresa el nombre de operación del permiso y elige el objeto permiso a asociar y hace click en Agregar Permiso.	
	4. El sistema da de alta el permiso en LDAP y muestra un mensaje de éxito.

Cuadro 17: Flujo Principal.

Usuario	Sistema
1.1.1 El usuario ingresa una asociación nombre de operación de permiso y objeto ya existente en el sistema.	
	1.1.2 El sistema muestra un mensaje de error.

Cuadro 18: Flujo alternativo: Asociacion Permiso Objeto ya existente.

7.1.2.9. Baja de Permiso

Este caso especifica la baja de un permiso en el sistema. Mediante esta funcionalidad se dan de baja permisos en el LDAP. El usuario ingresa a la pantalla administración de permisos, en esta se muestra un panel para dar de baja permisos y la lista de permisos existente en el sistema. Se selecciona un permiso y se muestra el panel para eliminar el permiso seleccionado, luego al dar Eliminar se elimina el permiso de LDAP.

Postcondiciones

- Se da de baja el permiso en el sistema.

Usuario	Sistema
1. El usuario con permiso para dar de baja permisos, ingresa al sistema y ejecuta la funcionalidad de baja de permiso.	
	2. El sistema muestra una tabla con la lista de todos los permisos del sistema.
3. El usuario elige un permiso.	
	4. El sistema muestra en un panel el permiso seleccionado con el botón eliminar.
5. El usuario da eliminar	
	6. Se elimina el permiso del sistema y se muestra un mensaje de éxito.

Cuadro 19: Flujo Principal.

Usuario	Sistema
	6.1.1 El permiso no pudo ser eliminado. El sistema muestra un mensaje de error.

Cuadro 20: Flujo alternativo: El permiso no pudo ser eliminado

8. Conclusiones

En este proyecto de grado para la carrera de Ingeniería en Computación de la Universidad de la República (UdelaR) se logró desarrollar un sistema informático de autenticación y autorización basado en herramientas de código abierto.

Tras un estudio teórico se realizó el análisis del estado del arte en herramientas que se encuentran dentro de la categoría Identity Access Manager (IAM). Se concluyó que Apache Fortress, que implementa el modelo RBAC, es la herramienta más adecuada sobre la cual se basó el desarrollo de este proyecto porque brinda una solución más estable y de mejor calidad que interactuando directamente con el servidor OpenLDAP sin ninguna herramienta IAM. Fortress cuenta con el respaldo del proyecto Apache, es una herramienta relativamente nueva y no cuenta con gran cantidad de información en Internet, durante el desarrollo del proyecto se pudo observar que la curva de aprendizaje es bastante alta, sin embargo, una vez salteado ese obstáculo probó ser una herramienta confiable y amigable.

Al haber seleccionado Fortress que es de código abierto es posible realizarle cambios de manera de poder cumplir con requerimientos que están fuera del estándar RBAC y de las posibilidades de Fortress. En este trabajo se optó por desarrollar una aplicación propia basada en las interfaces programables Java que brinda Fortress para poder tener una mayor flexibilidad a la hora de cumplir con todos los requerimientos del proyecto. Esta forma de trabajo resulta más fácil y mejora la claridad y calidad del proyecto, aunque se genera una arquitectura más heterogénea con base de datos auxiliares y comunicaciones directas con el servidor OpenLDAP. Este servicio de directorio es de código abierto y está ampliamente utilizado en la industria, desarrollado y mantenido por una gran comunidad en Internet. Demostró ser una herramienta robusta y confiable.

Este diseño de la arquitectura y del sistema permitió implementar uno de los principales requerimientos como es el de restringir los roles de un usuario según la ubicación donde está ingresando al sistema. Este caso de uso está fuera del estándar de RBAC y de Fortress por lo que se implementaron nuevas entidades en el sistema, administradas por la aplicación y almacenadas en la base de datos MySQL fuera del OpenLDAP y de Fortress. Si bien esto, complicó el desarrollo y mantenimiento del sistema, gracias al diseño realizado no resultó de gran dificultad su implementación y contribuyó en gran medida al potencial del sistema.

9. Trabajos futuros

En este capítulo se presentan posibles líneas de trabajos a futuro que pueden resultar interesantes pero que no formaron parte del alcance del proyecto.

Restricción basada en la ubicación Existe un estándar mejorado de RBAC que corresponde al ANSI RBAC INCITS 494-2012 el cual contempla y permite este tipo de restricciones sobre atributos arbitrarios. Específicamente menciona la restricción basada en ubicación [34]. Por lo que es posible que Fortress implemente INCITS 494 en algún momento futuro.

Por lo tanto se debe hacer un seguimiento al proyecto Apache Fortress atento a las nuevas versiones de este sistema. También, por tratarse de un proyecto de código abierto, es posible integrarse al equipo de desarrollo e ir construyendo las mejoras propuestas por la actualización del estándar.

Aplicación web La aplicación web desarrollada para este proyecto cumple con las necesidades básicas del mismo, y permite la validación de los casos de uso. Sin embargo, es posible y deseable mejorar su estética, usabilidad, eficiencia, etc. La capa de presentación ha sido cambiada en varias ocasiones durante el proyecto gracias a la separación entre las capas de presentación y de negocio.

Integración con base de datos Si bien muchas de las operaciones necesarias para administrar la base de datos MySQL han sido implementadas, dentro del alcance del proyecto no se encontraba dicha tarea. Este manejo era realizado directamente sobre la base de datos, por lo que una mejora importante sería crear pantallas o procesos capaces de administrar completamente las tablas de la base de datos.

Integración con sistemas externos Dentro del proyecto se validaron las interacciones con Samba, Sudo y correo electrónico. El módulo para redes inalámbricas (Radius) está implementado sin embargo no pudo probarse. El sistema está previsto para que este tipo de interacciones sea modular, de manera que sean programables nuevos módulos para otro tipo de sistemas con los que se tengan que interactuar.

Referencias

[1] D. d. C. UBA. Control de acceso. [Online]. Available: www-2.dc.uba.ar/materias/seginf/material/Clase02-Unidad2_vf.pdf

[2] Rick Kuhn, *Role Based Access Control*, American National Standard for Information Technology Std. INCITS 359, April 2003. [Online]. Available: http://csrc.nist.gov/rbac/rbac-std-ncits.pdf

[3] O. Foundation. OpenLDAP. [Online]. Available: http://www.openldap.org/

[4] C. IIS. Modelo de procesos. [Online]. Available: https://eva.fing.edu.uy/course/view.php?id=613

[5] Virginia Mosqueda, Felix Poggioli. (2004, Marzo) Implementacion de un componente de un sistema de seguridad para el servicio de publicaciones en una intranet academica. Universidad Catolica Andres Bello. [Online]. Available: http://biblioteca2.ucab.edu.ve/anexos/biblioteca/marc/texto/AAQ0197.pdf

[6] I. o. N. C. Department of Computer Science. Bell LaPadula Model. [Online]. Available: http://www.cs.unc.edu/~dewan/242/f96/notes/prot/node13.html

[7] ANSI INCITS, *An Introduction to Role-Based Access Control (ANSI INCITS 359-2004)*, Std. 359, 2004.

[8] *Lightweight Directory Access Protocol (LDAP): The Protocol*, Novell, Inc Std. 4511, 2006.

[9] D. W. Chadwick, *Understanding X.500 - The Directory*, Std., 1996. [Online]. Available: http://sec.cs.kent.ac.uk/x500book/

[10] M. Wahl, "Lightweight Directory Access Protocol (v3)," Internet Requests for Comments, RFC Editor, RFC 2251, December 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2251.txt

[11] G. Good, "LDAP Data Interchange Format (LDIF)," Internet Requests for Comments, RFC Editor, RFC 2849, June 2000. [Online]. Available: https://tools.ietf.org/html/rfc2849

[12] T. Kukuk. Network Information Service. [Online]. Available: http://www.linux-nis.org/

[13] R. Srinivasan, "Remote Procedure Call," Internet Requests for Comments, RFC Editor, RFC 1831, August 1995. [Online]. Available: https://tools.ietf.org/html/rfc1831

[14] Novell. Netware Directory Services. [Online]. Available: https://support.novell.com/techcenter/articles/ana19930402.html

[15] TechTarget. iam. [Online]. Available: http://searchsecurity.techtarget.com/definition/identity-access-management-IAM-system

[16] IBM. IBM identity manager. [Online]. Available: http://www-03.ibm.com/software/products/es/identity-manager

[17] A. S. Foundation. Apache Frotress OSS IAM Java SDK. [Online]. Available: http://directory.apache.org/fortress/

[18] S. Team. SAMBA. [Online]. Available: http://www.samba.org/

[19] GNA. SAMBA-TOOLS. [Online]. Available: https://gna.org/projects/smbldap-tools/

[20] sudo. SUDO man. [Online]. Available: http://www.sudo.ws/man/1.8.12/sudo.man.html

[21] Z. Inc. Zimbra. [Online]. Available: https://www.zimbra.com/

[22] T. P. H. Page. postfix. [Online]. Available: http://www.postfix.org/

[23] R. J. L. Alan C. O'Connor, *Economic Analysis of Role-Based Access Control*, RTI International Std., 2010.

[24] S. S. Rosanna Lee, *JNDI API Tutorial and Reference: Building Directory-Enabled Java Applications*, Std., 2000.

[25] NIST. The NIST Model for Role-Based Access Control: Towards a Unified Standard.

[26] S. Z. Beguelin, *Especificación formal del modelo de seguridad de MIDP 2.0 en el Cálculo de Construcciones Inductivas.*, Universidad Nacional de Rosario Std., 2006.

[27] I. Red Hat. jboss. [Online]. Available: http://www.jboss.org/

[28] ——. richfaces. [Online]. Available: http://richfaces.jboss.org/

[29] O. Corporation. mysql. [Online]. Available: https://www.mysql.com/

[30] T. A. S. Foundation. svn. [Online]. Available: https://subversion.apache.org/

[31] D. B. T. H. Benito van der Zander, Jan Sundermeyer. textstudio. [Online]. Available: http://www.texstudio.org/

[32] T. Kukuk. Pluggable Authentication Modules for Linux. [Online]. Available: http://www.linux-pam.org/

[33] ——. Name Service Switch. [Online]. Available: http://man7.org/linux/man-pages/man5/nsswitch.conf.5.html

[34] *Role Based Access Control Policy Enhanced (ANSI INCITS 494-2012)*, ANSI INCITS Std. 494, 2012.

Anexos

A. Instalación de la aplicación

En este anexo se especifica el proceso de instalación de la aplicación desarrollada.

A.1. Pre-requisitos para la instalación

En esta sección se detallan los diferentes aplicativos con los que debemos contar.

- Servidor de aplicaciones *Jboss 6* .
- Servidor de Base de datos *MySQL 5.6*.

A.2. Instalación de Fortress

A continuación se describe el procedimiento para realizar la instalación del software fortress. La misma incluye también la instalación del servidor openldap y de los paquetes a utilizar.

A.2.1. Pre-requisitos

Como requisitos de la instalación del software tenemos:

- Java Version 7 o superior
- Maquina con 1GB, o mas, de memoria RAM

A.2.2. Procedimiento de instalación

1. Descomprimir el paquete de instalación.
   - unzip fortressBuilder-Redhat-Silver-x86-64-1.0-RC37.zip
2. Configurar permisos
   - *chmod a+x -Rf **
3. Configurar el java a utilizar
   - Editar el archivo *b.sh* y hacer apuntar la variable JAVA_HOME al directorio donde esta instalado java, por ejemplo *JAVA_HOME=/usr/java/jdk1.7.0_60*
4. Compilar el paquete base
   - Ejecutar el comando *./b.sh*
5. Configurar nuestro árbol de ldap y el servicio de ldap. Editamos el archivo *build.properties*

- Cambiar el valor de la variable enable.mgr.impl.rest a true
- Configurar cual sera nuestro servidor de ldap
  - ldap.host=IP_SERVIDOR
- Modificar la base de ldap,
  - suffix.name=asse
  - suffix.dc=uy

6. Por ultimo se instala y cargar ldap
   - ejecutar el comando *./b.sh init-slapd* Luego de finalizado el proceso tendremos instalado la versión de openldap distribuida por fortress. El directorio de instalación es /opt/symas.

Luego de la instalación copiar el archivo ./config/fortress.properties a un lugar que luego sea accesible.
Para levantar el servicio de openldap se debe ejecutar ./b.sh start-slapd.

A.3. Instalación y Configuración de la aplicación

A continuación se detallan los pasos a seguir.

A.3.1. Configuración openldap

Por defecto, el servidor openldap que se instalo en los pasos anteriores, no viene configurado para soportar objetos samba o sudo. A continuación se detallan los pasos a seguir para dejar operativas estas funcionalidades. Los esquemas que se mencionan están incluidos en el paquete de instalación.

- Actualizar el esquema rfc230bis para que soporte el objectClass namedObject. Puede utilizarse el que se incluye en el paquete de instalación, o descargarse el schema o copiarlo de una instalación de openldap, por defecto se encuentra en la ubicación /etc/openldap/schema. El directorio donde se localizan los esquemas en nuestra instalación es /opt/symas/etc/openldap/schema. Incluir dicho esquema en la configuración de openldap.
- Por defecto se incluye el esquema nis, en nuestro caso y para no generar conflictos debemos quitar dicha inclusión.
- Incluir los esquemas correspondientes de sudo (sudo.schema) y samba (samba3.schema).
- Reiniciar el servidor ldap, en el directorio donde se instalo fortress ejecutar *./b.sh stop-slapd; ./b.sh start-slapd*

Ahora estamos en condiciones de cargar los objetos para cubrir las funcionalidades necesarias para los servicios de samba, sudo y administración de la aplicación, para esto se incluyen en el paquete de instalación los archivos ldif necesarios. A continuación se listan dichos archivos, recordar adaptarlos a su entorno.

- samba.ldif

- sudo.ldif
- groups.ldif
- adminUser.ldif
- ldif_completo.ldif, este archivo incluye toda la base necesaria para poder manejar fortress y la aplicación, es destinado a aquellos usuarios que quieran realizar su propia instalación de ldap y cargar los datos manualmente.

A.3.2. Carga de datos

En el paquete de instalación se incluyen los datos iniciales de la base de datos, los mismo se encuentran dentro del directorio *carga de datos*. Realizar la siguiente ejecución:

- mysql –host=localhost –user=root –password –port=3306 –default-character-set=utf8 –comments <ïnitBD.sql"

A.3.3. Configuracion del war a utilizar

Cada organización tendrá su propia estructura en el servidor openldap, el war distribuido en este paquete de instalación hace referencia como base de ldap a dc=asse,dc=uy. En este punto necesitaremos contar con dos archivos para adaptar la aplicación a su entorno, *extras.properties* y *fortress.properties*, en el paquete de instalación se incluyen dichos archivos con las configuraciones por defecto. Recordar el archivo fortress.properties que se menciono cuando termino la instalación de fortress, el mismo debe sustiruir al que se incluye en el paquete, y por otra parte se debe modificar el archivo extras.properties según sus necesidades. Luego de realizadas las configuraciones pertinentes ejecutar el script *config.sh* para linux, y el ejecutable *config.bat* para windows.

A.3.4. Acceso a la base de datos desde Jboss

Debemos configurar nuestro servidor de Jboss para que pueda acceder a la base de datos que estamos utilizando. En el paquete de instalación se incluye la configuración básica, *mysql-ds.xml*, este archivo debe ser copiado dentro del directorio *deploy* del servidor de jboss que estemos utilizando, en nuestro caso la ubicación del directorio es *server/default/deploy*. Esto puede variar dependiendo de su instalación. Se debe editar dicho archivo para que aplique al entorno actual.

A.3.5. Despliegue de la aplicación en Jboss

Hay varias formas de desplegar el software en Jboss, en este caso copiaremos el war generado en el punto 4.3 a la carpeta server/default/deploy, con esto el propio Jboss sera el encargado de desplegar el war. Por ultimo para acceder a la aplicación la url a utilzar es *http://server_jboss:8080/PrototipoFortress/login.xhtml*, con el usuario *admin* y contraseña *admin*.

B. Prototipo Java para LDAP

En este anexo se describe el prototipo alternativo que se utilizó para probar la conexión con OpenLDAP a partir de las librerías de Java que brinda Oracle del paquete Java Naming and Directory Interface (JNDI).

La configuración de la variable de ambiente que prepara el contexto para la conexión con el LDAP es:

```
private Hashtable<String, String> env = new Hashtable<String, String>();

env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://localhost:389");
env.put(Context.SECURITY_PRINCIPAL, "cn=Manager,dc=openldap,dc=org");
env.put(Context.SECURITY_CREDENTIALS, "secret");
```

Las operaciones implementadas para el prototipo fueron: textttinsert, edit, delete y search. Se eligieron las operaciones clásicas para un ABM de un usuario en el LDAP más la operación para realizar la búsqueda del mismo.

Para la prueba creamos una entidad textttPersona que contiene atributos simulando un usuario real del sistema. A continuación transcribimos los atributos de la clase:

```
public class Persona {

private String nombre;
private String direccion;
private String password;

....

}
```

Una vez definida la clase Java del elemento que deseamos manejar en el OpenLDAP codificamos las operaciones antes mencionadas de la siguiente manera:

```
private boolean insert(Persona person) {
    try {

        DirContext dctx = new InitialDirContext(env);
        Attributes matchAttrs = new BasicAttributes(true);
        matchAttrs.put(new BasicAttribute("uid", person.getNombre()));
        matchAttrs.put(new BasicAttribute("cn", person.getNombre()));
        matchAttrs.put(new BasicAttribute("street", person.getDireccion()));
        matchAttrs.put(new BasicAttribute("sn", person.getNombre()));
```

```
        matchAttrs.put(new BasicAttribute("userpassword", encryptLdapPassword("SH
        matchAttrs.put(new BasicAttribute("objectclass", "top"));
        matchAttrs.put(new BasicAttribute("objectclass", "person"));
        matchAttrs.put(new BasicAttribute("objectclass", "organizationalPerson"))
        matchAttrs.put(new BasicAttribute("objectclass", "inetOrgPerson"));
        String name = "uid=" + person.getNombre() + ",ou=People,dc=openldap,dc=or
        InitialDirContext iniDirContext = (InitialDirContext) dctx;
        iniDirContext.bind(name, dctx, matchAttrs);

        System.out.println("Se ha insertado "+person.getNombre());
        return true;
    } catch (Exception e) {
        System.out.println(e);
        return false;
    }
}

private boolean edit(Persona person) {
    try {

        DirContext ctx = new InitialDirContext(env);
        ModificationItem[] mods = new ModificationItem[2];
        Attribute mod0 = new BasicAttribute("street", person.getDireccion());
        Attribute mod1 = new BasicAttribute("userpassword", encryptLdapPassword("
        mods[0] = new ModificationItem(DirContext.REPLACE_ATTRIBUTE, mod0);
        mods[1] = new ModificationItem(DirContext.REPLACE_ATTRIBUTE, mod1);

        ctx.modifyAttributes("uid=" + person.getNombre() + ",ou=People,dc=openlda

        System.out.println("success editing "+person.getNombre());
        return true;
    } catch (Exception e) {
        System.out.println(e);
        return false;
    }
}

private boolean delete(Persona person) {
    try {

        DirContext ctx = new InitialDirContext(env);
        ctx.destroySubcontext("uid=" + person.getNombre() + ",ou=People,dc=openld

        System.out.println("success deleting "+person.getNombre());
        return true;
    } catch (Exception e) {
        System.out.println(e);
        return false;
    }
```

```
}

private boolean search(Persona person) {
    try {

          DirContext ctx = new InitialDirContext(env);
          String base = "dc=openldap,dc=org";

          SearchControls sc = new SearchControls();
          sc.setSearchScope(SearchControls.SUBTREE_SCOPE);

          String filter = "(&(objectclass=person)(uid="+person.getNombre()+"))";

          NamingEnumeration results = ctx.search(base, filter, sc);


          while (results.hasMore()) {
                SearchResult sr = (SearchResult) results.next();
                Attributes attrs = sr.getAttributes();

                Attribute attr = attrs.get("uid");
                if(attr != null)
                      System.out.println("record found "+attr.get());
          }

          ctx.close();
          return true;

          } catch (Exception e) {
                System.out.println(e);
                return false;
          }
}
```

Definidas las operaciones se genera un textttMain para probar cada una de las funciones, a continuación describimos un ejemplo de código para ejecutar:

```
Persona person = new Persona();
person.setDireccion("MiCalle 1695/1101");
person.setNombre("Juan Perez");
person.setPassword("password123");

// insert
main.insert(person);

// edit
main.edit(person);

// select
main.search(person);
```

```
// delete
main.delete(person);
```

C. Configuraciones

En este anexo se encuentra las configuraciones o contenidos de archivos relevantes.

C.1. LDAP

Se realizaron configuraciones tanto a nivel de servidores como de clientes.

C.1.1. Configuraciones de Cliente

Archivo /etc/nssswitch.conf

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Legal entries are:
#
#       compat                  Use compatibility setup
#       nisplus                 Use NIS+ (NIS version 3)
#       nis                     Use NIS (NIS version 2), also called YP
#       dns                     Use DNS (Domain Name Service)
#       files                   Use the local files
#       [NOTFOUND=return]       Stop searching if not found so far
#
# For more information, please read the nsswitch.conf.5 manual page.
#

# passwd: files nis
# shadow: files nis
# group:  files nis

passwd: compat ldap
group:  compat ldap

hosts:          files dns
networks:       files dns

services:       files
```

```
protocols:      files
rpc:            files
ethers:         files
netmasks:       files
netgroup:       files nis
publickey:      files

bootparams:     files
automount:      files nis
aliases:        files
```

Archivo /etc/ldap.conf

```
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#

# Your LDAP server. Must be resolvable without using LDAP.
# Multiple hosts may be specified, each separated by a
# space. How long nss_ldap takes to failover depends on
# whether your LDAP client library supports configurable
# network or connect timeouts (see bind_timelimit).
host    127.0.0.1

# The distinguished name of the search base.
base    dc=asse,dc=uy

# Another way to specify your LDAP server is to provide an
# uri with the server name. This allows to use
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
#ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
binddn  cn=Administrator,dc=asse,dc=uy

# The credentials to bind with.
# Optional: default is no credential.
bindpw  password

# Must be set or sudo will ignore LDAP; may be specified multiple times.
```

```
sudoers_base   ou=SUDOers,dc=asse,dc=uy
#
# verbose sudoers matching from ldap
#sudoers_debug 2
#
# Enable support for time-based entries in sudoers.
#sudoers_timed yes


# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
#rootbinddn cn=manager,dc=example,dc=com

# The port.
# Optional: default is 389.
#port 389

# The search scope.
#scope sub
#scope one
#scope base

# Search timelimit
#timelimit 30

# Bind/connect timelimit
#bind_timelimit 30

# Reconnect policy:
#  hard_open: reconnect to DSA with exponential backoff if
#             opening connection failed
#  hard_init: reconnect to DSA with exponential backoff if
#             initializing connection failed
#  hard:      alias for hard_open
#  soft:      return immediately on server failure
bind_policy     soft

# Connection policy:
#  persist:   DSA connections are kept open (default)
#  oneshot:   DSA connections destroyed after request
#nss_connect_policy persist

# Idle timelimit; client will close connections
# (nss_ldap only) if the server has not been contacted
# for the number of seconds specified below.
#idle_timelimit 3600

# Use paged rseults
```

```
#nss_paged_results yes

# Pagesize: when paged results enable, used to set the
# pagesize to a custom value
#pagesize 1000

# Filter to AND with uid=%s
#pam_filter objectclass=account

# The user ID attribute (defaults to uid)
#pam_login_attribute uid

# Search the root DSE for the password policy (works
# with Netscape Directory Server). Make use of
# Password Policy LDAP Control (as in OpenLDAP)
pam_lookup_policy       yes

# Check the 'host' attribute for access control
# Default is no; if set to yes, and user has no
# value for the host attribute, and pam_ldap is
# configured for account management (authorization)
# then the user will not be allowed to login.
#pam_check_host_attr yes

# Check the 'authorizedService' attribute for access
# control
# Default is no; if set to yes, and the user has no
# value for the authorizedService attribute, and
# pam_ldap is configured for account management
# (authorization) then the user will not be allowed
# to login.
#pam_check_service_attr yes

# Group to enforce membership of
#pam_groupdn cn=PAM,ou=Groups,dc=example,dc=com

# Group member attribute
#pam_member_attribute uniquemember

# Specify a minium or maximum UID number allowed
#pam_min_uid 0
#pam_max_uid 0

# Template login attribute, default template user
# (can be overriden by value of former attribute
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody
```

```
# HEADS UP: the pam_crypt, pam_nds_passwd,
# and pam_ad_passwd options are no
# longer supported.
#
# Do not hash the password at all; presume
# the directory server will do it, if
# necessary. This is the default.
#pam_password clear

# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
#pam_password crypt

# Remove old password first, then update in
# cleartext. Necessary for use with Novell
# Directory Services (NDS)
#pam_password nds

# RACF is an alias for the above. For use with
# IBM RACF
#pam_password racf

# Update Active Directory password, by
# creating Unicode password and updating
# unicodePwd attribute.
#pam_password ad

# Use the OpenLDAP password change
# extended operation to update the password.
pam_password    exop

# Redirect users to a URL or somesuch on password
# changes.
#pam_password_prohibit_message Please visit http://internal to change your passwo

# Use backlinks for answering initgroups()
#nss_initgroups backlink

# returns NOTFOUND if nss_ldap's initgroups() is called
# for users specified in nss_initgroups_ignoreusers
# (comma separated)
nss_initgroups_ignoreusers      root,ldap

# Enable support for RFC2307bis (distinguished names in group
# members)
```

```
nss_schema      rfc2307bis

# RFC2307bis naming contexts
# Syntax:
# nss_base_XXX          base?scope?filter
# where scope is {base,one,sub}
# and filter is a filter to be &'d with the
# default filter.
# You can omit the suffix eg:
# nss_base_passwd       ou=People,
# to append the default base DN but this
# may incur a small performance impact.
#nss_base_passwd        ou=People,dc=example,dc=com?one
#nss_base_shadow        ou=People,dc=example,dc=com?one
#nss_base_group         ou=Group,dc=example,dc=com?one
#nss_base_hosts         ou=Hosts,dc=example,dc=com?one
#nss_base_services      ou=Services,dc=example,dc=com?one
#nss_base_networks      ou=Networks,dc=example,dc=com?one
#nss_base_protocols     ou=Protocols,dc=example,dc=com?one
#nss_base_rpc           ou=Rpc,dc=example,dc=com?one
#nss_base_ethers        ou=Ethers,dc=example,dc=com?one
#nss_base_netmasks      ou=Networks,dc=example,dc=com?ne
#nss_base_bootparams    ou=Ethers,dc=example,dc=com?one
#nss_base_aliases       ou=Aliases,dc=example,dc=com?one
#nss_base_netgroup      ou=Netgroup,dc=example,dc=com?one

# attribute/objectclass mapping
# Syntax:
#nss_map_attribute      rfc2307attribute        mapped_attribute
#nss_map_objectclass    rfc2307objectclass      mapped_objectclass

# configure --enable-nds is no longer supported.
# NDS mappings
nss_map_attribute       uniqueMember member

# Services for UNIX 3.5 mappings
#nss_map_objectclass posixAccount User
#nss_map_objectclass shadowAccount User
#nss_map_attribute uid msSFU30Name
#nss_map_attribute uniqueMember msSFU30PosixMember
#nss_map_attribute userPassword msSFU30Password
#nss_map_attribute homeDirectory msSFU30HomeDirectory
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_objectclass posixGroup Group
#pam_login_attribute msSFU30Name
#pam_filter objectclass=User
#pam_password ad

# configure --enable-mssfu-schema is no longer supported.
```

```
# Services for UNIX 2.0 mappings
#nss_map_objectclass posixAccount User
#nss_map_objectclass shadowAccount user
#nss_map_attribute uid msSFUName
#nss_map_attribute uniqueMember posixMember
#nss_map_attribute userPassword msSFUPassword
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_attribute shadowLastChange pwdLastSet
#nss_map_objectclass posixGroup Group
#nss_map_attribute cn msSFUName
#pam_login_attribute msSFUName
#pam_filter objectclass=User
#pam_password ad

# RFC 2307 (AD) mappings
#nss_map_objectclass posixAccount user
#nss_map_objectclass shadowAccount user
#nss_map_attribute uid sAMAccountName
#nss_map_attribute homeDirectory unixHomeDirectory
#nss_map_attribute shadowLastChange pwdLastSet
#nss_map_objectclass posixGroup group
#nss_map_attribute uniqueMember member
#pam_login_attribute sAMAccountName
#pam_filter objectclass=User
#pam_password ad

# configure --enable-authpassword is no longer supported
# AuthPassword mappings
#nss_map_attribute userPassword authPassword

# AIX SecureWay mappings
#nss_map_objectclass posixAccount aixAccount
#nss_base_passwd ou=aixaccount,?one
#nss_map_attribute uid userName
#nss_map_attribute gidNumber gid
#nss_map_attribute uidNumber uid
#nss_map_attribute userPassword passwordChar
#nss_map_objectclass posixGroup aixAccessGroup
#nss_base_group ou=aixgroup,?one
#nss_map_attribute cn groupName
#nss_map_attribute uniqueMember member
#pam_login_attribute userName
#pam_filter objectclass=aixAccount
#pam_password clear

# For pre-RFC2307bis automount schema
#nss_map_objectclass automountMap nisMap
#nss_map_attribute automountMapName nisMapName
#nss_map_objectclass automount nisObject
```

```
#nss_map_attribute automountKey cn
#nss_map_attribute automountInformation nisMapEntry

# Netscape SDK LDAPS
#ssl on

# Netscape SDK SSL options
#sslpath /etc/ssl/certs

# OpenLDAP SSL mechanism
# start_tls mechanism uses the normal LDAP port, LDAPS typically 636
ssl     no
ldap_version    3
pam_filter      objectClass=posixAccount
nss_base_passwd ou=people,dc=asse,dc=uy
nss_base_shadow ou=people,dc=asse,dc=uy
nss_base_group  ou=group,dc=asse,dc=uy
tls_checkpeer   no
#ssl on

# OpenLDAP SSL options
# Require and verify server certificate (yes/no)
# Default is to use libldap's default behavior, which can be configured in
# /etc/openldap/ldap.conf using the TLS_REQCERT setting.  The default for
# OpenLDAP 2.0 and earlier is "no", for 2.1 and later is "yes".
#tls_checkpeer yes

# CA certificates for server certificate verification
# At least one of these are required if tls_checkpeer is "yes"
#tls_cacertfile /etc/ssl/ca.cert
#tls_cacertdir /etc/ssl/certs

# Seed the PRNG if /dev/urandom is not provided
#tls_randfile /var/run/egd-pool

# SSL cipher suite
# See man ciphers for syntax
#tls_ciphers TLSv1

# Client certificate and key
# Use these, if your server requires client authentication.
#tls_cert
#tls_key

# Disable SASL security layers. This is needed for AD.
#sasl_secprops maxssf=0

# Override the default Kerberos ticket cache location.
#krb5_ccname FILE:/etc/.ldapcache
```