UNIVERSIDAD DE GUAYAQUIL

Facultad de Ciencias Matemáticas y Física

Carrera de Ingeniería en Sistemas Computacionales

"Servicio de Directorio Seguro"

PROYECTO DE GRADO

Previo a la Obtención del Titulo de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Autores:

Elsa del Rosario Aparicio Serrano

Wendy Sofía Espinosa Moya

Katiuska Paola Sánchez Luis

GUAYAQUIL – ECUADOR

Año: 2007

AGRADECIMIENTO

A Dios ante todo por siempre brindarme su amor.

A mis Padres, por su incondicional amor, apoyo económico, emocional y espiritual.

A mis hermanos y familia por su comprensión y cariño.

A mis amigos que son mi otra familia, con quienes he compartido alegrías y tristezas durante estos años de estudios.

A mis tutores de proyecto y coordinador de seminario.

A todas y cada una de las personas que de alguna manera han colaborado con este proyecto.

Mil gracias a todos por ayudarme en éste, el último esfuerzo de mi carrera universitaria, más el primero de los esfuerzos del resto de mi vida.

Elsa Aparicio Serrano

AGRADECIMIENTO

Mis infinitas gracias a Dios, por su cuidado amoroso durante toda mi vida.

A mis queridos padres, los cuales han sido participes activos en mi formación y en mis valores. Gracias por tener tanta confianza en mí.

A las personas que me han brindado su amistad, confianza, aprecio y apoyo. Convirtiéndose en verdaderos amigos.

A profesores y tutores del Seminario de Graduación, por su apoyo, tiempo y paciencia.

Finalmente a todas las personas que han sido participes en la realización de este proyecto.

Sofia Espinosa Moya

AGRADECIMIENTO

A Dios por siempre guiar mi camino e iluminar mi vida con sus bendiciones.

A mis Padres, por todo el amor, la paciencia y el apoyo tanto económico como emocional, ya que ellos han sido mi mayor fuente de inspiración para todo lo que hago.

A mis dos hermanas, que con sus travesuras y ocurrencias le han dado fuerza a mi vida para seguir adelante, pese a todos los inconvenientes que podamos tener.

A dos de mis Primas, por su apoyo incondicional, ya que siempre estuvieron pendientes de mi, han sido más que primas unas hermanas.

A mi familia por su infinito cariño y amor.

A mis compañer@s que a lo largo de estos años de estudio se fueron convirtiendo en verdaderos amig@s, con quienes he compartimos momentos inolvidables.

A todos mil gracias.

Paola Sánchez Luis

DEDICATORIA

Este proyecto se lo dedico a Dios, por levantarme en cada caída y amarme. A mi madre Estilita Serrano, mi padre David Aparicio; mis hermanos David, Alex y Andrea; sobrin@s David Alberto, Emily Daniela y mis dos angelitos que están por nacer. A mis primas queridas Natasha y Sandra; familia y seres queridos que ya no están entre nosotros. Para los eternos amig@s Sofía, Paola, Allisón, Jennifer, Sara, Mafer, Joe, Luis, Lenín, Francis y todos aquellos que me conocen por todo lo que juntos compartimos.

Elsa Aparicio Serrano

DEDICATORIA

Este proyecto está dedicado a mis padres Pablo Espinosa y María Moya por la fuerza y apoyo incondicional que me ha dado en mis años de carrera universitaria.

A mis tíos Carmen, Gladys y Nicolás que me han dado sus consejos en el momento apropiado.

A mis amigos que siempre están en mi corazón.

Sofía Espinosa Moya

DEDICATORIA

Este proyecto va dedicado primeramente a Dios, por ser la luz en mi camino, el que me permite día a día gozar de los bellos dones que me brinda.

A mi Padre Ubaldo Sánchez. A mi madre Margarita Luis; a mis hermanas Katherine y María José. Mis Primas Janeth y Gina; a mi abuelo que ya no nos acompaña pero se que el estaría feliz en este momento. A todos mis amigos aquellos que viven lejos, a los que están cerca pues han dejado huellas en mi corazón.

Paola Sánchez Luis

TRIBUNAL DE GRADUACIÓN

Presidente del Tribunal	Primer Vocal
Segundo Vocal	Secretario

DECLARACIÓN EXPRESA

“La autoría de la tesis de grado corresponde exclusivamente a los suscritos, perteneciendo a la Universidad de Guayaquil los derechos que generen la aplicación de la misma”.

(Reglamento de Graduación de la Carrera de Ingeniería en sistemas Computacionales, Art. 26)

_____

Elsa del Rosario Aparicio Serrano

_____

Wendy Sofía Espinosa Moya

_____

Katiuska Paola Sánchez Luis

RESUMEN

El Proyecto de Servicio de Directorio Seguro, ha sido desarrollado con la finalidad de que empresas, organizaciones, instituciones educativas como la Universidad de Guayaquil, etc. empleen una aplicación cliente para realizar consultas de sus diferentes departamentos y personal, comunicándose con un Servidor de Directorio (LDAP) que contiene dicha información.

La aplicación permite consultas rápidas y sencillas por departamentos, apellido, dirección, teléfono, etc. Posee una interfaz amigable lo que ayuda a la interacción de los usuarios con ella. El objetivo del uso de este proyecto radica en que la disponibilidad de la información ayude en la administración de la institución.

Nuestro proyecto es denominado Servicio de Directorio Seguro debido a que la aplicación se comunica con el servidor por medio del protocolo SSL, el garantiza la seguridad utilizando la encriptación en

las comunicaciones Cliente - Servidor, empleando de un certificado autofirmado en el Servidor de Directorio (LDAP).

ÍNDICE GENERAL

ABREVIATURA

ACO.-	Análisis del Comportamiento de Objetos
AEO	Análisis de la Estructura de Objetos
API.-	Interface de Aplicación del Programa
CN	Common Name
DIT	Directory Information Tree
DN.-	Nombre Distintivo
FSF.-	Free Software Foundation.
GPL.-	General Public License.
IP.-	Internet Protocol.
JNDI.-	Java Naming and Directory Interface
LAN.-	Local Area Network.
LDAP	Protocolo de Acceso Ligero a Directorios, (Lightwight Directory Access Protocol)
LDIF	Formato de Intercambio Ldap
ObjectClass	Clase de Objetos

SDS.-	Servicio de Directorio Seguro
SSL.	Nivel de Zócalo Seguro.
O	organizationName
OU	organizationalUnitName
SN	surname
TCP.-	Transport Control Protocol.

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

CAPITULO 1

SERVICIO DE DIRECTORIO SEGURO

1.1 Antecedentes

Servicio de Directorio

El Servicio de Directorio es un término, que se refiere a la información contenida, al conjunto hardware/software que gestiona

dicha información, a las aplicaciones cliente/servidor que utilizan esta información, etc. Es decir, el Servicio de Directorio es un conjunto complejo de componentes que trabajan de manera cooperativa para prestar un servicio.

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

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

Los directorios están para proporcionar una respuesta rápida a operaciones de búsqueda o consulta. Poseen la capacidad de replicar información de forma amplia, con el propósito de aumentar la disponibilidad y fiabilidad, y a la vez reducir tiempo de respuesta. Cuando se duplica la información de un directorio, pueden aceptarse

inconsistencias temporales entre la información que hay en las réplicas, siempre que finalmente exista una sincronización si se requiere de su utilización.

Hay muchas formas de proporcionar un Servicio de Directorio, existen métodos que admiten almacenar en el directorio diferentes tipos de información, establecer requisitos distintos para hacer referencias a la información, consultarla y actualizarla, la forma en que protege al directorio de accesos no autorizados.

Un Servicio de Directorio accesible por multitud de aplicaciones, se convierte en una parte vital de un sistema, al proporcionar un acceso uniforme a las personas, recursos y otros objetos del sistema, es decir, el directorio se ve como un todo uniforme, en lugar de un conjunto de partes independientes.

¿Para qué puede utilizarse el Directorio?

Una de las principales utilidades de los directorios ha sido la de buscar información, de hecho, el prototipo de directorio siempre ha sido la guía de teléfonos, en la cual los abonados se encuentran ordenados alfabéticamente.

La ventaja de los directorios electrónicos está en que permiten una escalabilidad no disponible en los directorios tradicionales, basta imaginarse el espacio necesario para almacenar la guía de usuarios de la compañía telefónica para comprender dicha escalabilidad.

Además, el hecho de ser directorios electrónicos permite acceder a la información contenida en ellos de maneras distintas a las tradicionales. Por ejemplo, se pueden realizar búsquedas por apellido, por dirección, teléfono, etc.

Muchas veces no basta con tener la información almacenada en un directorio electrónico, es muy importante que el directorio sea accesible desde todas las aplicaciones que son susceptibles de utilizarlo.

LDAP y OPENLDAP

LDAP (“Lightweight Directory Access Protocol", «Protocolo Ligero de Acceso a Directorios») es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio.

LDAP utiliza TCP/IP en lugar de los protocolos OSI. TCP/IP requiere menos recursos y está más disponible, especialmente en ordenadores de escritorio. El modelo funcional de LDAP es más simple y ha eliminado opciones raramente utilizadas en X.500. LDAP es más fácil de comprender e implementar. LDAP representa la información mediante cadenas de caracteres en lugar de complicadas estructuras.

LDAP es un tipo de base de datos, pero no es una base de datos relacional. No está diseñada para procesar cientos o miles de cambios por minuto como los sistemas relacionales, sino para realizar lecturas de datos de forma muy eficiente.

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

un servidor LDAP hace referencia a la misma entrada a la que haría referencia en otro servidor LDAP.

Tal vez la mayor ventaja es que se puede acceder al directorio LDAP desde casi cualquier plataforma de computación, desde cualquier número creciente de aplicaciones fácilmente disponibles para LDAP. Es también fácil personalizar las aplicaciones internas de alguna empresa para añadirles soporte LDAP.

A diferencia de las bases de datos relacionales, no se tiene que pagar por cada conexión de software cliente o por licencia. La mayoría de los servidores LDAP son simples de instalar, fácilmente mantenibles, y fácilmente optimizables.

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

- Es muy rápido en la lectura de registros.
- Permite replicar el servidor de forma muy sencilla y económica.
- Muchas aplicaciones de todo tipo tienen interfaces de conexión a LDAP y se pueden integrar fácilmente.
- Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.

- Usa un sistema jerárquico de almacenamiento de información.
- Permite múltiples directorios independientes.
- Funciona sobre TCP/IP y SSL.
- La mayoría de aplicaciones disponen de soporte para LDAP.
- La mayoría de servidores LDAP son fáciles de instalar, mantener y optimizar.

Tradicionalmente se han usado las estructuras de árbol para jerarquizar la información contenida en un medio. El ejemplo más claro es la estructura de carpetas (directorios) de un sistema operativo. Esta organización nos permite ordenar la información en subdirectorios que contienen información muy específica.

LDAP utiliza:

Entradas: El modelo de información de LDAP está basado en entradas. Una entrada es una colección de atributos que tienen un único y global Nombre Distintivo (DN). El DN se utiliza para referirse a una entrada sin ambigüedades. Cada atributo de una entrada posee un tipo y uno o más valores.

Atributos: Los datos del directorio se representan mediante pares de atributo y su valor. Por ejemplo el atributo commonName, o cn (nombre de pila), se usa para almacenar el nombre de una persona.

Los atributos requeridos son aquellos que deben estar presentes en las entradas que utilicen en la clase de objetos. Todas las entradas precisas de los atributos permitidos son aquellos que pueden estar presentes en las entradas que utilicen la clase de objetos.

Tipos de Atributos: Una definición de tipo de atributo especifica la sintaxis de un atributo y cómo se ordenan y comparan los atributos de ese tipo.

Los tipos de atributos en el directorio forman un árbol de clases. Por ejemplo, el tipo de atributo "commonName" es una subclase del tipo de atributo "name".

LDIF: Para importar y exportar información de directorio entre servidores de directorios basados en LDAP, o para describir una serie de cambios que han de aplicarse al directorio, se usa en general el fichero de formato conocido como LDIF (formato de intercambio de LDAP).

Objetos: En LDAP, una clase de objetos define la colección de atributos que pueden usarse para definir una entrada. El estándar LDAP proporciona estos tipos básicos para las clases de objetos: Grupos en el directorio, Unidades Organizacionales, Organizaciones, Personas.

OpenLDAP

OpenLDAP nació como la continuación de la versión 3.3 del servidor LDAP de la Universidad de Michigan cuando dejaron de desarrollarlo.

OpenLDAP es un servidor LDAP que se distribuye bajo licencia GNU (OpenSource), que permite que el software se pueda usar de forma gratuita tanto de forma educativa como profesional. Además disponemos del código fuente para poder realizar nuestras propias modificaciones.

1.2 Situación Conflicto

La Universidad de Guayaquil, cuenta con una gran cantidad de Personal Administrativo y de Servicio que labora en los diversos departamentos de la Administración Central y los pertenecientes a

cada facultad. Asimismo cuenta con un amplio personal Docente y alumn@s. Debido a esto se manifiestan situaciones de falta de localización de los miembros de la universidad impidiendo desde una administración idónea de recursos (humanos y materiales), hasta la inasistencia de convocatorias a reuniones de trabajo.

El problema nace por la carencia de un Servidor de Directorio que ofrezca un servicio que permita realizar búsqueda o consultas de miembros (Autoridades, Docentes, Alumn@s, Personal Administrativo y de Servicio) de los integrantes de la Universidad de Guayaquil.

1.3 Causas

- Falta de interés, investigación y propuesta de un proyecto de Directorio de Servidor por parte del Departamento de Computo de la Universidad de Guayaquil.

- Poca Iniciativa en el desarrollo de proyectos que satisfagan las necesidades existentes en la universidad. Estos proyectos pueden ser llevados a cabo por estudiantes de la Carrera de Ingeniería en Sistemas Computacionales de la Facultad de Ciencias Matemáticas y Físicas de la Universidad de Guayaquil.

1.4 Consecuencia

Carencia de consultas, búsquedas o localización de los miembros que pertenecen a la Universidad de Guayaquil, cuando estos son requeridos para realizar alguna actividad (capacitaciones, juntas, reuniones extraordinarias, inicio de clases etc.).

No puede realizarse una administración idónea de los recursos (miembros y objetos) que la universidad posee. Si un miembro esta activo o no, a que organización (facultad, departamento) pertenece, etc.

1.5 Misión

La misión de este proyecto es desarrollar un Servicio de Directorio que contenga información de los miembros que integran la Universidad de Guayaquil. Se podrá accesar, manipular y descartar la información del servidor realizando operaciones de ingreso, búsqueda, actualizaciones y eliminación por medio de la interacción de una aplicación CLIENTE con el Servidor con el propósito de llevar a cabo una Administración efectiva de los componentes que permitan alcanzar la optimización de actividades (reuniones, cursos, inicio de clases, etc.).

1.6 Visión

Los Usuarios (miembros de la universidad) de la Aplicación CLIENTE podrán realizar consultas al Servidor de Directorio para obtener información como direcciones, teléfonos, correo electrónico de los objetos (Unidades Académicas, Administrativas, departamentos, Docentes, Alumnos, Personal Administrativo y de Servicio) almacenados en el servidor siempre y cuando estos usuarios posean privilegios o accesos autorizados necesarios. Un usuario que sea Administrador de alguna dependencia de las diversas Facultades de la Universidad será capaz de visualizar los componentes existentes en las unidades a ellos encomendadas, resolviendo los problemas como conseguir un numero telefónico, conocer trabajadores activos, etc. De manera segura.

1.7 Objetivos

1.7.1 Objetivo General

Implementar el Servicio de Directorio Open Source LDAP (Lightwight Directory Access Protocol), para realizar consultas de nombres de los integrantes de la Universidad de Guayaquil.

1.7.2 Objetivos Específicos

- Administrar de forma centralizada la información de los miembros y objetos de la universidad (por medio de un Servidor OPENLDAP).

- Almacenar y organizar la información en estructuras de datos denominadas entradas en la ejecución del Servidor.

- Crear una aplicación CLIENTE, por medio de la cual los usuarios podrán acceder al Servicio de Directorio para realizar las operaciones con los permisos correspondientes de lectura, escritura, actualización sobre todos los atributos, crear nuevas entradas, etc.

- Establecer la conexión Cliente – Servidor que permita la interacción entre el servicio de directorio y usuarios al receptar las repuestas de las operaciones (lectura, escritura, modificaciones, eliminaciones).

- Proporcionar una respuesta rápida a operaciones de búsqueda o consulta que se realicen a través de la aplicación CLIENTE.

- Impedir que la información viaje de manera desprotegida a través de la red, cuando se produzca la comunicación entre el cliente y el servidor. Esto se logrará iniciando el servidor bajo SSL.

1.8 Alcances

1. Antecedentes, situación conflicto, causas, consecuencias, misión, visión, objetivos y Administración del Proyecto. (Capítulo I)

2. Análisis para el desarrollo de un Servidor de Directorio Seguro y Diseño de una aplicación para el uso del mismo.(Capítulo II)

3. Diseño de árbol para la base de datos que utilizaremos en el Servidor de Directorio. (Capítulo III)

4. Diseño de una Aplicación CLIENTE que interactué con el servidor de directorio para poder obtener la información requerida del mismo. (Capítulo III).

   Por medio de la aplicación se podrá:

   - Crear o Añadir los siguientes objetos (personas, unidades organizacionales, organizaciones).
   - Modificar, mover y eliminar objetos (Dominio, unidades organizativas y usuarios).
   - Realizar búsqueda de objetos (Dominio, unidades organizativas, usuarios) y visualización de estos.

5. Instalación, Configuración y Ejecución del Servidor OPENLDAP.(Capítulo IV)

6. Codificación del Diseño de la Aplicación Cliente que interactuará con el Servidor de Directorio.(Capítulo IV)

7. Utilizar protocolo SSL para mantener la seguridad en la comunicación del CLIENTE/SERVIDOR, impidiendo que la información viaje desprotegida por la red, al crear una llave y certificado para OpenLdap. (Capítulo IV)

8. Pruebas de funcionamiento, operatividad y seguridad del Servidor de Directorio con la Aplicación Cliente. (Capítulo IV)

1.9 Arquitectura General del Servicio de Directorio Seguro

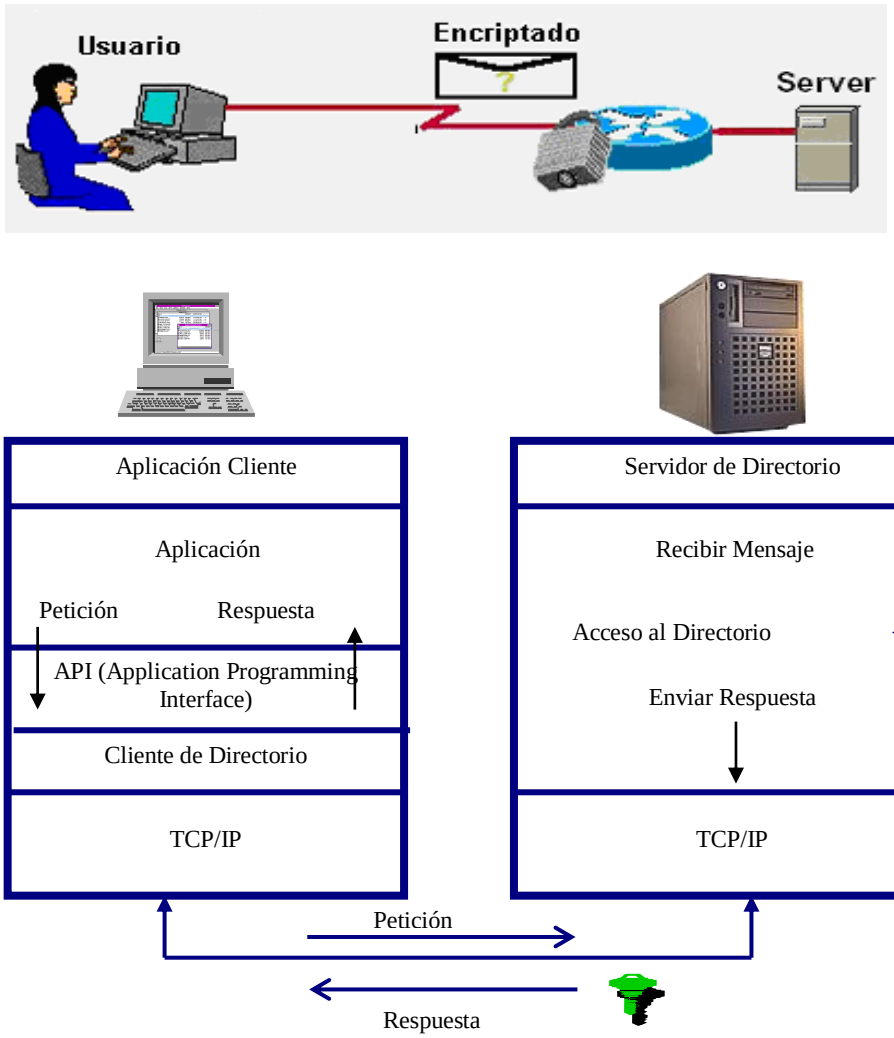


Gráfico #1
Arquitectura de Servicio de Directorio

Los servicios de directorio se implementan siguiendo el modelo cliente-servidor (Gráfico #1):

- La aplicación cliente accede al directorio mediante una función de la API (Interface de Aplicación del Programa), la cual es el conjunto de rutinas para la gestión de entrada/salida de datos.

- La función API envía un mensaje a un proceso en el servidor.

- Dicho proceso accede al directorio y devuelve el resultado de la operación.

- Ya que el Servidor OpenLDAP utiliza TCP/IP en lugar de los protocolos OSI. TCP/IP requiere menos recursos y está más disponible.

- En cuanto a las operaciones de autenticación y control, OpenLDAP establece una autenticación básica, en la que la contraseña del usuario viaja sin seguridades por la red. Para solucionar este grave problema se pueden utilizar el protocolo SSL para cifrar las conexiones extremo a extremo por medio de la utilización de un certificado autofirmado por el Servidor.

1.10 Modelo

Se ha escogido el **MODELO ESPIRAL** (gráfico#2) para el desarrollo del proyecto debido a que este posee una combinación del Modelo Lineal y el de Construcción de Prototipos, proporcionando el potencial desarrollo rápido de versiones del software.

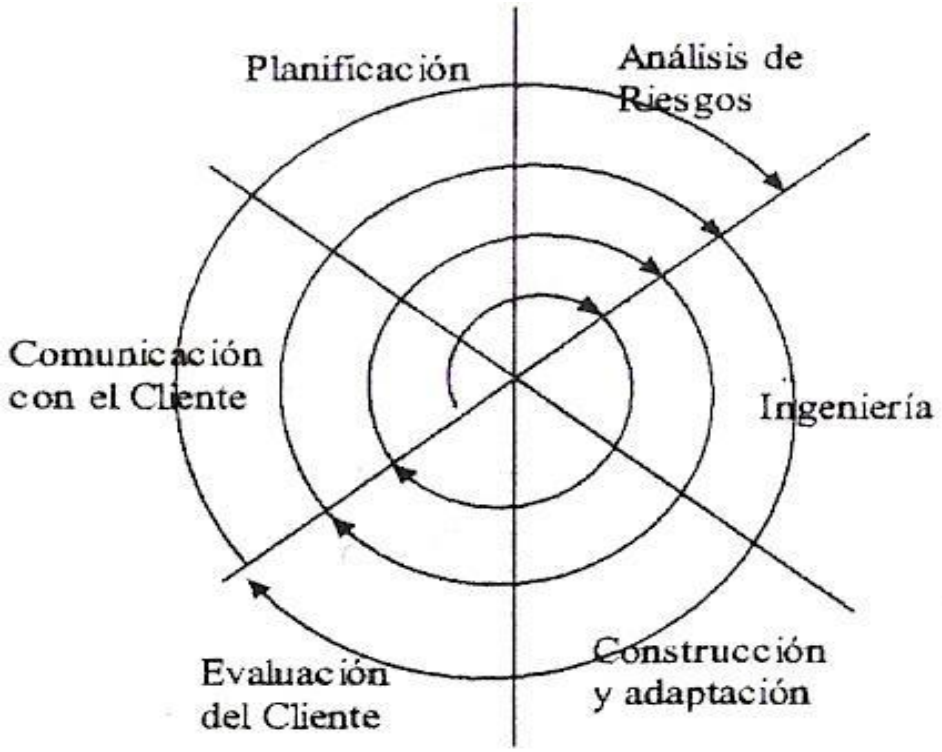


Gráfico #2
Modelo

Este modelo esta compuesto de Planificación, Análisis de Riesgos, Ingeniería, Construcción y Adaptación, Evaluación del Cliente y Comunicación con el Cliente.

1.11 Metodología

Para el proyecto “Servicio de Directorio Seguro” se ha escogido la tecnología orientada a objetos ya que el mundo orientado a objetos tiene una mayor disciplina que el de las técnicas de estructura convencional. Esto nos lleva a un mundo de clases reutilizables, donde la mayor parte del proceso de construcción de software consistirá en el ensamblaje de clases ya existentes y probadas.

Algunos de los beneficios de la metodología orientada a objetos son:

- **Reutilización:** Las clases están diseñadas para que se reutilicen en muchos sistemas.
- **El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel:** El encapsulado oculta detalles y hace que las clases complejas sean fáciles de utilizar.
- **Confiabilidad:** Es probable que el software construido a partir de clases estables ya probadas tenga menos fallas que el software elaborado a partir de cero.
- **Sencillez en la programación y en el mantenimiento:** Los programas se conjuntan a partir de piezas pequeñas. El

programador crea un método para una clase a la vez. En el mantenimiento se cambia un método de clase a la vez. Cada clase efectúa sus funciones independientemente de las demás.

- **Independencia del diseño:** Las clases están diseñadas para ser independientes del ambiente de plataformas de hardware y software.

1.12 Planeación de Proyecto y Cronograma

1.12.1 Procesos de Planeación

Permite desarrollar y mantener un esquema de trabajo para completar la necesidad para la cual el proyecto será creado

Planeación de Alcances: Desarrollar un alcance escrito como base para decisiones futuras.

Definición de Alcances: Subdividir los paquetes de entrega de un proyecto en componentes mas pequeños y manejables (subdividir en varios alcances).

Corresponde a los ALCANCES del proyecto expresados en este documento. (pág. 15)

Definición de Actividades: Identificar las actividades específicas que deben de ser ejecutadas para producir los diferentes alcances que se definieron (Tabla # 1).

Alcance	Actividades en cada alcance
1. Antecedentes, situación conflicto, causas, consecuencias, misión, visión, objetivos y Administración del Proyecto.	Recopilación de información
	Selección de la información recopilada para antecedentes
	Definición de situación conflicto, causas, consecuencias
	Administración del Proyecto (Planeación)
2. Análisis para el desarrollo de un Servidor de Directorio Seguro y Diseño de una aplicación para el uso del mismo.	**Análisis de la Estructura de Objetos (AEO)**
	Definición de tipos de Objetos
	Definición de Generalizaciones
	Diagrama de Relación de Objetos
	Análisis del Comportamiento de Objetos (ACO)
	Estados de Objetos y cambios de los mismos
	Definición de Eventos, pre estados, post estados
	Diagrama de Operaciones (Definición de Operaciones, condiciones de control, etc.)
	Diagrama de flujo de Objetos Diagrama General
3. Diseño de árbol para la base de datos que utilizaremos en el Servidor de Directorio.	**Diseño de Estructura de Base de Datos del OPENLDAP**
	Modelo de Información
	Modelo de Nombrado
	Modelo de Funcional
	Modelo de Seguridad
4. Diseño de una Aplicación CLIENTE que interactué con el servidor de directorio para poder obtener la información requerida del mismo.	**Diseño de Interfaces**
	Ingreso al Servidor por medio de la aplicación
	Interfaz de Ingreso de Objetos.
	Interfaz de Consulta de Objetos.
	Interfaz de Actualizaciones de Objetos.
	Interfaz de Eliminación de Objetos.

5. Instalación, Configuración y Ejecución del Servidor OPENLDAP	**Descargar la herramienta OPENLDAP**
	Instalación de paquetes para el Servidor
	Configuración del Servidor, Cliente y Directorio
	Ejecución del Servidor, Cliente y Directorio
	Pruebas del Servidor, Cliente y Directorio
6. Codificación del Diseño de la Aplicación Cliente que interactuará con el Servidor de Directorio.	**Codificación del Diseño propuesto**
	Ingreso al Servidor por medio de la aplicación
	Interfaz de Ingreso de Objetos.
	Interfaz de Consulta de Objetos.
	Interfaz de Actualizaciones de Objetos.
	Interfaz de Eliminación de Objetos.
7. Mantener la seguridad en la comunicación del CLIENTE/SERVIDOR. Utilizando SSL para impedir que la información viaje desprotegida por la red.	**Descarga de paquetes necesarios.**
	Instalación de OpenSSL
	Configuración de SSL
	Ejecución y Pruebas de funcionamiento
8. Implementación y Pruebas de funcionamiento, operatividad y seguridad del Servidor de Directorio con la Aplicación Cliente.	**Implementación de:**
	Servidor
	Aplicación Cliente
	Encriptación
	Elaboración de Pruebas de:
	Conexión al Servidor por medio de la aplicación
	Ingreso de Objetos del Directorio
	Consulta de Objetos del Directorio
	Actualizaciones de Objetos del Directorio
	Eliminación de Objetos del Directorio
	Encriptación para la seguridad

Tabla # 1
Definición de Actividades

Secuencia de Actividades: Identificar y documentar las dependencias entre actividades cual es primera, segunda, etc. (Tabla #2)

Estimación de la Duración de la Actividad: Estimar la cantidad de tiempo que se requiere para completar cada actividad. (Tabla #2)

Ubicación en el documento	Alcance	Actividades en cada alcance	Secuencia de la actividad	Estimación de duración de actividad
Capítulo I	1. Antecedentes, situación conflicto, causas, consecuencias, misión, visión, objetivos y Administración del Proyecto.	**Recopilación de información**	1	7 Semanas
		Selección de la información recopilada para antecedentes	2	
		Definición de situación conflicto, causas, consecuencias, Objetivos, Alcances, Arquitectura, Modelo, Metodología, etc.	3	
		Administración del Proyecto: Planeación de Actividades de alcances, Recursos, Costos, Riesgos.	4	
Capítulo II	2. Análisis para el desarrollo de un Servidor de Directorio Seguro y Diseño de una aplicación para el uso del mismo.	**Análisis de la Estructura de Objetos (AEO)**	5	2 Semanas
		Definición de tipos de Objetos	5.1	
		Definición de Generalizaciones	5.2	
		Diagrama de Relación de Objetos	5.3	
		Análisis del Comportamiento de Objetos (ACO)	6	3 Semanas
		Estados de Objetos y cambios de los mismos	6.1	
		Definición de Eventos, pre estados, post estados	6.2	
		Diagrama de Operaciones (Definición de Operaciones, condiciones de control, etc.)	6.3	
		Diagrama de flujo de Objetos Diagrama General	6.4	

Capítulo III	3. Diseño de árbol para la base de datos que utilizaremos en el Servidor de Directorio.	**Diseño de Estructura de Base de Datos del OPENLDAP**	7	3 Semana
		Modelo de Información	7.1	
		Modelo de Nombrado	7.2	
		Modelo de Funcional	7.3	
		Modelo de Seguridad	7.4	
	4. Diseño de una Aplicación CLIENTE que interactué con el servidor de directorio para poder obtener la información requerida del mismo.	**Diseño de Interfaces**	8	2 Semana
		Ingreso al Servidor por medio de la aplicación	8.1	
		Interfaz de Ingreso de Objetos.	8.2	
		Interfaz de Consulta de Objetos.	8.3	
		Interfaz de Actualizaciones de Objetos.	8.4	
		Interfaz de Eliminación de Objetos.	8.5	
Capítulo IV	5. Instalación, Configuración y Ejecución del Servidor OPENLDAP.	**Descargar la herramienta OPENLDAP**	9	1 Semana
		Instalación de paquetes para el Servidor	10	
		Configuración del Servidor y Cliente	11	
		Ejecución del Servidor y Cliente	12	
Capítulo IV	6. Codificación del Diseño de la Aplicación Cliente que interactuará con el Servidor de Directorio.	**Codificación del Diseño propuesto**	13	8 Semanas
		Ingreso al Servidor por medio de la aplicación	13.1	
		Interfaz de Ingreso de Objetos.	13.2	
		Interfaz de Consulta de Objetos.	13.3	
		Interfaz de Actualizaciones de Objetos.	13.4	
		Interfaz de Eliminación de Objetos.	13.5	

Capítulo IV	7. Mantener la seguridad en la comunicación del CLIENTE/SERVIDOR. Utilizando SSL para impedir que la información viaje desprotegida por la red.	**Descarga de paquetes necesarios para la aplicación.**	14	2 Semanas
		Instalación de OpenSSL	15	
		Configuración de SSL	16	
		Ejecución y Pruebas de funcionamiento	17	
Capítulo V	8. Implementación y Pruebas de funcionamiento, operatividad y seguridad del Servidor de Directorio con la Aplicación Cliente.	**Implementación de:**	18	2 Semanas
		Servidor	18.1	
		Aplicación Cliente	18.2	
		Encriptación	18.3	
		Elaboración de Pruebas de:	19	
		Conexión al Servidor por medio de la aplicación	19.1	
		Ingreso de Objetos del Directorio	19.2	
		Consulta de Objetos del Directorio	19.3	
		Actualizaciones de Objetos del Directorio	19.4	
		Eliminación de Objetos del Directorio	19.5	
		Encriptación para la seguridad	19.6	

Tabla # 2
Estimación de la duración de la actividad

1.12.2 Cronograma

Tareas	Sept				Oct				Nov				Dic				Ene				Feb				Mar				Abr				May	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Introducción																																		
Análisis																																		
Análisis de la Estructura de Objetos																																		
Análisis del Comportamiento de Objetos																																		
Diseño																																		
Diseño de Estructura de OPENLDAP																																		
Diseño de Interfaces																																		
Instalación, Codificación y Seguridad																																		
Instalación, Config. Y Ejecución del Servidor																																		
Codificación del Diseño propuesto																																		
Seguridad en la comunicación.																																		
Implementación y Pruebas.																																		

Indicadores

Resumen de tareas
Resumen de Actividades
Feriados

Tabla # 3
Cronograma

1.13 Recursos del Proyecto

1.13.1 Planeación de Recursos

Determinar que recursos (personas, equipos, materiales) y en que cantidad se deben usar para ejecutar las actividades.

A continuación detallamos los recursos utilizados y presentamos un cuadro de los recursos utilizados en cada actividad del proyecto.

Recursos:

- Investigación Internet y biblioteca.
- Equipos, dispositivos y utilitarios
  - Laptop TOSHIBA
  - Impresoras hp 1400, hp Laser Jet 1300
  - Laboratorio de Graduación de CISC
  - 3 Pen drivers
  - Microsoft Office(Utilitario)
- Personal del Proyecto:
  - 3 Personas
- Material de Apoyo:

  - Libro (Análisis y diseño orientado a Objetos de j. Martin y J. Odel).
  - Manuales OpenLdap.
- Material de Oficina
- Tutor del Proyecto:
  - Ing. Carlos Montes
  - Ing. Carlos Salazar
- Herramienta para el desarrollo de la aplicación:
  - JDK 1.5
  - Eclipse SDK 3.2.1
- Sistema Operativo
  - Linux Fedora Core 4,
  - Windows XP
- Herramienta OPEN SURCE:
  - OPENLDAP.

Alcance	Actividades	Recursos
1. Antecedentes, situación conflicto, causas, consecuencias, misión, visión, objetivos y Administración del Proyecto.	Recopilación de información	▪ Investigación Internet y biblioteca. ▪ Equipos, dispositivos y utilitarios (laptop, laboratorio, impresora, pen drivers Microsoft Office). ▪ Personal del Proyecto. ▪ Material de Apoyo – Libro (Análisis y diseño orientado a Objetos de j. Martin y J. Odel). ▪ Material de Oficina (plumas, hojas, lápices, etc.)
	Selección de la información recopilada para antecedentes	
	Definición de situación conflicto, causas, consecuencias	
	Administración del Proyecto (Planeación)	
2. Análisis para el desarrollo de un Servidor de Directorio Seguro y Diseño de una aplicación para el uso del mismo.	**Análisis de la Estructura de Objetos (AEO)**	
	Definición de tipos de Objetos	
	Definición de Generalizaciones	
	Diagrama de Relación de Objetos	
	Análisis del Comportamiento de Objetos (ACO)	
	Estados de Objetos y cambios de los mismos	
	Definición de Eventos, pre estados, post estados	
	Diagrama de Operaciones (Definición de Operaciones, condiciones de control, etc.)	
	Diagrama de flujo de Objetos	
	Diagrama General	
3. Diseño de árbol para la base de datos que utilizaremos en el Servidor de Directorio.	**Diseño de Estructura de Base de Datos del OPENLDAP**	
	Modelo de Información	
	Modelo de Nombrado	
	Modelo de Funcional	
	Modelo de Seguridad	

4. Diseño de una Aplicación CLIENTE que interactué con el servidor de directorio para poder obtener la información requerida del mismo.	**Diseño de Interfaces**	▪ Tutor del Proyecto.
	Ingreso al Servidor por medio de la aplicación	
	Interfaz de Ingreso de Objetos.	
	Interfaz de Consulta de Objetos.	
	Interfaz de Actualizaciones de Objetos.	
	Interfaz de Eliminación de Objetos.	
5. Instalación, Configuración y Ejecución del Servidor OPENLDAP.	**Descargar la herramienta OPENLDAP**	▪ Investigación Internet y biblioteca. ▪ Equipos (laptop, computadores del laboratorio, impresoras). ▪ Personal del Proyecto. ▪ Material de Apoyo – Libro (Análisis y diseño orientado a Objetos de j. Martin y J. Odel).
	Instalación de paquetes para el Servidor	
	Configuración del Servidor, Cliente y Directorio	
	Ejecución del Servidor, Cliente y Directorio	
6. Codificación del Diseño de la Aplicación Cliente que interactuará con el Servidor de Directorio.	**Codificación del Diseño propuesto**	
	Ingreso al Servidor por medio de la aplicación	
	Interfaz de Ingreso de Objetos.	
	Interfaz de Consulta de Objetos.	
	Interfaz de Actualizaciones de Objetos.	
	Interfaz de Eliminación de Objetos.	

7. Mantener la seguridad en la comunicación del CLIENTE/SERVIDOR. Utilizando SSL para impedir que la información viaje desprotegida por la red	**Descarga de paquetes necesarios para la aplicación.**	▪ Material de Oficina (plumas, hojas, lápices, etc.) ▪ Tutor del Proyecto. ▪ Herramienta de programación S1studio JDK 1.5 ▪ Sistema Operativo Linux Fedora Core 4, Windows XP ▪ Herramienta OPENLDAP.
	Instalación de OpenSSL	
	Configuración de SSL	
	Ejecución y Pruebas de funcionamiento	
8. Pruebas de funcionamiento, operatividad y seguridad del Servidor de Directorio con la Aplicación Cliente.	**Implementación de:**	
	Servidor	
	Aplicación Cliente	
	Encriptación	
	Elaboración de Pruebas de:	
	Conexión al Servidor por medio de la aplicación	
	Ingreso de Objetos del Directorio	
	Consulta de Objetos del Directorio	
	Actualizaciones de Objetos del Directorio	
	Eliminación de Objetos del Directorio	
	Encriptación para la seguridad	

Tabla # 4
Recursos

1.13.1.1 Recursos de Hardware

Equipo		Descripción
Servidor	Procesador	Normalmente servidores multiprocesador.
	Discos duro	Para OpenLDAP los más óptimo es que uses un disco duro para el sistema operativo (preferiblemente en RAID) y un disco separado para la base de datos (normalmente sin RAID). Elige discos duros muy rápidos.
	Tamaño de la memoria	Dependerá del número de entradas que almacenaremos y del número de atributos que use cada entrada como de las pruebas de carga que realicemos y sus resultados. Normalmente entre 1 GB y 4 GB.
Cliente	Procesador	Pentium IV
	Discos duros	Desde 40 GB
	Tamaño de la memoria	Desde 512MB

Tabla # 5
Recursos de Hardware

1.13.1.2 Recursos de Software

Equipo		Descripción
Paquetes de OpenLDAP		El servidor (slapd.conf), el cliente (ldap.conf) y el directorio (schema). Herramientas adicionales para el mantenimiento del sistema (slapcat, slapadd, slapindex)
Servidor	Sistema Operativo LINUX distribución FEDORA CORE 4	Elegir una instalación simple, sólo con los complementos imprescindibles.
	Sistema de Archivos Ext3 para Linux	Elegir un sistema de archivos adecuado.
Cliente	Sistema Operativo Linux o Windows	Puede ser una estación de trabajo con sistema operativo LINUX o WINDOWS.

Tabla # 6
Recursos de Software

1.14 Estimación de Costos

Desarrollar una aproximación de los costos de los recursos que se necesita para completar las actividades de los alcances del proyecto (Tabla #7).

RECURSOS	COSTOS
Equipo de Desarrollo	
Laptop TOSHIBA	$800.00
Impresoras Canon Pixma	$60.00
Gastos Generales	
Investigación Internet y biblioteca.	$30.00
3 Pen drivers	$75.00
Fotocopias de Material de Apoyo	$40.00
Material de Oficina	$32.15
Gastos Varios	
Luz	$80.00
Agua	$48.00
Teléfono	$48.00
TOTAL DEL COSTO	**$1.213.15**

Tabla # 7
Estimación de Costos

1.15 Riesgos

Los riesgos de un proyecto son eventos o condiciones inciertas que, en caso de ocurrir, tienen efectos positivos o negativos sobre los objetivos y alcances del proyecto.

Un riesgo tiene una causa y, si ocurre (evento de riesgo), una consecuencia (efecto).

Riesgos conocidos: aquellos que han sido identificados y analizados durante la planificación del proyecto.

Habitualmente se gestionan los riesgos con efecto negativo, es decir, aquellos que suponen una amenaza para el éxito del proyecto.

1.15.1 Identificación de Riesgos

En esta sección se determina que riesgos potenciales pueden ocurrir y que tienen la posibilidad de afectar el proyecto.

Riesgos en la Elaboración de la Planificación

- Las definiciones de la planificación, de los recursos y del producto no sean equilibrados.
- Información innecesaria.
- Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).
- La planificación no incluye tareas necesarias.
- No se puede construir la aplicación en el tiempo asignado.
- El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.).
- Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.

En los requisitos del proyecto

- Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.
- Se añaden requisitos extra.
- Las partes del proyecto que no se han especificado claramente consumen más tiempo del esperado.

Diseño, Interfaz y Desarrollo del proyecto

- Desconocimiento de conceptos de análisis y diseño orientado a objetos.
- Un mal diseño implica volver a diseñar e implementar.
- La utilización de metodologías desconocidas deriva en un periodo extra de formación y tener que volver atrás para corregir los errores iniciales cometidos en la metodología.
- El desarrollo de una interfaz de usuario inadecuada requiere volver a diseñarla y a implementarla.
- Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación.
- El desarrollo de funciones software innecesarias alarga la planificación.
- Las herramientas de desarrollo no están disponibles en el momento deseado.
- Las herramientas de desarrollo no funcionan como se esperaba; el personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas.
- Las herramientas de desarrollo no se han elegido en función de sus características técnicas, y no proporcionan las prestaciones previstas.

- El aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.

1.15.2 Cualificación, Cuantificación y priorización de los Riesgos

1.15.2.1 Cuantificación y Cualificación

Una vez identificados los riesgos, el siguiente paso es analizar cada riesgo para cuantificarlo y determinar su impacto (efecto sobre los objetivos del proyecto en caso de ocurrir) en el proyecto. Tanto la estimación de la probabilidad que ocurra el riesgo como de el impacto de estos se realizan de forma subjetiva. Usaremos adjetivos (muy improbable, probable, improbable) y asignar un valor numérico a cada uno (Tabla # 8).

ADJETIVO	VALOR
Muy probable	3
Probable	2
Improbable	1

Tabla # 8
Cuantificación Cualificación de Riesgos

Nº	RIESGOS	CUALIFICACIÓN (ADJETIVO)	CUANTIFICACIÓN (VALOR)
	Riesgos en la Elaboración de la Planificación		
1	Las definiciones de la planificación, de los recursos y del producto no sean equilibrados.	PROBABLE	2
2	Información innecesaria.	IMPROBABLE	1
3	Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).	MUY PROBABLE	3
4	La planificación no incluye tareas necesarias.	PROBABLE	2
5	No se puede construir la aplicación en el tiempo asignado.	PROBABLE	2
6	El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.).	PROBABLE	2
7	Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.	MUY PROBABLE	1
	En los requisitos del proyecto		
8	Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.	PROBABLE	2
9	Se añaden requisitos extra.	IMPROBABLE	1
10	Las partes del proyecto que no se han especificado claramente consumen más tiempo del esperado.	PROBABLE	2

Diseño, Interfaz y Desarrollo del proyecto			
11	Desconocimiento de conceptos de análisis y diseño orientado a objetos.	MUY PROBABLE	3
12	Un mal diseño implica volver a diseñar e implementar.	PROBABLE	2
13	La utilización de metodologías desconocidas deriva en un periodo extra de formación y tener que volver atrás para corregir los errores iniciales cometidos en la metodología.	PROBABLE	2
14	El desarrollo de una interfaz de usuario inadecuada requiere volver a diseñarla y a implementarla.	IMPROBABLE	1
15	Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación.	PROBABLE	2
16	El desarrollo de funciones software innecesarias alarga la planificación.	MUY PROBABLE	3
17	Las herramientas de desarrollo no están disponibles en el momento deseado.	MUY PROBABLE	3
18	Las herramientas de desarrollo no funcionan como se esperaba; el personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas.	MUY PROBABLE	3

Tabla # 9
Riesgos

1.15.2.2 Priorización

Nº	RIESGOS	PROBABILIDAD DE PERDIDA	MAGNITUD / PERDIDA (Semanas)
	Riesgos en la Elaboración de la Planificación		
1	Las definiciones de la planificación, de los recursos y del producto no sean equilibrados.	35%	1 Semana y 3 días
2	Información innecesaria.	10%	1 día
3	Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).	50%	3 Semanas
4	La planificación no incluye tareas necesarias.	25%	3días
5	No se puede construir la aplicación en el tiempo asignado.	40%	2 Semanas
6	El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.).	35%	1 Semana y 3 días
7	Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.	60%	4 Semanas
	En los requisitos del proyecto		
8	Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.	30%	1 Semana
9	Se añaden requisitos extra.	10%	1 día
10	Las partes del proyecto que no se han especificado claramente consumen más tiempo del esperado.	45%	2 Semana y 3 días

	Diseño, Interfaz y Desarrollo del proyecto		
11	Desconocimiento de conceptos de análisis y diseño orientado a objetos.	50%	3 semanas
12	Un mal diseño implica volver a diseñar e implementar.	45%	2 Semana y 3 días
13	La utilización de metodologías desconocidas deriva en un periodo extra de formación y tener que volver atrás para corregir los errores iniciales cometidos en la metodología.	30%	1 Semana
14	El desarrollo de una interfaz de usuario inadecuada requiere volver a diseñarla y a implementarla.	10%	1 día
15	Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación.	40%	2 Semana
16	El desarrollo de funciones software innecesarias alarga la planificación.	50%	3 semanas
17	Las herramientas de desarrollo no están disponibles en el momento deseado.	55%	3 Semanas y 3 días
18	Las herramientas de desarrollo no funcionan como se esperaba; el personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas.	60%	4 Semanas

Tabla # 10
Priorización

1.15.3 Desarrollo de las Respuestas a los Riesgos

Una vez identificados los riesgos, y realizada la estimación de la “exposición a riesgos”, se construye un plan de contingencia para cada uno de los riesgos identificados, que se han elegido como significativos para gestionarlos.

Qué hacer en este punto depende mucho del tipo de riesgo. Sin embargo, las acciones o estrategias que se pueden seguir son las siguientes:

- Evitar el riesgo: es decir, no realizar actividades arriesgadas cambiando el plan del proyecto.
- Conseguir información acerca del riesgo cuando éste no es muy conocido.
- Eliminar el origen del riesgo.
- Mitigar el riesgo. No se elimina el origen pero se cambia el plan para que su efecto sea menor.
- Comunicar el riesgo al resto del equipo, al cliente y a la dirección, para que estén prevenidos.
- Aceptar que el riesgo puede ocurrir y hacer un plan de contingencias para minimizar la posibilidad que se dé.

La colección de todos los planes de contingencia se suele agrupar en el llamado *Plan de Respuestas a Riesgos* – (Respuestas previstas para aquellos riesgos que se han considerado prioritarios y que serán planificados).

Plan de Contingencia de cada Riesgo

A continuación desarrollamos el Plan de contingencia de los riesgos con mayor probabilidad de ocurrencia y que representan mayor retraso en el proyecto.

Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).

Pasos a seguir:

1. Darle mayor amplitud en tiempo a tareas críticas.
2. Realizar una optimización del tiempo considerando los alcances definidos en el inicio del proyecto.
3. Tener claro las actividades a realizar en cada alcance para que el proyecto sea planificado de manera realista.

Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.

Pasos a seguir:

1. Elaborar cronogramas de actividades en Project ya que esta herramienta nos permite respetar las horas y los días laborables, dándonos una exactitud en el tiempo.
2. Respetar el cronograma de actividades y el tiempo que se le distribuye a cada una de ellas, tomando en cuenta feriados y holguras.
3. En lo que respecta a la codificación, desarrollar módulos independientes e integrarlos posteriormente.

Desconocimiento de conceptos de análisis y diseño orientado a objeto.

Pasos a seguir:

1. Breve capacitación sobre conceptos de análisis orientados a Objetos a aquellos integrantes que no conozcan esta metodología de análisis y diseño.
2. Tener a la mano Materiales de Apoyo – Libro (Análisis y diseño orientado a Objetos de j. Martin y J. Odel).

El desarrollo de funciones software innecesarias alarga la planificación.

Pasos a seguir:

1. Identificar todas las funciones que se pueden realizar en el proyecto.
2. Analizar el resultado que dará cada una de las funciones identificadas.
3. Definir solo aquellas funciones que me permitan llegar a mis alcances planteados.

Las herramientas de desarrollo no están disponibles en el momento deseado.

Pasos a seguir:

1. Elaborar una lista de todas las herramientas que se van a utilizar en la elaboración del proyecto.
2. Obtener los instaladores de las herramientas, que van a ser utilizadas incluyendo sus librerías y componentes.
3. Todos los integrantes del grupo deberán tener los instaladores de las herramientas que sean necesarios para el desarrollo.

Las herramientas de desarrollo no funcionan como se esperaba; el personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas.

Pasos a seguir:

1. Obtener manuales de uso y ayudas de las herramientas de desarrollo.
2. Realizar un cronograma de actividades en la que indique el tiempo que se va a tomar en resolver el problema.
3. Utilizar las Técnicas de Estimación de Tiempo, tomando en cuenta el número de personal, líneas de código y la herramienta de programación a utilizar, para que de esta manera podamos tener un valor mas aproximado de tiempo que llevará en desarrollar cada actividad.

1.16 Análisis Legal

En el desarrollo del Servicio de Directorio se utilizará OpenLdap, un servidor Ldap, que al ser un protocolo independiente de la plataforma, varias distribuciones Linux lo incluyen.

OpenLdap tiene su propia licencia la ”OpenLDAP Public License”, la cual puede considerarse de software libre permisiva, sin copyleft y compatible con la GNU GPL.

El Derecho de autor de OpenLdap (Copyright 1998-2004 The OpenLDAP Foundation), menciona que se permite la redistribución y el uso de código fuente con o sin modificación, además que los titulares del copyright no pueden ser usados para respaldar o promover productos provenientes del software sin previo permiso específico por escrito.

Ya que la licencia de OpenLdap es sin copyleft no nos obliga como usuarios a distribuir un software derivado como software libre. Podríamos redistribuirlo como semilibre o incluso a través de licencias de software propietario, incluso reservarnos en exclusiva los derechos de modificación y distribución sobre el software derivado.

Copia de la Licencia la ”OpenLDAP Public License”

Se permiten la versión pública 2.8 de la licencia de OpenLDAP, la redistribución del 17 de agosto de 2003 y el uso de este software y documentación asociada (“software”), con o sin la modificación, a condición de que se resuelven las condiciones siguientes: 1. Las redistribuciones en forma de la fuente deben conservar las declaraciones y los avisos, 2. Del copyright. Las redistribuciones en forma binaria deben reproducir declaraciones aplicables del copyright y los avisos, esta lista de condiciones, y la negación siguiente en la documentación y/o otros materiales proporcionados la distribución, y 3. Las redistribuciones deben contener una copia de este documento. La fundación de OpenLDAP puede revisar esta licencia de vez en cuando. Cada revisión es distinguida por un número de versión. Puedes utilizar este software bajo términos de esta revisión de la licencia o de conformidad con cualquier revisión subsecuente de la licencia. ESTE SOFTWARE ES PROPORCIONADO POR LA FUNDACIÓN DE OPENLDAP Y SUS CONTRIBUIDORES “COMO ES” Y CUALQUIER GARANTÍA EXPRESADA O IMPLICADA, INCLUYENDO, PERO NO LIMITADO A, LAS GARANTÍAS IMPLICADAS DEL MERCHANTABILITY Y APTITUD PARA UN PROPÓSITO PARTICULAR SE NIEGAN. NUNCA LA FUNDACIÓN DE OPENLDAP, TUS CONTRIBUIDORES, O LOS AUTORES O LOS DUEÑOS DEL SOFTWARE SEAN OBLIGADOS PARA LOS DAÑOS DIRECTOS, INDIRECTOS, FORTUITOS, ESPECIALES, EJEMPLARES, O

CONSECUENTES CUALESQUIERA (INCLUYENDO, PERO NO LIMITADO A, CONSECUCIÓN DE MERCANCÍAS SUBSTITUTAS O LOS SERVICIOS; PÉRDIDA DE USO, DE DATOS, O DE BENEFICIOS; O INTERRUPCIÓN DEL NEGOCIO) SIN EMBARGO CAUSADA Y EN CUALQUIER TEORÍA DE LA RESPONSABILIDAD, SI EN CONTRATO, RESPONSABILIDAD TERMINANTE, O EL AGRAVIO (NEGLIGENCIA INCLUYENDO O DE OTRA MANERA) QUE SE PRESENTA EN CUALQUIER SALIDA DEL USO DE ESTE SOFTWARE, AUNQUE ACONSEJADO DE LA POSIBILIDAD DE TAL DAÑO. Los nombres de los autores y de los sostenedores del copyright no deben ser utilizados en la publicidad o promover de otra manera la venta, el uso u otro repartir en este software sin el específico, escrito el permiso anterior. Seguirá habiendo el título al copyright en este software siempre con los sostenedores del copyright. OpenLDAP es una marca registrada de la fundación de OpenLDAP. Copyright 1999-2003 la fundación de OpenLDAP, ciudad de la secoya, California, los E.E.U.U. Todos los derechos reservados. Se concede el permiso de copiar y de distribuir copias in extenso de este documento.

CAPITULO 2

ANÁLISIS

2.1 Levantamiento de información

Los esfuerzos de recopilación deben enfocarse en los hechos que permitan conocer y analizar información específica y verdaderamente útil para el proyecto

y su desarrollo, pues de lo contrario se puede incurrir en interpretaciones erróneas, lo cual genera retraso y desperdicio de recursos empleados.

Esta actividad exige mantener una relación constante con las fuentes emisoras de la información, así como con las áreas u organizaciones con otra ubicación física.

Para recabar la información en forma ágil y ordenada utilizamos las siguientes técnicas de recopilación:

Investigación documental:

Esta técnica permite la selección y análisis de aquellos escritos que contienen datos de interés relacionados con el proyecto, consultamos diversos proyectos en la biblioteca de la Carrera.

La Entrevista:

Consiste básicamente en celebrar reuniones individuales o grupales en las cuales se cuestiona orientadamente a los participantes para obtener información. Este medio es posiblemente el más usado y el que puede brindar información más completa y precisa, puesto que el entrevistador, al tener contacto con el entrevistado, además de

obtener respuestas, puede percibir actitudes y recibir comentarios. Realizamos entrevistas con el Ing. Carlos Montes para los requerimientos del proyecto.

2.2 Análisis de la Estructura de Objetos (AEO)

2.2.1 Definición de tipos de objetos

- DOMINIO
- ORGANIZACIÓN
  - ✓ EDUCATIVAS
  - ✓ ADMINISTRATIVAS
  - ✓ RECTORADO Y VICERECTORADO
- DEPARTAMENTOS
  - ✓ ACADEMICOS
  - ✓ ADMINISTRATIVOS
- USUARIO
  - ✓ DOCENTES
  - ✓ PERSONAL ADMINISTRATIVO Y DE SERVICIO
  - ✓ ALUMNOS
- GRUPOS
  - ✓ ANONIMO
  - ✓ ADMINISTRADOR
- S.D.S (Aplicación Servidor de Directorio Seguro)
- SERVIDOR SEGURO

2.2.2 Definición de Generalizaciones

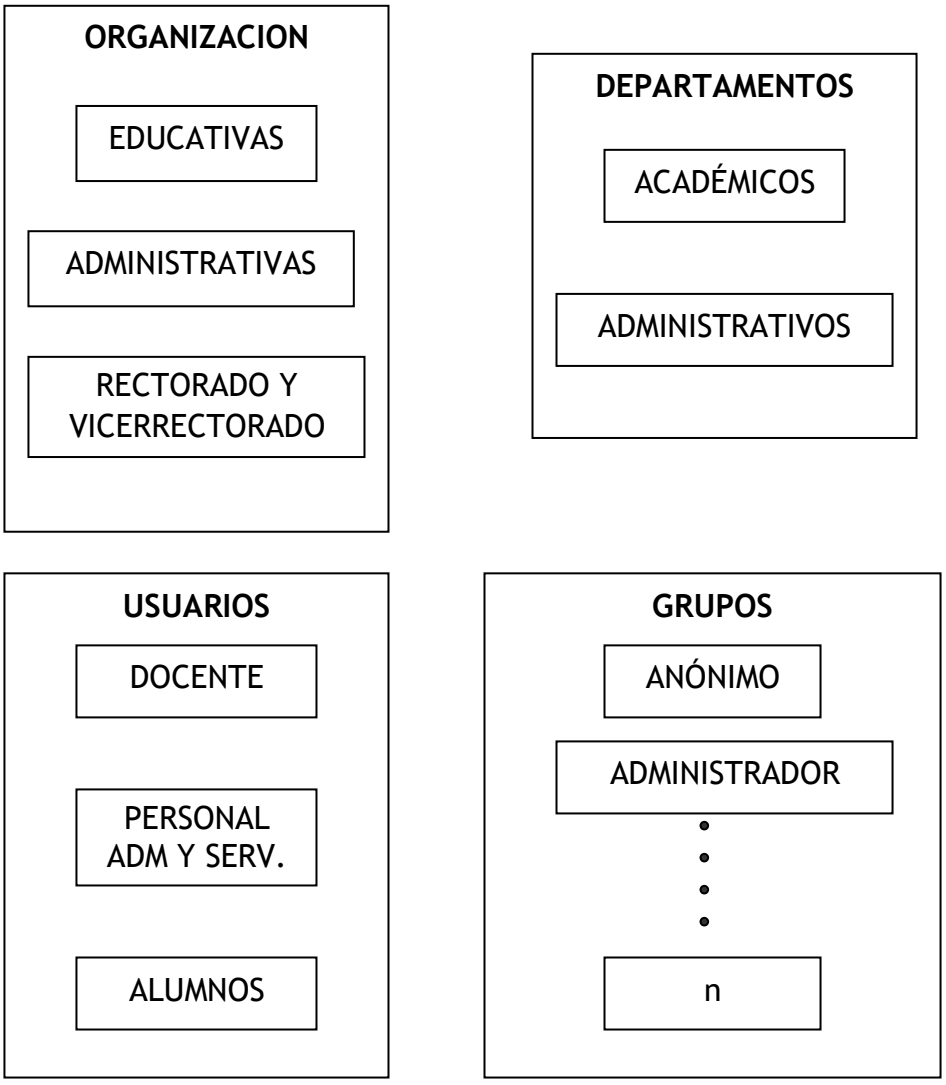


Gráfico #3
Definición de Generalizaciones

2.2.3 Diagrama de Relaciones de Objetos

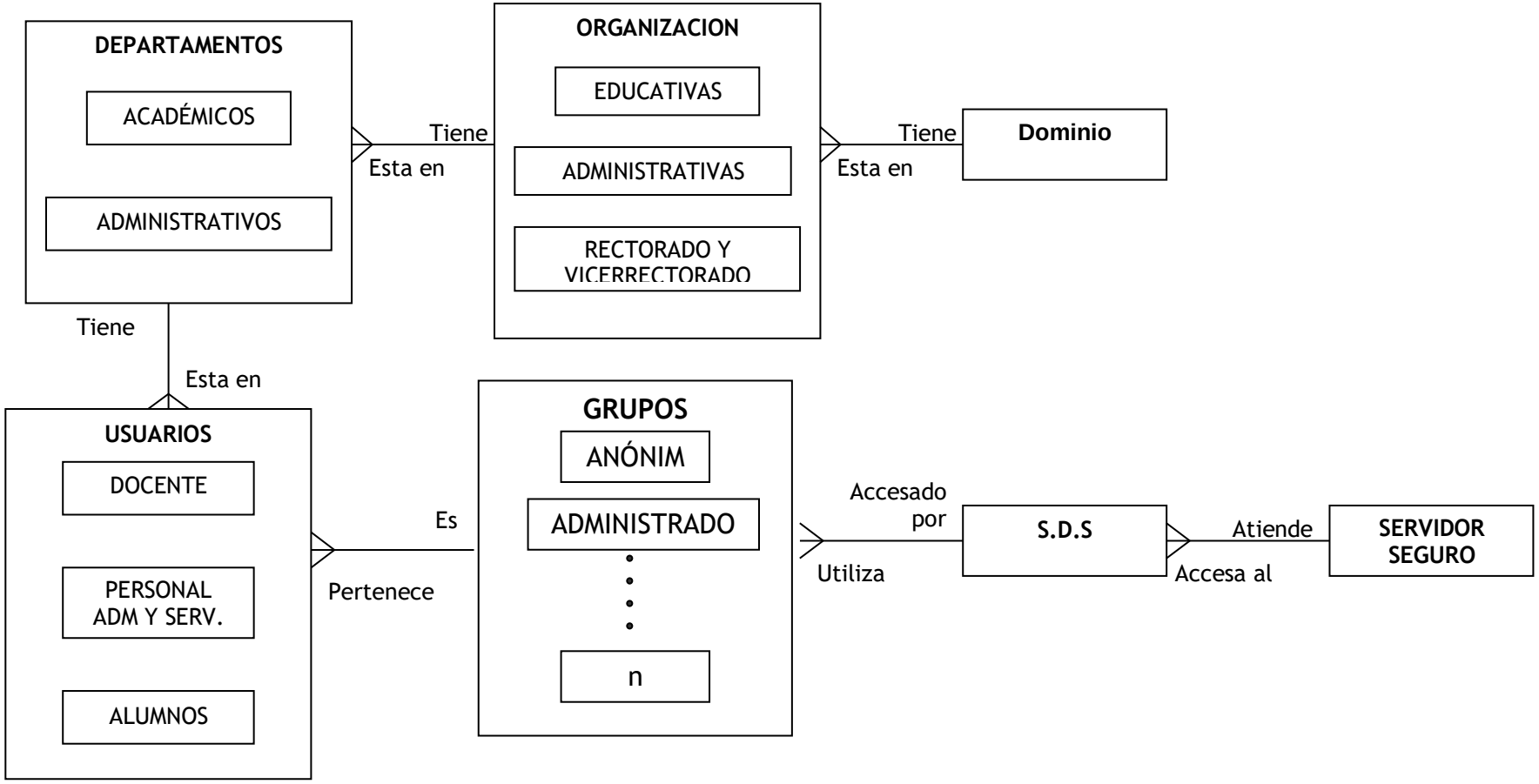
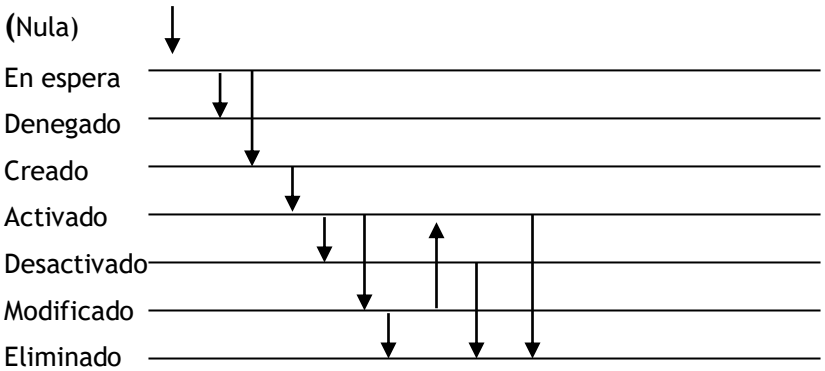


Gráfico #4
Diagrama de Relaciones de Objetos

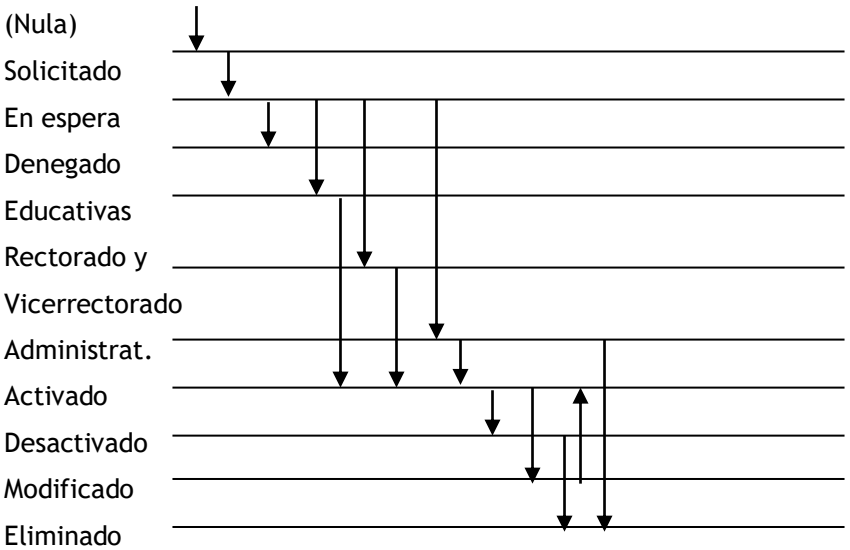
2.3 Análisis del Comportamiento de Objetos (ACO)

2.3.1 Estado de Objetos y cambio de los mismos

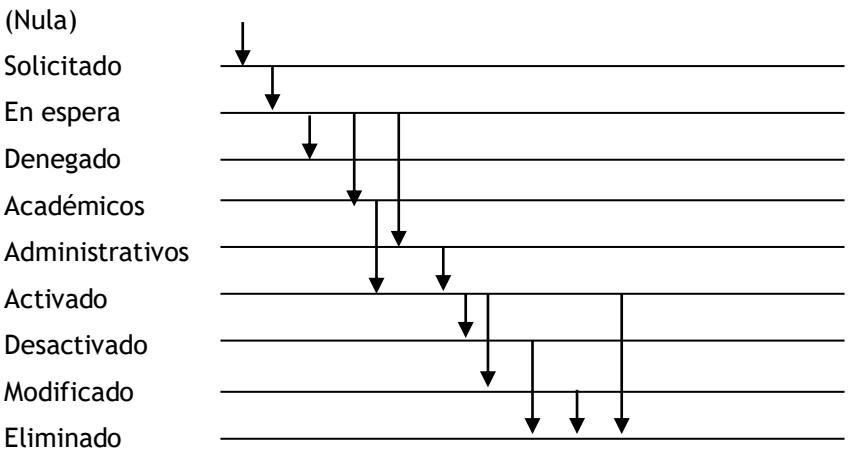
DOMINIO



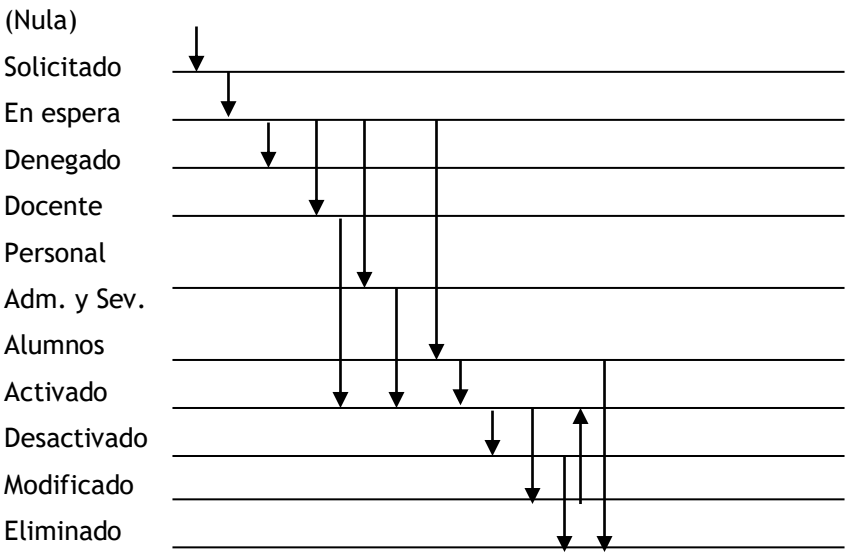
ORGANIZACION



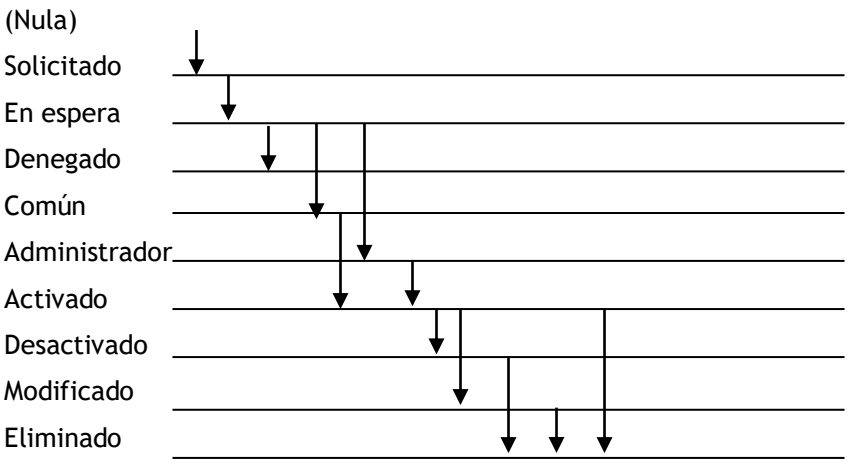
DEPARTAMENTOS



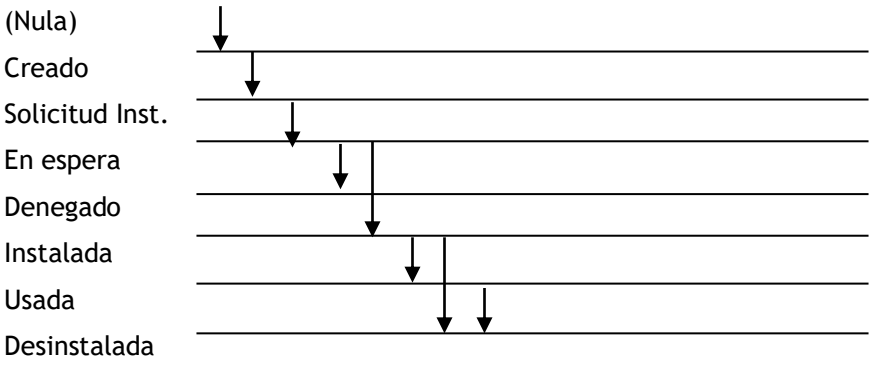
USUARIOS



GRUPOS



S.D.S.



SERVIDOR SEGURO

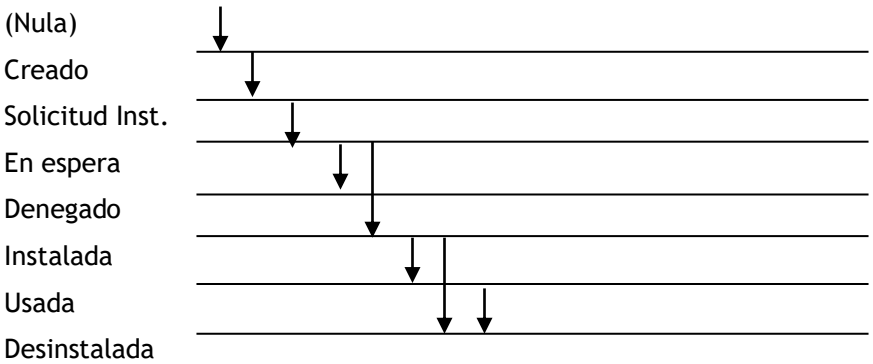


Gráfico #5
Estado de Objetos

2.3.2 Definición de Eventos, pre estados, post estados

TIPO DE EVENTO	NOMBRE DE EVENTO	PREESTADO DEL EVENTO	POST ESTADO DEL EVENTO
Requerir	Acceso Requerido	No existe Objeto alguno	Solicitud realizada
Solicitar	Solicitud realizada	No existe Objeto alguno	Solicitud en espera
Esperar	Solicitud en análisis	Solicitud en espera	Respuesta (Aceptar/Rechazar)
Crear	Dominio creado Solicitud creada	Solicitud en espera Solicitud en espera	Dominio activado Organización activada
Crear y clasificar	Grupos creados como Anónimo, Administrador Usuario creado como Docente, Personal Adm y Servicio, Alumnos	Solicitud aceptada Solicitud aceptada	Grupo: Común, Administrador,….n Usuario: Docente, Personal Adm. Y Serv, Alumnos
Activar	Objeto activado para consulta	Objeto creado	Objeto activado
Modificar	Objeto actualizado (Dominio, Organización, Usuario, Grupos)	Objeto Activado	Objeto Modificado
Eliminar	Objeto Eliminado (Dominio, Organización, Grupo, Usuario)	Objeto Activado/Desactivado	No existe Objeto

Tabla # 11
Definición de Eventos

2.3.3 Diagrama de Operaciones (Definición de Operaciones, condiciones de activación y Resumen)

2.3.3.1 Identificar operaciones

Nº	OPERACIONES
1	Solicitar acceso
2	Revisar Acceso
3	Llenar formulario
4	Enviar solicitud de acceso
5	Análisis de solicitud
6	Creación de acceso al Sistema
7	Enviar respuesta al Cliente
8	Acepta acceso conociendo privilegios
9	Realizar operaciones
10	Digita datos de objetos
11	Verificar datos
12	Almacenar datos de forma segura
13	Digita datos de objeto a buscar
14	Buscar datos de objetos de forma segura
15	Visualizar datos de objetos
16	Modifica datos de objetos
17	Eliminar Objetos del Servidor
18	Continuar Operaciones

Tabla # 12
Identificar Operaciones

2.3.3.2 Definición de condiciones de activación

Nº	CONDICIONES DE ACTIVACIÓN
1	Si requiere acceso
2	Si no existe dominio, Usuario , Grupo y/o Solicitud Rechazada
3	Si existe formulario
4	Envio de solicitud existente
5	Análisis de solicitud aceptada
6	Si existe en el Servidor
7	Si objeto (dominio, organización, grupo, usuario) pertenece al Servidor y tiene privilegios y/o Continua Operación
8	Operación realizar Ingreso
9	Si datos digitados
10	Datos digitados correctamente
11	Operación a realizar: Otra
12	Se escribe datos a buscar
13	Si encuentran datos objetos para: Consulta
14	Si encuentran datos objetos para: Actualización
15	Si encuentran datos objetos para: Eliminación
16	Si desea continuar / No desea continuar

Tabla # 13
Definición de Condiciones

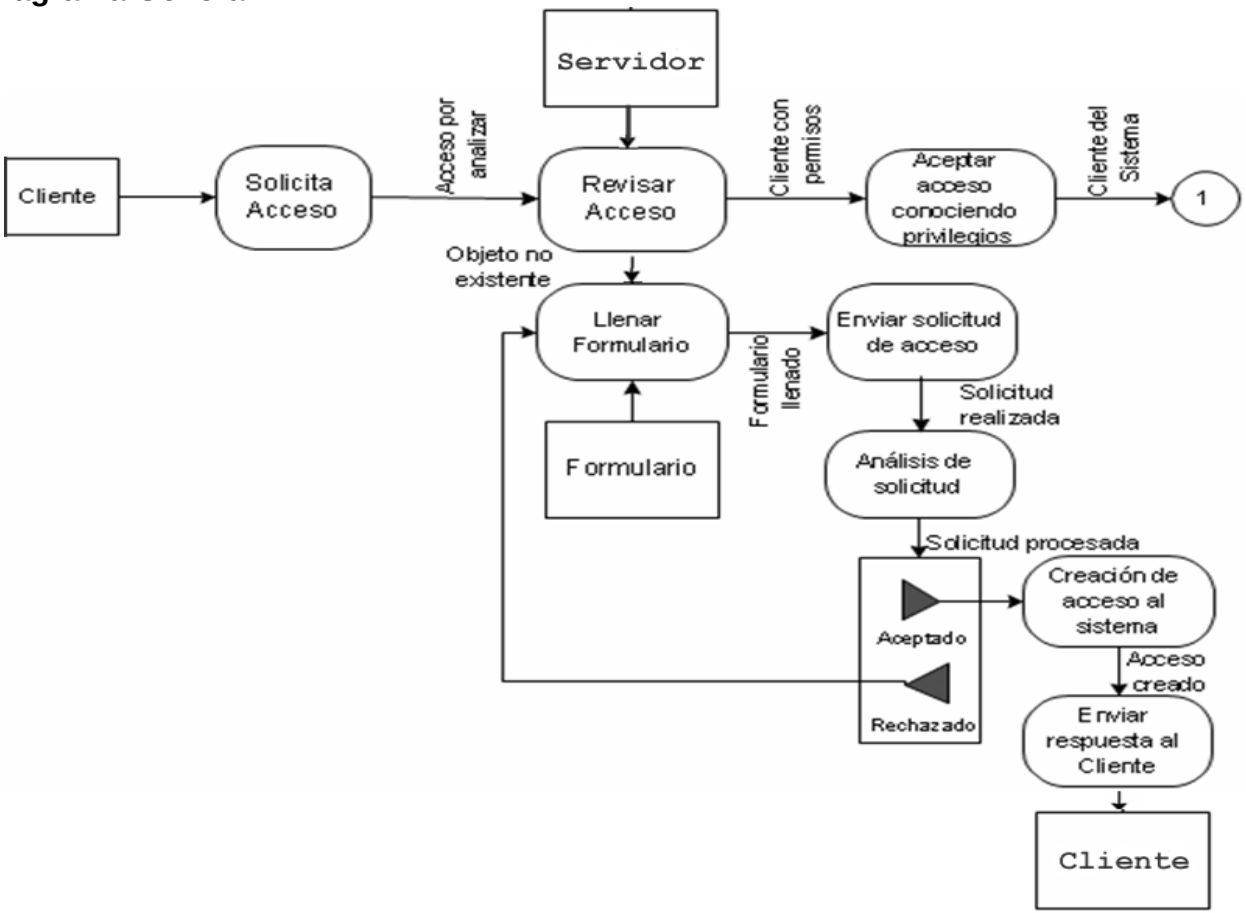
2.3.3.3 Resumen Eventos, Condición de Activación, Operaciones

EVENTOS	CONDICIONES DE ACTIVACION	OPERACIONES
-	-	Solicitar acceso
Acceso por analizar	Si requiere acceso	Revisar Acceso
Objeto no existente en Servidor	Si no existe dominio, Usuario , Grupo y/o Solicitud Rechazada	Llenar formulario
Formulario llenado	Si existe formulario	Enviar solicitud de acceso
Solicitud Realizada	Envío de solicitud existente	Análisis de solicitud
Acceso Creado	Si se crea el Acceso	Envía respuesta al cliente
Solicitud en análisis	Análisis de solicitud aceptada/ Rechazada	Creación de acceso al Sistema
Cliente del Sistema	Si existe en el Servidor	Acepta acceso conociendo privilegios
Cliente con Servicios	Si objeto (dominio, organización, grupo, usuario) pertenece al Servidor y tiene privilegios y/o Continua Operación	Realizar operaciones

EVENTOS	CONDICIONES DE ACTIVACION	OPERACIONES
No existe objeto	Operación realizar Ingreso	Digita datos de objetos
Objeto digitado	Si datos digitados	Verificar datos
Objeto correctamente digitado	Datos digitados correctamente	Almacenar datos de forma segura
Cliente con permiso	Operación a realizar: Otra	Digita datos de objeto a buscar
Objeto digitado por buscar	Se escribe datos a buscar	Busca datos de objetos de forma segura
Objeto activo	Si encuentran datos objetos para: Consulta	Visualizar datos de objetos
Objeto activo	Si encuentran datos objetos para: Actualización	Modifica datos de objetos
Objeto activo	Si encuentran datos objetos para: Eliminación	Eliminar Objetos del Servidor
Objeto activo / Objeto modificado/ Objeto eliminado / Objeto desactivado	Si desea continuar / No desea continuar	Continuar Operaciones

Tabla # 14
Resumen de Eventos, Condiciones y Operaciones

2.3.4 Diagrama General



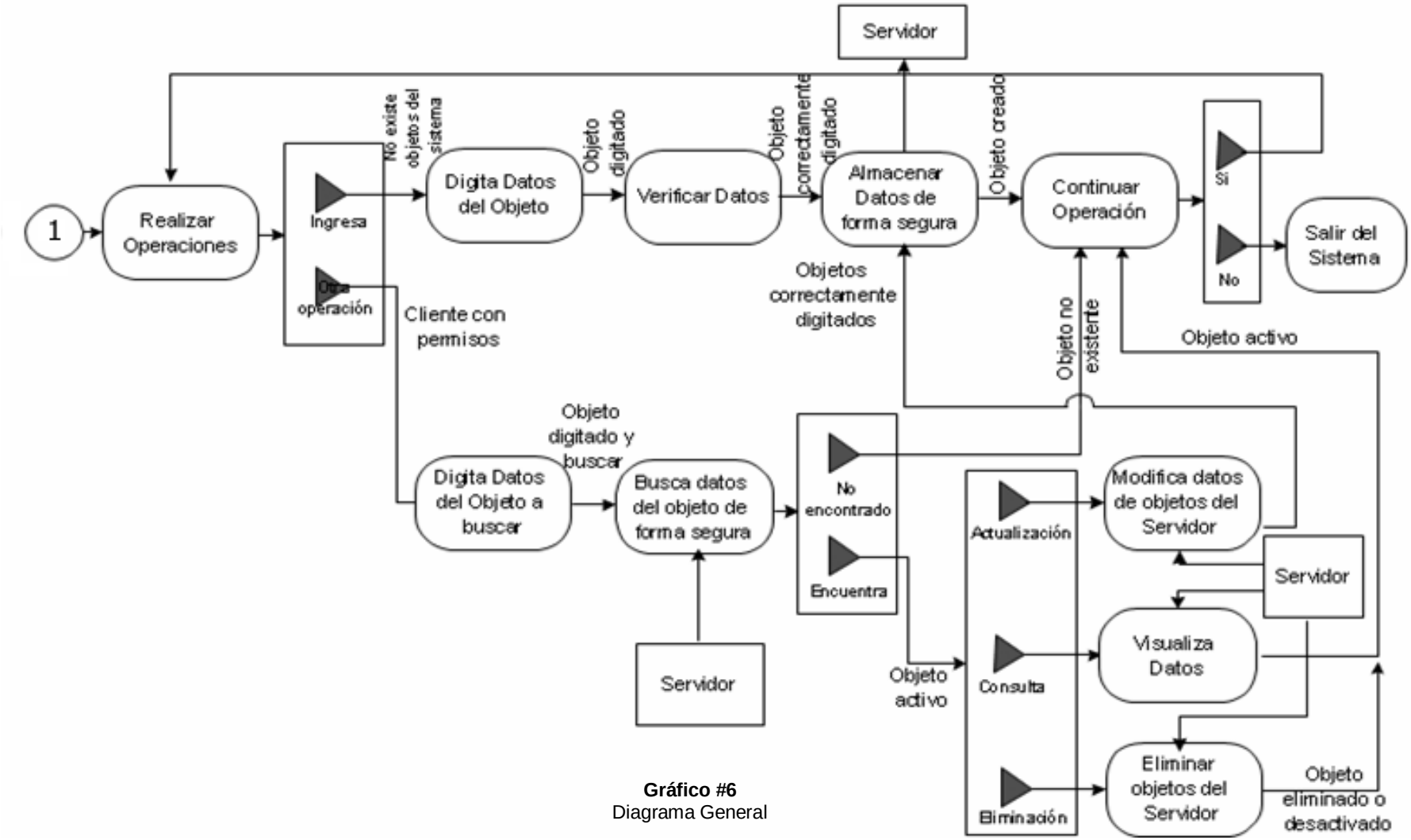


Gráfico #6
Diagrama General

CAPITULO 3

DISEÑO

3.1 Diseño de árbol para la base de datos.

Un árbol de directorio no es nada más que una manera organizada de proveer contenedores para almacenar diferentes tipos de información. Es como un sistema para que tus datos lo llenen.

El estándar LDAP define cuatro modelos que permiten entender mejor el servicio de directorio

- ✓ Modelo de información
- ✓ Modelo de nombrado
- ✓ Modelo funcional
- ✓ Modelo de seguridad

3.1.1 Modelo de Información

La unidad básica de información almacenada en el directorio es la entrada. Generalmente una entrada representa un objeto del mundo real (una persona, un servidor, etc), pero el modelo no exige este aspecto.

Se describe la estructura de la información almacenada en el directorio LDAP, presentando un árbol de directorio en el cual cada una de las cajas representa una entrada en el directorio. Toda la información que se guarda en el directorio se almacena de forma jerárquica, formando el árbol de directorio.

Una entrada se compone de un conjunto de atributos, cada uno de ellos tiene un tipo y uno o varios valores. El tipo define la clase de información que va a almacenar y los valores son la información en sí. Los tipos de los atributos tienen asociada una determinada sintaxis, la cual describe los tipos de datos que se van a almacenar como valores de este atributo, así como también define como se van a realizar las comparaciones en las búsquedas.

Árbol de Información del Directorio

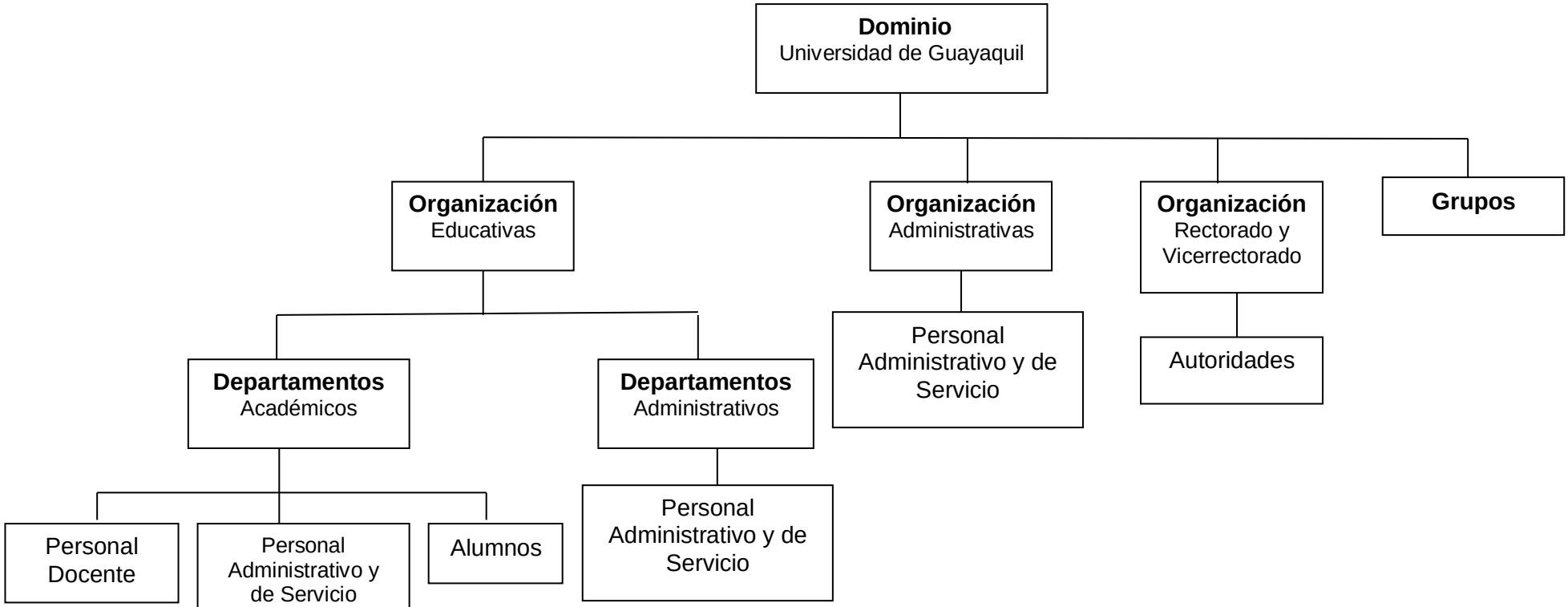


Gráfico #7
Árbol de Información de Directorio

Al diseñar el modelo de información se ha intentado ser lo más fiel posible a las entradas y atributos necesarios para el proyecto (Tabla #15).

ENTRADA	CLASE DE INFORMACION
Organización Educativas	Información de las diversas facultades que posee la universidad
Organización Administrativas	Información de las dependencias administrativas de la universidad
Organización Rectorado y Vicerrectorado	Información del Rectorado, Vicerrectorados General, Administrativo y Académico
Departamento Administrativos	Información correspondiente a los departamentos administrativos de cada Organización Educativa
Departamento Académicos	Información correspondiente a los departamentos académicos de cada Organización Educativa
Personal Docente	Datos informativos del personal docente que labora en la universidad
Personal Administrativo y de Servicio	Datos informativos del personal administrativo y de Servicio que labora en la universidad
Alumnos	Datos informativos del alumnado de la universidad
Autoridades	Datos informativos de las autoridades de la universidad
Grupos	Información de elementos que componen el Directorio con el propósito de asignarles privilegios

Tabla # 15
Entradas y Atributos

3.1.2 Modelo de Nombrado

Describe como se organiza e identifica la información en el directorio LDAP. Una vez organizadas las entradas formando una determinada estructura, este modelo nos indica como referenciarlas.

Las entradas son organizadas dentro del árbol de información del Directorio (“DIT” *Directory Information Tree*) en base a su nombre distintivo (“DN”*Distinguished Name*).

No puede existir ninguna entrada suelta, solo la entrada raíz puede no tener una entrada padre. En el caso de añadir una entrada en un punto inexistente en el directorio, el servidor devolverá un mensaje de error y no realizará la operación.

Esta flexibilidad permite al directorio almacenar la información de la forma más conveniente, se puede crear un grupo que contenga todas las personas de la organización y otro que contenga todos los grupos o se puede elegir una estructura que refleje la estructura jerárquica de la organización.

Esquema de Nombrado del Directorio

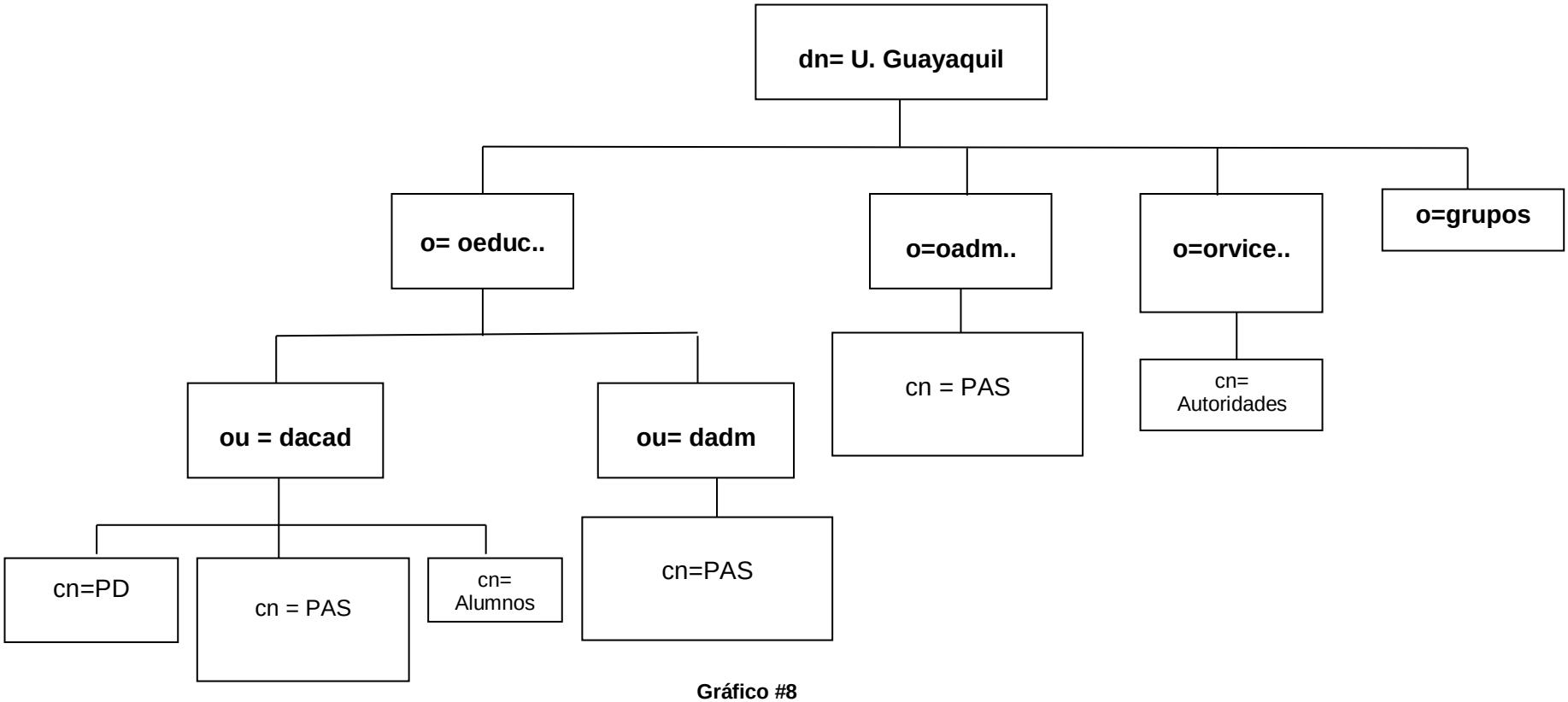


Gráfico #8
Esquema de Nombrado del Directorio

3.1.3 Modelo Funcional

Este modelo describe que operaciones pueden ser realizadas sobre la información almacenada en el directorio LDAP, definiendo un conjunto de operaciones divididas en tres grupos. Las operaciones de consulta permiten realizar búsquedas en el directorio y recuperar datos necesarios. Las operaciones de actualización o modificación permiten añadir, borrar, renombrar y cambiar entradas del directorio. Las operaciones de autenticación y control permiten la identificación de los clientes y del directorio, así como controlar ciertos aspectos de una sesión.

Operaciones de Consulta:

Existen dos operaciones de consulta que permiten buscar y obtener información almacenada en el directorio y son utilizadas en este proyecto:

Operación search

Permite buscar en el directorio las entradas que cumplen las especificaciones indicadas de atributos, entradas, etc. por ello se deben especificar los siguientes parámetros:

Base, DN que indica el punto de partida para la búsqueda.

Scope, ámbito de la búsqueda, puede ser:

Base, solo se busca en la entrada base.

One, se busca en el nivel inmediantemente inferior a la entrada base.

Subtree, se busca en todo el subárbol bajo la entrada base.

Filtro de búsqueda, indica el criterio de búsqueda.

Atributos a devolver, se puede indicar que atributos se devuelven y devuelve el valor del atributo o el tipo de dato contenido.

Límite, indica el número máximo de entradas que serán devueltas.

Operación compare

Esta operación es similar a la operación de consulta utilizando un filtro de equiparación, la diferencia se encuentra en que dada una entrada que cumple con las especificaciones pero que no tiene el atributo que se desea devolver, el directorio devuelve un valor especial, para indicar que la entrada cumple con los requisitos.

Operaciones de actualización

Existen cuatro operaciones que permiten añadir, borrar, renombrar (modificar el DN) y modificar. A continuación detallamos su funcion:

Operación add

Esta operación permite añadir nuevas entradas al directorio, recibe como parámetros el DN de la entrada a crear, los atributos y los valores asociados. Para poder realizar esta operación se debe cumplir que:

- El nodo padre de la entrada exista en el directorio.
- No debe haber otra entrada con el mismo DN.

- La entrada debe cumplir con los requisitos especificados en el esquema.
- El control de accesos permita esta operación.

Operación delete

Permite eliminar entradas del directorio, recibe como parámetro el DN de la entrada a borrar. Para poder realizar esta operación, se deben cumplir las siguientes condiciones:

- La entrada a borrar debe existir en el directorio.
- Dicha entrada no debe tener ningún hijo.
- El control de accesos debe permitir esta operación.

Operación rename

Esta operación permite modificar el DN de una entrada, para poder renombrar una entrada se deben cumplir las siguientes condificiones:

- La entrada a renombrar debe existir.
- No debe existir una entrada con el nuevo DN.
- El control de accesos debe permitir esta operación.

Operación modify

Permite la modificación de los atributos de una entrada. Para poder ejecutarse esta operación deben cumplirse las siguientes condiciones:

- La entrada a modificar debe existir.
- Las modificaciones de los atributos deben realizarse.
- La entrada resultante debe ser conforme al esquema.
- El control de accesos debe permitir la actualización.

En este punto se indica que las operaciones en LDAP son atómicas, si alguna de las modificaciones falla, toda la operación de actualización falla.

Operaciones de autenticación y control

Operación bind

Esta operación permite autenticar al cliente frente al directorio. Hay varios tipos de autenticación, desde una

sesión anónima, una sesión autenticada, en la que el usuario se ha identificando proporcionando la contraseña, La autenticación básica, en las que al establecer la conexión se envían al servidor el nombre distinguido del usuario y su contraseña en claro, el servidor considera que el cliente se ha autenticado si la contraseña coincide con la almacenada en el campo *userPassword*.

Operación unbind

Esta operación cierra la conexión con el servidor LDAP.

Operación abandon

Esta operación permite indicar al servidor LDAP que el cliente abandona la operación en curso.

3.1.4 Modelo de Seguridad

Describe como puede protegerse la información contenida en el directorio LDAP frente a accesos no autorizados.

Por medio de la comparación con la contraseña del administrador se comprueba si es efectivamente miembro del directorio, y luego por medio de la comparación de la entrada se determina los privilegios que le permitirá realizar las diferentes operaciones de consulta, inserción, actualización o eliminación.

También aplicaremos la encriptación para que la información no viaje en claro permitiendo de esta manera que cualquiera pueda capturar la información que se envía en la comunicación cliente – servidor utilizando openSSL.

3.2 Diseño de la Aplicación CLIENTE

3.2.1 Ingreso al Servidor por medio de la aplicación (Ejecución).

Gráfico #9
Interfaz de Ingreso al Servidor

Conexión con el Servidor.- Conectar al Servidor es la primera interfaz que presenta en la aplicación Cliente, en la que se visualizarán dos pestañas (Gráfico #9):

- Lista de Sesiones (el usuario tendrá la oportunidad de crear, editar, copiar, borrar y renombrar sesiones.)

- Conexión rápida (el usuario deberá ingresar datos de configuración como host, dn, puerto, en el caso que sea un usuario administrador deberá ingresar el usuario y contraseña)

Gráfico #10
Interfaz de Conexión

Si es un usuario Administrador este podrá realizar las operaciones de ingreso, modificación, eliminación, búsquedas de entradas del árbol de directorio. Si es un usuario anónimo este solo podrá realizar operaciones de búsquedas.

Además el usuario podrá conectarse mediante un puerto seguro 636 eligiendo la opción SSL, caso contrario se conectará por defecto mediante el puerto 389 (Gráfico #10).

3.2.2 Interfaz de aprobación de certificado autofirmado

Certificado Autofirmado.- Al conectarse al servidor mediante el protocolo SSL se muestra la siguiente ventana, en la que el usuario tendrá tres opciones: Aceptar el certificado en esta sesión, aceptarlo siempre, y no aceptarlo. En caso de elegir la tercera opción la aplicación no podrá conectarse en modo seguro (Gráfico #11).

Gráfico #11
Interfaz de Aprobación de Certificado

3.2.3 Visualización del árbol de directorio.

Una vez conectada la aplicación con el servidor, se observará en la parte izquierda de la ventana el árbol de directorio, a la derecha los atributos del nodo con sus respectivos valores; las barras de menú y de herramientas; que nos permitirán interactuar con el directorio (Gráfico #12).

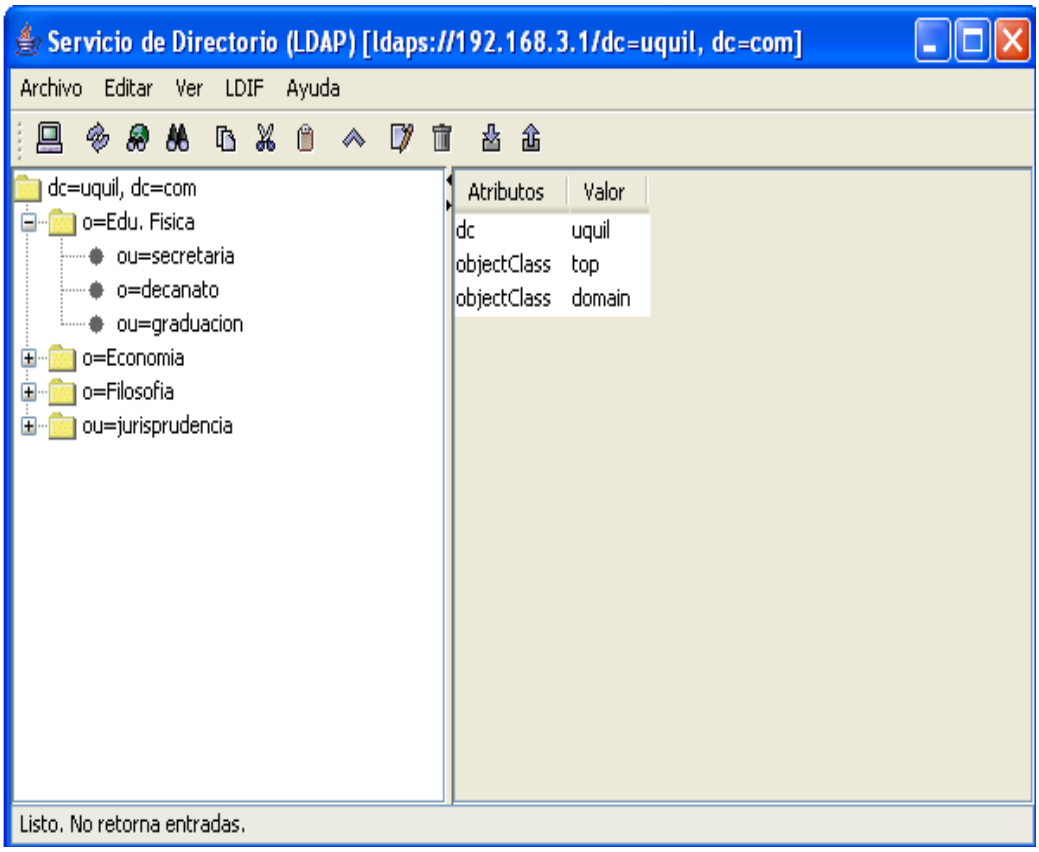


Gráfico #12
Pantalla de Visualización del Árbol de Directorio

3.2.4 Interfaz de Ingreso de Objetos.

Ingreso.- Mediante la aplicación se podrán crear entradas de organizaciones, unidades organizacionales, departamentos administrativos o personas (Gráfico #13).

Teniendo que ingresar datos como nombre, número de teléfono, descripción, fax, en este caso la nueva entrada es la facultad de filosofía.

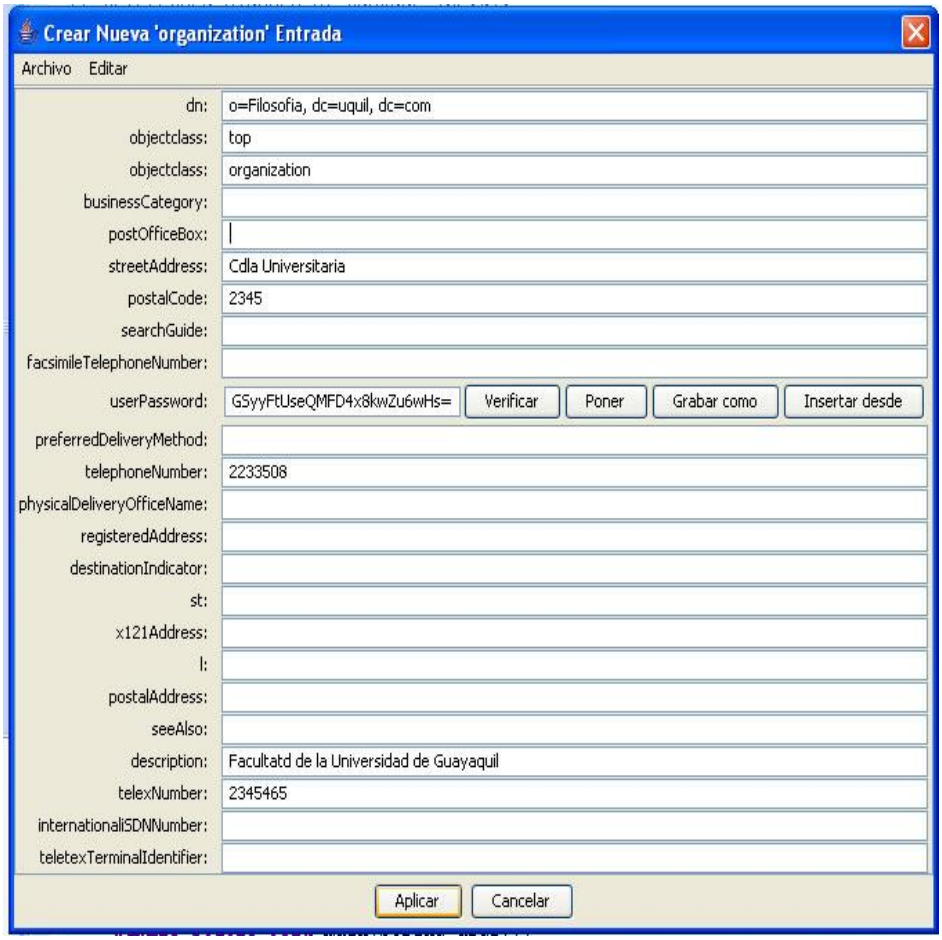


Gráfico #13
Interfaz de Ingreso de Objetos

Después de presionar el botón aplicar se observará el nodo ingresado con sus respectivos atributos y valores.

3.2.5 Interfaz de Búsqueda

Consulta o Búsqueda.- En la ventana se muestra automáticamente la raíz (DN: dc=uquil, dc=com) del árbol. En Filtrar se deberá indicar el criterio de busqueda, por ejemplo

objectclass=*, objectclass=top, objectclass=person,. objectclass=organization (Gráfico #14).

En atributos, se indicará los valores o atributos que se buscan de determinada entrada. Teniendo dos opciones de búsqueda:

1. Un nivel: Muestra en la parte inferior la raíz del nodo buscado.
2. Sub niveles de árbol: Muestra el nodo a buscar y los nodos del cual se deriva.

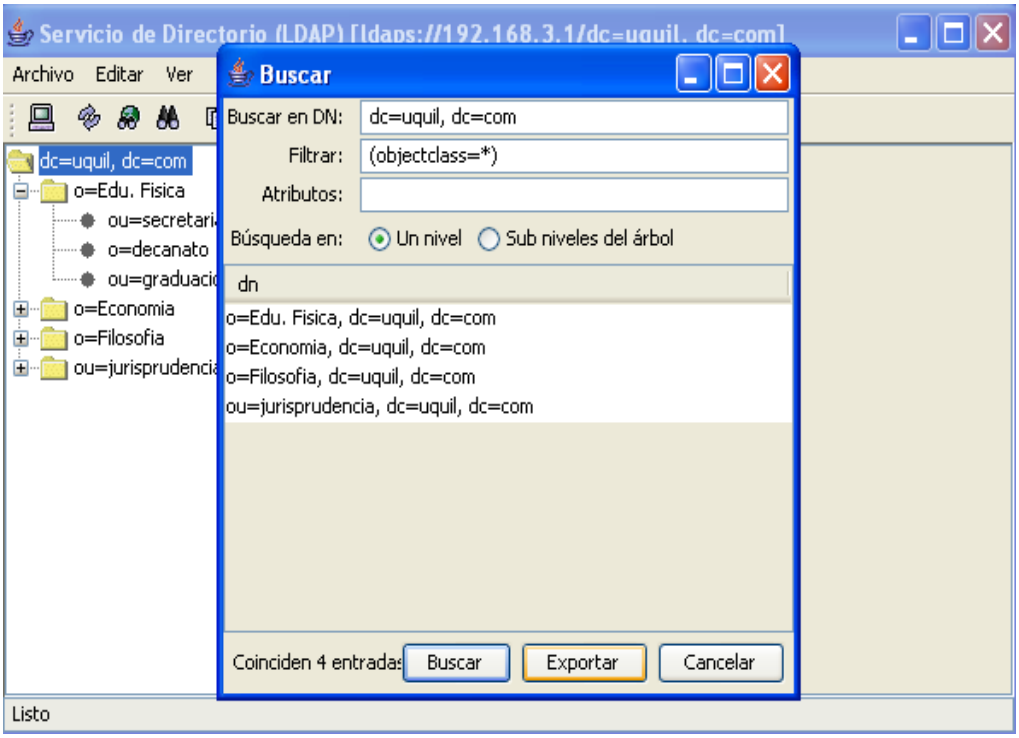


Gráfico #14
Interfaz de Búsqueda

3.2.6 Interfaz de Modificación

Modificación- Esta interfaz muestra la edición de los nodos del árbol de directorio, se podrá modificar los valores de los atributos anteriormente ingresados.

Al presionar el botón aplicar se observará los valores de los atributos modificados (Gráfico #15).

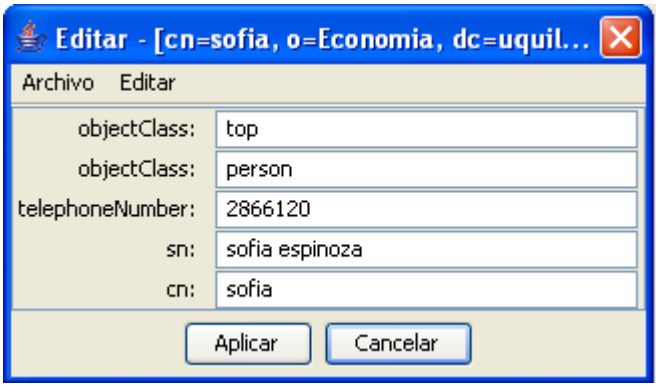


Gráfico #15
Interfaz de Modificación

3.2.7 Interfaz de Eliminación de Objetos

Eliminación:, Esta pantalla permitirá al usuario eliminar el nodo. Si el nodo a eliminar tiene sub niveles tiene que estar seleccionada la opción con hijos. Caso contrario la aplicación

mostrará un mensaje de error en la barra de estado y no permitirá la eliminación del nodo (Gráfico #16).

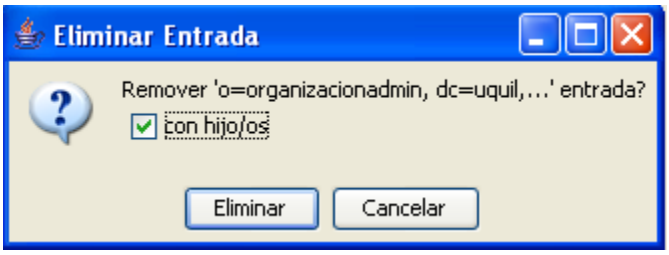


Gráfico #16
Interfaz de Eliminación

CAPÍTULO 4

DESARROLLO Y PRUEBA DEL SISTEMA

4.1 Instalación, configuración y ejecución de OpenLdap como servidor de directorio

4.1.1 Descarga e instalación de OpenLdap

Se descargó la versión disponible en Fedora Core 4,

Versión OpenLap 2.2.23-5, que contiene las librerías necesarias para ejecutar las aplicaciones necesarias del servidor y cliente OpenLdap.

Además de los siguientes paquetes:

- openldap-clients-2.2.23-5: Contiene herramientas de líneas de comando para visualizar y modificar directorios en un servidor Ldap.
- openldap-servers-2.2.23-5: Contiene los servidores y otras utilidades necesarias para configurar un servidor Ldap.
- authconfig-4.6.12-1
- authconfig-gtk-4.6.12-1 (opcional)

Para la instalación utilizamos los discos de Fedora Core 4, en los cuales como mencionamos al inicio del capítulo se encuentra la versión de OpenLdap con los paquetes detallados mencionados.

4.1.2 Configuración del Servidor OpenLdap

Editamos el fichero *slapd.conf* que se encuentra en el directorio etc/openldap, este archivo es el principal de OpenLDAP, donde se configuran los parámetros necesarios para el funcionamiento del servidor.

A continuación mencionamos los parámetros del archivo slapd.conf:

Parámetros Globales

Los parámetros que afectan el funcionamiento de todo el Servidor OpenLDAP son:

- *include:* Este parámetro indica otros archivos de configuración utilizados por el Servidor OpenLDAP,
- *referral:* Indica el URI (Identificador Uniforme de Recursos) del servicio de directorio superior como ldaps en lugar de ldap.
- *pidfile:* Contiene el número de proceso asignado al servidor LDAP al arranque.
- *argsfile:* Contiene parámetros utilizados en la línea de comandos al iniciar el servidor OpenLDAP.
- *access:* Parámetro utilizado para restringir acceso al servidor LDAP.

Parámetros por Base de Datos

Cada servidor LDAP posee bases de datos, es dentro de ellas donde residirá todos los datos que conforman la información del Servidor OpenLDAP.

En el sentido más estricto de la palabra OpenLDAP no utiliza una base de datos que se actualice constantemente o soporte miles de transacciones por segundo, sin embargo si utilizamos "base de datos" para almacenar la información. El tipo de base de datos que se utiliza en OpenLDAP es generalmente ldbm.

A continuación detallamos los parámetros que hacen referencia a la base de datos dentro del archivo *slapd.conf:*

- *database:* Indica el tipo de "base de datos" a utilizarse, generalmente del tipo ldbm.
- *suffix:* Este parámetro indica el nodo raíz de la base de datos.
- *rootdn:* Establece el nodo ("usuario") que tiene privilegios globales para modificar la "base de datos".
- *rootpw:* Indica la contraseña para el usuario rootdn.
- *directory:* Directorio dónde se guardarán todos los datos del directorio LDAP.

4.1.3 Ejecución y Comprobación del Servidor OpenLdap

Ejecución del Servidor

Posteriormente iniciamos el servicio de LDAP añadiendo éste al resto de los servicios que arrancan junto con el sistema, esto se lo hace vía comando con *service ldap Start.*

Creamos un fichero LDIF (LDAP Data interchange Format O Formato de Intercambio de LDAP), ya que el servidor de directorio trabaja con este fichero. El cual almacena información de forma jerárquica las entradas orientadas a objetos. Ldif es un fichero ASCII.

Comprobación del Servidor

Se ha verificado que todo funcione correctamente Antes de configurar el sistema para utilizar LDAP. Utilizando el comando *Ldapsearch.*

4.1.4 Configuración de Clientes

Configuramos el archivo ldap.conf, el cual permite la comunicación de los clientes con el servidor, para esto se ha definido los valores para los parámetros host y base, esto se hace con el fin de establecer hacia que servidor y a que directorio conectarse.

Además se configuró el paquete authconfig en modo texto para habilitar la autenticación y permitir que el cliente se integre a un dominio LDAP, También verificamos en modo gráfico que los datos del servidor y el directorio sean los correctos (Gráfico #17).

Gráfico #17
Configuración de Clientes

4.2 Creación de componentes

Para la creación de los componentes se utilizó la metodología orientada a objetos ya que tiene una mayor disciplina que el de las técnicas de estructura convencional. Este proyecto se ha realizado siguiendo el modelo de desarrollo en espiral.

El orden de desarrollo se detalla a continuación:

Conexión al servidor LDAP

Operaciones Básicas (buscar, añadir, eliminar, modificar)

Desconexión de un servidor

Operaciones LDIF.

Certificación SSL

Las clases SSL que existen en el proyecto están disponibles de manera OpenSource en Internet, las mismas que fueron reutilizadas y adaptadas al desarrollo de la seguridad. Tomando en consideración el crear certificados para asegurar la comunicación del cliente con el servidor Linux. Además obtuvimos liberarías necesarias para la aplicación como provider, ldapbp y jndi que es creada principalmente para la lectura, escritura y modificación de directorios; utilizando únicamente la propiedad de lectura para extraer los datos necesarios de Ldap. Las mismas se las puede encontrar de manera gratuita en http://www.openldap.org

4.3 Seguridades

4.3.1 Descarga e instalación de paquetes necesarios

OpenLDAP tiene una forma de comunicación segura, SSL; la cual realiza las comunicaciones encriptadas en modo seguro. Por lo que se descargó la versión OpenSSL 0.97f-7.

OpenSSL es un software de cifrado que permite tanto el establecimiento de sesiones utilizando el protocolo SSL como la emisión de certificados Este software esta accesible públicamente en http://www.openssl.org.

4.3.2 Configuración y Ejecución de SSL

Configuración

Para habilitar las conexiones SSL hacia el servidor, se necesita de un certificado en el mismo, este certificado proporciona una conexión segura y encriptada.

Se creó la llave y el certificado para OpenLdap, requiriendo utilizar una clave con algoritmo RSA de 1024 octetos y estructura

X509, ya que este certificado es autofirmado se le asignó 730 días (2 años).

Ejecución

A fin de que surtan efecto los cambios, es necesario reiniciar el servicio ldap.

4.3.3 Pruebas

Las pruebas del sistema son un mecanismo para comprobar el funcionamiento correcto de la aplicación con la intención de descubrir un error para luego proceder a su corrección.

En el proyecto realizamos varios tipos de pruebas entre las más importantes tenemos:

4.3.3.1 Pruebas del Sistema

Las pruebas del sistema son un mecanismo para comprobar el funcionamiento correcto del software.

Según el índice de fiabilidad del software, la aplicación se comporta de acuerdo a las especificaciones funcionales y a los

requisitos del rendimiento. En este sistema se han hecho varios tipos de pruebas entre las más importantes tenemos:

- Verificación y validación
- Pruebas de unidad
- Pruebas de integración
- Pruebas de seguridad

Verificación y validación

En este tipo de prueba, la verificación analiza que no existan errores en la implementación de la aplicación, y la validación analiza que lo que se define en las especificaciones iniciales guarde relación con lo desarrollado.

Se realizaron pruebas con el software totalmente ensamblado como un todo para comprobar si cumple los requisitos funcionales y de rendimiento, facilidad de mantenimiento, recuperación de errores, etc. Se comprobó que la aplicación cubre todas las especificaciones definidas en los alcances, verificando que el sistema realiza todas las operaciones básicas en el directorio LDAP

Se realizó este tipo de prueba, con los datos que ingresan en el directorio, y se comprobó todos los posibles errores de ingreso que se pudieran dar, corrigiendo con codificación extra en todos los casos.

Pruebas de Unidad

La prueba de unidad consiste en ejecutar el código para convencerse de que básicamente funciona. En esta fase se revisó la codificación utilizando el depurador para encontrar fallos, para posteriormente corregir estos errores.

Se realizó la prueba de caja blanca, en el siguiente orden:

En la conexión, se probaron las clases que dan acceso al directorio, comprobando su operabilidad, se permite la comunicación por medio de la configuración del servidor en base a la ip colocada en la interfaz eth0 de la maquina Linux.

En la autenticación, se verificó que las clases con sus métodos funcionen correctamente, permitiendo identificar el

usuario que manipulará el árbol de directorio dándole así los privilegios necesarios para realizar operaciones sobre el directorio LDAP.

Con lo que respecta a la codificación relacionada con los ficheros LDIF, se verificó que se realicen importaciones y exportaciones sin problemas.

En las clases correspondientes a SSL, se comprobó el ingreso por el puerto seguro 636, aceptando el certificado generado y autofirmado por el servidor.

La prueba de caja negra se centra directamente en la interfaz con el usuario. Se realizaron comprobaciones en los ingresos de datos para observar que tan sensible es la aplicación al momento de las entradas, verificando también que la sistema tolera un volumen de datos aceptable. Además no produce efectos negativos en el funcionamiento del sistema y directorio al momento de ingresar combinaciones de datos.

Pruebas de Integración

En nuestro proyecto se realizó la prueba de integración de forma ascendente, probando así la integración de nivel superior.

Como resultados las operaciones de eliminación, modificación no se pueden realizar sin el método correspondiente a la autenticación de usuarios con privilegios.

Pruebas de Seguridad

Se realizó las pruebas con la aplicación desarrollada, utilizando SSL y habilitando el puerto 636 en el servidor. Tras aceptar el certificado autofirmado, el servidor LDAP permitió completar la conexión y realizar cualquier tipo de consulta y/o manipulación de registros. Además se comprobó que la aplicación se conecta correctamente teniendo configuradas todas las seguridades desde la maquina Windows.

Las pruebas de seguridad realizadas en el servidor, se las hizo comprobando que solo los puertos necesarios estén

habilitados, razón por la cual el servidor establece comunicación solo por los puertos 636 y 389.

Siendo el puerto 636 el que nos provee confidencialidad en el transporte de datos y protección de la integridad de datos ya que trabaja bajo el protocolo SSL.

Durante la comunicación Cliente – Servidor, el servidor envía su certificado con estructura X.509 para verificar la identidad del Clliente y permitirle manipular el directorio LDAP.

CAPÍTULO 5

IMPLEMENTACIÓN DEL SISTEMA

5.1 Introducción

El Sistema está implementado bajo un modelo CLIENTE – SERVIDOR el mismo que es desarrollado en Eclipse SDK 3.2.1,

Además se utilizan las siguientes herramientas o aplicaciones.

- Sistema Operativo Windows XP
- Sistema Operativo Linux Fedora Core 4
- OpenLdap 2.2.23-5
- OpenSSL 0.97f-7
- VMware Workstation (Máquina Virtual)

La implementación del sistema la llevamos a cabo con el desarrollo orientado a objetos mediante clases reutilizables.

En esta sección se desarrollará, las pautas y pasos necesarios para la puesta en marcha de una aplicación desarrollada para ejecutarse sobre plataforma Windows y Linux.

Como fase inicial, el equipo que servirá como anfitrión de la aplicación debe tener un Sistema Operativo robusto y de características que se enfoquen principalmente en la disponibilidad en caso de alta demanda, seguridad, fiabilidad y sobre todo agilidad; se debe recordar que la aplicación a ejecutarse es altamente de utilidad para toda organización que desee usar y administrar un directorio de manera confiable por todas las seguridades que este ofrece.

Ya que la opción por consola para visualizar, insertar, borrar o modificar datos no es muy amigable, presentamos la aplicación Servicio de Directorio (LDAP) la cual permite a los usuarios ver los elementos almacenados en un directorio LDAP de manera jerárquica y añadir, modificar o eliminar los elementos si el usuario se ha conectado como administrador.

Por motivos de amplitud y generalidad en el desarrollo de esta aplicación se incluirá también una guía de pasos para saber como utilizarla.

5.2 Elementos Físicos

Para la puesta en marcha de este sistema se requieren los siguientes elementos:

- Dos PCs para realizar todas las pruebas necesarias.
- Una PC para la configuración del Servidor de Directorio.
- Una Laptop para desarrollar la aplicación e instalar Sistema Operativo Linux sobre una máquina virtual.
- 1 cable UTP cat 5e para red entre las PCs.
- 1 regulador por las variaciones de voltaje.

Los elementos de Software requeridos son los siguientes instaladores de:

- Sistema Operativo Windows XP
- Sistema Operativo Linux Fedora Core 4
- OpenLdap 2.2.23-5
- OpenSSL 0.97f-7
- VMware Workstation (Máquina Virtual)
- Eclpse SDK 3.2.1
- Parches necesarios.

5.3 Elemetos Lógicos

Los elementos lógicos que se requieren son:

- Direcciones IP
- Validaciones de las variables a utilizarse en el transcurso de la construcción de los módulos.
- Declaración de la funciones a utilizar en cada módulo.
- Realización de las interfaces necesarias en el sistema.
- Construcción de las clases y métodos para la elaboración de todo el sistema.

5.4 Elemento Humano

Este proyecto se llevo a cado por tres personas, las mismas que aportaron con el conocimiento adquirido, en investigaciones sobre el tema a desarrollar así como también los conocimientos aprendidos en el transcurso de los Nueve Semestres.

5.5 Infraestructuras

El Sistema se ejecuta sobre máquinas que se encuentran debidamente configuradas en red a un servidor de directorio. Para lo que utilizamos un máximo de n PCS y un mínimo de 1 PC. Además, se emplearán tarjetas de red que garanticen compatibilidad con sistemas Linux tipo 10/100, de preferencia marca 3-Com.

Si existen mas de dos maquinas, se necesitará configurar un hub o un switch, para conectar todas las maquinas.

5.6 Capacitación a los usuarios

La aplicación es muy intuitiva y fácil de entender por los usuarios finales, contando con opciones de ayuda, las cuales pueden

ser consultadas en cualquier momento, detallando paso a paso el funcionamiento de la aplicación, permitiendo dar ayuda mientras el sistema se encuentre en ejecución.

Además se elaboró un manual de usuario que contiene toda la información detallada para la ejecución del mismo.

CAPÍTULO 6

RECOMENDACIONES Y CONCLUSIONES

6.1 Recomendaciones.

A continuación se menciona recomendaciones para el funcionamiento del sistema en cuanto a hardware y software.

6.2 Hardware

En cuanto al hardware recomendamos como mínimo los siguientes elementos:

Servidor

- Procesadores Pentium IV, Celaron, AMD.
- Memoria de 768 a 1G o más para el servidor.
- Disco duro con espacio particionado de 10Gb mínimo para la instalación del sistema operativo Linux Fedora Core 4.0.

Cliente

- Procesadores Pentium IV, Celaron, AMD.
- Memoria de 128 RAM o más para el cliente.
- Disco duro con espacio desde 40 GB mínimo para la ejecución de la aplicación.

Otros elementos

- CD-ROM 52x para la instalación de aplicaciones necesarias.
- Tarjeta de red 10 / 100 de preferencia 3-Com para mejor perfomance, no necesariamente, se puede usar cualquiera,

pero debe comprobarse la compatibilidad con el sistema operativo instalado.

- Cable de red para la conexión con la LAN local, categoría 5e o superior.

6.3 Software

El software recomendado para el desarrollo del sistema es el siguiente:

- Sistema Operativo recomendado para la plicación es Windows 2000, XP, Millenium.
- Sistema Operativo Linux FedoraCore 4.0
- OpenLdap 2.2.23-5

6.4 Cableado

Se recomienda como mínimo los siguientes:

- Utiizar un Cable Up categoría 5e.
- La maquina Linux Fedora Core debe tener correctamente configurada y activa la interfaz eth0
- Hab o switch en caso que se conecte varias pc en red
- Las tarjetas deben ser compatibles con el Fedora Core, se recomienda 3-Com 10/100.

6.5 Puesta en marcha

Para la puesta en marcha de la aplicación se requiere el archivo ejecutable (archivo.jar). La maquina donde se ejecuta la aplicación deberá estar conectada en red con el servidor.

Si por alguna razón no arranca, se deberá revisar las configuraciones de red de nuevo de ser necesario, tomando en consideración lo que se menciona en el manual de usuario.

6.6 Conclusiones

- El Sistema Servicio de Directorio Seguro ha sido desarrollado con la finalidad de que empresas incluyendo la Universidad de Guayaquil puedan hacer uso de un sistema que le permitan realizar consultas rápidas y de manera amigable del personal que tienen laborando en sus instituciones y de esta manera ayudar a la administración de estos recursos humanos, además de ofrecernos un esquema de su estructura.

- Tomando en consideración que una de las principales utilidades de los directorios ha sido la de buscar información, la aplicación

Sistema de Directorio (LDAP) Seguro puede filtrar búsquedas por apellido, por dirección, teléfono, etc, permitiéndole al usuario de la aplicación encontrar la información de manera más especifica para el uso que el mismo requiera, ya sea una dirección para enviar un oficio o un teléfono para comunicar una reunión urgente o alguna designación de viaje.

- Además esta aplicación ha sido desarrollada con el uso del protocolo SSL, el cual da una mayor garantía de seguridad debido a que realiza de manera encriptada las comunicaciones Cliente - Servidor. Haciendo uso de un certificado autofirmado con estructura X.509 para verificar la identidad del Cliente y permitirle manipular el directorio LDAP.

- Específicamente el uso de esta aplicación en la Universidad de Guayaquil permitirá que todos sus componentes (Administración Central, Rectorado, Vicerrectorados, facultades, etc.) puedan obtener información sobre sus miembros para realizar las diversas actividades que se requieran y sean necesarias dentro de la institución.

GLOSARIO DE TÉRMINOS

Esta sección del documento ofrece un breve resumen de los significados de ciertos términos técnicos en orden alfabético.

A

Autenticación.- Verificación de que el cliente sea quien dice ser para lograr acceso al sistema.

Administrador.- Persona que tiene privilegios de acceso al sistema.

C

Código Fuente.- Lenguaje mediante el cual fue programado el sistema.

Conexión.- Comunicación entre varias maquinas.

CN.- (Nombre de pila), se usa para almacenar el nombre de una persona.

E

Eth0.- Interfaz de red en Linux

F

Firewall.- Conjunto de políticas de seguridad de acceso a computadoras.

Freeware.- Software gratuito.

G

GPL.- Licencia pública general aplicada al software gratuito.

GUI.- Interfaces gráficas de usuario.

H

Hardware.- Parte física de una PC.

Hub.- Elemento físico de conexión en red.

I

IP.- Identificador único que distingue una computadora de otra.

Interfaz.- Medio grafico de comunicación entre la PC y el usuario

J

JNDI.- Java Naming and Directory Interface. Librería creada principalmente para la lectura, escritura y modificación de directorios

L

LAN.- Red de área local.

LDAP .- Lightweight Directory Access Protocol. Protocolo Ligero de Acceso a Directorios

LDBM.- Una base de datos de gran rendimiento basada en disco.

M

Maquina Virtual.- Software que permite la simulación de un computador.

Multiplataforma.- Se puede trabajar sobre diferentes sistemas operativos sin que afecte su desempeño.

O

Clase de Objetos (ObjectClass.-) Define la colección de atributos que pueden usarse para definir una entrada.

Open Source. - Tecnología libre de distribución por la cual no se necesita licencia.

OrganizationalPerson.- Proporciona atributos para describir a una persona que pertenezca a una organización.

P

Passwords.- Contraseña de acceso al sistema.

Paquetes.- Segmento de datos que se transmite de un lugar a otro.

Parámetros.- Datos que se envía a una función para que cumpla con su propósito.

Person.- Proporciona atributos para describir a una persona.

Políticas de Seguridad.- Conjunto de normas y reglas para mantener seguro un sistema.

Protocolos. – Implementación de la lógica de una capa del modelo OSI.

R

Red.- Conjunto de maquinas que se comunican entre si.

Root.- Directorio raíz.

S

Schema (esquema).- Define que clases de objetos se pueden almacenar en el directorio, que atributos deben contener, que atributos son opcionales y el formato de los atributos.

Servidor.- Equipo de computo, el cual esta recibiendo constante peticiones de clientes para proveerle de algún servicio.

Software.- Programas en ejecución.

Switch.- Dispositivo de interconexión de redes de computadoras que opera en la capa 2 del modelo OSI, este interconecta dos o más segmentos de red.

SSL.- Nivel de Zocalo Seguro.

T

Telnet.- Servicio que permite abrir una interfaz de comunicación.

V

Vulnerable.- Sensibilidad para sufrir un ataque.

X

X.500.- Organiza las entradas en el directorio de manera jerárquica, capaz de almacenar gran cantidad de datos, con grandes capacidades de búsqueda y fácilmente escalable.

CAPÍTULO 1

MANUAL TÉCNICO

A continuación se presenta el funcionamiento del proyecto, definición de objetos, generalizaciones, diagrama de relación y generalización de los objetos, estado de los objetos y cambios en los mismos, eventos, estados, pre-estados, post-estados, condiciones de activación, diagrama de flujo de objetos, etc.

Además también ponemos a su disposición el diccionario de datos, codificación de los componentes más relevantes y con su respectiva explicación.

El objetivo de este manual técnico es dar a conocer la estructura y comportamiento de los objetos como el servidor, la aplicación y demás objetos que interactúan en el proyecto.

1.1 Análisis de la Estructura de Objetos (AEO)

1.1.1 Definición de tipos de objetos.

En el proyecto encontramos los siguientes objetos:

- DOMINIO
- ORGANIZACIÓN (organization)
  - EDUCATIVAS
  - ADMINISTRATIVAS
  - RECTORADO Y VICERECTORADO
- DEPARTAMENTOS (organization unit)
  - ACADEMICOS
  - ADMINISTRATIVOS
- USUARIO (person)
  - DOCENTES
  - PERSONAL ADMINISTRATIVO Y DE SERVICIO
  - ALUMNOS
- GRUPOS (person unit)
  - ANONIMO
  - ADMINISTRADOR

- S.D.S (Aplicación Servidor de Directorio Seguro)
- SERVIDOR SEGURO

1.1.2 Definición de Generalizaciones

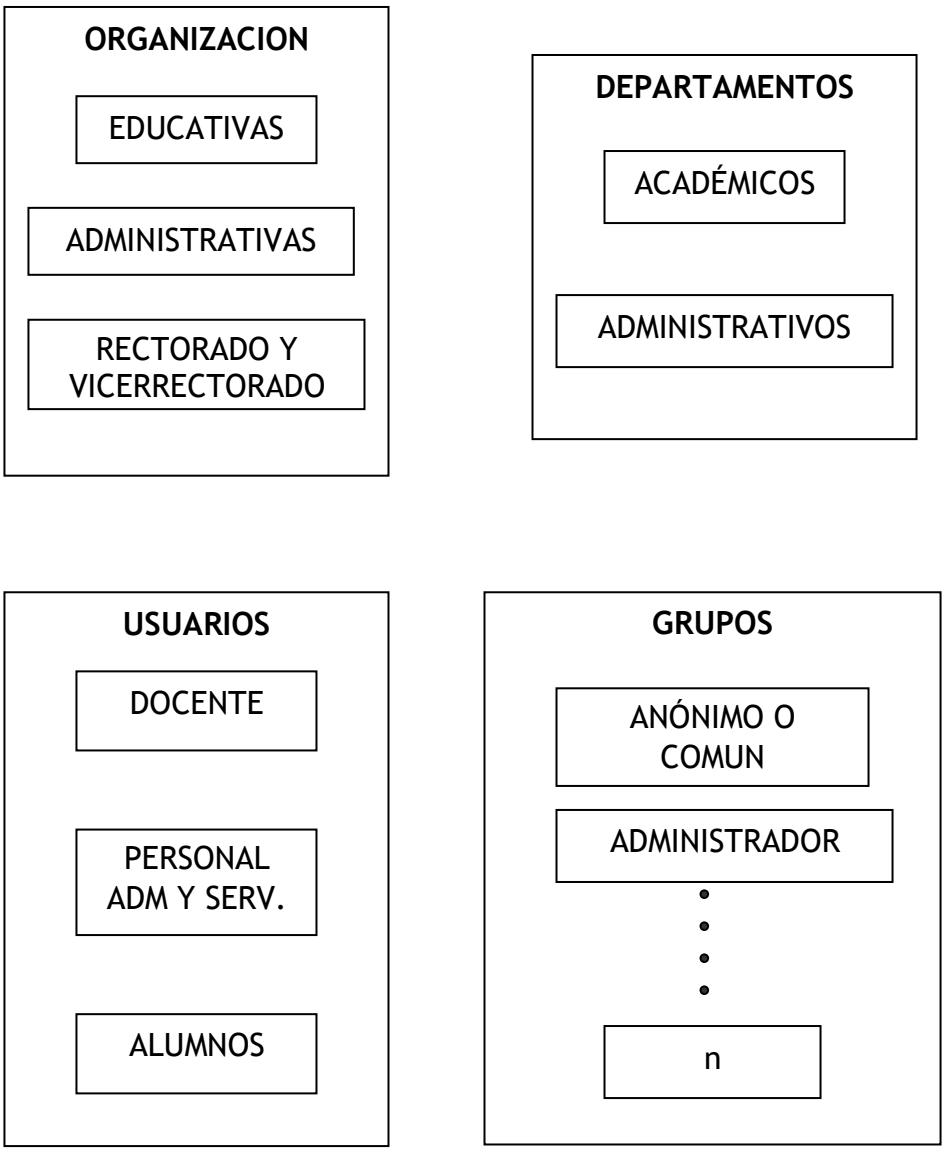


Gráfico -1-
Definición de Generalización

1.1.3 Diagrama de Relaciones de Objetos

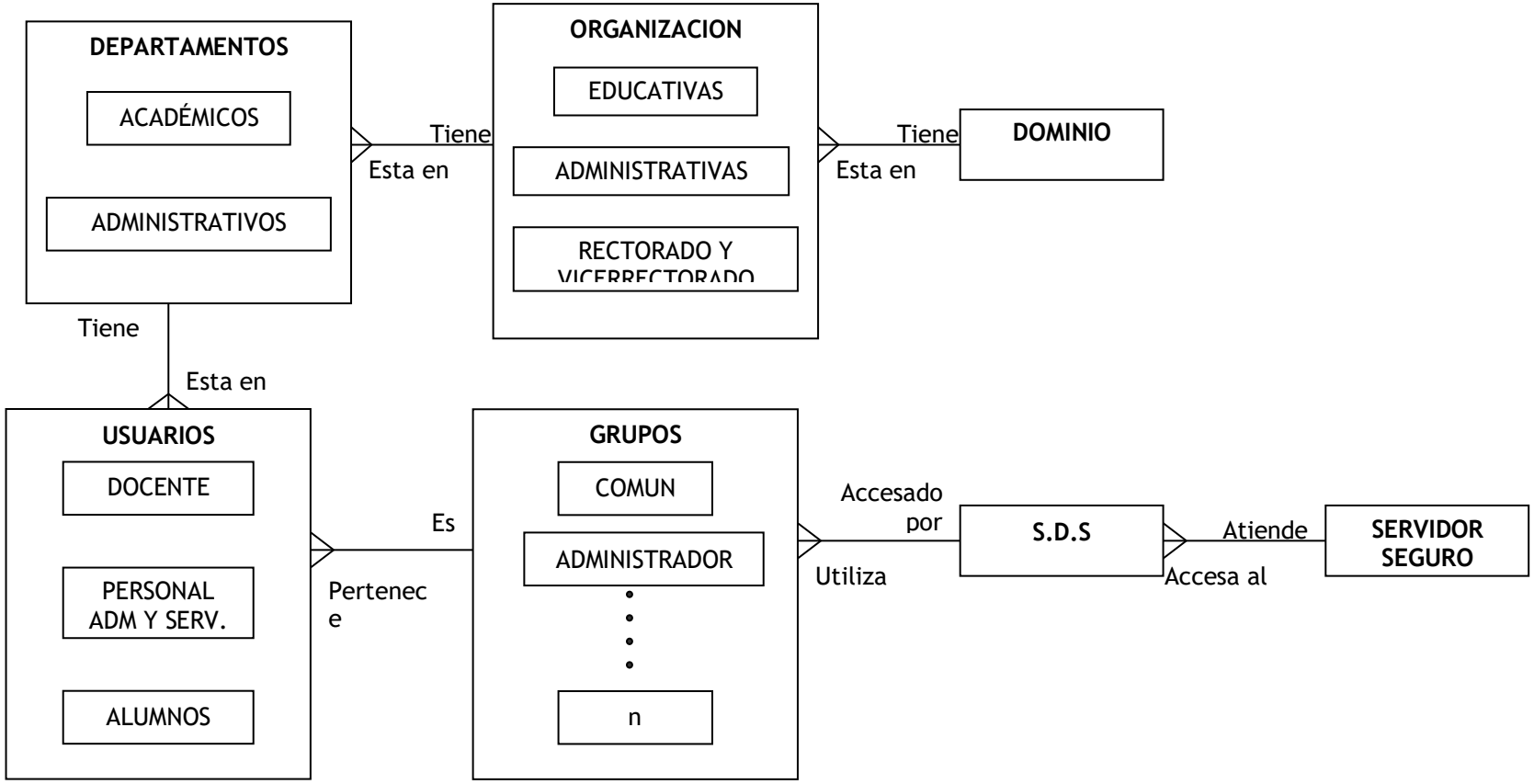
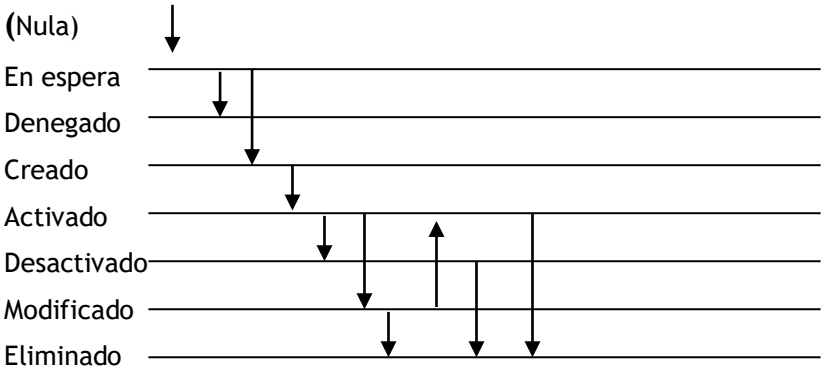


Gráfico -2-
Relación de Objetos

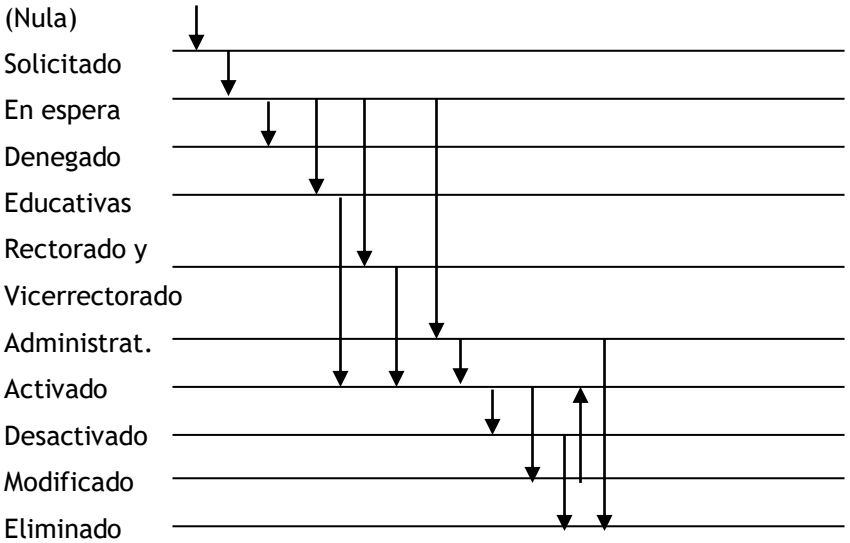
1.2 Análisis del Comportamiento de Objetos (ACO)

1.2.1 Estado de Objetos y cambio de los mismos

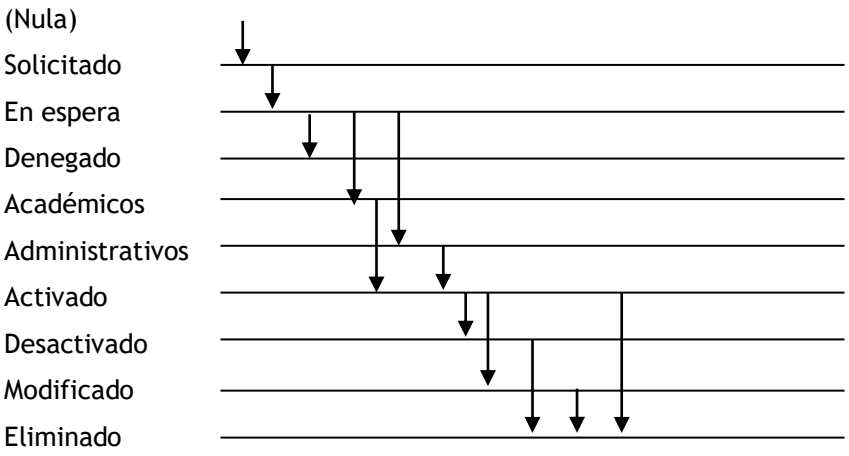
DOMINIO



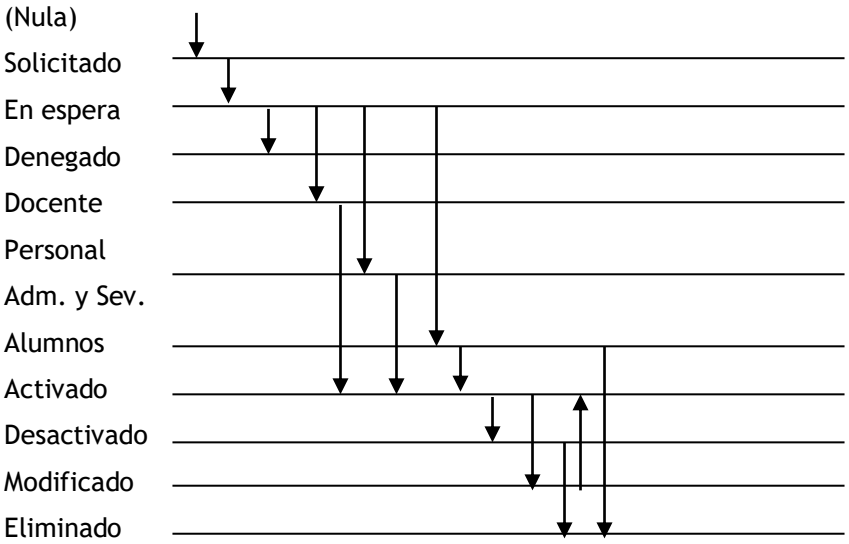
ORGANIZACION



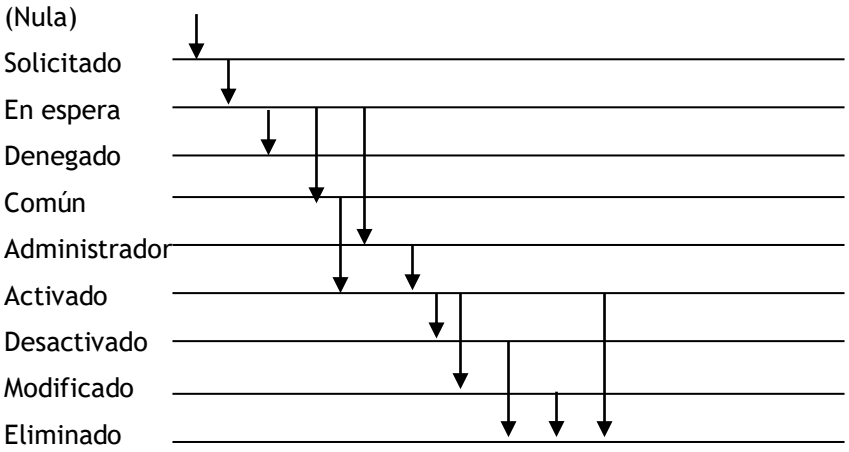
DEPARTAMENTOS



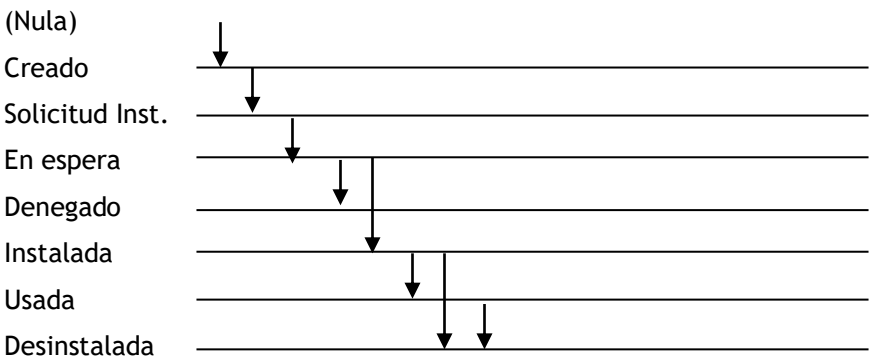
USUARIOS



GRUPOS



S.D.S.



SERVIDOR SEGURO

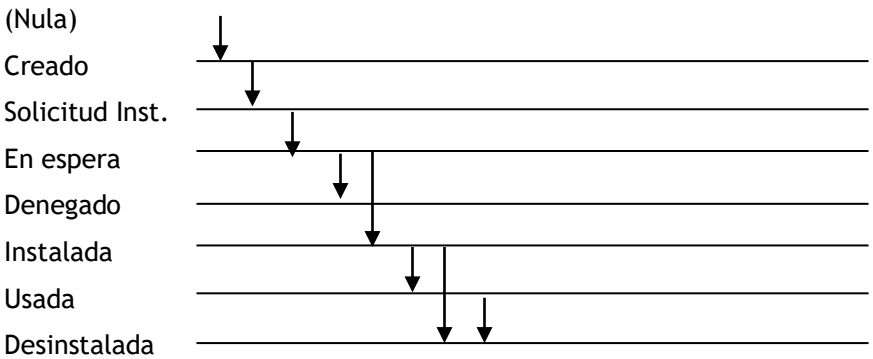


Gráfico -3-
Estados de Objetos

1.2.2 Definición de Eventos, pre estados, post estados

TIPO DE EVENTO	NOMBRE DE EVENTO	PREESTADO DEL EVENTO	POST ESTADO DEL EVENTO
Requerir	Acceso Requerido	No existe Objeto alguno	Solicitud realizada
Solicitar	Solicitud realizada	No existe Objeto alguno	Solicitud en espera
Esperar	Solicitud en análisis	Solicitud en espera	Respuesta (Aceptar/Rechazar)
Crear	Dominio creado Solicitud creada	Solicitud en espera Solicitud en espera	Dominio activado Organización activada
Crear y clasificar	Grupos creados como Común, Administrador Usuario creado como Docente, Personal Adm y Servicio, Alumnos	Solicitud aceptada Solicitud aceptada	Grupo: Común, Administrador,….n Usuario: Docente, Personal Adm. Y Serv, Alumnos
Activar	Objeto activado para consulta	Objeto creado	Objeto activado
Modificar	Objeto actualizado (Dominio, Organización, Usuario, Grupos)	Objeto Activado	Objeto Modificado
Eliminar	Objeto Eliminado (Dominio, Organización, Grupo, Usuario)	Objeto Activado/Desactivado	No existe Objeto

1.2.3 Diagrama de Operaciones (Definición de Operaciones, condiciones de activación y Resumen)

1.2.3.1 Identificar operaciones

En esta sección definimos las operaciones que se emplean en el proyecto desde la solicitud de acceso hasta las acciones que realiza la aplicación sobre el directorio.

Nº	OPERACIONES
1	Solicitar acceso
2	Revisar Acceso
3	Llenar formulario
4	Enviar solicitud de acceso
5	Análisis de solicitud
6	Creación de acceso al Sistema
7	Enviar respuesta al Cliente
8	Acepta acceso conociendo privilegios
9	Realizar operaciones
10	Digita datos de objetos
11	Verificar datos
12	Almacenar datos de forma segura
13	Digita datos de objeto a buscar
14	Buscar datos de objetos de forma segura
15	Visualizar datos de objetos
16	Modifica datos de objetos
17	Eliminar Objetos del Servidor
18	Continuar Operaciones

1.2.3.2 Definición de condiciones de activación

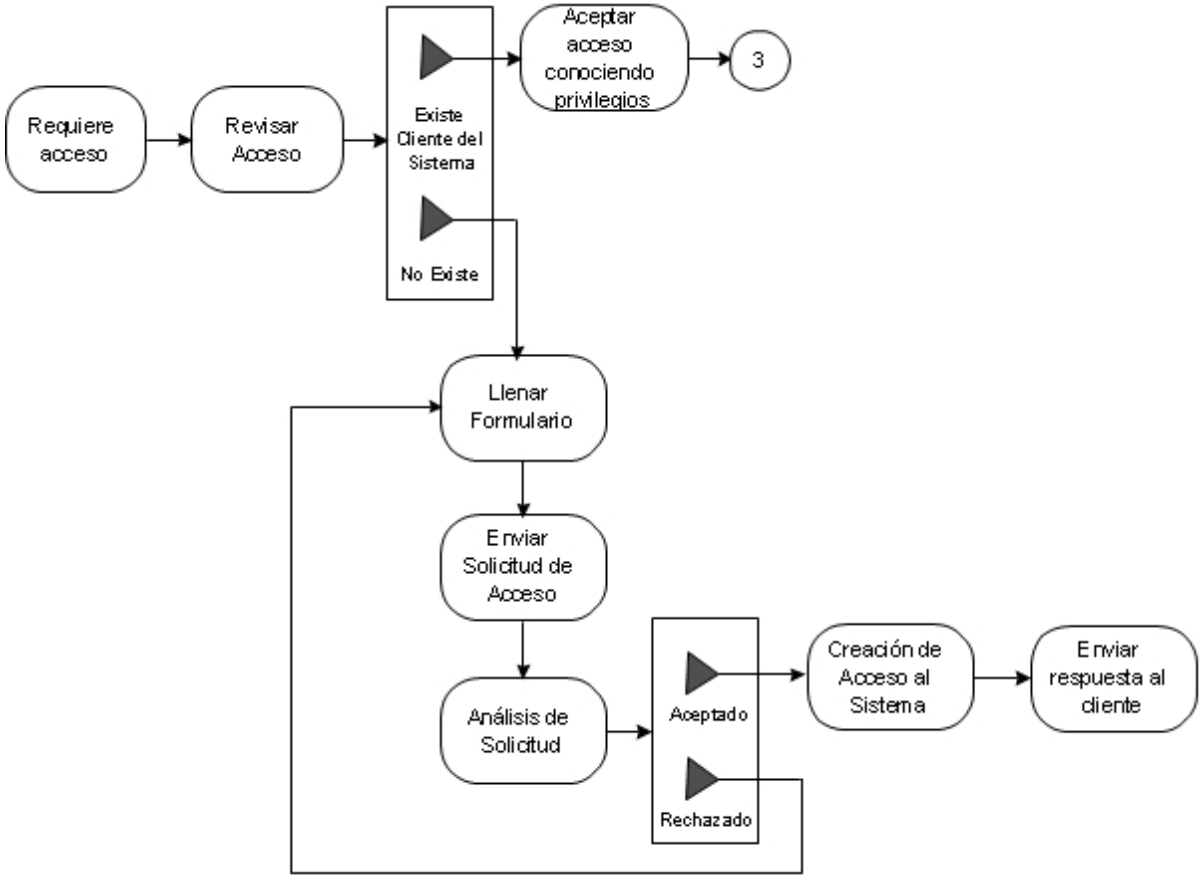
Nº	CONDICIONES DE ACTIVACIÓN
1	Si requiere acceso
2	Si no existe dominio, Usuario , Grupo y/o Solicitud Rechazada
3	Si existe formulario
4	Envio de solicitud existente
5	Análisis de solicitud aceptada
6	Si existe en el Servidor
7	Si objeto (dominio, organización, grupo, usuario) pertenece al Servidor y tiene privilegios y/o Continua Operación
8	Operación realizar Ingreso
9	Si datos digitados
10	Datos digitados correctamente
11	Operación a realizar: Otra
12	Se escribe datos a buscar
13	Si encuentran datos objetos para: Consulta
14	Si encuentran datos objetos para: Actualización
15	Si encuentran datos objetos para: Eliminación
16	Si desea continuar / No desea continuar

1.2.3.3 Resumen Eventos, Condiciones y Operaciones

EVENTOS	CONDICIONES DE ACTIVACION	OPERACIONES
-	-	Solicitar acceso
Acceso por analizar	Si requiere acceso	Revisar Acceso
Objeto no existente en Servidor	Si no existe dominio, Usuario , Grupo y/o Solicitud Rechazada	Llenar formulario
Formulario llenado	Si existe formulario	Enviar solicitud de acceso
Solicitud Realizada	Envío de solicitud existente	Análisis de solicitud
Acceso Creado	Si se crea el Acceso	Envía respuesta al cliente
Solicitud en análisis	Análisis de solicitud aceptada/ Rechazada	Creación de acceso al Sistema
Cliente del Sistema	Si existe en el Servidor	Acepta acceso conociendo privilegios
Cliente con Servicios	Si objeto (dominio, organización, grupo, usuario) pertenece al Servidor y tiene privilegios y/o Continua Operación	Realizar operaciones

EVENTOS	CONDICIONES DE ACTIVACION	OPERACIONES
No existe objeto	Operación realizar Ingreso	Digita datos de objetos
Objeto digitado	Si datos digitados	Verificar datos
Objeto correctamente digitado	Datos digitados correctamente	Almacenar datos de forma segura
Cliente con permiso	Operación a realizar: Otra	Digita datos de objeto a buscar
Objeto digitado por buscar	Se escribe datos a buscar	Busca datos de objetos de forma segura
Objeto activo	Si encuentran datos objetos para: Consulta	Visualizar datos de objetos
Objeto activo	Si encuentran datos objetos para: Actualización	Modifica datos de objetos
Objeto activo	Si encuentran datos objetos para: Eliminación	Eliminar Objetos del Servidor
Objeto activo / Objeto modificado/ Objeto eliminado / Objeto desactivado	Si desea continuar / No desea continuar	Continuar Operaciones

1.2.3.4 Diagrama Operaciones



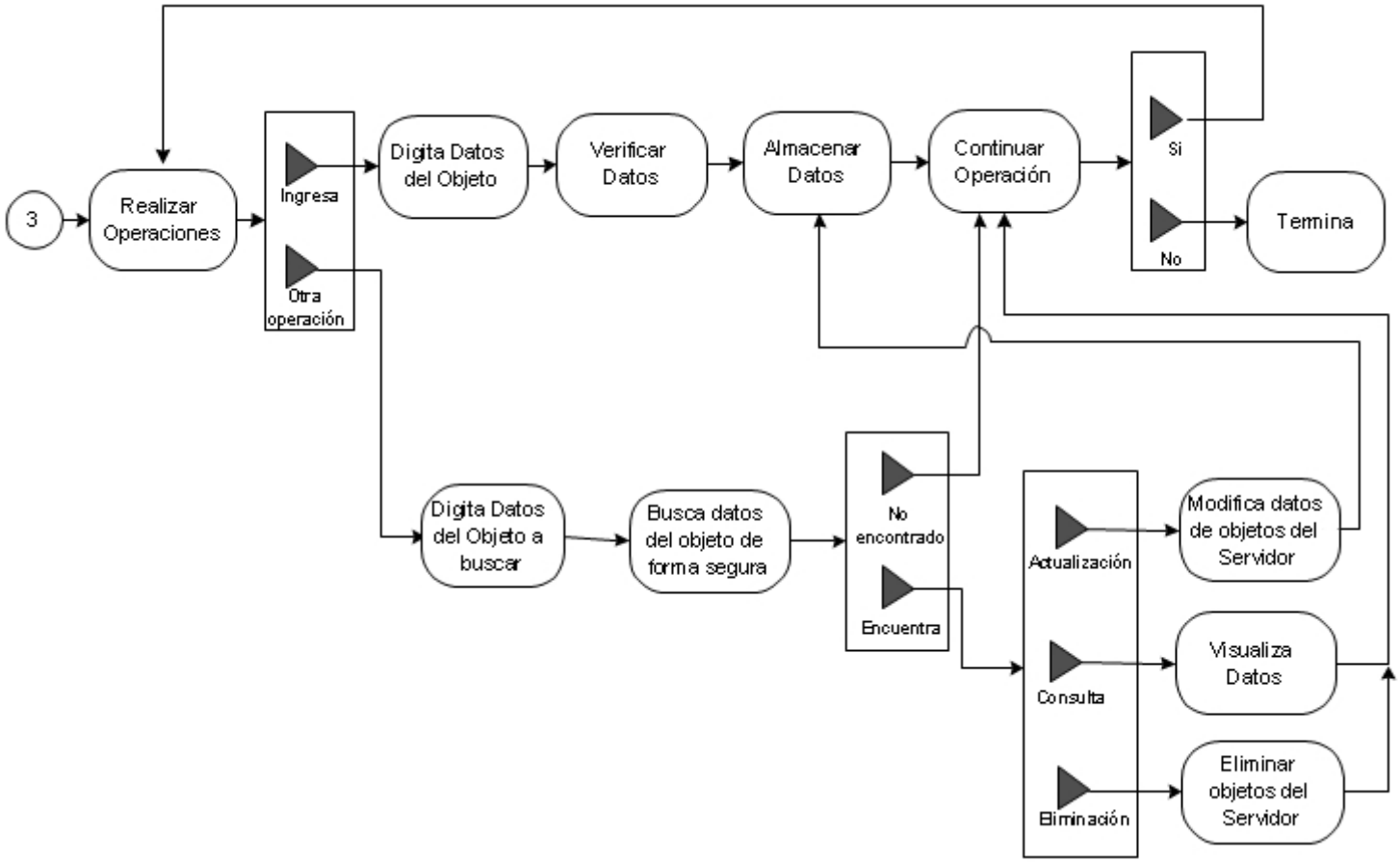
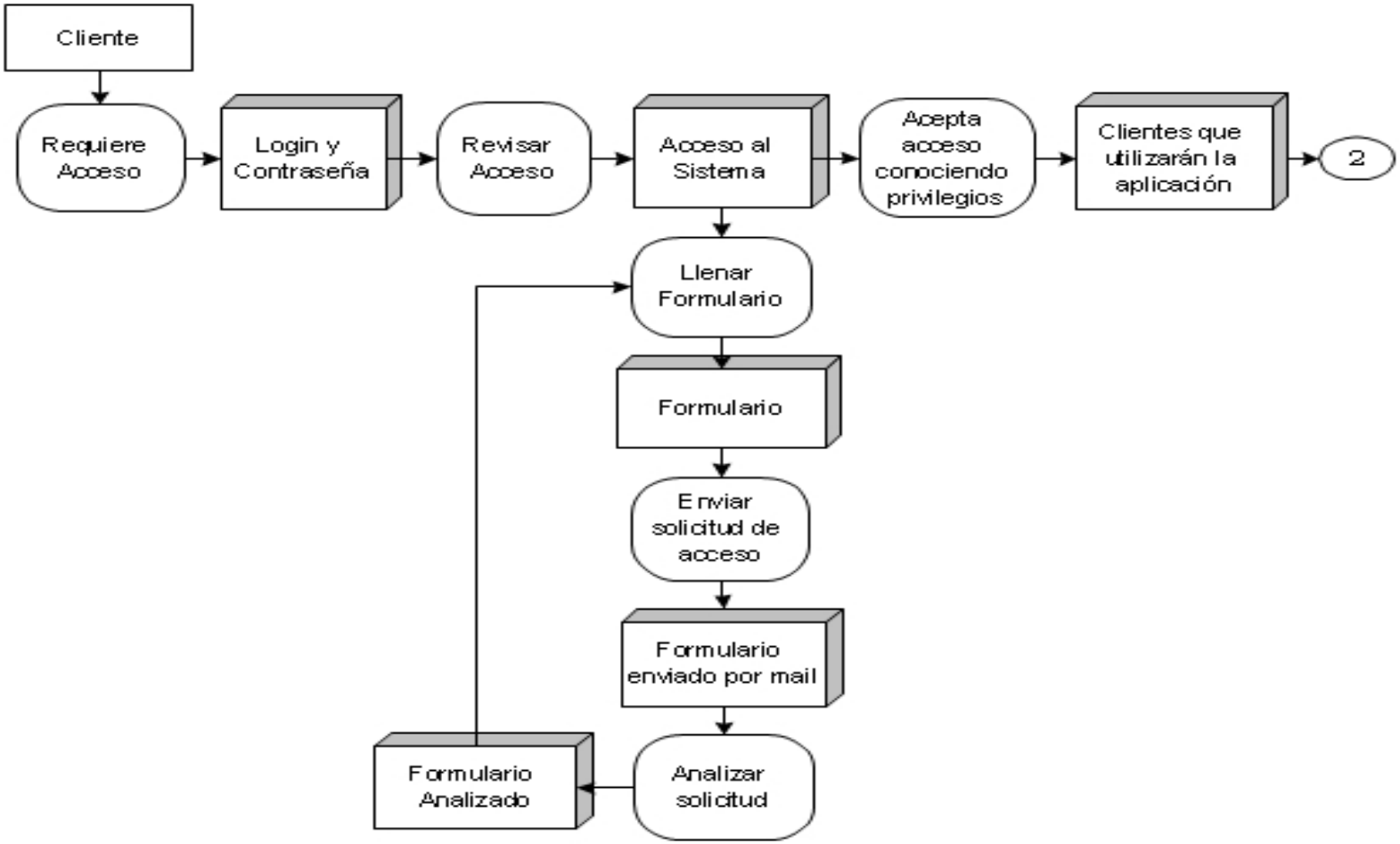


Gráfico -4-
Diagrama de Operaciones

1.2.4 Diagrama de flujo de Objetos



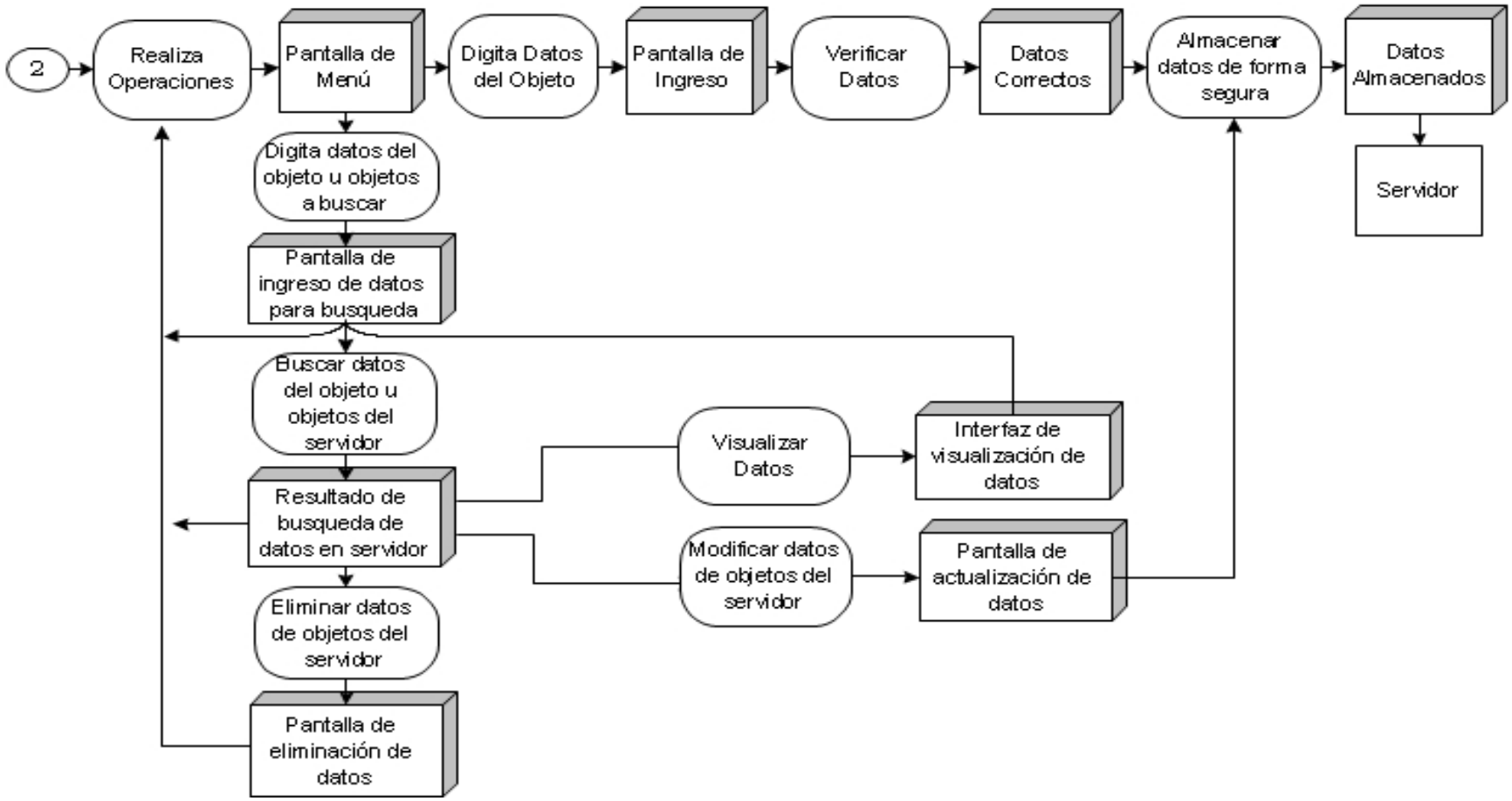
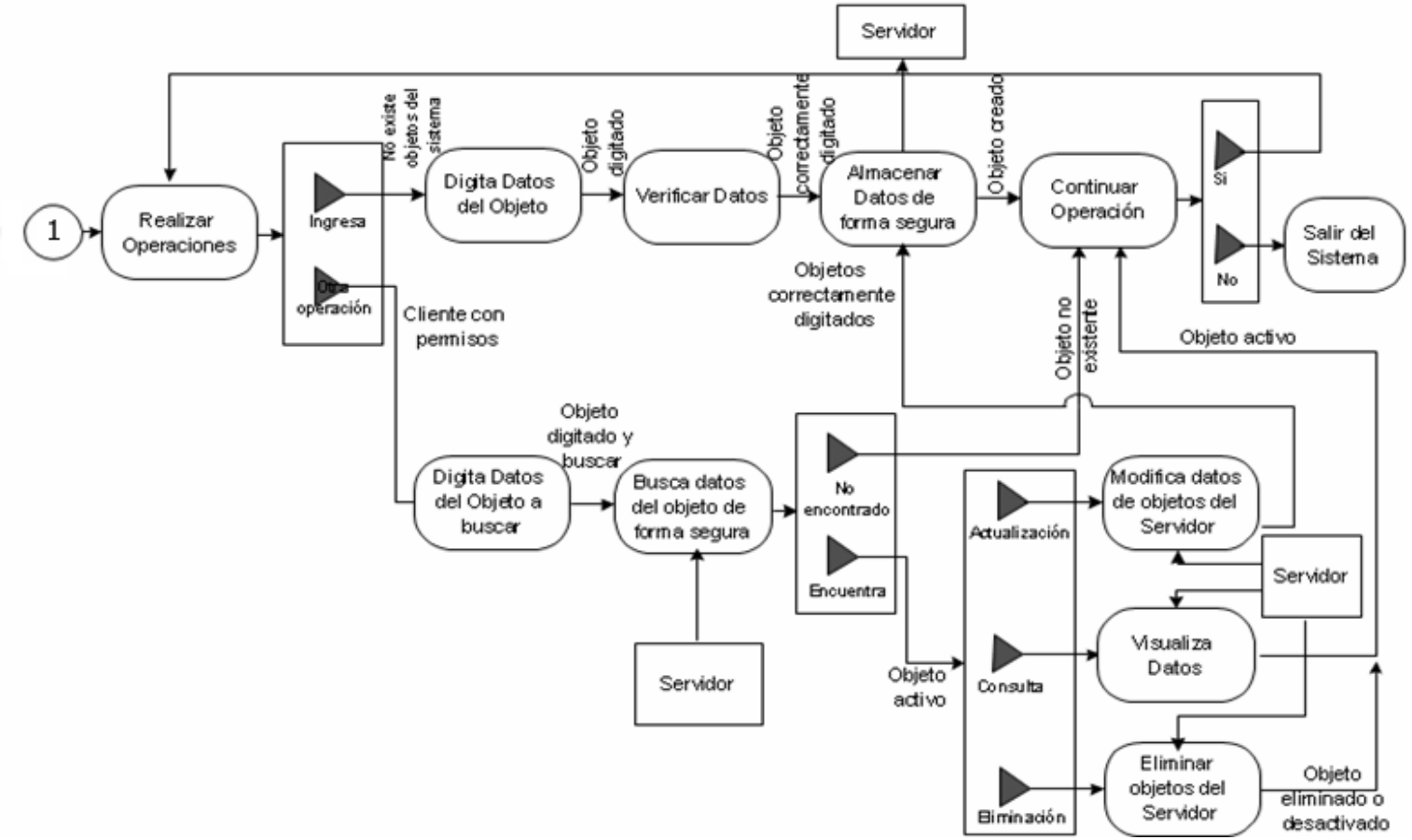


Gráfico -5-
Diagrama de Flujo de Objetos

1.2.5 Diagrama General



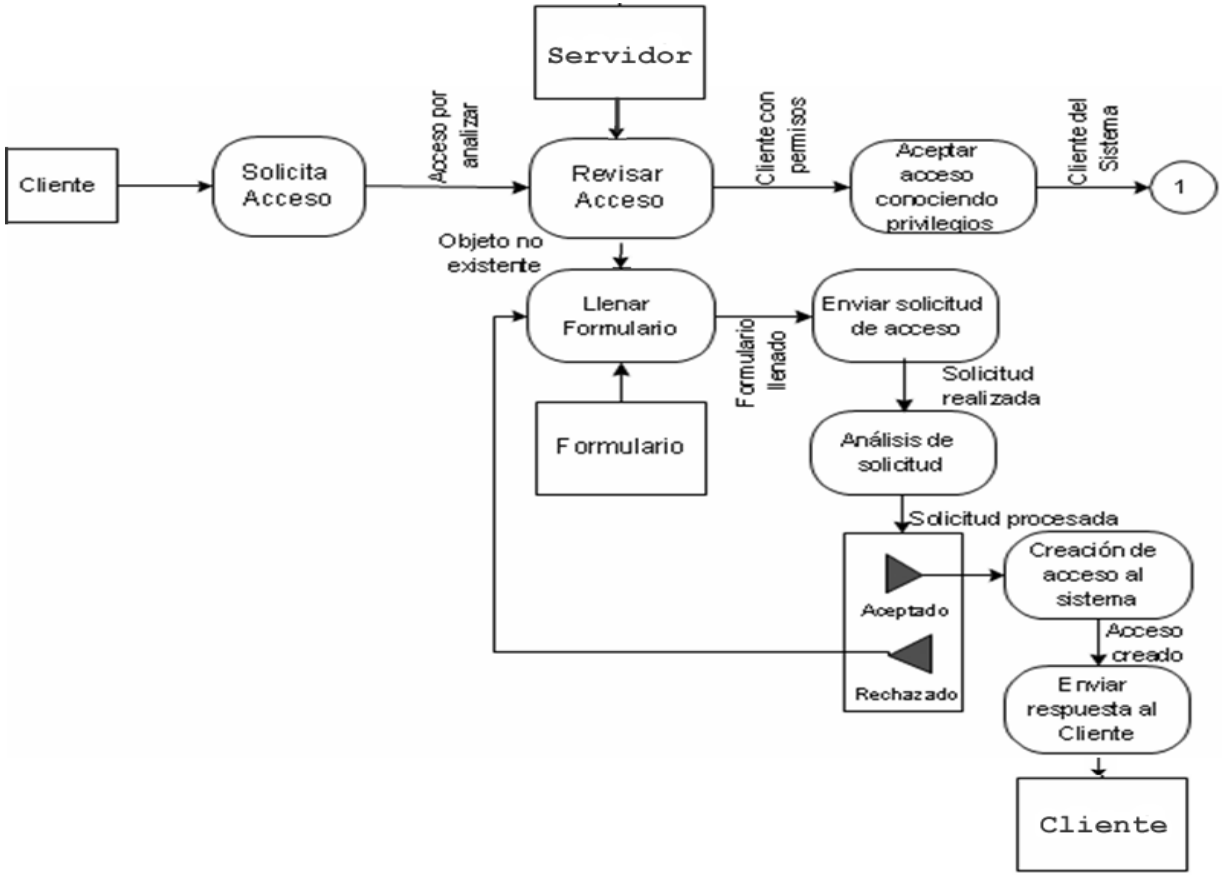


Gráfico -6-
Diagrama General

1.3. Diccionario de Datos

Clases/Archivos de Configuración	Descripción
Atributes.config	El archivo atributes.config carga las configuraciones de acción genérica de la configuración aplicable a cualquier sesión del ldap. Especifica los tipos de datos para almacenar certificados, contraseña, mail, etc.
Archivos.cfg	Son archivos que contiene información de configuración de los parámetros utilizados para la conexión con el servidor LDAP, por defecto encontramos browser.cfg la cual puede ser modificada. Cuando se crean nuevas sesiones se generan estos archivos de configuración.
BrowserApp (lbe.ui)	Clase principal mediante la cual se realiza la activación de las clases que desarrollan los componentes del árbol de directorio, menú principal y archivos de configuración, utilizando las clases **MenuManager.java** (Crea la interfaz principal: menú, tool bar, pop up). **DisplayHelpAction.java** (Crea las ayudas), **DnDDirTree.java** (llama a los componentes del árbol, se extiende de la clase **DirTree.java**).
Browser (lbe.ui)	Ubica los objetos de acuerdo a la posición. Define tamaño y dimensiones de la tabla y árbol en donde visualizaremos las entradas, atributos y valores. Permitiendo mediante la función init, levantar la ventana de conexión. Además asigna las funciones correspondientes a cada uno de los ítems del menú.

AttributeConfig (lbe.common)	Realiza el proceso de ingresos de los valores o atributos que no se encuentren definidos en la plantilla seleccionada y que se requieran añadir. Trabajando conjuntamente con la clase **EditWindow.java** que es donde se crea la ventana que contendrá las opciones de añadir atributos y valores, además permite el cambio de atributos.
DirTree (lbe.ui)	Permite la expanción y selección del árbol, ordenar entradas obtener el puerto y el host para posteriormente conectarse. Contiene funciones de creación del arbol (*createtree*), añadir un nodo (*addnode*), eliminación de un nodo (*deletenode*), mostrar los nodos hijos de las entradas (*expand*).
LdapURL (lbe.common)	Permite conocer con que dominio se conectará el cliente. Coloca el árbol de directorio en el JPanel, agregando además el scroll. En esta clase se encuentran las funciones para obtener host *(getHost)*, puerto *(getPort),* dn *(getDN)*, modificación de host, puerto y dn, guardar la configuración de acuerdo al puerto escogido.
JNDI (Java Naming and Directory Interface) (lbe.ldap)	Librería que contiene todas las funciones de manipulación del directorio permitiéndonos añadir atributos *(addAttribute)*, añadir entradas *(addEntry)*, comparaciones*(compare)*, leer atributos *(Attributesread)*, conexiones *(connect)*, desconectar *(disconnect)*, resetear conexión *(resetConnection)*, eliminar atributos *(deleteAttribute)*, eliminar entradas *(deleteEntry)*, eliminar el arbol *(deleteTree)*, buscar entradas *(findEntryName)*, modificar entradas *(modifyAttribute)*, Renombrar entrada *(renameEntry)*, actualizar atributo *(updateAttribute)*, actualizar entrada *(updateEntry)*, busquedas *(search)*, etc.
CertificateEditor2 (lbe.editor)	Permite la exhibición de los certificados **X.509**

PasswordEditor (lbe.editor)	Genera, guarda, inserta, verifica y encripta contraseñas utilizando el algoritmo de encriptación **SHA.**

1.4 Codificación de los Componentes

1.4.1 Componentes más destacados

BrowserApp.java

```
package lbe.ui;

import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
//import java.io.PrintStream;
import javax.swing.*;
import javax.swing.border.BevelBorder;
import lbe.common.*;
import lbe.interfaces.StartupListener;

// Referencia al package lbe.ui:
//            AboutWindow, Browser

public class BrowserApp
    implements StartupListener
{

    public BrowserApp()
    {
        currentProgressValue = 0;
        progressLabel = null;
        progressBar = null;
        progressLabel = new JLabel("Cargando,
espere...");
        progressLabel.setAlignmentX(0.5F);
        progressBar = new JProgressBar(0, 6);
        progressBar.setValue(0);
        progressBar.setStringPainted(true);
    }

    public void done()
    {
        processDone();
    }
```

```
    public JProgressBar getProgressBar()
    {
        return progressBar;
    }

    public JLabel getProgressLabel()
    {
        return progressLabel;
    }

    public static void main(String args[])
    {
        String vers = System.getProperty("java.version");
        if(vers.compareTo("1.1.2") < 0)
            System.out.println("!!!WARNING: Swing debe
trabajar con java 1.1.2 o version VM!!!");

        String file = null;
        String base = null;
        for(int i = 0; i < args.length; i++)
            if(args[i].equalsIgnoreCase("-f"))
            {
                if(i + 1 >= args.length)
                {
                    System.err.println("-f requiere un
argumento");
                    System.exit(1);
                }
                file = args[++i];
            } else
            if(args[i].equalsIgnoreCase("-base"))
            {
                if(i + 1 >= args.length)
                {
                    System.err.println("-base requiere un
argumento");
                    System.exit(1);
                }
                base = args[++i];
            }

        boolean rs = false;
        if(base == null)
            rs = Config.setBaseLocation("file:///" +
System.getProperty("user.dir") + File.separator);
        else
            rs = Config.setBaseLocation(base);
        if(!rs)
        {
            System.err.println("No se puede colocar base
en ese lugar. Tiene que termina con '\\' o '/'");
            System.exit(1);
        }
```

```
        if(file == null)
            Config.loadConfig(Config.defaultConfFile);
        else
            Config.loadConfig(file);

        String
lookAndFeel="com.sun.java.swing.plaf.windows.WindowsLookA
ndFeel";
        try
        {
            UIManager.setLookAndFeel(lookAndFeel);
        }
        catch(Exception ex)
        {
            System.err.println("Fallo carga L&F: " +
lookAndFeel);
            System.err.println(ex);
            System.exit(1);
        }
        Window splash = new Window(new Frame());
        JFrame frame = new JFrame("Servicio de Directorio
(LDAP)");
        frame.setForeground(Color.black);
        JOptionPane.setRootFrame(frame);
        JPanel pp = new JPanel() {

            public Insets getInsets()
            {
                return new Insets(10, 15, 10, 15);
            }

        };
        BrowserApp app = new BrowserApp();
        JLabel head = new JLabel("Servicio de Directorio
(LDAP)");
        head.setForeground(Color.black);
        head.setAlignmentX(0.5F);
        pp.setBorder(new BevelBorder(0));
        pp.setLayout(new BorderLayout());
        pp.add(head, "North");
        pp.add(app.getProgressLabel(), "Center");
        pp.add(app.getProgressBar(), "South");
        splash.add(pp);
        Dimension iSize = new Dimension(250, 100);
        splash.setSize(iSize.width, iSize.height);
        UITools.center(null, splash);
        splash.show();
        frame.setCursor(Browser.waitCursor);
        final Browser b = new Browser(app, frame);
        b.setContainer(frame);
        java.awt.event.WindowListener l = new
WindowAdapter() {

            public void windowClosing(WindowEvent e)
```

```
            {
                b.exit();
            }

        };
        frame.addWindowListener(l);
        frame.getContentPane().setLayout(new
BorderLayout());
        frame.getContentPane().add(b, "Center");
        int height =
Resource.getResourceAsInt("window.height", 370);
        int width =
Resource.getResourceAsInt("window.width", 600);
        frame.setSize(width, height);
        UITools.center(null, frame);
        frame.show();
        frame.setCursor(Browser.normCursor);
        splash.dispose();
        b.init();
    }

    private void processDone()
    {
        try
        {
            SwingUtilities.invokeLater(new Thread() {

                public void run()
                {

progressBar.setValue(++currentProgressValue);
                    progressBar.repaint();
                }

            });
        }
        catch(Exception exception) { }
    }

    private void setMessage(final String msg)
    {
        try
        {
            SwingUtilities.invokeLater(new Thread() {

                public void run()
                {
                    progressLabel.setText(msg);
                    progressLabel.repaint();
                }

            });
        }
        catch(Exception exception) { }
```

```
    }

    public void setProcess(int id)
    {
        String msg = null;
        if(id != 1)
            processDone();
        switch(id)
        {
        case 1: // '\001'
            msg = "Configuracion...";
            break;

        case 2: // '\002'
            msg = "menus...";
            break;

        case 3: // '\003'
            msg = "componentes del arbol...";
            break;

        case 4: // '\004'
            msg = "cargando templates...";
            break;

        case 5: // '\005'
            msg = "componentes de tabla...";
            break;

        case 6: // '\006'
            msg = "inicializando...";
            break;

        default:
            msg = "Inicializado..";
            break;
        }
        setMessage(msg);
    }

    private int currentProgressValue;
    private JLabel progressLabel;
    private JProgressBar progressBar;



}
```

Browser.java

```
package lbe.ui;

import java.awt.*;
import java.awt.datatransfer.*;
import java.awt.event.*;
import java.io.File;
import java.io.PrintStream;
import java.util.*;
import javax.naming.Name;
import javax.naming.directory.*;
import javax.swing.*;
import javax.swing.Timer;
import javax.swing.text.JTextComponent;
import javax.swing.tree.DefaultMutableTreeNode;
import lbe.common.*;
import lbe.interfaces.*;
import lbe.ldap.JNDI;
import lbe.ui.connect.ConnectWindow2;
import lbe.util.*;

// Referencia a clases del package lbe.ui:
//            AboutWindow, AddAttributeWindow,
AttributeTable, CustomFileFilter,
//            DeleteEntryWindow, DirTree,
DisplayHelpAction, DnDAttributeTable,
//            DnDDirTree, EditPanel, EditWindow,
HistoryWindow,
//            LDIFExportWindow, LDIFImportWindow,
PasteDNWindow, ReadThread,
//            ReferralHandler, RenameEntryWindow,
SearchWindow, TemplateAction,
//            TransferTreeWindow, ViewWindow2

public class Browser extends JPanel
    implements ActionListener, ClipboardOwner
{

      private static final long serialVersionUID = 1L;

      public Browser()
    {
        this(null, null);
    }

    public Browser(StartupListener progress, JFrame
inFrame)
    {
        clipboard = null;
        clipboardOwner = false;
        clipboardCut = false;
        sourceNode = null;
```

```
        menuManager = null;
        keyStroke = null;
        ldap = null;
        templates = null;
        aConfig = null;
        currAttrib = null;
        tree = null;
        table = null;
        history = null;
        viewWindow = null;
        editWindow = null;
        connectWindow = null;
        container = null;
        frame = null;
        propertiesMask = 0;
        rd = null;
        frame = inFrame;
        setLayout(new BorderLayout());
        if(progress != null)
            progress.setProcess(1);
        initClipboard();
        aConfig = new AttributeConfig();
        Common.fixSize = Config.getFixLocation();
        if(Debug.debugMode)
        {
            Config.printSettings();
            aConfig.print();
        }
        ldap = new JNDI();
        ldap.setConnectionHandler(new
ReferralHandler(frame));
        ldap.addErrorListener(new ErrorListener() {

            public void error(String msg, String detail)
            {
                setError(msg, detail);
            }

        });
        if(progress != null)
            progress.setProcess(2);
        menuManager = new MenuManager(this);
        DisplayHelpAction helpAction = new
DisplayHelpAction(inFrame);
        menuManager.registerAction("usage", helpAction);
        menuManager.registerAction("general",
helpAction);
        menuManager.registerAction("notes", helpAction);
        menuManager.registerAction("about", helpAction);
        javax.swing.JMenuBar mb =
menuManager.getMenuBar();
        final JPopupMenu treePopup =
menuManager.getTreePopupMenu();
```

```
        final JPopupMenu tablePopup =
menuManager.getTablePopupMenu();
        if(progress != null)
            progress.setProcess(3);
        tree = new DnDDirTree(this);
        tree.addTreeMouseListener(new MouseAdapter() {

            public void mouseClicked(MouseEvent e)
            {
                if(((e.getModifiers() & 8) != 0 ||
(e.getModifiers() & 4) != 0) && tree.isSelected())
                    treePopup.show(e.getComponent(),
e.getX(), e.getY());
            }

        });
        if(progress != null)
            progress.setProcess(4);
        templates = new Templates();
        TemplateAction templateAction = new
TemplateAction(this);
        menuManager.registerAction("Plantillas",
templateAction);
        String tt[] = templates.listOfTemplates();
        adds = menuManager.getAddEntryMenu();
        for(int i = 0; i < tt.length; i++)
        {
            JMenuItem tmp = adds.add(new
JMenuItem(tt[i]));
            tmp.addActionListener(templateAction);
        }

        mb.validate();
        if(progress != null)
            progress.setProcess(5);
        table = new DnDAttributeTable(this);
        table.addTableMouseListener(new MouseAdapter() {

            public void mouseClicked(MouseEvent e)
            {
                TreeNode2 node = tree.getSelected();
                if(node == null)
                    return;
                if((e.getModifiers() & 8) != 0 ||
(e.getModifiers() & 4) != 0)
                    tablePopup.show(e.getComponent(),
e.getX(), e.getY());
                else
                if(e.getClickCount() >= 2)
                {
                    Attributes at =
getSelected2(table.getSelectedAttributes());
                    if(at == null)
                        return;
```

```
                    if(ldap.anonymousBind())
                        view(frame, node.getURL(), at);
                    else
                        edit(frame, node.getURL(), at,
3);
                }
            }

        });
        if(progress != null)
            progress.setProcess(6);
        int height = Resource.getResourceAsInt("height",
200);
        int tableWidth =
Resource.getResourceAsInt("table.width", 250);
        int treeWidth =
Resource.getResourceAsInt("tree.width", 250);
        Dimension tableSize = new Dimension(tableWidth,
height);
        Dimension treeSize = new Dimension(treeWidth,
height);
        table.setPreferredSize(tableSize);
        tree.setPreferredSize(treeSize);
        table.setMinimumSize(tableSize);
        tree.setMinimumSize(treeSize);
        JPanel p1 = new JPanel();
        p1.setLayout(new BorderLayout());
        p1.add(status = new JTextField(30), "Center");
        status.setEditable(false);
        JSplitPane sp = new JSplitPane(1, false, tree,
table);
        sp.setOneTouchExpandable(true);
        add(mb, "North");
        JPanel top = new JPanel();
        top.setLayout(new BorderLayout());
        top.add(menuManager.getToolBar(), "North");
        top.add(sp, "Center");
        add(top, "Center");
        add(p1, "South");
        status.setText("Creado por:U.G.\\CISC\\3SG04");
        setConnectionButtons(true);
        resetMenus(false);
        history = new HistoryWindow("Error Log", 350,
200, Config.getLogSize());
        UITools.center(frame, history);
        status.addMouseListener(new MouseAdapter() {

            public void mouseClicked(MouseEvent e)
            {
                if(e.getClickCount() >= 2)
                    history.setVisible(true);
            }

        });
```

```
            rd = new ReadThread(this, ldap);
            rd.start();
            if(progress != null)
                progress.done();
        }

        public void actionPerformed(ActionEvent e)
        {
            String cmd = e.getActionCommand();
            if(cmd.equals("Conectar"))
                openConnectWindow();
            else
            if(cmd.equals("Salir"))
                exit();
            else
            if(cmd.equals("Desconectar"))
                disconnect();
            else
            if(cmd.equals("Reconectar"))
                reconnect();
            else
            if(cmd.equals("NewWindow"))
            {
                JFrame f = new JFrame();
                Browser p = new Browser(null, f);
                p.setContainer(f);
                f.getContentPane().add(p);
                f.pack();
                f.setVisible(true);
                p.init();
            } else
            if(cmd.equals("TableA"))
                table.sort(true);
            else
            if(cmd.equals("TableD"))
                table.sort(false);
            else
            if(cmd.equals("ErrorLog"))
                history.setVisible(true);
            else
            if(cmd.equals("Grabar config"))
                saveConfiguration();
            else
            if(cmd.equals("Abrir config"))
                readConfiguration();
            else
            if(cmd.equals("VerAttributes"))
            {
                AbstractButton ct =
(AbstractButton)e.getSource();
                ldap.showOpAttributes(ct.isSelected());
            } else
            if(cmd.equals("Buscar DN"))
            {
```

```
            Object val = table.getSelectedValue();
            if(val == null)
            {
                setWarning("Atributo no seleccionado.",
"Seleccionar un atributo.");
                return;
            }
            if(val instanceof String)
            {
                tree.selectDN(val.toString());
            } else
            {
                setWarning("El valor seleccionado no es
string.", "Seleccionar un valor string.");
                return;
            }
        } else
        {
            LDAPURL url = null;
            TreeNode2 selectedNode = tree.getSelected();
            if(selectedNode == null)
            {
                setWarning("Entrada no selecccionada. Por
favor seleccionar una entrada.", "");
                return;
            }
            url = selectedNode.getURL();
            if(cmd.equals("UP"))
                tree.selectParent(selectedNode);
            else
            if(cmd.equals("TreeA"))
                tree.sort(true);
            else
            if(cmd.equals("TreeD"))
                tree.sort(false);
            else
            if(cmd.equals("Eliminar Entrada"))
            {
                DeleteEntryWindow w = new
DeleteEntryWindow(ldap, tree.getSelectedNodes(), tree);
                UITools.center(frame, w);
                w.setVisible(true);
            } else
            if(cmd.equals("Ver Entrada"))
                view(frame, url, currAttrib);
            else
            if(cmd.equals("ldifE"))
            {
                LDIFExportWindow w = new
LDIFExportWindow(ldap, tree.getSelectedEntries());
                UITools.center(frame, w);
                w.setVisible(true);
            } else
            if(cmd.equals("ldifI"))
```

```
            {
                LDIFImportWindow w = new
LDIFImportWindow(ldap, url, tree);
                UITools.center(frame, w);
                w.setVisible(true);
            } else
            if(cmd.equals("Refrescar"))
                tree.rebuildNode(selectedNode);
            else
            if(cmd.equals("Agregar Atributo"))
                addAttribute(url);
            else
            if(cmd.equals("Renombrar Entrada"))
                renameEntry(selectedNode, url);
            else
            if(cmd.equals("Editar Entrada"))
                edit(frame, url, currAttrib, 3);
            else
            if(cmd.equals("Editar Atributo"))
            {
                String at[] =
table.getSelectedAttributes();
                if(testSelectedAttributes(at))
                {
                    Attributes attr = getSelected2(at);
                    if(attr != null)
                        edit(frame, url, attr, 2);
                }
            } else
            if(cmd.equals("Crear Plantilla"))
                createTemplate(url, currAttrib);
            else
            if(cmd.equals("Copiar Entrada"))
                transferTreeUI(url, false);
            else
            if(cmd.equals("Mover Entrada"))
                transferTreeUI(url, true);
            else
            if(cmd.equals("Eliminar Atributo"))
            {
                String attributes[] =
table.getSelectedAttributes();
                Object values[] =
table.getSelectedValues();
                if(testSelectedAttributes(attributes))
                    deleteAttribute(url, attributes,
values);
            } else
            if(cmd.equals("Ver Atributo"))
            {
                String attributes[] =
table.getSelectedAttributes();
                if(testSelectedAttributes(attributes))
                {
```

```
                    Attributes at =
getSelected2(attributes);
                    if(at != null)
                        view(frame, url, at);
                }
            } else
            if(cmd.equals("Buscar"))
            {
                SearchWindow w = new SearchWindow(this,
ldap, url, tree);
                UITools.center(frame, w);
                w.setVisible(true);
            } else
            if(cmd.equals("Copiar"))
                copy(url, false, null);
            else
            if(cmd.equals("Cortar"))
                copy(url, true, selectedNode);
            else
            if(cmd.equals("Pegar"))
                paste(url, selectedNode);
        }
    }

    public void addAttribute(LDAPURL url)
    {
        AddAttributeWindow w = new
AddAttributeWindow(frame);
        w.setVisible(true);
        String name = w.getAttributeName();
        String type = w.getAttributeType();
        if(name == null)
            return;
        Attributes at = new BasicAttributes(true);
        at.put(new BasicAttribute(name,
w.getActualAttributeType()));
        if(editWindow == null)
            editWindow = new EditWindow(frame, aConfig);
        editWindow.setTitle("Agregar Atributo - [" +
url.getDN() + "]");
        editWindow.init(url, at, 2);
        Attributes newAt = editWindow.getChanges();
        if(newAt == null)
            return;
        if(type.equals("binary"))
        {
            aConfig.addAttribute(name);
            aConfig.save();
            ldap.setAttributeConfig(aConfig);
        }
        Attribute a = newAt.get(name);
        if(ldap.addAttribute(url, a))
            read(url);
    }
```

```
    public void autoconnect()
    {
        tree.setSortOption(Config.getTreeSortOption());
        ldap.init(null);
        ldap.setAttributeConfig(aConfig);

menuManager.setOpAttributesCheckbox(Config.areAttributesS
et());
        if(Config.verifyHostInfo())
            connectTo(Config.getHost(), Config.getPort(),
Config.getBaseDN(), Config.useSSL());
    }

    public void busy(String msg)
    {
        setFeedback(msg);
        container.setCursor(waitCursor);
    }

    private void connectTo(String host, String port,
String baseDN, boolean ssl)
    {
        int prt = Common.toInt(port, 389);
        String urlString = LDAPURL.toUrl(host, prt,
baseDN, ssl);
        LDAPURL url = new LDAPURL(host, prt, baseDN);
        busy("Conectando a " + urlString);
        if(ldap.mainConnect(url))
        {
            resetMenus(ldap.anonymousBind() ^ true);
            setConnectionButtons(false);
            tree.createTree(url);
            setFeedback("Conectado.");
            setTitle("Servicio de Directorio (LDAP) [" +
urlString + "]");
        }
    }

    private void copy(LDAPURL url, boolean cut, TreeNode2
node)
    {
        String msg = null;
        if(clipboard == null)
        {
            setError(" No permite el acceso al sistema.",
"");
            return;
        }
        clipboardCut = false;
        if(tree.isFocused())
        {
            if(cut)
            {
```

```
                clipboardCut = true;
                sourceNode = node;
            }
            msg = CopyManager.toString(url, null, null);
        } else
        {
            msg = CopyManager.toString(url,
table.getSelectedAttributes(),
table.getSelectedValues());
            if(cut)
                cutAttributes(url,
table.getSelectedAttributes(),
table.getSelectedValues());
        }
        StringSelection data = new StringSelection(msg);
        clipboard.setContents(data, this);
        clipboardOwner = true;
    }

    public void createTemplate(LDAPURL url, Attributes
entry)
    {
        if(url == null || entry == null)
            return;
        Attribute objectClass = null;
        String prefix = null;
        String dn = url.getDN();
        objectClass = getObjectClass(entry);
        if(objectClass == null)
        {
            setError("Fallo en la creacion de la
plantilla.", "Especificaciones necesarias del
Objectclass.");
            return;
        }
        prefix = dn.substring(0, dn.indexOf("="));
        Vector names = new Vector(objectClass.size());
        try
        {
            for(Enumeration enume = objectClass.getAll();
enume.hasMoreElements();
names.addElement(enume.nextElement()));
        }
        catch(Exception _ex)
        {
            setError("Fallo valor del objectclass", "");
            return;
        }
        Object comp[] = new Object[2];
        JComboBox cb = new JComboBox(names);
        cb.setEditable(true);
        cb.setSelectedItem(names.lastElement());
        comp[0] = "Ingresar o seleccionar el nombre de la
plantilla:";
```

```
        comp[1] = cb;
        int rs = JOptionPane.showOptionDialog(frame,
((Object) (comp)), "Crear Plantilla", 2, 3, null, null,
null);
        if(rs != 0)
            return;
        String name = (String)cb.getSelectedItem();
        name = name.trim();
        if(name.length() == 0)
        {
            setError("Error Crear Plantilla: Nombre de
plantilla no es valida.", "");
            return;
        }
        if(templates.checkTemplateName(name))
        {
            setError("Error Crear Plantilla : plantillas
" + name + " ya existe.", "");
            return;
        }
        if(templates.addTemplate(name, names, prefix,
entry))
        {
            JMenuItem mu = adds.add(new JMenuItem(name));

mu.addActionListener(menuManager.getAction("Plantillas"))
;
            setFeedback("Plantillas'" + name + "' creada
y agregada.");
        } else
        {
            setError("Error al crear nueva plantilla.",
"");
        }
    }

    private void cutAttributes(LDAPURL url, String
attributes[], Object values[])
    {
        for(int i = 0; i < attributes.length; i++)
        {
            Attribute at = new
BasicAttribute(attributes[i], values[i]);
            ldap.deleteAttribute(url, at);
        }

        read(url);
    }

    public void deleteAttribute(LDAPURL url, String
attributes[], Object values[])
    {
        JCheckBox cb1 = new JCheckBox("Todos los
valores?");
```

```
        JCheckBox cb2 = new JCheckBox("Seleccionar
valor?");
        cb2.setSelected(true);
        JPanel p1 = new JPanel();
        p1.setLayout(new GridLayout(2, 1));
        p1.add(cb1);
        p1.add(cb2);
        Object msg[] = new Object[2];
        msg[1] = p1;
        ButtonGroup p = new ButtonGroup();
        p.add(cb1);
        p.add(cb2);
        if(attributes.length == 1)
            msg[0] = "Remover '" + attributes[0] + "' con
atributo :";
        else
            msg[0] = "Remover " + attributes.length + "
con atributos:";
        int rs = JOptionPane.showConfirmDialog(this,
((Object) (msg)), "Eliminar Atributo", 0);
        if(rs != 0)
            return;
        Attribute at = null;
        boolean failed = false;
        int i = 0;
        busy("Eliminando...");
        for(i = 0; i < attributes.length; i++)
        {
            if(cb1.isSelected())
                at = new BasicAttribute(attributes[i]);
            else
                at = new BasicAttribute(attributes[i],
values[i]);
            if(!ldap.deleteAttribute(url, at))
                failed = true;
        }

        if(failed)
        {
            rd.synchRead(url);
            setWarning("Errores encontrados durante la
eliminacion. Revisar el error log.", "");
        } else
        {
            read(url);
        }
    }

    public void disconnect()
    {
        ldap.disconnect();
        setConnectionButtons(true);
        resetMenus(false);
        tree.clear();
```

```
        table.clear();
        menuManager.setOpAttributesCheckbox(false);
        setTitle("Servicio de Directorio (LDAP)");
        setFeedback("Desconectado.");
    }

    public void done()
    {
        done("Listo");
    }

    public void done(String msg)
    {
        setFeedback(msg);
        container.setCursor(normCursor);
    }

    private LDAPURL url_tmp;

    public void edit(JFrame frame, LDAPURL url, final
Attributes attribs, final int mode)
    {
      url_tmp=url;
        if(attribs == null)
            return;
        if(editWindow == null)
            editWindow = new EditWindow(frame, aConfig);
        Timer t = new Timer(20, new ActionListener() {

            public void actionPerformed(ActionEvent e)
            {
                editWindow.setTitle("Editar - [" +
url_tmp.getDN() + "]");
                editWindow.init(url_tmp, attribs, mode);
                Attributes newat =
editWindow.getChanges();
                if(newat == null)
                    return;
                busy("Actualizando entrada...");
                if(ldap.updateEntry(url_tmp, newat))
                    read(url_tmp);
            }

        });
        t.setRepeats(false);
        t.start();
    }

    public void exit()
    {
        rd.stop();
        ldap.disconnect();
        if(viewWindow != null)
            viewWindow.dispose();
```

```
        if(editWindow != null)
            editWindow.dispose();
        if(history != null)
            history.dispose();
        if((propertiesMask & 0x10) == 0)
            System.exit(0);
    }

    private JFileChooser getFileChooser()
    {
        if(filechooser == null)
        {
            filechooser = new JFileChooser();
            filechooser.setFileFilter(new
CustomFileFilter(".cfg", "Archivo conf (*.cfg)"));
        }
        return filechooser;
    }

    public String getName(String dn)
    {
        return ldap.getName(dn);
    }

    private Attribute getObjectClass(Attributes entry)
    {
        String id = null;
        for(Enumeration enume = entry.getIDs();
enume.hasMoreElements();)
        {
            id = (String)enume.nextElement();
            if(id.equalsIgnoreCase("objectclass"))
                return entry.get(id);
        }

        return null;
    }

    public Attributes getSelected2(String attributes[])
    {
        int size = attributes.length;
        if(size == 0)
            return null;
        Attributes temp = new BasicAttributes(true);
        for(int i = 0; i < size; i++)
            temp.put(currAttrib.get(attributes[i]));

        return temp;
    }

    public void init()
    {
        init(".");
    }
```

```
    public void init(String dir)
    {
        if(connectWindow == null)
            connectWindow = new ConnectWindow2(frame,
dir, ldap);
        if(Config.autoconnect())
            autoconnect();
        else
        if((propertiesMask & 2) == 0)
            openConnectWindow();
    }

    private void initClipboard()
    {
        SecurityManager s = System.getSecurityManager();
        if(s != null)
            try
            {
                s.checkSystemClipboardAccess();
                clipboard =
Toolkit.getDefaultToolkit().getSystemClipboard();
            }
            catch(SecurityException _ex)
            {
                System.err.println("Unable to access
system clipboard. Internal clipboard will be used
instead.");
                clipboard = new Clipboard("myclipboard");
            }
        else
            clipboard =
Toolkit.getDefaultToolkit().getSystemClipboard();
    }

    public void list(TreeNode2 node)
    {
        busy("Retrieving...");
        if(ldap.list(node, node.getURL()))
        {
            int nentries = node.getChildCount();
            if(nentries == 0)
                done("Listo. No retorna entradas.");
            else
            if(nentries == 1)
                done("Listo. Retorna una entrada.");
            else
                done("Listo. " + nentries + " entradas
retornadas .");
        }
    }

    public void lostOwnership(Clipboard clipboard,
Transferable tr)
```

```
    {
        clipboardOwner = false;
    }

    public void openConnectWindow()
    {

connectWindow.setProperties(Config.getSettings());
        UITools.center(frame, connectWindow);
        connectWindow.setVisible(true);
        java.util.Properties props =
connectWindow.getProperties();
        if(props != null)
        {
            disconnect();
            Config.setSettings(props);
            autoconnect();
        }
    }

    public Name parseDN(String dn)
    {
        return ldap.parse(dn);
    }

    private void paste(LDAPURL url, TreeNode2 dstNode)
    {
        if(clipboard == null)
        {
            setError("No permite acceso al sistema.",
"");
            return;
        }
        Transferable clipData =
clipboard.getContents(clipboard);
        if(clipData == null)
        {
            setWarning("No pegó dato.", "");
            return;
        }
        CopyData dirData = null;
        try
        {
            String s =
(String)clipData.getTransferData(DataFlavor.stringFlavor)
;
            dirData = CopyManager.fromString(s);
        }
        catch(Exception e)
        {
            setError("Error: " + e.getMessage(), "");
            return;
        }
        if(dirData == null)
```

```
        {
            setError("Dato invalido.", "");
            return;
        }
        if(dirData.attributes == null)
        {
            if(tree.isFocused())
            {
                boolean move = false;
                TreeNode2 srcNode = null;
                if(clipboardOwner && clipboardCut)
                {
                    move = true;
                    srcNode = sourceNode;
                }
                transferTreeUI(dirData.url, srcNode, url,
dstNode, move);
            } else
            {
                String selectedAttrib =
table.getSelectedAttribute();
                if(selectedAttrib != null)
                {
                    PasteDNWindow w = new
PasteDNWindow(this, ldap, url, dirData.url,
selectedAttrib);
                    UITools.center(frame, w);
                    w.setVisible(true);
                }
            }
        } else
        {
            pasteAttribute(url, dirData.attributes);
        }
    }

    public void pasteAttribute(LDAPURL toUrl, Attributes
attributes)
    {
        Attribute at = null;
        int i = 0;
        boolean failed = false;
        boolean res = false;
        JCheckBox cb1 = new JCheckBox("Agregar
Atributos");
        JCheckBox cb2 = new JCheckBox("Reemplazar
Atributos");
        cb2.setSelected(true);
        Object msg[] = new Object[3];
        msg[0] = "Pegar opciones :";
        msg[1] = cb1;
        msg[2] = cb2;
        ButtonGroup p = new ButtonGroup();
        p.add(cb1);
```

```
        p.add(cb2);
        int rs = JOptionPane.showConfirmDialog(this,
((Object) (msg)), "Pegar Atributos", 2);
        if(rs != 0)
            return;
        for(Enumeration enume = attributes.getAll();
enume.hasMoreElements();)
        {
            at = (Attribute)enume.nextElement();
            if(cb1.isSelected())
                res = ldap.addAttribute(toUrl, at);
            else
                res = ldap.updateAttribute(toUrl, at);
            if(!res)
                failed = true;
        }

        read(toUrl);
    }

    public void read(LDAPURL url)
    {
        rd.read(url);
    }

    public void readConfiguration()
    {
        if((propertiesMask & 4) != 0)
            return;
        filechooser = getFileChooser();
        filechooser.setCurrentDirectory(new File("."));
        filechooser.setApproveButtonText("Abrir");
        filechooser.setDialogTitle("Abrir
configuracion");
        if(filechooser.showOpenDialog(this) != 0)
            return;
        File file = filechooser.getSelectedFile();
        if(Config.load(file))
        {
            setFeedback("Leer configuracion.");
            disconnect();
            init();
        } else
        {
            setError("fallo al leer archivo de
configuracion.", "");
        }
    }

    public void rebuildRootTree()
    {
        tree.rebuildRoot();
    }
```

```
    public void reconnect()
    {
        disconnect();
        autoconnect();
    }

    public void renameEntry(TreeNode2 node, LDAPURL url)
    {
        RenameEntryWindow w = new RenameEntryWindow(ldap,
url, tree, node);
        UITools.center(frame, w);
        w.setVisible(true);
    }

    public void resetCursor()
    {
        container.setCursor(normCursor);
    }

    public void resetMenus(boolean manager)
    {
        menuManager.resetMenus(manager);
        if(keyStroke == null)
            keyStroke = KeyStroke.getKeyStroke(127, 0);
        if(manager)
        {
            tree.registerKeyboardAction(this, "Eliminar
Entrada", keyStroke, 1);
            table.registerKeyboardAction(this, "Eliminar
Atributo", keyStroke, 1);
        } else
        {
            table.unregisterKeyboardAction(keyStroke);
            tree.unregisterKeyboardAction(keyStroke);
        }
    }

    public void saveConfiguration()
    {
        if((propertiesMask & 8) != 0)
            return;
        filechooser = getFileChooser();
        filechooser.setCurrentDirectory(new File("."));
        filechooser.setApproveButtonText("Guardar");
        filechooser.setDialogTitle("Guardar
configuracion");
        if(filechooser.showOpenDialog(this) != 0)
            return;
        File file = filechooser.getSelectedFile();
        String fname = file.getAbsolutePath();
        if(!fname.endsWith(".cfg"))
        {
            fname = fname + ".cfg";
            file = new File(fname);
```

```
        }
        if(Config.save(file))
            setFeedback("Configuracion grabada.");
        else
            setError("Fallo al grabar archivo de
configuracion.", "");
    }

    public void selectUrl(LDAPURL url)
    {
        tree.selectDN(url.getDN());
    }

    private void setConnectionButtons(boolean
enableConnect)
    {
        menuManager.setConnectionButtons(enableConnect);
    }

    public void setContainer(Container con)
    {
        container = con;
    }

    public void setError(String m, String e)
    {
        status.setForeground(Resource.getErrorColor());
        setStatus(m, e);
        StringBuffer buf = new StringBuffer(m);
        buf.append("\nReason: ");
        buf.append(e.toString());
        buf.append("\n");
        history.log(buf.toString());
        container.setCursor(normCursor);
        if(Config.popupErrorWindow())
            history.setVisible(true);
    }

    public void setFeedback(String m)
    {
        status.setForeground(Resource.getMsgColor());
        setStatus(m, null);
    }

    protected void setProperties(int props)
    {
        propertiesMask = props;
    }

    public void setStatus(String m, String e)
    {
        status.setText(m);
        status.setScrollOffset(0);
        status.setToolTipText(e);
```

```
    }

    public void setTitle(String title)
    {
        if(frame != null)
            frame.setTitle(title);
    }

    public void setWarning(String m, String e)
    {
        status.setForeground(Resource.getWarningColor());
        setStatus(m, e);
    }

    public void synchRead(LDAPURL url)
    {
        rd.synchRead(url);
    }

    private boolean testSelectedAttributes(String attr[])
    {
        if(attr == null || attr != null && attr.length ==
0)
        {
            setWarning("Atributo(s) no seleccionado.",
"Seleccione attribute(s) antes utilizar esta funcion.");
            return false;
        } else
        {
            return true;
        }
    }

    public void transferTreeUI(LDAPURL srcUrl, TreeNode2
srcNode, LDAPURL destUrl, TreeNode2 destNode, final
boolean move)
    {
        if(destUrl != null)
        {
            int rc = TransferTreeWindow.verify(frame,
ldap, srcUrl, destUrl);
            if(rc == 1 || rc == 2)
                return;
        }
        TransferTreeWindow w = new
TransferTreeWindow(ldap, srcUrl, destUrl, move);
        UITools.center(frame, w);
        final LDAPURL dstUrl = destUrl;
        final TreeNode2 dstNode = destNode;
        final TreeNode2 soNode = srcNode;
        w.addActionCompletedListener(new
ActionCompletedListener() {

            public void canceled()
```

```
            {
            }

            public void completed(boolean success)
            {
                if(dstUrl == null)
                    tree.rebuildRoot();
                else
                    tree.rebuildNode(dstNode);
                if(move && soNode != null)
                    tree.updateParentNode(soNode);
            }

        });
        w.setVisible(true);
    }

    public void transferTreeUI(LDAPURL srcUrl, boolean
move)
    {
        transferTreeUI(srcUrl, null, null, null, move);
    }

    public void update(Attributes attrib, Vector all, int
maxA, int maxV, LDAPURL currEntry)
    {
        currAttrib = attrib;
        table.updateTable(all, maxA, maxV, currEntry);
    }

    public void view(JFrame frame, LDAPURL url,
Attributes attribs)
    {
        if(attribs == null)
            attribs = ldap.read(url);
        if(viewWindow == null)
            viewWindow = new ViewWindow2(frame, aConfig);
        viewWindow.init(url, attribs);
        viewWindow.setVisible(true);
    }

    public static Cursor waitCursor = new Cursor(3);
    public static Cursor normCursor = new Cursor(0);
    private Clipboard clipboard;
    private boolean clipboardOwner;
    private boolean clipboardCut;
    private TreeNode2 sourceNode;
    private JTextField status;
    private JMenu adds;
    private MenuManager menuManager;
    private KeyStroke keyStroke;
    protected JNDI ldap;
    protected Templates templates;
    protected AttributeConfig aConfig;
```

```
    private Attributes currAttrib;
    protected DirTree tree;
    protected AttributeTable table;
    protected HistoryWindow history;
    protected ViewWindow2 viewWindow;
    protected EditWindow editWindow;
    protected ConnectWindow2 connectWindow;
    protected JFileChooser filechooser;
    private Container container;
    protected JFrame frame;
    private int propertiesMask;
    private ReadThread rd;
}
```

AttributeConfig

```
package lbe.common;

import java.io.*;
import java.util.Enumeration;
import java.util.Hashtable;

// Referencia clases del package lbe.common:
//            AttributeProperties, Common, Debug

public class AttributeConfig
{

    public AttributeConfig()
    {
        this(defaultConfig);
    }

    public AttributeConfig(String filename)
    {
        attrProp = null;
        changed = false;
        attrProp = read(filename);
    }

    public boolean addAttribute(String name)
    {
        if(attrProp == null)
            attrProp = new Hashtable();
        if(attrProp.get(name) == null)
        {
            attrProp.put(name, new
AttributeProperties(true));
            changed = true;
            return true;
        } else
        {
```

```
            return false;
        }
    }

    public String getBinaryList()
    {
        String attribs = new String();
        for(Enumeration e = attrProp.keys();
e.hasMoreElements();)
        {
            String key = (String)e.nextElement();
            AttributeProperties p = getProperties(key);
            if(p.isBinary())
                attribs = attribs + key + " ";
        }

        return attribs;
    }

    public AttributeProperties getProperties(String
attrib)
    {
        return
(AttributeProperties)attrProp.get(attrib.toLowerCase());
    }

    public boolean isBinary(String attrib)
    {
        AttributeProperties prop = getProperties(attrib);
        return prop != null && prop.isBinary();
    }

    public void print()
    {
        Enumeration e = attrProp.keys();
        System.out.println("# ATTRIBUTE CONFIG #");
        AttributeProperties prop;
        for(; e.hasMoreElements(); System.out.println("
(" + prop.toString() + ")"))
        {
            String attrID = (String)e.nextElement();
            prop =
(AttributeProperties)attrProp.get(attrID);
            System.out.print("Atributo: " + attrID);
        }

    }

    public Hashtable read(String filename)
    {
        Hashtable attrProp = new Hashtable();
        String editorName = null;
        String editorArgs = null;
        try
```

```
        {
            BufferedReader d = Common.openFile(filename);
            String line;
            while((line = d.readLine()) != null)
            {
                line = line.trim();
                if(line.startsWith("#") || line.length()
== 0)
                    continue;
                int index = line.indexOf(',');
                String part1;
                String part2;
                if(index == -1)
                {
                    part1 = line;
                    part2 = null;
                } else
                {
                    part1 = line.substring(0,
index).trim();
                    part2 = line.substring(index +
1).trim();
                }
                index = part1.indexOf('=');
                if(index == -1)
                {
                    Debug.error("= necesitado");
                    continue;
                }
                String attrName = part1.substring(0,
index).trim();
                String attrType = part1.substring(index +
1).trim();
                editorName = editorArgs = null;
                if(part2 != null)
                {
                    index = part2.indexOf(' ');
                    if(index == -1)
                    {
                        editorName = part2;
                        editorArgs = null;
                    } else
                    {
                        editorName = part2.substring(0,
index).trim();
                        editorArgs =
part2.substring(index + 1).trim();
                    }
                }
                AttributeProperties prop = new
AttributeProperties(editorName, editorArgs, false);
                if(attrType.startsWith("bin"))
                    prop.setBinary(true);
                else
```

```
                if(attrType.startsWith("str"))
                {
                    prop.setBinary(false);
                } else
                {
                    Debug.error("Tipo de dato
desconocido");
                    continue;
                }
                attrProp.put(attrName.toLowerCase(),
prop);
            }
        }
        catch(Exception e)
        {
            Debug.error("Error " + e.getMessage());
        }
        return attrProp;
    }

    public void save()
    {
        save(defaultConfig);
    }

    public void save(String file)
    {
        Enumeration k = attrProp.keys();
        try
        {
            PrintWriter d = Common.writeFile(file);
            for(; k.hasMoreElements(); d.println())
            {
                String name = (String)k.nextElement();
                AttributeProperties prop =
(AttributeProperties)attrProp.get(name);
                d.print(name + "=");
                if(prop.isBinary())
                    d.print("binary");
                else
                    d.print("string");
                if(prop.getEditorName() != null)
                {
                    d.print("," + prop.getEditorName());
                    if(prop.getEditorArgs() != null)
                        d.print(" " +
prop.getEditorArgs());
                }
            }

            d.close();
        }
        catch(Exception e)
        {
```

```
            Debug.error("Error en AttriuteConfig: " +
e.getMessage());
        }
    }

    public static String defaultConfig =
"attributes.config";
    private Hashtable attrProp;
    private boolean changed;

}
```

EditWindow.java

```
package lbe.ui;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.naming.directory.Attributes;
import javax.swing.*;
import lbe.common.*;

// Referencia clases del package lbe.ui:
//            EditPanel, ViewPanel2

public class EditWindow extends JDialog
    implements ActionListener
{

    public EditWindow(JFrame frame, AttributeConfig
attrProp)
    {
        super(frame, "Editar", true);
        newAttribs = null;
        this.frame = frame;
        getContentPane().setLayout(new BorderLayout());
        JPanel p1 = new JPanel();
        applyButton = new JButton("Aplicar");
        applyButton.addActionListener(this);
        cancelButton = new JButton("Cancelar");
        cancelButton.addActionListener(this);
        p1.add(applyButton);
        p1.add(cancelButton);
        editPanel = new EditPanel(frame, attrProp);
        getContentPane().add(initMenuBar(), "North");
        getContentPane().add(editPanel, "Center");
        getContentPane().add(p1, "South");
    }

    public void actionPerformed(ActionEvent e)
```

```
    {
        String cmd = e.getActionCommand();
        if(cmd.equals("Eliminar valor"))
            editPanel.deleteSelectedValue();
        else
        if(cmd.equals("Añadir Atributo"))
            editPanel.addNewAttribute();
        else
        if(cmd.equals("Añadir Valor"))
            editPanel.addValueToSelected();
        else
        if(cmd.equals("Aplicar"))
        {
            newAttribs = editPanel.getChanges();
            setVisible(false);
        } else
        if(cmd.equals("Cancelar"))
        {
            newAttribs = null;
            setVisible(false);
        } else
        if(cmd.equals("addnoneempty"))
            editPanel.addNoneEmptyOnly =
addNoneEmptyAttribCB.isSelected();
    }

    public Attributes getChanges()
    {
        return newAttribs;
    }

    public void init(LDAPURL url, Attributes attrs, int
mode)
    {
        editPanel.init(url, attrs, mode);
        pack();
        String label = mode != 1 ? "Actualizar" :
"Agregar";
        addNoneEmptyAttribCB.setText(label + " non-empty
attributes only");
        addNoneEmptyAttribCB.setSelected(true);
        editPanel.addNoneEmptyOnly =
addNoneEmptyAttribCB.isSelected();
        int height = 0;
        Dimension sbSize = editPanel.getScrollBarSize();
        if(sbSize != null)
            height = sbSize.height;
        Common.fixSize(this);
        Dimension newSize = getSize();
        newSize.height += height;
        setSize(newSize);
        UITools.center(frame, this);
        setVisible(true);
    }
```

```
    private JMenuBar initMenuBar()
    {
        JMenuBar mb = new JMenuBar();
        JMenu file = new JMenu("Archivo");
        file.add(newMenuItem("Aplicar", 'A', null));
        file.add(newMenuItem("Cancelar", 'C', null));
        mb.add(file);
        JMenu edit = new JMenu("Editar");
        edit.add(newMenuItem("Añadir Valor", 'A', null));
        edit.add(newMenuItem("Eliminar valor", 'E',
null));
        edit.add(newMenuItem("Añadir Atributo", 'D',
null));
        edit.add(new JSeparator());
        addNoneEmptyAttribCB =
newCheckboxMenuItem("Añadir solo atributos no
inicializados", 't', "addnoneempty");
        edit.add(addNoneEmptyAttribCB);
        mb.add(edit);
        return mb;
    }

    public JCheckBoxMenuItem newCheckboxMenuItem(String
label, char men, String cmd)
    {
        JCheckBoxMenuItem tmp = new
JCheckBoxMenuItem(label);
        tmp.setMnemonic(men);
        tmp.addActionListener(this);
        if(cmd != null)
            tmp.setActionCommand(cmd);
        return tmp;
    }

    public JMenuItem newMenuItem(String label, char men,
String cmd)
    {
        JMenuItem tmp = new JMenuItem(label);
        tmp.setMnemonic(men);
        tmp.addActionListener(this);
        if(cmd != null)
            tmp.setActionCommand(cmd);
        return tmp;
    }

    public void setTitle(String title)
    {
        super.setTitle(title);
    }

    private EditPanel editPanel;
    private JButton applyButton;
    private JButton cancelButton;
```

```
    private JCheckBoxMenuItem addNoneEmptyAttribCB;
    private AttributeConfig attrProp;
    private Attributes newAttribs;
    private JFrame frame;
}
```

DirTree.java

```
public class DirTree extends JPanel
    implements TreeExpansionListener,
TreeSelectionListener
{

    public DirTree(Browser p)
    {
        super(true);
        sortOption = 0;
        parent = p;
        root = new TreeNode2("root");
        treeModel = new DefaultTreeModel(root);
        tree = new JTree(treeModel);
        tree.putClientProperty("JTree.lineStyle",
"Angled");
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setPreferredSize(new Dimension(200,
200));
        scrollPane.getViewport().add(tree);
        setLayout(new GridLayout(1, 0));
        add(scrollPane);
        tree.setRootVisible(false);
    }

    public void addNode(LDAPURL url)
    {
        TreePath path = tree.getSelectionPath();
        TreeNode2 selected = getSelectedNode(path);
        TreeNode2 first = null;
        if(selected.isUpdated())
        {
            String name = parent.getName(url.getDN());
            first = new TreeNode2(name, url);
            first.add(new TreeNode2("nothing here"));
            selected.add(first);
            treeModel.reload(selected);
            TreePath p = path.pathByAddingChild(first);
            tree.setSelectionPath(p);
            tree.scrollPathToVisible(p);
        } else
```

```
        {
            expand(selected);
            selectNode(url);
        }
    }

    public void addTreeMouseListener(MouseAdapter e)
    {
        tree.addMouseListener(e);
    }

    public void clear()
    {
        enableListeners(false);
        root.removeAllChildren();
        treeModel.reload(root);
        tree.setRootVisible(false);
        tree.clearSelection();
    }

    public void createTree(LDAPURL url)
    {
        root.setUrl(url);
        String rootName = url.getDN();
        if(rootName.length() == 0)
            rootName = "Root DSE";
        root.setUserObject(rootName);
        tree.setRootVisible(true);
        parent.list(root);
        sort(root);
        treeModel.reload(root);
        enableListeners(true);
    }

    public void deleteNode(TreeNode2 node)
    {
        tree.removeTreeSelectionListener(this);
        treeModel.removeNodeFromParent(node);
        tree.addTreeSelectionListener(this);
    }

    private void enableListeners(boolean enable)
    {
        if(enable)
        {
            tree.addTreeExpansionListener(this);
            tree.addTreeSelectionListener(this);
        } else
        {
            tree.removeTreeExpansionListener(this);
            tree.removeTreeSelectionListener(this);
        }
    }
```

```
    public void expand(TreeNode2 tmp)
    {
        tmp.removeAllChildren();
        parent.list(tmp);
        sort(tmp);
        treeModel.reload(tmp);
    }

    private String getName(Name name, int index)
    {
        String nm = name.get(index);
        return nm.trim();
    }

    public TreeNode2 getSelected()
    {
        TreePath path = tree.getSelectionPath();
        if(path == null)
            return null;
        else
            return
(TreeNode2)path.getLastPathComponent();
    }

    public LDAPURL[] getSelectedEntries()
    {
        TreePath path[] = tree.getSelectionPaths();
        if(path == null)
            return null;
        LDAPURL urls[] = new LDAPURL[path.length];
        for(int i = 0; i < path.length; i++)
        {
            TreeNode2 tmp =
(TreeNode2)path[i].getLastPathComponent();
            urls[i] = tmp.getURL();
        }

        return urls;
    }

    private TreeNode2 getSelectedNode(TreePath path)
    {
        return (TreeNode2)path.getLastPathComponent();
    }

    public TreeNode2[] getSelectedNodes()
    {
        TreePath path[] = tree.getSelectionPaths();
        if(path == null)
            return null;
        TreeNode2 nodes[] = new TreeNode2[path.length];
        for(int i = 0; i < path.length; i++)
            nodes[i] =
(TreeNode2)path[i].getLastPathComponent();
```

```
        return nodes;
    }

    public boolean isFocused()
    {
        return tree.hasFocus();
    }

    public boolean isSelected()
    {
        return tree.getSelectionCount() != 0;
    }

    public void rebuildNode(TreeNode2 node)
    {
        enableListeners(false);
        expand(node);
        enableListeners(true);
    }

    public void rebuildRoot()
    {
        enableListeners(false);
        expand(root);
        enableListeners(true);
    }

    public void selectDN(String dn)
    {
        Name name = parent.parseDN(dn);
        if(name == null)
        {
            parent.setWarning("El valor seleccionado es
probablemente invalido DN", "");
            return;
        }
        int crsize = name.size();
        if(crsize == 0)
        {
            parent.setWarning("Zero length DN", "");
            return;
        }
        Name rt = parent.parseDN(root.getURL().getDN());
        int rtsize = rt.size();
        if(crsize < rtsize)
        {
            parent.setWarning("No hay suficientes
componentes", "");
            return;
        }
        String rtName = null;
        String crName = null;
        for(int i = 0; i < rtsize; i++)
```

```
        {
            rtName = getName(rt, i);
            crName = getName(name, i);
            if(!rtName.equalsIgnoreCase(crName))
            {
                parent.setWarning("No en este árbol",
"");
                return;
            }
        }

        TreePath t = new TreePath(root);
        if(rtsize == crsize)
            tree.setSelectionPath(t);
        else
            selectNode(t, name, rtsize);
    }

    public void selectNode(TreePath path, Name name, int
index)
    {
        TreeNode2 tmp =
(TreeNode2)path.getLastPathComponent();
        Name crNode = null;
        String crName = null;
        LDAPURL url = null;
        String selName = getName(name, index);
        for(int i = 0; i < tmp.getChildCount(); i++)
        {
            TreeNode2 t = (TreeNode2)tmp.getChildAt(i);
            url = t.getURL();
            crNode = parent.parseDN(url.getDN());
            if(crNode == null)
                continue;
            crName = getName(crNode, crNode.size() - 1);
            Debug.debug("revisando: " + crName + " " +
selName);
            if(!crName.equalsIgnoreCase(selName))
                continue;
            TreePath pa = path.pathByAddingChild(t);
            if(index + 1 == name.size())
            {
                tree.setSelectionPath(pa);
                tree.scrollPathToVisible(pa);
            } else
            {
                if(tree.isCollapsed(pa))
                    tree.expandPath(pa);
                selectNode(pa, name, index + 1);
            }
            break;
        }

    }
```

```
    public void selectNode(LDAPURL url)
    {
        selectDN(url.getDN());
    }

    public void selectParent(TreeNode2 node)
    {
        TreeNode2 parent = node.getParentNode();
        if(parent != null)
        {
            javax.swing.tree.TreeNode nodes[] =
parent.getPath();
            TreePath path = new TreePath(nodes);
            tree.setSelectionPath(path);
            tree.scrollPathToVisible(path);
        }
    }

    public void setSortOption(int sortType)
    {
        sortOption = sortType;
    }

    private void sort(TreeNode2 node)
    {
        switch(sortOption)
        {
        case 1: // '\001'
            node.sort(false);
            break;

        case 2: // '\002'
            node.sort(true);
            break;
        }
    }

    public void sort(boolean ascending)
    {
        TreeNode2 node = getSelected();
        if(node == null)
        {
            return;
        } else
        {
            node.sort(ascending ^ true);
            treeModel.reload(node);
            return;
        }
    }

    public void treeCollapsed(TreeExpansionEvent
treeexpansionevent)
```

```
    {
    }

    public void treeExpanded(TreeExpansionEvent event)
    {
        TreeNode2 tmp =
(TreeNode2)event.getPath().getLastPathComponent();
        TreeNode2 t = (TreeNode2)tmp.getFirstChild();
        if(!t.isUpdated())
            expand(tmp);
    }

    public void updateParentNode(TreeNode2 node)
    {
        enableListeners(false);
        tree.clearSelection();
        TreeNode2 pr = (TreeNode2)node.getParent();
        expand(pr);
        enableListeners(true);
    }

    public void valueChanged(TreeSelectionEvent e)
    {
        TreeNode2 tmp =
(TreeNode2)e.getPath().getLastPathComponent();
        parent.read(tmp.getURL());
    }

    private int sortOption;
    protected JTree tree;
    private TreeNode2 root;
    protected Browser parent;
    protected DefaultTreeModel treeModel;
}
```

DnDDirTree.java

```
package lbe.ui;

import java.awt.Point;
import java.awt.datatransfer.*;
import java.awt.dnd.*;
import java.net.MalformedURLException;
import javax.swing.*;
import javax.swing.tree.TreePath;
import lbe.common.*;
import lbe.interfaces.ActionCompletedListener;
import lbe.util.FinishDragThread;

// Referencia clases del of package lbe.ui:
//            DirTree, Browser, TransferTreeWindow

public class DnDDirTree extends DirTree
    implements DragGestureListener, DropTargetListener,
DragSourceListener
{

    public DnDDirTree(Browser p)
    {
        super(p);
        dragSource = null;
        dragSource = DragSource.getDefaultDragSource();
        DragGestureRecognizer dgr =
dragSource.createDefaultDragGestureRecognizer(super.tree,
3, this);
        dgr.setSourceActions(dgr.getSourceActions() & -
5);
        DropTarget dropTarget = new
DropTarget(super.tree, this);
    }

    public void dragDropEnd(DragSourceDropEvent event)
    {
        if(event.getDropSuccess())
        {
            int action = event.getDropAction();
            if(action == 2 &&
!super.parent.ldap.anonymousBind())
            {

super.tree.removeTreeSelectionListener(this);

super.treeModel.removeNodeFromParent(sourceNode);

super.tree.addTreeSelectionListener(this);
            }
        }
    }
```

```
    public void dragEnter(DragSourceDragEvent
dragsourcedragevent)
    {
    }

    public void dragEnter(DropTargetDragEvent
droptargetdragevent)
    {
    }

    public void dragExit(DragSourceEvent dragsourceevent)
    {
    }

    public void dragExit(DropTargetEvent droptargetevent)
    {
    }

    public void dragGestureRecognized(DragGestureEvent
dragGestureEvent)
    {
        sourceNode = null;
        TreePath path = super.tree.getSelectionPath();
        if(path == null)
            return;
        TreeNode2 node =
(TreeNode2)path.getLastPathComponent();
        String op = null;
        if(super.parent.ldap.anonymousBind() &&
dragGestureEvent.getDragAction() == 2)
            op = "c;";
        else
            op = "-;";
        StringSelection selection = new
StringSelection(op + node.getURL().getUrl());
        sourceNode = node;
        dragSource.startDrag(dragGestureEvent, null,
selection, this);
    }

    public void dragOver(DragSourceDragEvent
dragsourcedragevent)
    {
    }

    public void dragOver(DropTargetDragEvent
droptargetdragevent)
    {
    }

    public synchronized void drop(DropTargetDropEvent
event)
    {
```

```
        if(super.parent.ldap.anonymousBind())
        {
            event.rejectDrop();
            SwingUtilities.invokeLater(new Runnable() {

                public void run()
                {
                    String msg = "Deberia conectarse como
un administrador para realizar actualizaciones.";
                    JOptionPane.showMessageDialog(parent,
msg, "DnD Error", 0);
                }

            });
            return;
        }
        try
        {
            Transferable tr = event.getTransferable();

if(tr.isDataFlavorSupported(DataFlavor.stringFlavor))
            {
                event.acceptDrop(event.getDropAction());
                Object data =
tr.getTransferData(DataFlavor.stringFlavor);
                handleDropData(event, data.toString());
            } else
            {
                event.rejectDrop();
            }
        }
        catch(Exception e)
        {
            event.rejectDrop();
        }
    }

    public void dropActionChanged(DragSourceDragEvent
dragsourcedragevent)
    {
    }

    public void dropActionChanged(DropTargetDragEvent
droptargetdragevent)
    {
    }

    private void handleDropData(final DropTargetDropEvent
event, String data)
    {
        Point p = event.getLocation();
        TreePath path =
super.tree.getPathForLocation(p.x, p.y);
        boolean complete = false;
```

```
        if(path == null)
        {
            Thread t = new Thread(new
FinishDragThread(event, false));
            SwingUtilities.invokeLater(t);
            return;
        }
        final TreeNode2 node =
(TreeNode2)path.getLastPathComponent();
        LDAPURL destUrl = node.getURL();
        LDAPURL srcUrl = null;
        try
        {
            srcUrl = new LDAPURL(data.substring(2));
        }
        catch(MalformedURLException e)
        {
            Debug.error("Invalido url: " + e.getMessage()
+ " " + data);
            Thread t = new Thread(new
FinishDragThread(event, false));
            SwingUtilities.invokeLater(t);
            return;
        }
        boolean move = false;
        if(data.charAt(0) != 'c')
            move = event.getDropAction() == 2;
        int rc =
TransferTreeWindow.verify(super.parent.frame,
super.parent.ldap, srcUrl, destUrl);
        if(rc == 1 || rc == 2)
        {
            Thread t = new Thread(new
FinishDragThread(event, false));
            SwingUtilities.invokeLater(t);
            return;
        } else
        {
            TransferTreeWindow w = new
TransferTreeWindow(super.parent.ldap, srcUrl, destUrl,
move);
            UITools.center(super.parent.frame, w);
            w.addActionCompletedListener(new
ActionCompletedListener() {

                public void canceled()
                {
                    Thread t = new Thread(new
FinishDragThread(event, false));
                    SwingUtilities.invokeLater(t);
                }

                public void completed(boolean success)
                {
```

```
                    if(success)
                        expand(node);
                    Thread t = new Thread(new
FinishDragThread(event, success));
                    SwingUtilities.invokeLater(t);
                }

            });
            w.setVisible(true);
            return;
        }
    }

    private DragSource dragSource;
    private TreeNode2 sourceNode;
}
```

LdapURL.java

```
/ permite la conexion con el servidor
package lbe.common;

import java.net.MalformedURLException;
import java.net.URLDecoder;

public class LDAPURL
{

    public LDAPURL()
    {
        host = dn = base = null;
        port = 389;
    }

    public LDAPURL(String url)
        throws MalformedURLException
    {
        try
        {
            url = URLDecoder.decode(url);
        }
        catch(Exception e)
        {
            throw new
MalformedURLException(e.getMessage());
        }
        int p1 = url.indexOf("://");
        if(p1 == -1)
            throw new MalformedURLException("necesitando
'[protocol]://'");
        String protocol = url.substring(0, p1);
        p1 += 3;
        int p2 = url.indexOf('/', p1);
        String base = null;
        if(p2 == -1)
        {
            base = url.substring(p1);
            parseHostPort(base);
            dn = "";
        } else
        {
            base = url.substring(p1, p2);
            p2++;
            dn = url.substring(p2);
            int p3 = dn.indexOf('?');
            if(p3 != -1)
                dn = dn.substring(0, p3);
            parseHostPort(base);
        }
    }

    public LDAPURL(String host, int port, String dn)
```

```
    {
        this.host = host;
        this.port = port;
        this.dn = dn;
    }

    public static String encode(String toEncode)
    {
        StringBuffer encoded = new
StringBuffer(toEncode.length() + 10);
        for(int currPos = 0; currPos < toEncode.length();
currPos++)
        {
            char currChar = toEncode.charAt(currPos);
            if(currChar >= 'a' && currChar <= 'z' ||
currChar >= 'A' && currChar <= 'Z' || currChar >= '0' &&
currChar <= '9' || "$-_.+!*'(),".indexOf(currChar) > 0)
            {
                encoded.append(currChar);
            } else
            {
                encoded.append("%");
                encoded.append(hexChar((currChar & 0xf0)
>> 4));
                encoded.append(hexChar(currChar & 0xf));
            }
        }

        return encoded.toString();
    }

    public String getBase()
    {
        if(base == null)
            base = "ldap://" + host + ":" + port;
        return base;
    }

    public String getDN()
    {
        return dn;
    }

    public String getEncodedUrl()
    {
        return getBase() + "/" + encode(dn);
    }

    public String getHost()
    {
        return host;
    }

    public int getPort()
```

```
    {
        return port;
    }

    public String getUrl()
    {
        return getBase() + "/" + dn;
    }

    private static char hexChar(int hexValue)
    {
        if(hexValue < 0 || hexValue > 15)
            return 'x';
        if(hexValue < 10)
            return (char)(hexValue + 48);
        else
            return (char)((hexValue - 10) + 97);
    }

    private void parseHostPort(String str)
        throws MalformedURLException
    {
        int p1 = str.indexOf(':');
        if(p1 == -1)
        {
            host = str;
            port = 389;
        } else
        {
            host = str.substring(0, p1);
            String pp = str.substring(p1 + 1);
            try
            {
                port = Integer.parseInt(pp);
            }
            catch(NumberFormatException _ex)
            {
                throw new MalformedURLException("Invalid
port number: " + pp);
            }
        }
    }

    public boolean sameHosts(LDAPURL url)
    {
        return getHost().equalsIgnoreCase(url.getHost())
&& getPort() == url.getPort();
    }

    public void setDN(String dn)
    {
        this.dn = dn;
    }
```

```
    public void setHost(String host)
    {
        this.host = host;
        base = null;
    }

    public void setPort(int port)
    {
        this.port = port;
        base = null;
    }

    public static String toUrl(String host, int port,
String dn, boolean ssl)
    {
        StringBuffer msg = new StringBuffer();
        msg.append(ssl ? "ldaps://" : "ldap://");
        msg.append(host);
        if(ssl && port != 636 || !ssl && port != 389)
        {
            msg.append(":");
            msg.append(String.valueOf(port));
        }
        msg.append("/");
        msg.append(dn);
        return msg.toString();
    }

    private String host;
    private int port;
    private String dn;
    private String base;
```

JNDI.java

```
package lbe.ldap;

import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;
//import javax.swing.tree.DefaultMutableTreeNode;
import lbe.common.*;
import lbe.interfaces.*;

public class JNDI
{

    public JNDI()
    {
        connections = new Hashtable();
        connector = null;
        limit = 0;
        timeout = 0;
        version = 2;
        anonymousBind = true;
        indicatorType = 0;
        indicatorAttr = null;
        listAttrs = null;
        listFilter = null;
        attributesList = null;
        parser = null;
        showOpAttributes = false;
        listener = null;
        env = new Properties();
        env.put("java.naming.factory.initial",
DEFAULT_CTX);
    }

    public boolean addAttribute(LDAPURL url, Attribute
at)
    {
        try
        {
            ModificationItem mods[] = new
ModificationItem[1];
            mods[0] = new ModificationItem(1, at);
            return modifyAttribute(url, mods);
        }
        catch(NamingException e)
        {
            error("Fallo al agregar'" + at.getID() + "'
atributo por " + url.getUrl(), e);
        }
        return false;
    }
```

```
    public boolean addEntry(LDAPURL url, Attributes at)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        try
        {
            ctx.createSubcontext(url.getDN(), at);
        }
        catch(ReferralException e)
        {
            LDAPURL myurl = getReferralUrl(e);
            return addEntry(myurl, at);
        }
        catch(NamingException e)
        {
            error("Fallo al agregar nueva entraeda " +
url.getDN(), e);
            return false;
        }
        return true;
    }

    public void addErrorListener(ErrorListener ls)
    {
        listener = ls;
    }

    public boolean anonymousBind()
    {
        return anonymousBind;
    }

    private void checkLeaf(TreeNode2 node, Attributes
attrs)
    {
        Attribute n = null;
        TreeNode2 more = new TreeNode2("nothing here");
        if(indicatorAttr == null || attrs == null)
        {
            node.add(more);
            return;
        }
        n = attrs.get(indicatorAttr[0]);
        if(n == null)
        {
            node.add(more);
            return;
        }
        try
        {
            if(indicatorType == 0)
            {
```

```
                int i =
Integer.parseInt(n.get().toString());
                if(i > 0)
                    node.add(more);
            } else
            if(indicatorType == 1)
            {
                Boolean b = new
Boolean(n.get().toString());
                if(b.booleanValue())
                    node.add(more);
            } else
            {
                node.add(more);
            }
        }
        catch(NamingException _ex)
        {
            node.add(more);
        }
    }

    public int compare(LDAPURL srcUrl, LDAPURL dstUrl)
    {
        if(!srcUrl.sameHosts(dstUrl))
            return 0;
        Name src = parse(srcUrl.getDN());
        Name dst = parse(dstUrl.getDN());
        if(dst.compareTo(src) == 0)
            return 1;
        if(dst.startsWith(src))
            return 2;
        Name prefix = src.getPrefix(src.size() - 1);
        return dst.compareTo(prefix) != 0 ? 0 : 3;
    }

    public DirContext connect(LDAPURL url)
    {
        String base = url.getBase();
        DirContext ctx =
(DirContext)connections.get(base);
        if(ctx != null)
            return ctx;
        setDefaultEnv();
        env.put("java.naming.provider.url", base);
        do
        {
            try
            {
                ctx = new InitialDirContext(env);
                connections.put(base, ctx);
                return ctx;
            }
            catch(AuthenticationException e)
```

```
            {
                error("Error de autenticacion: " + base,
e);
                if(connector == null)
                    return null;
                Properties pr =
connector.referralConnection(env, url, anonymousBind());
                if(pr != null)
                {
                    env = pr;
                    continue;
                }
            }
            catch(CommunicationException e)
            {
                error("Error de comunicacion: " + base,
e);
                if(connector == null)
                    return null;
                if(connector.connectionFailed(url))
                {
                    resetConnection(url);
                    continue;
                }
            }
            catch(NamingException e)
            {
                error("Fallo al conectarse a " + base,
e);
            }
            return ctx;
        } while(true);
    }

    public boolean deleteAttribute(LDAPURL url, Attribute
at)
    {
        try
        {
            ModificationItem mods[] = new
ModificationItem[1];
            mods[0] = new ModificationItem(3, at);
            return modifyAttribute(url, mods);
        }
        catch(NamingException e)
        {
            error("Fallo al eliminar '" + at.getID() + "'
atributo por " + url.getUrl(), e);
        }
        return false;
    }

    public boolean deleteEntry(LDAPURL url)
    {
```

```
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        try
        {
            ctx.destroySubcontext(url.getDN());
        }
        catch(ReferralException e)
        {
            LDAPURL myurl = getReferralUrl(e);
            return deleteEntry(myurl);
        }
        catch(NamingException e)
        {
            error("Fallo al eliminar entrada " +
url.getDN(), e);
            return false;
        }
        return true;
    }

    public boolean deleteTree(LDAPURL url, Cancelable
cancelable, ProgressListener listener)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        String entryDN = null;
        LDAPURL myurl = null;
        String baseDN = url.getDN();
        try
        {
            for(NamingEnumeration results = search(ctx,
baseDN, "(objectclass=*)", DEFAULT_ATTR, 1, false);
results.hasMore();)
            {
                if(cancelable != null &&
cancelable.isCanceled())
                {
                    results.close();
                    return false;
                }
                SearchResult si =
(SearchResult)results.next();
                entryDN = getFixedDN(si.getName(),
baseDN);
                myurl = new LDAPURL(url.getHost(),
url.getPort(), entryDN);
                if(!deleteTree(myurl, cancelable,
listener))
                {
                    results.close();
                    return false;
                }
```

```
            }

        }
        catch(NamingException e)
        {
            error("Fallo al eliminar árbol", e);
            return false;
        }
        if(listener != null)
            listener.msg(entryDN, null);
        return deleteEntry(url);
    }

    public boolean disconnect()
    {
        DirContext ctx = null;
        for(Enumeration enume = connections.elements();
enume.hasMoreElements();)
            try
            {
                ctx = (DirContext)enume.nextElement();
                ctx.close();
            }
            catch(NamingException e)
            {
                error("fallo desconexion", e);
            }

        connections.clear();
        return true;
    }

    public void error(String msg, Exception e)
    {
        String detail;
        if(e == null)
            detail = null;
        else
            detail = e.getMessage();
        if(listener != null)
            listener.error(msg, detail);
    }

    public boolean exists(LDAPURL url)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        try
        {
            NamingEnumeration results = search(ctx,
url.getDN(), "(objectclass=*)", DEFAULT_ATTR, 0, false);
            return true;
        }
```

```
        catch(NameNotFoundException _ex)
        {
            return false;
        }
        catch(NamingException _ex)
        {
            return false;
        }
    }

    public boolean export(LDAPURL url, String filter,
String attributes[], int scope, Cancelable cancelable,
Exportable exporter)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        String entryDN = null;
        String baseDN = url.getDN();
        boolean moreReferrals = true;
        while(moreReferrals)
            try
            {
                LDAPURL myurl;
                SearchResult si;
                for(NamingEnumeration results =
search(ctx, baseDN, filter, attributes, scope, false);
results.hasMore(); exporter.export(myurl,
si.getAttributes()))
                {
                    if(cancelable.isCanceled())
                    {
                        results.close();
                        return false;
                    }
                    si = (SearchResult)results.next();
                    entryDN = getFixedDN(si.getName(),
baseDN);
                    myurl = new LDAPURL(url.getHost(),
url.getPort(), entryDN);
                }

                moreReferrals = false;
            }
            catch(ReferralException e)
            {
                LDAPURL myurl = getReferralUrl(e);
                int subscope = scope != 1 ? scope : 0;
                boolean rs = export(myurl, filter,
attributes, subscope, cancelable, exporter);
                if(!rs)
                    return rs;
                moreReferrals = e.skipReferral();
                try
```

```
                {
                    ctx =
(DirContext)e.getReferralContext();
                }
                catch(NamingException _ex) { }
            }
            catch(NamingException e)
            {
                error("Fallo durante exportacion ´", e);
                return false;
            }
        return true;
    }

    public LDAPURL findEntryName(LDAPURL url)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return null;
        Name name = parse(url.getDN());
        String base = name.getPrefix(name.size() -
1).toString();
        String dn = url.getDN();
        String rdn = name.get(name.size() -
1).toString();
        int i = 1;
        boolean foundName = true;
        while(foundName)
            try
            {
                NamingEnumeration results = search(ctx,
dn, "(objectclass=*)", DEFAULT_ATTR, 0, false);
                results.close();
                if(i == 1)
                    rdn = rdn + " copy";
                else
                if(i == 2)
                    rdn = rdn + " " + i;
                else
                if(i >= 3)
                    rdn = rdn.substring(0, rdn.length() -
1) + i;
                dn = rdn + ", " + base;
                i++;
            }
            catch(NameNotFoundException _ex)
            {
                foundName = false;
                return new LDAPURL(url.getHost(),
url.getPort(), dn);
            }
            catch(NamingException _ex)
            {
                return null;
```

```
            }
        return null;
    }

    private String fixName(String name)
    {
        if(name.length() > 0 && name.charAt(0) == '"')
        {
            int size = name.length() - 1;
            StringBuffer buf = new StringBuffer();
            for(int i = 1; i < size; i++)
            {
                if(name.charAt(i) == '/')
                    buf.append("\\");
                buf.append(name.charAt(i));
            }

            return buf.toString();
        } else
        {
            return name;
        }
    }

    private String getDN(String rdn, String base)
    {
        if(rdn.length() == 0)
            return base;
        if(base.length() == 0)
            return rdn;
        else
            return rdn + ", " + base;
    }

    public LDAPURL getDestination(LDAPURL fromUrl,
LDAPURL toUrl, boolean rename)
    {
        if(rename)
        {
            return toUrl;
        } else
        {
            String newDN = getName(fromUrl.getDN()) + ",
" + toUrl.getDN();
            return new LDAPURL(toUrl.getHost(),
toUrl.getPort(), newDN);
        }
    }

    private String getFixedDN(String rdn, String base)
    {
        return getDN(fixName(rdn), base);
    }
```

```
    public String getName(String dn)
    {
        try
        {
            Name nm = parser.parse(dn);
            return nm.get(nm.size() - 1).toString();
        }
        catch(NamingException _ex)
        {
            return null;
        }
    }

    public LDAPURL getReferralUrl(ReferralException e)
    {
        String url = (String)e.getReferralInfo();
        try
        {
            return new LDAPURL(url);
        }
        catch(Exception ex)
        {
            Debug.error("Invalid url: " + ex.getMessage()
+ " " + url);
        }
        return null;
    }

    public int importEntry(LDAPURL url, String dn,
Attributes entry, int type)
    {
        boolean rs = false;
        LDAPURL myurl = new LDAPURL(url.getHost(),
url.getPort(), dn);
        if(type == 0)
            rs = addEntry(myurl, entry);
        else
        if(type == 1)
            rs = updateEntry(myurl, entry);
        else
        if(type == 2)
            rs = synchEntry(myurl, entry);
        else
            return 0;
        return !rs ? -1 : 1;
    }

    public void init(Config config)
    {
    }

    public boolean list(TreeNode2 node, LDAPURL url)
    {
        DirContext ctx = connect(url);
```

```
        if(ctx == null)
            return false;
        String name = null;
        TreeNode2 t = null;
        LDAPURL myurl = null;
        String baseDN = url.getDN();
        String entryDN = null;
        boolean moreReferrals = true;
        while(moreReferrals)
            try
            {
                for(NamingEnumeration results =
search(ctx, baseDN, listFilter, listAttrs, 1);
results.hasMore(); node.add(t))
                {
                    SearchResult si =
(SearchResult)results.next();
                    name = fixName(si.getName());
                    entryDN = getDN(name, baseDN);
                    myurl = new LDAPURL(url.getHost(),
url.getPort(), entryDN);
                    t = new TreeNode2(name, myurl);
                    checkLeaf(t, si.getAttributes());
                }

                moreReferrals = false;
            }
            catch(ReferralException e)
            {
                myurl = getReferralUrl(e);
                if(myurl.getDN().length() == 0)
                {
                    myurl.setDN(baseDN);
                    name = url.getDN();
                } else
                {
                    name = getName(myurl.getDN());
                }
                name = name + " [" + myurl.getHost() +
":" + myurl.getPort() + "]";
                t = new TreeNode2(name, myurl);
                checkLeaf(t, null);
                node.add(t);
                moreReferrals = e.skipReferral();
                try
                {
                    ctx =
(DirContext)e.getReferralContext();
                }
                catch(NamingException _ex) { }
            }
            catch(NamingException e)
            {
                error("Fallo lista", e);
```

```
                return false;
            }
        return true;
    }

    public boolean mainConnect(LDAPURL url)
    {
        setDefaultEnv();
        String base = url.getBase();
        env.put("java.naming.provider.url", base);
        try
        {
            DirContext ctx = new InitialDirContext(env);
            connections.put(base, ctx);
            if(version == 3 && indicatorAttr != null)
                listAttrs = indicatorAttr;
            else
                listAttrs = DEFAULT_ATTR;
            if(parser == null)
                parser = ctx.getNameParser("");
            return true;
        }
        catch(NamingException e)
        {
            error("Fallo al concectar a " + base, e);
        }
        return false;
    }

    private boolean modifyAttribute(LDAPURL url,
ModificationItem mods[])
        throws NamingException
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        try
        {
            ctx.modifyAttributes(url.getDN(), mods);
        }
        catch(ReferralException e)
        {
            LDAPURL myurl = getReferralUrl(e);
            return modifyAttribute(myurl, mods);
        }
        return true;
    }

    public Name parse(String dn)
    {
        try
        {
            return parser.parse(dn);
        }
```

```
            catch(NamingException _ex)
            {
                return null;
            }
        }

        public Attributes read(LDAPURL url)
        {
            DirContext ctx = connect(url);
            if(ctx == null)
                return null;
            Attributes attrs = null;
            try
            {
                if(showOpAttributes)
                    attrs = ctx.getAttributes(url.getDN(),
attributesList);
                else
                    attrs = ctx.getAttributes(url.getDN());
            }
            catch(ReferralException e)
            {
                LDAPURL myurl = getReferralUrl(e);
                if(myurl.getDN().length() == 0)
                    myurl.setDN(url.getDN());
                return read(myurl);
            }
            catch(CommunicationException e)
            {
                if(connector == null)
                {
                    error("Error de comunicacion : " +
url.getBase(), e);
                    return null;
                }
                if(connector.connectionFailed(url))
                    resetConnection(url);
            }
            catch(NamingException e)
            {
                error("Fallo al leer entrada " + url.getDN(),
e);
                return null;
            }
            return attrs;
        }

        public boolean renameEntry(LDAPURL url, String newDN)
        {
            DirContext ctx = connect(url);
            if(ctx == null)
                return false;
            try
            {
```

```
            ctx.rename(url.getDN(), newDN);
        }
        catch(ReferralException e)
        {
            error("Fallo al renonbrar entrada (no
soportado por referrals)", e);
            return false;
        }
        catch(NamingException e)
        {
            error("Fallo al renombrar entrada " +
url.getDN(), e);
            return false;
        }
        return true;
    }

    private void resetConnection(LDAPURL url)
    {
        connections.remove(url.getBase());
    }

    private NamingEnumeration search(DirContext ctx,
String dn, String filter, String attribs[], int type)
        throws NamingException
    {
        return search(ctx, dn, filter, attribs, type,
true);
    }

    private NamingEnumeration search(DirContext ctx,
String dn, String filter, String attribs[], int type,
boolean setLimits)
        throws NamingException
    {
        SearchControls constraints = new
SearchControls();
        constraints.setSearchScope(type);
        constraints.setReturningAttributes(attribs);
        if(setLimits)
        {
            constraints.setCountLimit(limit);
            constraints.setTimeLimit(timeout);
        }
        NamingEnumeration results = ctx.search(dn,
filter, constraints);
        return results;
    }

    public Vector search(LDAPURL url, String filter,
String attribs[], boolean subTreeScope, Cancelable
cancel)
    {
        Vector results = new Vector();
```

```
        String attrs[] = new String[attribs.length + 1];
        attrs[0] = "objectclass";
        System.arraycopy(attribs, 0, attrs, 1,
attribs.length);
        int scope = subTreeScope ? 2 : 1;
        subSearch(url, filter, attrs, scope, results,
cancel);
        return results;
    }

    public void setAttributeConfig(AttributeConfig
config)
    {
        String binary = config.getBinaryList();
        if(connections.size() == 0)
        {
            env.put("java.naming.ldap.attributes.binary",
binary);
        } else
        {
            DirContext ctx = null;
            for(Enumeration enume =
connections.elements(); enume.hasMoreElements();)
                try
                {
                    ctx =
(DirContext)enume.nextElement();

ctx.removeFromEnvironment("java.naming.ldap.attributes.bi
nary");

ctx.addToEnvironment("java.naming.ldap.attributes.binary"
, binary);
                }
                catch(NamingException ex)
                {
                    error("Fallo al utilizar variable",
ex);
                }

        }
    }

    public void setConnectionHandler(Connector con)
    {
        connector = con;
    }

    private void setDefaultEnv()
    {
        timeout = Config.getTimeout();
        limit = Config.getLimit();
        listFilter = Config.getListFilter();
        if(listFilter == null)
```

```
            listFilter = "(objectclass=*)";
        attributesList = Config.getAttributesList();
        showOpAttributes = attributesList != null;
        env.put("java.naming.referral",
Config.manageReferrals() ? "ignore" : "throw");
        env.put("java.naming.batchsize",
String.valueOf(Config.getBatchSize()));
        String userdn = null;
        String userpwd = null;
        if(Config.adminLogin())
        {
            userdn = Config.getManagerDN();
            userpwd = Config.getPassword();
        }
        if(userdn != null && userpwd != null)
        {
            env.put("java.naming.security.principal",
userdn);
            env.put("java.naming.security.credentials",
userpwd);
            anonymousBind = false;
        } else
        {
            env.remove("java.naming.security.principal");

env.remove("java.naming.security.credentials");
            anonymousBind = true;
        }
        env.put("java.naming.security.authentication",
Config.getSecurityAuthentication());
        String saslClientPkgs =
Config.getSaslClientPkgs();
        if(saslClientPkgs != null)
            env.put("javax.security.sasl.client.pkgs",
saslClientPkgs);
        else

env.remove("javax.security.sasl.client.pkgs");
        env.put("java.naming.ldap.derefAliases",
Config.getAliasingOption());
        env.put("java.naming.ldap.deleteRDN",
Config.deleteOldDN() ? "true" : "false");
        version = Config.getVersion();
        env.put("java.naming.ldap.version",
String.valueOf(version));
        String securityProtocol =
Config.getSecurityProtocol();
        if(securityProtocol != null)
        {
            env.put("java.naming.security.protocol",
securityProtocol);
            if(securityProtocol.equalsIgnoreCase("ssl"))
```

```
env.put("java.naming.ldap.factory.socket",
Config.getLdapSocketFactory());
        } else
        {
            env.remove("java.naming.security.protocol");

env.remove("java.naming.ldap.factory.socket");
        }
        if(Debug.ldapDebug())
            env.put("com.sun.jndi.ldap.trace.ber",
System.err);
        String leafAttr = Config.getLeafIndicator();
        if(leafAttr != null)
        {
            indicatorAttr = (new String[] {
                leafAttr
            });
            indicatorType =
Config.getLeafIndicatorType();
        }
        String provider = Config.getJndiProvider();
        env.put("java.naming.factory.initial", provider
!= null ? ((Object) (provider)) : ((Object)
(DEFAULT_CTX)));
    }

    public void showOpAttributes(boolean show)
    {
        showOpAttributes = show;
    }

    private boolean subSearch(LDAPURL url, String filter,
String attribs[], int scope, Vector rs, Cancelable
cancel)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        String entryDN = null;
        Attributes at = null;
        Attribute a = null;
        LDAPURL myurl = null;
        int subscope = 0;
        String baseDN = url.getDN();
        boolean moreReferrals = true;
        while(moreReferrals)
            try
            {
                Vector vl;
                for(NamingEnumeration results =
search(ctx, baseDN, filter, attribs, scope);
results.hasMore(); rs.addElement(vl))
                {
```

```
                    SearchResult si =
(SearchResult)results.next();
                    if(cancel.isCanceled())
                    {
                        results.close();
                        return false;
                    }
                    vl = new Vector(attribs.length);
                    entryDN = getFixedDN(si.getName(),
baseDN);
                    myurl = new LDAPURL(url.getHost(),
url.getPort(), entryDN);
                    vl.addElement(myurl);
                    at = si.getAttributes();
                    for(int i = 1; i < attribs.length;
i++)
                    {
                        a = at.get(attribs[i]);
                        if(a == null)
                        {
                            vl.addElement("N/A");
                        } else
                        {
                            Object v = a.get();
                            if(v instanceof byte[])

vl.addElement(Common.format((byte[])v));
                            else

vl.addElement(a.get().toString());
                        }
                    }

                }

                moreReferrals = false;
            }
            catch(ReferralException e)
            {
                myurl = getReferralUrl(e);
                subscope = scope != 1 ? scope : 0;
                boolean error = subSearch(myurl, filter,
attribs, subscope, rs, cancel);
                if(!error)
                    return error;
                moreReferrals = e.skipReferral();
                try
                {
                    ctx =
(DirContext)e.getReferralContext();
                }
                catch(NamingException _ex) { }
            }
            catch(NamingException e)
```

```
            {
                error("Fallo Busqueda", e);
                return false;
            }
        return true;
    }

    public boolean synchEntry(LDAPURL url, Attributes
ats)
    {
        DirContext ctx = connect(url);
        if(ctx == null)
            return false;
        try
        {
            ctx.modifyAttributes(url.getDN(), 2, ats);
        }
        catch(ReferralException e)
        {
            LDAPURL myurl = getReferralUrl(e);
            return synchEntry(url, ats);
        }
        catch(NameNotFoundException _ex)
        {
            try
            {
                ctx.createSubcontext(url.getDN(), ats);
            }
            catch(NamingException _ex2)
            {
                return false;
            }
        }
        catch(NamingException e)
        {
            error("Fallo al sincronizar entradas ", e);
            return false;
        }
        return true;
    }

    public boolean transfer(LDAPURL fromUrl, LDAPURL
toUrl, boolean delete, boolean replace, boolean
withChildren, Cancelable cancelable, ProgressListener
listener)
    {
        LDAPURL dstUrl = toUrl;
        int rc = compare(fromUrl, toUrl);
        if(rc == 1)
            dstUrl = findEntryName(dstUrl);
        if(withChildren)
            return transferTreeSub(fromUrl, dstUrl,
delete, replace, cancelable, listener);
        else
```

```
            return transferEntry(fromUrl, dstUrl, delete,
replace);
    }

    public boolean transferEntry(LDAPURL fromUrl,
Attributes ats, LDAPURL toUrl, boolean delete, boolean
replace)
    {
        if(delete && !deleteEntry(fromUrl))
            return false;
        if(updateEntry(toUrl, ats, replace))
            return true;
        if(delete)
            addEntry(fromUrl, ats);
        return false;
    }

    public boolean transferEntry(LDAPURL fromUrl, LDAPURL
toUrl, boolean delete, boolean replace)
    {
        Attributes ats = read(fromUrl);
        if(ats == null)
            return false;
        else
            return transferEntry(fromUrl, ats, toUrl,
delete, replace);
    }

    private boolean transferTreeSub(LDAPURL fromUrl,
LDAPURL toUrl, boolean delete, boolean replace,
Cancelable cancelable, ProgressListener listener)
    {
        DirContext ctx = connect(fromUrl);
        if(ctx == null)
            return false;
        Attributes ats = read(fromUrl);
        if(ats == null)
            return false;
        String srcDN = fromUrl.getDN();
        String dstDN = toUrl.getDN();
        boolean createdBase = false;
        boolean rc = false;
        boolean moreReferrals = true;
        while(moreReferrals)
            try
            {
                NamingEnumeration results = search(ctx,
srcDN, "(objectclass=*)", DEFAULT_ATTR, 1, false);
                if(!results.hasMore())
                {
                    if(!transferEntry(fromUrl, ats,
toUrl, delete, replace))
                        return false;
                } else
```

```
                {
                    String name = null;
                    if(!createdBase)
                    {
                        if(!updateEntry(toUrl, ats,
replace))
                            return false;
                        createdBase = true;
                    }
                    LDAPURL srcUrl;
                    LDAPURL dstUrl;
                    for(; results.hasMore();
transferTreeSub(srcUrl, dstUrl, delete, replace,
cancelable, listener))
                    {
                        if(cancelable != null &&
cancelable.isCanceled())
                        {
                            results.close();
                            return false;
                        }
                        SearchResult si =
(SearchResult)results.next();
                        name = fixName(si.getName());
                        String tmpSrcDN = getDN(name,
srcDN);
                        srcUrl = new
LDAPURL(fromUrl.getHost(), fromUrl.getPort(), tmpSrcDN);
                        String tmpDstDN = getDN(name,
dstDN);
                        dstUrl = new
LDAPURL(toUrl.getHost(), toUrl.getPort(), tmpDstDN);
                        if(listener != null)
                            listener.msg(tmpSrcDN,
tmpDstDN);
                    }

                    if(delete && !deleteEntry(fromUrl))
                        return false;
                }
                moreReferrals = false;
            }
            catch(ReferralException e)
            {
                if(delete)
                {
                    moreReferrals = false;
                } else
                {
                    if(!createdBase)
                    {
                        if(!updateEntry(toUrl, ats,
replace))
                            return false;
```

```
                        createdBase = true;
                    }
                    LDAPURL srcUrl = getReferralUrl(e);
                    String tmpDstDN =
getName(srcUrl.getDN()) + ", " + dstDN;
                    LDAPURL dstUrl = new
LDAPURL(toUrl.getHost(), toUrl.getPort(), tmpDstDN);
                    boolean rs = transferTreeSub(srcUrl,
dstUrl, delete, replace, cancelable, listener);
                    if(!rs)
                        return false;
                    moreReferrals = e.skipReferral();
                    try
                    {
                        ctx =
(DirContext)e.getReferralContext();
                    }
                    catch(NamingException _ex) { }
                }
            }
            catch(NamingException e)
            {
                if(listener != null)
                    listener.error("Transferencia
fallida: " + e.getMessage(), null);
                error("Fallo en transferencia de arbol",
e);
                return false;
            }
        return true;
    }

    public boolean updateAttribute(LDAPURL url, Attribute
at)
    {
        try
        {
            ModificationItem mods[] = new
ModificationItem[1];
            mods[0] = new ModificationItem(2, at);
            return modifyAttribute(url, mods);
        }
        catch(NamingException e)
        {
            error("Fallo al actualizar '" + at.getID() +
"' atributo por " + url.getUrl(), e);
        }
        return false;
    }

    public boolean updateEntry(LDAPURL url, Attributes
at)
    {
        DirContext ctx = connect(url);
```

```
        if(ctx == null)
            return false;
        try
        {
            ctx.modifyAttributes(url.getDN(), 2, at);
        }
        catch(ReferralException e)
        {
            LDAPURL myurl = getReferralUrl(e);
            return updateEntry(myurl, at);
        }
        catch(NamingException e)
        {
            error("Fallo al actualizar entrada " +
url.getDN(), e);
            return false;
        }
        return true;
    }

    public boolean updateEntry(LDAPURL url, Attributes
ats, boolean replace)
    {
        return replace ? synchEntry(url, ats) :
addEntry(url, ats);
    }

    private static final String DEFAULT_ATTR[] = {
        "objectclass"
    };
    private static final String DEFAULT_LIST_FILTER =
"(objectclass=*)";
    public static final String LDAP_VERSION =
"java.naming.ldap.version";
    private static final String LDAP_ALIAS_OPTION =
"java.naming.ldap.derefAliases";
    private static final String LDAP_DELETE_RDN =
"java.naming.ldap.deleteRDN";
    private static final String LDAP_BINARY_ATTRIBUTES =
"java.naming.ldap.attributes.binary";
    private static final String LDAP_SOCKET_FACTORY =
"java.naming.ldap.factory.socket";
    private static final String SASL_CLIENT_PKGS =
"javax.security.sasl.client.pkgs";
    private static String DEFAULT_CTX =
"com.sun.jndi.ldap.LdapCtxFactory";
    public static final int BASE = 0;
    public static final int ONE = 1;
    public static final int SUB = 2;
    private Hashtable connections;
    private Connector connector;
    private int limit;
    private int timeout;
    private int version;
```

```
    private boolean anonymousBind;
    private int indicatorType;
    private String indicatorAttr[];
    private String listAttrs[];
    private String listFilter;
    private String attributesList[];
    private NameParser parser;
    private boolean showOpAttributes;
    private Properties env;
    protected ErrorListener listener;

}
```

CertificateEditor2.java

```
package lbe.editor;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.text.MessageFormat;
import javax.swing.*;
import lbe.common.Common;
import lbe.interfaces.AttributeEditor;

public class CertificateEditor2 extends JPanel
    implements AttributeEditor, ActionListener
{

      private static final long serialVersionUID = -
9119579697924218043L;
      public CertificateEditor2()
    {
        value = null;
        subjectLB = createLabel();
        issuerLB = createLabel();
        versionLB = createLabel();
        algorithmLB = createLabel();
        vadilityFLB = createLabel();
        vadilityTLB = createLabel();
        serialnLB = createLabel();
        setLayout(new BorderLayout());
        sizeLB = new JLabel();
        saveF = new JButton("Guardar como");
        saveF.addActionListener(this);
        loadF = new JButton("Insertar desde");
        loadF.addActionListener(this);
        viewB = new JButton("Ver");
        viewB.addActionListener(this);
        buttonP = new JPanel();
        buttonP.add(sizeLB);
        buttonP.add(saveF);
        buttonP.add(loadF);
        add(createCertPanel(), "Center");
        add(buttonP, "South");
    }

    public void actionPerformed(ActionEvent e)
    {
        String cmd = e.getActionCommand();
        if(cmd.equals("Guardar como"))
```

```
            saveOption();
        else
        if(cmd.equals("Insertar desde"))
            loadOption();
        else
        if(cmd.equals("Ver"))
            externalView();
    }

    public boolean checkType(Object value)
    {
        return value instanceof byte[];
    }

    private JPanel createCertPanel()
    {
        JPanel p1 = new JPanel();
        p1.setBorder(BorderFactory.createTitledBorder("
Información de Certificado "));
        GridBagLayout gbl = new GridBagLayout();
        GridBagConstraints gbc = new
GridBagConstraints();
        p1.setLayout(gbl);
        gbc.weightx = 0.0D;
        gbc.anchor = 13;
        Common.add(p1, createLabel("Asunto: "), gbl, gbc,
1, 0, 1, 1);
        Common.add(p1, createLabel("Issuer: "), gbl, gbc,
1, 1, 1, 1);
        Common.add(p1, createLabel("Valido desde: "),
gbl, gbc, 1, 2, 1, 1);
        Common.add(p1, createLabel("a: "), gbl, gbc, 1,
3, 1, 1);
        Common.add(p1, createLabel(" Sig. Algorithm: "),
gbl, gbc, 1, 4, 1, 1);
        Common.add(p1, createLabel("Numero Serial: "),
gbl, gbc, 3, 4, 1, 1);
        Common.add(p1, createLabel("Version: "), gbl,
gbc, 5, 4, 1, 1);
        gbc.weightx = 1.0D;
        gbc.anchor = 17;
        Common.add(p1, subjectLB, gbl, gbc, 2, 0, 5, 1);
        Common.add(p1, issuerLB, gbl, gbc, 2, 1, 5, 1);
        Common.add(p1, vadilityFLB, gbl, gbc, 2, 2, 5,
1);
        Common.add(p1, vadilityTLB, gbl, gbc, 2, 3, 5,
1);
        Common.add(p1, algorithmLB, gbl, gbc, 2, 4, 1,
1);
        Common.add(p1, serialnLB, gbl, gbc, 4, 4, 1, 1);
        Common.add(p1, versionLB, gbl, gbc, 6, 4, 1, 1);
        return p1;
    }
```

```
    private JLabel createLabel()
    {
        JLabel lb = new JLabel();
        lb.setForeground(Color.black);
        return lb;
    }

    private JLabel createLabel(String text)
    {
        JLabel lb = new JLabel(text);
        return lb;
    }

    private void externalView()
    {
        File tmpF = null;
        try
        {
            tmpF = File.createTempFile("certviewer",
".crt");
        }
        catch(IOException e)
        {
            System.err.println("Unable to create tmp
file: " + e.getMessage());
            return;
        }
        tmpF.deleteOnExit();
        Common.saveAsBytes(tmpF, value);
        String viewcmd[] = null;
        if(cmdstr == null)
        {
            viewcmd = (new String[] {
                "rundll32.exe",
"cryptext.dll,CryptExtOpenCER", tmpF.getAbsolutePath()
            });
        } else
        {
            String args = MessageFormat.format(cmdstr,
new Object[] {
                tmpF.getAbsolutePath()
            });
            viewcmd = Common.parseArgs(args);
        }
        if(viewcmd == null)
        {
            System.err.println("No view cmd specified");
            return;
        }
        Process child;
        try
        {
            child = Runtime.getRuntime().exec(viewcmd);
        }
```

```
        catch(IOException e)
        {
            System.err.println("Error starting external
viewer: " + e.getMessage());
            e.printStackTrace();
        }
    }

    private X509Certificate getCertObject()
    {
        if(value == null)
            return null;
        try
        {
            ByteArrayInputStream bais = new
ByteArrayInputStream(value);
            CertificateFactory cf =
CertificateFactory.getInstance("X.509");
            return
(X509Certificate)cf.generateCertificate(bais);
        }
        catch(Exception e)
        {
            System.err.println(e);
        }
        return null;
    }

    private File getFile(String label)
    {
        JFileChooser filechooser = new JFileChooser();
        filechooser.setApproveButtonText(label);
        filechooser.setCurrentDirectory(new File("."));
        if(filechooser.showOpenDialog(this) == 0)
        {
            File f = filechooser.getSelectedFile();
            return f;
        } else
        {
            return null;
        }
    }

    public Object getValue()
    {
        if(value == null)
            value = new byte[0];
        return value;
    }

    public boolean isEmpty()
    {
        return value == null || value.length == 0;
    }
```

```
    public boolean isRequired()
    {
        return false;
    }

    private void loadOption()
    {
        File f = getFile("Insert");
        if(f == null)
        {
            return;
        } else
        {
            setValue(Common.readAsBytes(f));
            return;
        }
    }

    public void requestFocus()
    {
        super.requestFocus();
    }

    private void saveOption()
    {
        File f = getFile("Save");
        if(f == null)
        {
            return;
        } else
        {
            Common.saveAsBytes(f, value);
            return;
        }
    }

    public void setArguments(String args)
    {
        if(args == null)
            return;
        boolean externalView = false;
        String argsV[] = Common.parseArgs(args);
        int size = argsV.length;
        for(int i = 0; i < size; i++)
        {
            String arg = argsV[i];
            if(arg.equalsIgnoreCase("-ext"))
                externalView = true;
            else
            if(arg.equalsIgnoreCase("-extcmd") && i + 1 <
size)
            {
                i++;
```

```
                cmdstr = argsV[i];
            } else
            {
                System.err.println("CertificateEditor:
Argumento desconocido: " + arg);
            }
        }

        if(externalView)
            buttonP.add(viewB);
        else
            buttonP.remove(viewB);
    }

    public void setEditMode(boolean edit)
    {
        if(edit)
            buttonP.add(loadF);
        else
            buttonP.remove(loadF);
    }

    private void setFields(X509Certificate cert)
    {
        String text = null;
        text = cert != null ?
cert.getSubjectDN().toString() : "N/A";
        subjectLB.setText(text);
        text = cert != null ?
cert.getIssuerDN().toString() : "N/A";
        issuerLB.setText(text);
        text = cert != null ?
String.valueOf(cert.getVersion()) : "N/A";
        versionLB.setText(text);
        text = cert != null ? cert.getSigAlgName() :
"N/A";
        algorithmLB.setText(text);
        text = cert != null ?
cert.getNotBefore().toString() : "N/A";
        vadilityFLB.setText(text);
        text = cert != null ?
cert.getNotAfter().toString() : "N/A";
        vadilityTLB.setText(text);
        text = cert != null ?
cert.getSerialNumber().toString() : "N/A";
        serialnLB.setText(text);
    }

    public void setRequired(boolean flag)
    {
    }

    public void setValue(Object new_value)
    {
```

```
            if(new_value == null)
                value = new byte[0];
            else
                value = (byte[])new_value;
            sizeLB.setText(Common.format(value, false));
            X509Certificate cert = getCertObject();
            setFields(cert);
        }

        public boolean verify()
        {
            return true;
        }

        private JLabel subjectLB;
        private JLabel issuerLB;
        private JLabel versionLB;
        private JLabel algorithmLB;
        private JLabel vadilityFLB;
        private JLabel vadilityTLB;
        private JLabel serialnLB;
        private JPanel buttonP;
        private JLabel sizeLB;
        protected JButton saveF;
        protected JButton loadF;
        protected JButton viewB;
        protected byte value[];
        private String cmdstr;
    }
```

PasswordEditor.java

```
package lbe.editor;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
//import java.io.PrintStream;
import java.util.Hashtable;
import javax.swing.*;
//import javax.swing.text.JTextComponent;
import lbe.common.Common;
import lbe.common.UITools;
import lbe.interfaces.AttributeEditor;

// Referenced classes of package lbe.editor:
//            PasswordHandler

public class PasswordEditor extends JPanel
    implements AttributeEditor, ActionListener
{

    /**
       *
       */
      private static final long serialVersionUID =
2877713845034403490L;
      public PasswordEditor()
    {
        value = null;
        editMode = false;
        nobuttons = false;
        forceAlgorithm = false;
        defaultAlgorithm = "SHA-1";
        debugMode = false;
        setLayout(new FlowLayout(0, 2, 2));
    }

    public void actionPerformed(ActionEvent e)
    {
        String cmd = e.getActionCommand();
        if(cmd.equals("Guardar como"))
            saveOption();
        else
        if(cmd.equals("Insertar desde"))
            loadOption();
        else
        if(cmd.equals("Verificar"))
            verifyOption();
        else
        if(cmd.equals("Encriptar"))
            setOption();
```

```
}

public boolean checkType(Object value)
{
    return value instanceof byte[];
}

private File getFile()
{
    JFileChooser filechooser = new JFileChooser();
    filechooser.setCurrentDirectory(new File("."));
    if(filechooser.showOpenDialog(this) == 0)
    {
        File f = filechooser.getSelectedFile();
        return f;
    } else
    {
        return null;
    }
}

private Frame getFrame()
{
    if(frame == null)
        frame = new JFrame();
    return frame;
}

public Object getValue()
{
    if(editMode)
    {
        String vl = pwdTF.getText();
        return vl.getBytes();
    } else
    {
        return value;
    }
}

public boolean isEmpty()
{
    return value == null || value.length == 0;
}

public boolean isRequired()
{
    return false;
}

private void loadOption()
{
    File f = getFile();
    if(f == null)
```

```
            {
                return;
            } else
            {
                setValue(Common.readAsBytes(f));
                return;
            }
        }

        public void requestFocus()
        {
            super.requestFocus();
        }

        private void saveOption()
        {
            File f = getFile();
            if(f == null)
            {
                return;
            } else
            {
                Common.saveAsBytes(f, value);
                return;
            }
        }

        private void set()
        {
            String vl = new String(value);
            if(editMode)
                pwdTF.setText(vl);
            else
                pwdLabel.setText(vl);
        }

        public void setArguments(String args)
        {
            nobuttons = false;
            defaultAlgorithm = "SHA-1";
            forceAlgorithm = false;
            debugMode = false;
            if(args != null)
            {
                Hashtable h = Common.args(args, new String[]
{
                    "-algorithm"
                }, new String[] {
                    "-nobuttons", "-force", "-debug"
                });
                if(h.get("-nobuttons") != null)
                    nobuttons = true;
                if(h.get("-force") != null)
                    forceAlgorithm = true;
```

```
            if(h.get("-debug") != null)
                debugMode = true;
            String tmp = (String)h.get("-algorithm");
            if(tmp == null)
                defaultAlgorithm = "SHA-1";
            else
            if(tmp.equalsIgnoreCase("SHA"))
                defaultAlgorithm = "SHA-1";
            else
            if(tmp.equalsIgnoreCase("MD5"))
                defaultAlgorithm = "MD5";
            else
                defaultAlgorithm = tmp;
        }
        if(debugMode)
        {
            String msg = "PasswordEditor: ";
            if(forceAlgorithm)
                msg = msg + "Forzando ";
            else
                msg = msg + "Usando ";
            msg = msg + defaultAlgorithm + " algorithm.";
            System.out.println(msg);
        }
        if(!nobuttons)
        {
            saveF = new JButton("Guardar como");
            loadF = new JButton("Insertar desde");
            saveF.addActionListener(this);
            loadF.addActionListener(this);
            verifyP = new JButton("Verificar");
            verifyP.addActionListener(this);
            generateP = new JButton("Encriptar");
            generateP.addActionListener(this);
        }
    }

    public void setEditMode(boolean edit)
    {
        if(edit)
        {
            pwdTF = new JTextField(20);
            add(pwdTF);
            if(!nobuttons)
            {
                add(verifyP);
                add(generateP);
                add(saveF);
                add(loadF);
            }
        } else
        {
            pwdLabel = new JLabel();
            add(pwdLabel);
```

```
            if(!nobuttons)
            {
                add(verifyP);
                add(saveF);
            }
        }
        editMode = edit;
    }

    private void setOption()
    {
        JButton setB = new JButton("Encriptar");
        JButton cancelB = new JButton("Cancelar");
        JLabel msgL = new JLabel(" ");
        final JPasswordField passTF = new
JPasswordField(20);
        JPanel p1 = new JPanel();
        p1.add(new JLabel("Ingresar contraseña"));
        p1.add(passTF);
        JPanel buttonP = new JPanel();
        buttonP.add(setB);
        buttonP.add(cancelB);
        JPanel main = new JPanel();
        main.setLayout(new BorderLayout());
        main.add(p1, "North");
        main.add(buttonP, "South");
        msgL.setForeground(Color.black);
        final JDialog frame = new JDialog(getFrame(),
"Generar Contraseña", true);
        frame.getContentPane().add(main);
        cancelB.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e)
            {
                frame.dispose();
            }

        });
        ActionListener actionListener = null;
        setB.addActionListener(actionListener = new
ActionListener() {

            public void actionPerformed(ActionEvent e)
            {
                String pwd = new
String(passTF.getPassword());
                String alg = defaultAlgorithm;
                if(!forceAlgorithm)
                {
                    String digest = new String(value);
                    if(digest.regionMatches(true, 0,
"{CRYPT}", 0, 7))
                        alg = "CRYPT";
                    else
```

```
                    if(digest.regionMatches(true, 0,
"{SHA}", 0, 5))
                        alg = "SHA-1";
                    else
                    if(digest.regionMatches(true, 0,
"{SSHA}", 0, 6))
                        alg = "SHA-1";
                    else
                    if(digest.regionMatches(true, 0,
"{MD5}", 0, 5))
                        alg = "MD5";
                    else
                    if(digest.regionMatches(true, 0,
"{SMD5}", 0, 6))
                        alg = "MD5";
                }
                String password =
PasswordHandler.generateDigest(pwd, null, alg);
                if(password != null)
                {
                    value = password.getBytes();
                    set();
                }
                frame.dispose();
            }

        });
        passTF.addActionListener(actionListener);
        frame.pack();
        UITools.center(getFrame(), frame);
        frame.setVisible(true);
    }

    public void setRequired(boolean flag)
    {
    }

    public void setValue(Object new_value)
    {
        if(new_value == null)
            value = new byte[0];
        else
            value = (byte[])new_value;
        set();
    }

    public boolean verify()
    {
        return true;
    }

    private void verifyOption()
    {
        JButton verifyB = new JButton("Verificar");
```

```
        JButton okB = new JButton("Salir");
        final JLabel msgL = new JLabel(" ", 0);
        final JPasswordField passTF = new
JPasswordField(20);
        JPanel p1 = new JPanel();
        p1.add(new JLabel("Ingresar contraseña"));
        p1.add(passTF);
        JPanel buttonP = new JPanel();
        buttonP.add(verifyB);
        buttonP.add(okB);
        JPanel main = new JPanel();
        main.setLayout(new BorderLayout());
        main.add(p1, "North");
        main.add(msgL, "Center");
        main.add(buttonP, "South");
        msgL.setForeground(Color.black);
        final JDialog frame = new JDialog(getFrame(),
"Verificar contraseña", true);
        frame.getContentPane().add(main);
        okB.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e)
            {
                frame.dispose();
            }

        });
        final String crValue = new String(value);
        ActionListener actionListener = null;
        verifyB.addActionListener(actionListener = new
ActionListener() {

            public void actionPerformed(ActionEvent e)
            {
                String pwd = new
String(passTF.getPassword());
                boolean rs =
PasswordHandler.verifyPassword(crValue, pwd);
                if(rs)
                    msgL.setText("Contraseña
verificada");
                else
                    msgL.setText("Contraseña no
verificada");
            }

        });
        passTF.addActionListener(actionListener);
        frame.pack();
        UITools.center(getFrame(), frame);
        frame.setVisible(true);
    }

    private JFrame frame;
```

```
    protected JLabel pwdLabel;
    protected JTextField pwdTF;
    protected JButton saveF;
    protected JButton loadF;
    protected JButton verifyP;
    protected JButton generateP;
    protected byte value[];
    protected boolean editMode;
    protected boolean nobuttons;
    protected boolean forceAlgorithm;
    protected String defaultAlgorithm;
    protected boolean debugMode;

}
```

CAPÍTULO 2

MANUAL DE USUARIO

2.1 Recomendaciones Generales

Se recomienda leer por completo esta guía, en la que se detallan algunas especificaciones con respecto al Sistema Servicio de Directorio LDAP. Si se requiere más información acerca del funcionamiento, remítase al manual aquí proporcionado.

2.2 Elementos de Hardware

Para poder utilizar el sistema de Servicio de Directorio Seguro, requerimos:

- Un Servidor
- Terminal o terminales en red

Descripciones detalladas del Servidor:

Procesador: Debe ser multiprocesador

Disco Duro: Para utilizar OpenLDAP lo más optimo es que utilice un disco duro para el sistema operativo y otro para el directorio donde se guarde la base.

Memoria: El tamaño de la memoria depende de las entradas que almacenaremos y los atributos que estas contengan, como mínimo podemos tener 768 MB.

Descripciones detalladas del Terminal (es)

Procesador: Lo más recomendable es un Petium IV.

Disco Duro: Desde 40GB.

Memoria: El tamaño de la memoria recomendable para el Terminal que contendrá la aplicación “Servicio de Directorio (LDAP)” que interactúa con el servidor.

2.3 Elementos de Software

Los requisitos de software que se emplean en el proyecto son los siguientes:

En el servidor:

- Sistema Operativo Linux Fedora Core 4
- OpenLdap 2.2.23-5
- OpenSSL 0.97f-7
- Parches necesarios

En el Terminal (les):

- Sistema Operativo Windows XP
- Parches necesarios.

2.4 Elementos Lógicos

Los elementos lógicos que se requieren son:

- Direcciones IP, configuración de la red que se va a emplear.

  Ej. Red 192.168.3.0/24

  Mascara 255.255.255.0

2.5 Instalación de Fedora Core 4.0

Antes de comenzar, asegurarse de tener una partición o un disco duro nuevo. Aconsejamos que imprima esta guía o por lo menos apunte los pasos más importantes en un papel. Arrancamos desde el CD.

Empezamos a instalar:

Presiona "Enter" para comenzar la instalación gráfica, observará como se cargan algunos módulos y como Linux reconoce el hardware.

A continuación aparece una pantalla (Gráfico #7) en la cual pregunta si desea analizar los CD para comprobar si funcionan correcto, es recomendable que "escanee" los 3 CDs, (si no desea hacerlo seleccione "Skip") cuando termine la acción seleccione "Continue" se cargará "Anaconda" que es instalador gráfico de Red Hat, ahora ya puede usar el ratón, haga clic en "Next".

Gráfico -7-
Verificación de discos

Select lenguaje:

En la siguiente pantalla selecciona el idioma (Spanish) y vuelve a pulsar "Next".

Configuración del teclado**:**

Selecciona "Spanish" de la lista y pulsa en "Siguiente" (Gráfico #8).

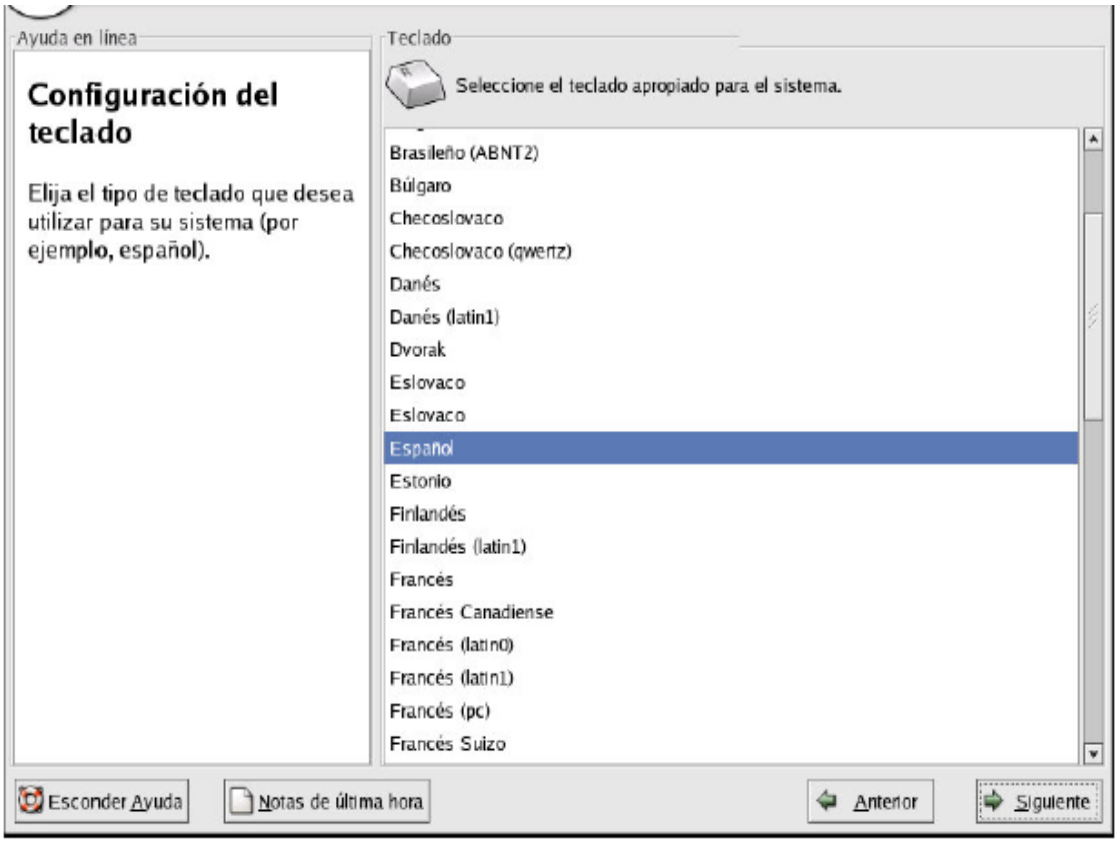


Gráfico -8-
Configuración del Teclado

Configuración del mouse:

Ahora, ya en español, selecciona su mouse de la lista, si no ve su modelo deje el genérico que viene seleccionado por defecto, y pulse, como no, "Siguiente" (Gráfico #9).

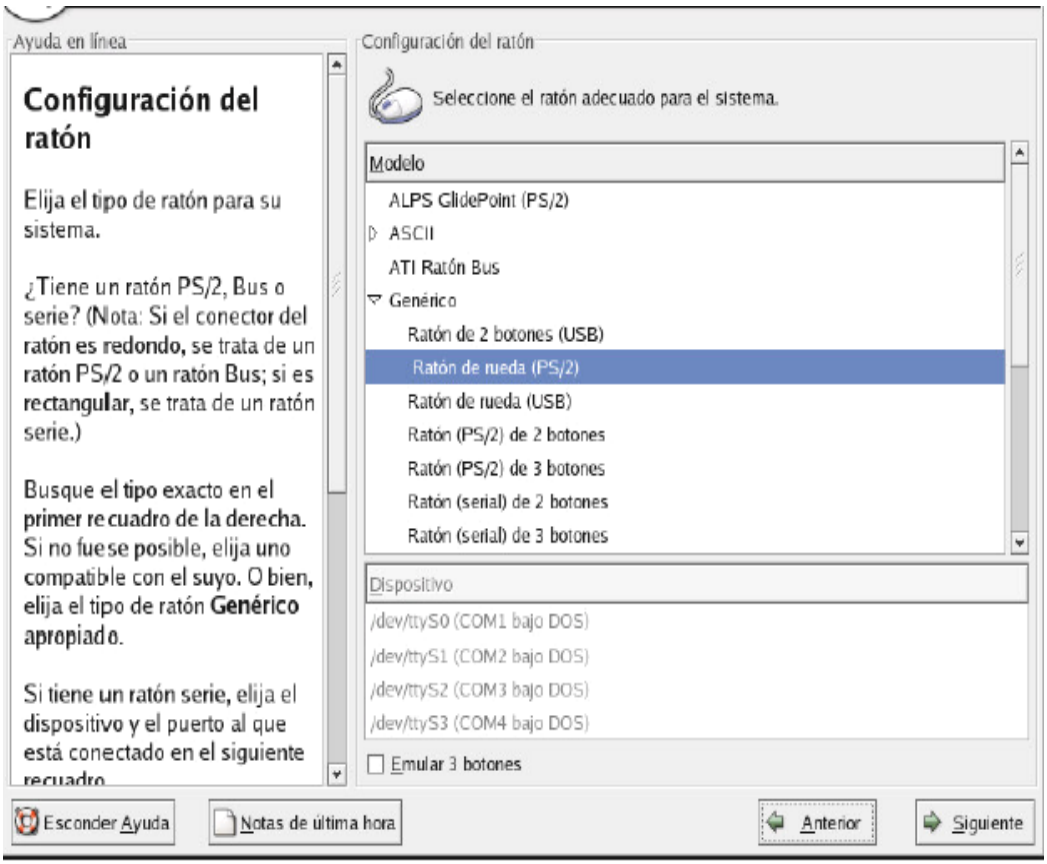


Gráfico -9-
Configuración del Ratón

Tipo de instalación:

Existen cuatro tipos de instalaciones:

- Escritorio personal.
- Estación de trabajo
- Servidor
- Personalizada

Escogemos la opción Servidor. Pulsamos "Siguiente" (Gráfico #10).

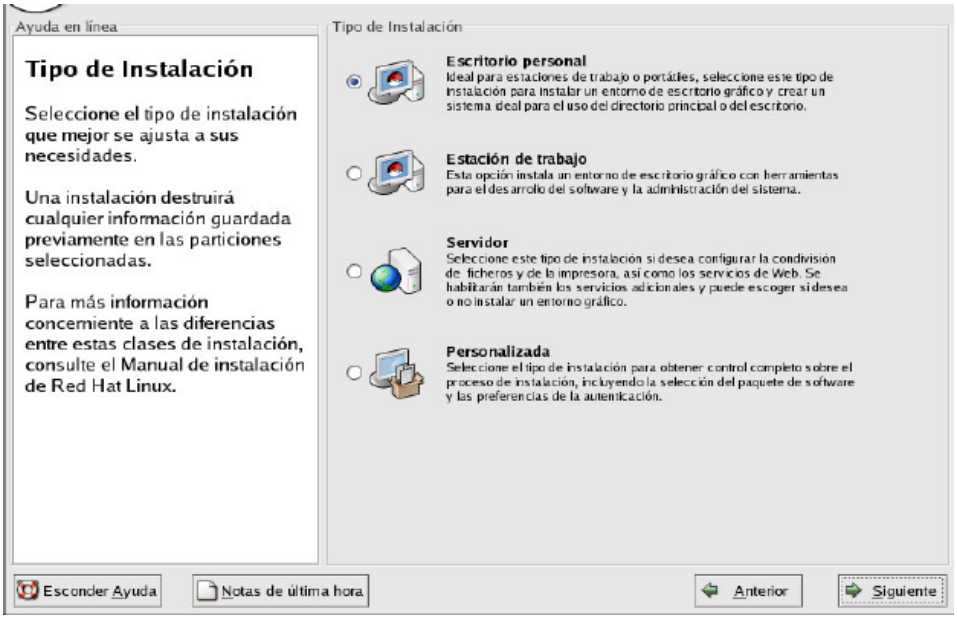


Gráfico -10-
Tipo de Instalación

Configuración de la partición:

Existen dos formas:

- Automática
- Manual (con Disk Druid)

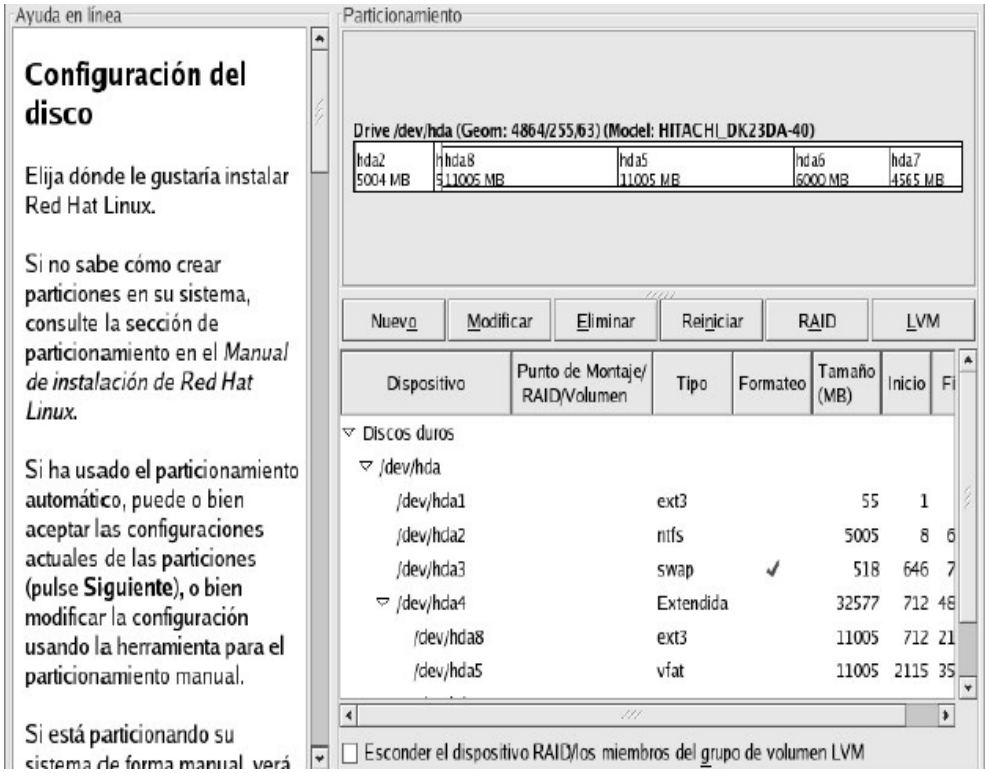


Gráfico -11-
Configuración de la Partición

¿Cómo identifico la unidad donde voy a instalar Linux?

Cuando vemos un disco duro o unidades dentro de Windows, se presentan como letras del abecedario (C, D, E, etc.) pero dentro de Linux, es diferente, ya que su estructura semeja un árbol donde cada partición y dispositivo de lectura/escritura se representa como un directorio, los nombres de las unidades de disco duro son:

- hda: disco duro principal
- hdb: disco duro secundario
- hda1: primera partición del disco duro principal.
- hdb2: segunda partición del disco secundario

Ahora, para dar un ejemplo de todo este proceso, supongamos que tienen un disco duro de 20 GB y generan dos particiones, uno de 5 GB para Windows y el resto para Linux, entonces es hda1 (Windows) y hda2 (Linux), siendo en este último donde crearíamos las particiones del sistema.

Directorios de Linux

Estos directorios se crean en el momento de la instalación del sistema.

- **/boot .-** Donde se leen los parámetros para iniciar el sistema.

  Requiere al menos 75 MB en Red Hat™ Enterprise Linux 3.0 y White Box Enterprise Linux 3.0. Asignar más espacio puede considerarse desperdicio.

- **/ o raíz .-** Es donde se instalarán los componentes del sistema operativo. Requiere de 350 a 512 MB.

- **Swap.-** Espacio físico para la memoria virtual del sistema. Debe asignarse el doble del tamaño del RAM físico.

- **/usr.-** Contiene la mayoría de los binarios (ejecutables), bibliotecas compartidas, manuales, datos de aplicaciones e imágenes que utiliza el sistema, cabeceras de desarrollo, el árbol del kernel y documentación.
- **/tmp.-** En éste se almacenan todos los ficheros temporales que generan los distintos programas.

- **/var.-** Corresponde a la partición de datos de servicios.

- **/home.-** Es donde se colocan los directorios para cada usuario con los perfiles de cada cuenta.

Haz clic sobre manual y pulsaremos "Siguiente".

Configuración del Cortafuegos

No configure cortafuegos en este momento. La herramienta utilizada para tal fin, system-configsecuritylevel, crea un cortafuegos simple y con muchas limitaciones. Se recomienda considerar otras alternativas como Firestarter o Shorewall. Deje activo SELinux, ya que éste proveerá al sistema de seguridad adicional. Al terminar, haga clic sobre el botón «Siguiente» (Gráfico #12).

Gráfico -12-
Configuración del Cortafuegos

Haga clic sobre el botón «Proceder» a fin de saltar la configuración del cortafuegos.

Configuración de red:

Para configurar los parámetros de red del sistema, haga clic sobre el botón «Modificar» para la interfaz eth0. En la ventana emergente para modificar la interfaz eth0, desactive la casilla «Configurar usando DHCP» y especifique la dirección IP y máscara de subred que utilizará en adelante el sistema.

Confirme con el administrador de la red donde se localice que estos datos sean correctos antes de continuar. Al terminar, haga clic sobre el botón «Aceptar»(Gráfico #13).

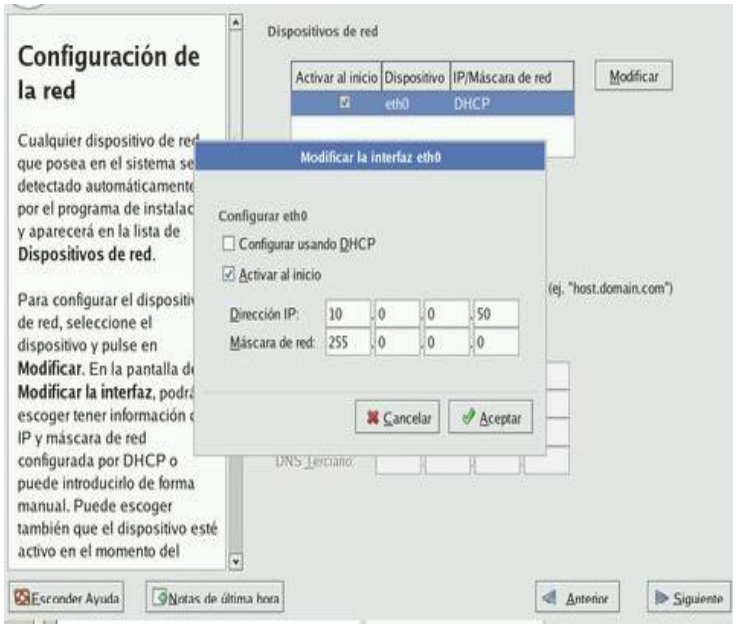


Gráfico -13-
Configuración de Red

Asigne un nombre de anfitrión (HOSTNAME) para el sistema. Al terminar, haga clic sobre el botón «Siguiente» (Gráfico #14).

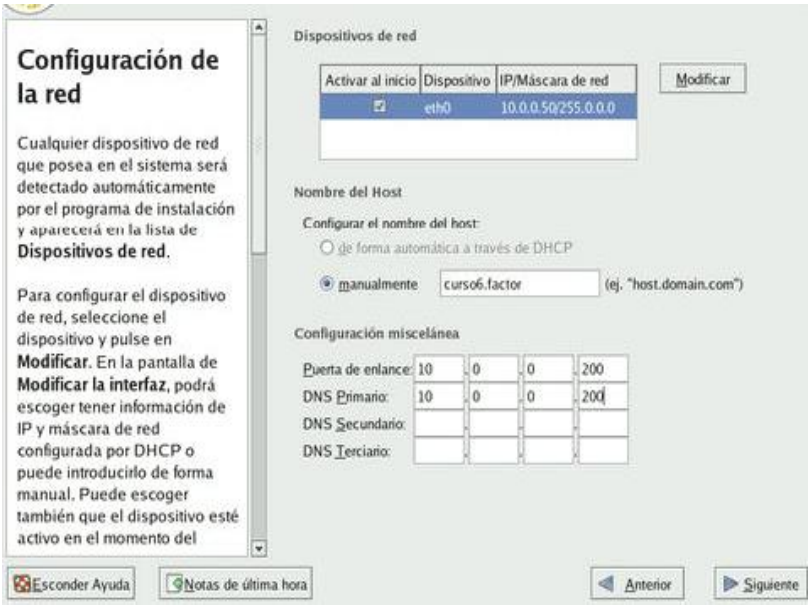


Gráfico -14-
Configuración de Red

Seleccionar el uso horario:

Seleccione la casilla «El sistema horario usará UTC», que significa que el reloj del sistema utilizará UTC (Tiempo Universal Coordinado), que es el sucesor de GMT (b>Greenwich Mean Time, que significa Tiempo Promedio de Greenwich), y es la zona horaria de referencia respecto a la cual se calculan todas las otras zonas del mundo. Haga clic con el ratón sobre la región que corresponda en el mapa mundial o seleccione en el siguiente campo la zona horaria

que corresponda a la región donde se hospedará físicamente el sistema (Gráfico #15).

Gráfico -15-
Configuración de Uso Horario

En el mapa seleccione America - Guayaquil

Pulse "Siguiente"

Contraseña del Root:

Asigne una clave de acceso al usuario root. Debe escribirla dos veces a fin de verificar que esta coincide con lo que realmente se

espera. Por razones de seguridad, se recomienda asignar una clave de acceso que evite utilizar palabras provenientes de cualquier diccionario, en cualquier idioma, así como cualquier combinación que tenga relación con datos personales (Gráfico #16), una buena clave contiene ocho o más caracteres entre los cuales tenemos letras, números y caracteres especiales, para evitar que sea fácil de descifrar.

Gráfico -16-
Configuración Contraseña del Root

Al terminar, haga clic sobre el botón «Siguiente», y espere a que el sistema haga la lectura de información de los grupos de paquetes.

Selección de paquetes:

En la siguiente pantalla podrá seleccionar los grupos de paquetes que quiera instalar en el sistema. Añada o elimine a su conveniencia (Gráfico #17).

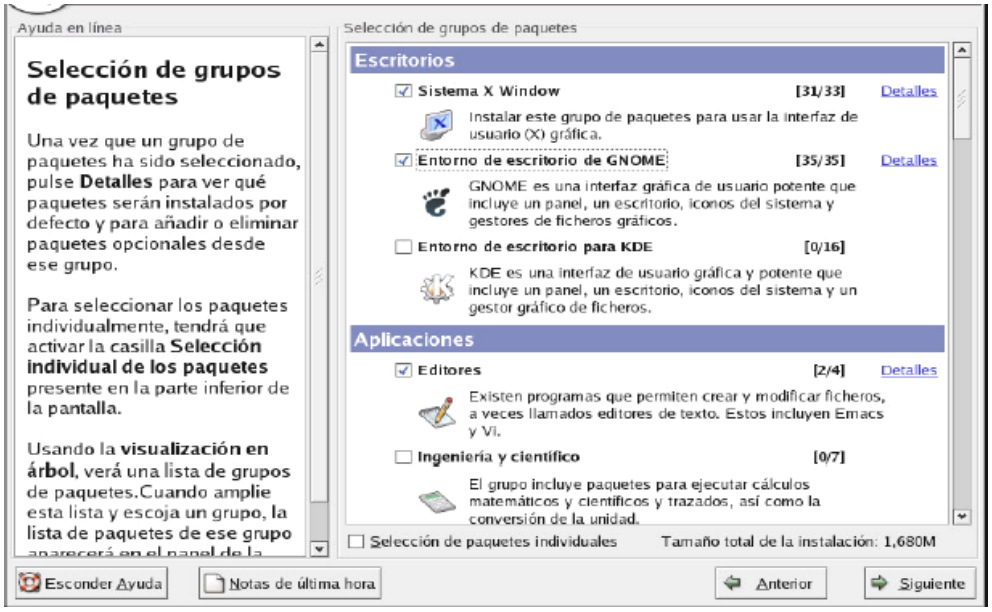


Gráfico -17-
Selección de Grupo de Paquetes

En esta parte es necesario recalcar, que se deberá escoger solo los paquetes necesarios para el servidor, entre ellos tenemos: OpenLdap-servers, OpenLdap-clients, Telnet, OpenSSL, dependencias opcionales de Java, Ldap jdk , Ldap jdk-javadoc,

Una vez terminada la selección de paquetes, haga clic sobre el botón «Siguiente» a fin de iniciar el proceso.

Instalando...

Si iniciará de forma automática el proceso de formato de las particiones que haya creado para instalar el sistema operativo.

Gráfico -18-
Instalación de Paquetes

Iniciará la instalación de los paquetes necesarios para el funcionamiento del sistema operativo. Espere algunos minutos hasta que concluya el proceso.

Una vez finalizada la instalación de los paquetes, haga clic sobre el botón «Reiniciar» (Gráfico #18).

2.6 Instalación, configuración y ejecución de OpenLdap como servidor de directorio

2.6.1. Descarga e instalación de OpenLdap

Si no instalamos los paquetes de OpenLDAP en la instalación de Linux Distribución Fedora Core4 debe descargar la versión 2.2.23-5.

Además se requerirá los siguientes paquetes:

- openldap-clients-2.2.23-5
- openldap-servers-2.2.23-5
- authconfig-4.6.12-1
- authconfig-gtk-4.6.12-1 (opcional)

Instalación de OpenLdap vía líneas de comando

1. Una vez obtenido el archivo Tar que contiene OpenLDAP, este debe ser descomprimido en un directorio temporal (/tmp por lo general) para poder iniciar la instalación.
2. Dentro del directorio temporal (/tmp) donde fue descomprimido OpenLDAP ejecute el comando: ./configure , este comando configura los archivos de instalación de acuerdo a su sistema.
3. Posteriormente debe ejecutar *make depend* seguido de *make*, esto genera OpenLDAP dentro del mismo directorio temporal.
4. Debe ejecutar ciertas pruebas para garantizar que OpenLDAP funcione correctamente, colóquese dentro del directorio *tests* y ejecute *make*
5. Ahora si debe instalar OpenLDAP en el sistema, descienda del directorio *tests* y como root ejecute: *make install*
6. El comando anterior instala OpenLDAP bajo el directorio /usr/local/etc/openldap.

2.6.2. Configuración del Servidor OpenLdap

Se procede ahora a editar el fichero slapd.conf, este es el archivo principal de OpenLDAP y es aquí donde se configuran todos sus parámetros, este fichero se encuentra dentro del directorio /etc/openldap.

Parámetros Globales

Los parámetros dentro de esta sección afectan el funcionamiento de todo el Servidor OpenLDAP, cualquier definición antes de un parámetro database es considerado global.

Se verifica que los ficheros de esquema mínimos requeridos estén presentes.

- *include:* Este parámetro indica otros archivos de configuración utilizados por el Servidor OpenLDAP, la declaración anterior carga los archivos core.schema, cosine.schema, inetorgperson.schema, nis.schema. Estos parámetros no serán modificados, de tal modo, debe quedar algo así:

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
```

- *referral:* Opcionalmente se puede descomentar la directiva referral para indicar el URI (Identificador Uniforme de Recursos) del servicio de directorio superior como ldaps en lugar de ldap.

```
referral        ldaps://uquil.com
```

- *pidfile:* Contiene el número de proceso asignado al servidor LDAP al arranque.

- *argsfile:* Contiene parámetros utilizados en la línea de comandos al iniciar el servidor OpenLDAP.

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

- *access:* Parámetro utilizado para restringir acceso al servidor LDAP.

```
access to *
       by dn="cn=root,dc=uquil,dc=com" write
       by * read
```

Parámetros por Base de Datos

Dentro de cada servidor LDAP se pueden encontrar varias base de datos, es dentro de estas bases de datos que residirá toda información del Servidor OpenLDAP.

En el sentido más estricto de la palabra OpenLDAP no utiliza una base de datos, la "base de datos" utilizada en OpenLDAP es generalmente ldbm.

- *database:* Indica el tipo de "base de datos" a utilizarse, generalmente del tipo ldbm (Otras alternativas: shell, passwd).

- *suffix:* Este parámetro indica el nodo raíz de la base de datos, esto es, el nodo sobre el cual será derivada toda la información, en este caso dc=uquil, dc=com. Lo anterior indica que toda información dentro de esta "base de datos" LDAP descenderá de la jerarquía dc=uquil, dc=com.

- *rootdn :* Establece el nodo ("usuario") que tiene privilegios globales para modificar la "base de datos" LDAP ,en este caso cn=root, nótese que desciende del nodo raíz (suffix) dc=uquil, dc=com.

```
#######################################
# ldbm and/or bdb database definitions
#######################################

database        ldbm
suffix          "dc=uquil,dc=com"
rootdn          "cn=root,dc=uquil,dc=com"
```

- *rootpw :* Indica la contraseña para el usuario rootdn. Esta contraseña también puede ser encriptada.

```
rootpw          secret
#rootpw         {MD5}pFl+EwLEUsTFiWOX52ySYA==
```

- *directory:* Directorio dónde se guardarán todos los datos del directorio LDAP.

```
directory       /var/lib/ldap
```

2.6.3. Ejecución y Comprobación del Servidor OpenLdap

Ejecución del Servidor:

Se inicia el servicio de LDAP añadiendo éste al resto de los servicios que arrancan junto con el sistema:

service ldap start

chkconfig ldap on

Se edita el fichero:

/usr/share/openldap/migration/migrate_common.ph

Se modifica los valores de las variables:

$DEFAULT_MAIL_DOMAIN y $DEFAULT_BASE.

Quedando del siguiente modo:

#Default DNS domain

$DEFAULT_MAIL_DOMAIN = "uquil.com"

#Default base

$DEFAULT_BASE = "dc=uquil,dc=com";

A continuación se genera un fichero base.ldif que a su vez contendrá el resto de los datos en el directorio.

/usr/share/openldap/migration/migarte_base.pl > base.ldif

Se procede a insertar la información generada en el directorio utilizando lo siguiente:

Ldapadd –x –W –D 'cn=root, dc=uquil, dc=com' –h 127.0.0.1 –f base.ldif

Comprobación del Servidor:

Antes de configurar el sistema para utilizar LDAP, es conveniente verificar que todo funciona correctamente. El siguiente mandato debe devolver toda la información del directorio solicitado (dc=uquil, dc=com).

Ldapsearch –x –b 'dc=uquil,dc=com' '(objectclass=)'*

2.6.4. Configuración de Clientes

Se definen los valores para los parámetros host y base a fin de establecer hacia que servidor y a que directorio

conectarse. Para fines prácticos, estos parámetros se modifican en /etc/openldap/ldap.conf

```
HOST 192.168.3.1
BASE dc=uquil,dc=com
TLS_CACERTDIR /etc/openldap/cacerts
```

Authconfig (modo texto):

Se habilitan las casillas utilizar LDAP y utilizar autenticación LDAP, (Gráfico #19) pulse la tecla Tab hasta Siguiente y pulse la tecla Enter y verifique que los datos del servidor y el directorio a utilizar sean los correctos (Gráfico #20).

Gráfico -19-
Authconfig, Pantalla Principal

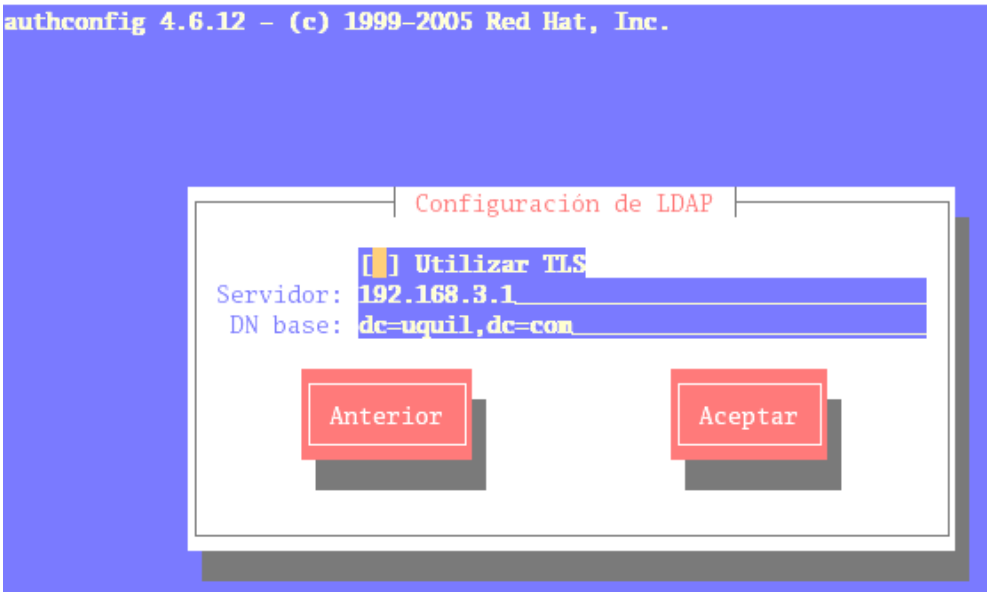


Gráfico -20-
Authconfig, Configuración LDAP

Authconfig -gtk (modo gráfico)

Utilizando authconfig-gtk, se deben habilitar las casillas de Soporte LDAP. Antes de cerrar la ventana en la pestañas de Información del usuario y Autenticación. (Gráfico #21) Antes de dar clic en Aceptar, hacer clic en Configurar LDAP y verificar que los datos del servidor y el directorio a utilizar sean los correctos (Gráfico #22).

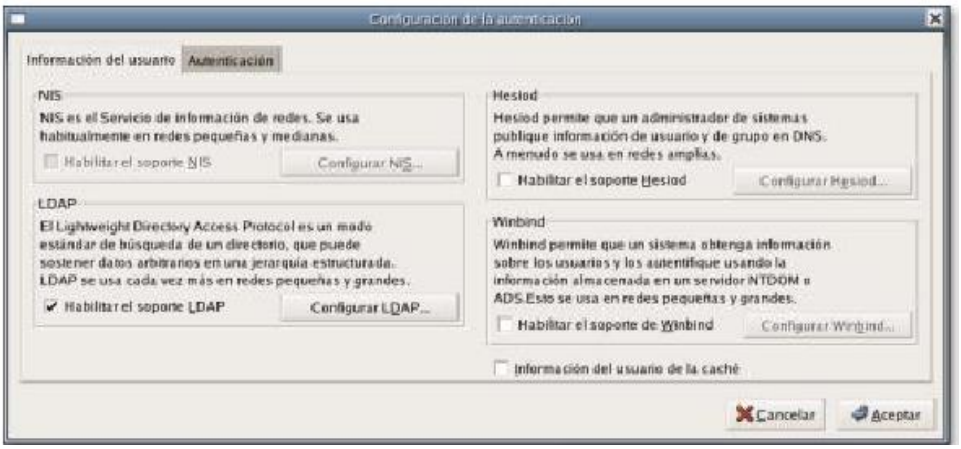


Gráfico -21-
Authconfig-gtk, Pestaña de Información del Usuario

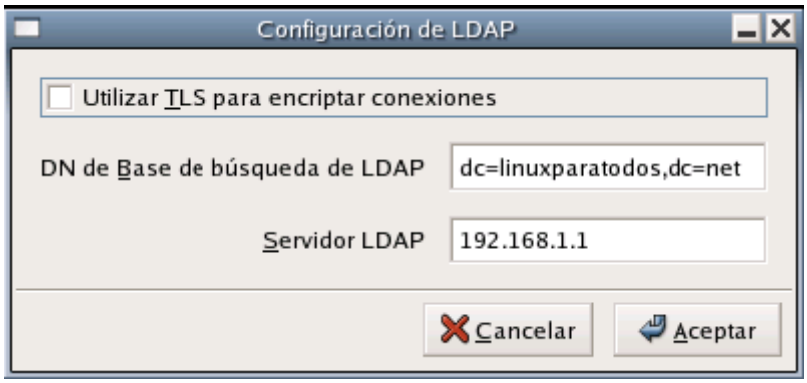


Gráfico -22-
Authconfig-gtk, Configuración LDAP

2.6.4. Configuración de OpenLdap en modo seguro

Hasta ahora lo que hemos hecho ha sido configurar un servidor OpenLDAP para que funcione por el puerto estándar, es decir, el puerto 389, que realiza las conexiones sin cifrar. Para configurar el servidor en modo seguro, debemos tenerlo funcionando correctamente por el puerto 636.

OpenLDAP tiene dos formas de comunicación segura, SSL y TLS. SSL opera en otro puerto, el 636 y, desde el principio, realiza las comunicaciones encriptadas en modo seguro. TLS, utiliza el puerto estándar 389 para iniciar las comunicaciones y que, después, cambia a modo seguro a través del citado puerto 636.

Es conveniente tener los dos tipos activados, ya que no todas las aplicaciones cliente soportan este método, e incluso hay aplicaciones que no soportan las transmisiones seguras.

Para activar los puertos 389 y 636 es necesario digitar las siguientes líneas:

slapd –h "ldaps:// ldap://127.0.0.1:978"

Se debe verificar que slapd este funcionando por ambos puertos y para ellos se hace lo siguiente*:*

netstat –alpn | *grep slapd*

Si sale algo similar al gráfico #23, significa que esta escuchando por ambos puertos.

```
[root@localhost ~]# netstat -alpn |grep slapd
tcp        0      0 0.0.0.0:389                 0.0.0.0:*
EN      7274/slapd
tcp        0      0 0.0.0.0:636                 0.0.0.0:*
EN      7274/slapd
tcp        0      0 :::389                      :::*
EN      7274/slapd
tcp        0      0 :::636                      :::*
EN      7274/slapd
unix  2      [ ]         DGRAM                    166325 7274/slapd
```

Gráfico -23-
Comprobación de Puertos abiertos

Para las transmisiones seguras hace falta un certificado en formato PEM para la llave SSL. Normalmente, se necesitaría que una organización dedicada a generar certificados nos proporcione uno, pero como, normalmente, el OpenLDAP lo vamos a usar en la red local, no a través de internet, ni para hacer comercio electrónico, basta con que la generemos nosotros mismos. Es decir vamos a generar un Certificado autofirmado.

Para generar el certificado, desde línea de comandos vamos a acceder a la ruta etc/openldap/cacerts y digitaremos lo siguiente:

Openssl req – x509 –nodes -newkey –days 730 –out ***slapd.crt*** *-keyout* ***slapd.key***

Lo anterior solicitará se ingresen varios datos:

- Código de dos letras para el país
- Estado o Provincia
- Ciudad
- Nombre de la empresa o razón Social
- Unidad o sección
- Nombre del anfitrión
- Dirección de correo

La salida de Vuelta sería similar al siguiente gráfico:

```
Generating a 1024 bit RSA private key
.................++++++
.++++++
writing new private key to 'dovecot.key'
-----
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.

There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]: EC
State or Province Name (full name) [Berkshire]: Guayas
Locality Name (eg, city) [Newbury]: Guayaquil
Organization Name (eg, company) [My Company Ltd]:
 sds
Organizational Unit Name (eg, section) []: servidor
Common Name (eg, your name or your server's hostname) []:
 uquil.com
Email Address []: elweka@hotmail.com
```

Gráfico -24-
Creación de certificado

El certificado solo será válido cuando el servidor LDAP sea invocado con el nombre definido en el campo **Comon Name.** Es decir solo podrá utilizarlo cuando se defina uquil.com como servidor LDAP con soporte SSL/TLS

Es necesario que todos los ficheros de claves y certificados tengan permisos de acceso de solo lectura para el usuario ldap:

*chown ldap.ldap /etc/openldap/cacerts/slapd.**

*chmod 400 /etc/openldap/cacerts/slapd.**

Parámetros slapd.conf:

Se debe descomentar los parámetros TLSCACertificate, TLSCertificateFile y TLSCertificateKeyFile estableciendo las rutas hacia el certificado y clave. Editar el archivo slapd.conf que se encuentra en la ruta /etc/openldap/ y hacer las siguientes modificaciones:

```
TLSCACertificateFile /etc/openldap/cacerts/slapd.crt
TLSCertificateFile /etc/openldap/cacerts/slapd.crt
TLSCertificateKeyFile /etc/openldap/cacerts/slapd.key
```

Es necesario reiniciar el servicio ldap para que hagan efectos los cambios realizados.

service ldap restart

2.7 Breve resumen de la funcionalidad de la aplicación “Servicio de Direcorio (LDAP)”

El Servicio de Directorio (LDAP) proporciona un interfaz de uso fácil. Permite que los usuarios vean un árbol de directorio LDAP de una manera jerárquica. Así como también permite hacer inserciones, eliminaciones y modificaciones siempre y cuando el usuario sea administrador. Los objetos de LDAP se exhiben bajo la forma de árbol y todos los atributos de las entradas bajo la forma de tabla.

El estado actual de la aplicación se observa en la barra de estado. En esta barra se muestran mensajes al seleccionar una entrada, añadirla, modificarla o borrarla. Los mensajes de estado son de color negro, las advertencias en amarillo y mensajes de error en rojo.

2.8. Aplicación “Servicio de Directorio (LDAP) ” paso a paso

Para ejecutar la aplicación SDS (Servicio de Directorio seguro), se debe dar doble clic en el archivo ejecutable *browser.jar,* que se encuentra en la carpeta que se proporciona en el cd, posteriormente observaremos una pantalla de carga de la aplicación (Gráfico #25).

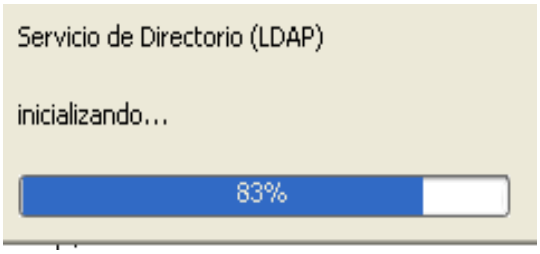


Gráfico -25-
Cargando la aplicación

Una vez que la aplicación está inicializada, se presenta la pantalla de conexión en la cual se captura parámetros de comunicación de la aplicación con el servidor de directorio LDAP, definiendo una sesión.

Conexión con el Servidor.- Se observa dos pestañas Lista de sesiones y Cconexión rápida.

- *Lista de Sesiones*: Aquí el usuario tiene la oportunidad de crear, editar, copiar, borrar y renombrar sesiones;

colocando en su sesión los datos necesarios para la conexión con el servidor de directorio donde se encuentra almacenada la información.(Gráfico #26).

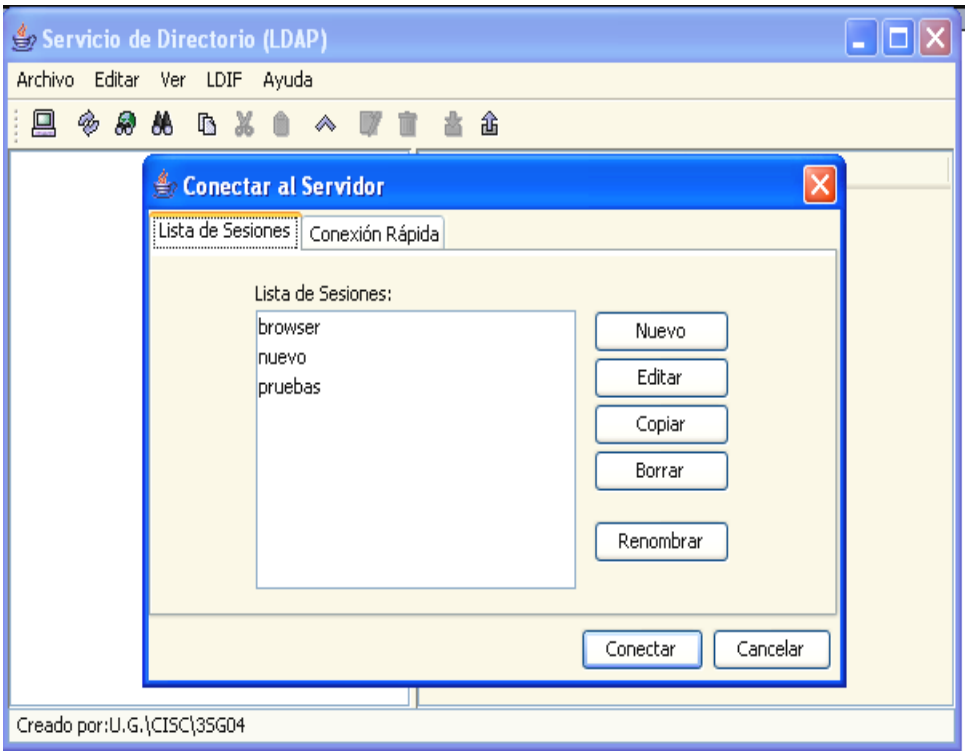


Gráfico -26-
Conectar con el Servidor – Pestaña Lista de Sesiones

Cuando seleccionamos nuevo o editar aparece "Editar Sesión", en esta ventana colocaremos los datos necesarios para la conexión, si lo haremos de forma segura seleccionando SSL o con por el puerto por defecto el 389; además deberemos especificar si somos usuarios anónimos o administradores, luego de digitar

estos datos debemos guardar estos parámetros y procedemos a conectarnos (Gráfico # 27).

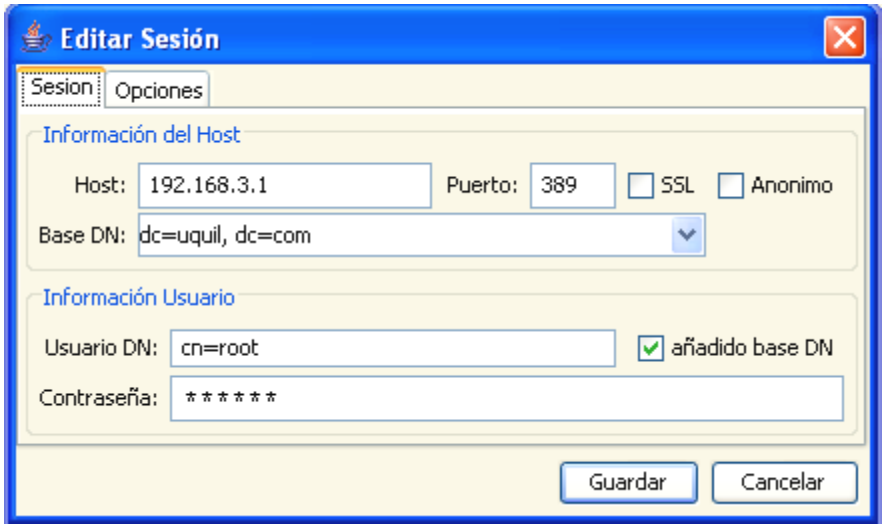


Gráfico -27-
Editar Sesión

- *Conexión rápida:* El usuario se conecta sin necesidad de crear una sesión, pero deberá ingresar los datos de configuración como: host (ejemplo: 192.168.3.1), dn (ejemplo: dc=uquil, dc=com), puerto (ejemplo: 389 o 636), los mismos que pudimos observar en “Editar Sesión” (Gráfico #28).

Gráfico -28-
Conectar con el Servidor – Pestaña Conexión Rápida

Si es un usuario Administrador este podrá realizar las operaciones de ingreso, modificación, eliminación, búsquedas de entradas del árbol de directorio. Si es un usuario anónimo este solo podrá realizar operaciones de búsquedas y deberá activar la casilla Anónimo.

Además el usuario podrá conectarse mediante un puerto seguro 636 eligiendo la opción SSL, caso contrario se conectará por defecto mediante el puerto 389.

Si nos conectamos de manera segura por el puerto 636 nos aparecerá una ventana, en la que debemos aprobar un certificado autofirmado que esta creado en el servidor

Certificado Autofirmado.- En esta ventana el usuario tendrá tres opciones: Aceptar el certificado en esta sesión, aceptarlo siempre, y no aceptarlo. En caso de elegir la tercera opción la aplicación no podrá conectarse en modo seguro, y nos mandará un error en la conexión (Gráfico #29).

Gráfico -29-
Certificado Autofirmado

Visualización del árbol de directorio: Una vez conectado el usuario por medio de la aplicación con el servidor, podemos observar la jerarquía del árbol del Servidor LDAP, sus entradas en forma visual, siendo de esta manera más amigable que la visualización que ofrece directamente el servidor.

En el lado izquierdo de la ventana podemos apreciar la estructura del árbol y a la derecha visualizamos los atributos y valores correspondiente al nodo del árbol. Además,

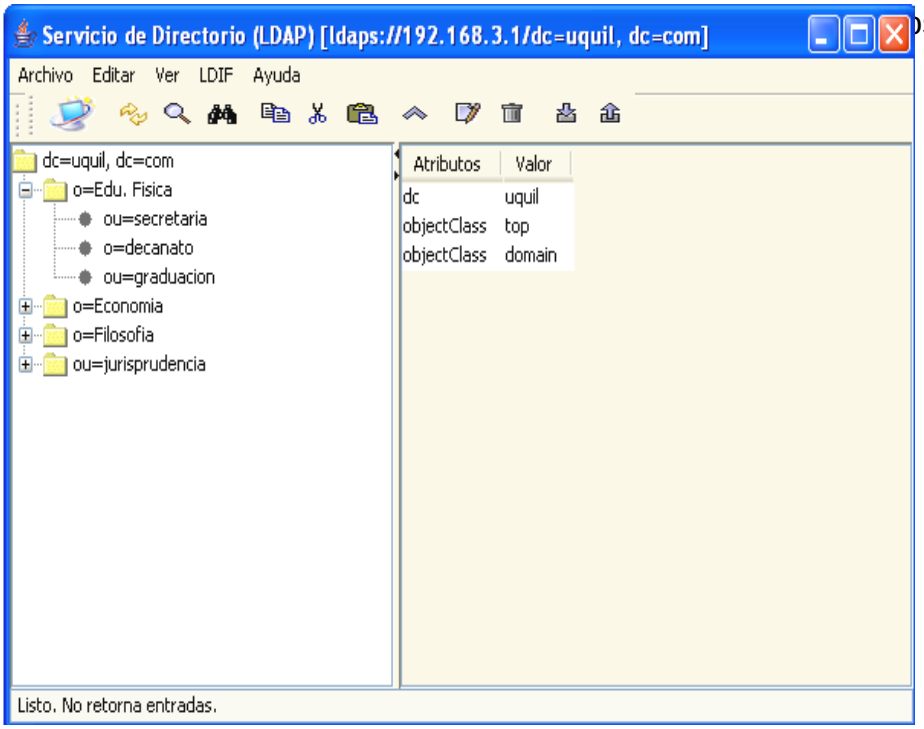


Gráfico -30-
Visualización del Árbol de Directorio

En el menú podemos ver diferentes opciones desplegables, que a continuación explicaremos.

Opción Archivo: Esta opción nos permite conectarnos, desconectarnos, reconectar, y grabar configuración. (Gráfico #31).

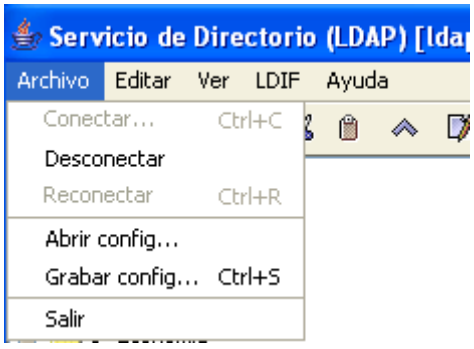


Gráfico -31-
Opciones de Menú Archivo

Opción Editar: Permite agregar, eliminar atributos y entradas o editarlos si ya están creados. También puedes copiar una entrada o moverla a otra rama del árbol (Gráfico #32).

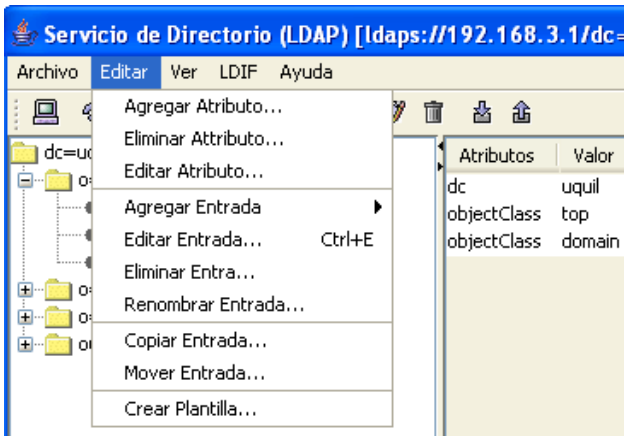


Gráfico -32-
Opciones de Menú Editar

Opción Ver: Permite ver las características de las entradas o atributos seleccionados. También permite buscar un DN, refrescar una entrada, ordenar la visualización del árbol o de la tabla en modo ascendente o descendente (Gráfico #33).

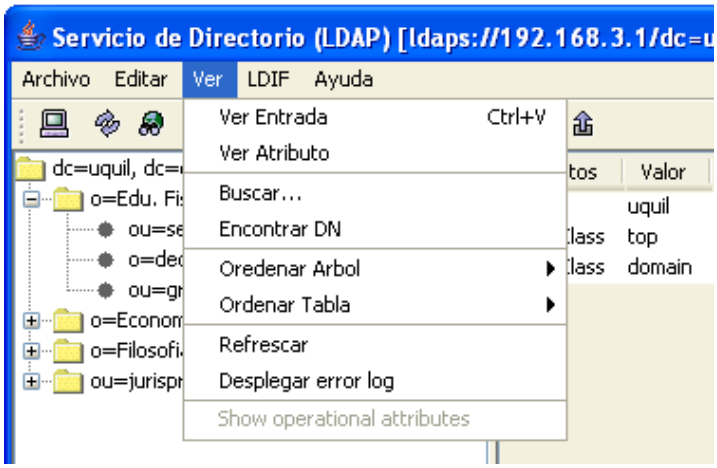


Gráfico -33-
Opciones de Menú Ver

Opción LDIF: La entrada LDIF permite importar o exportar un fichero LDIF (Gráfico #34).

Gráfico -34-
Opciones de Menú LDIF

Si se exporta un archivo LDIF se puede elegir entre exportar sólo la entrada seleccionada, sólo la entrada con sus hijos intermedios o la entrada con todos los hijos. Si se importa tendremos las opciones de solo añadir, solo actualizar o añadir y actualizar (Gráfico #35),

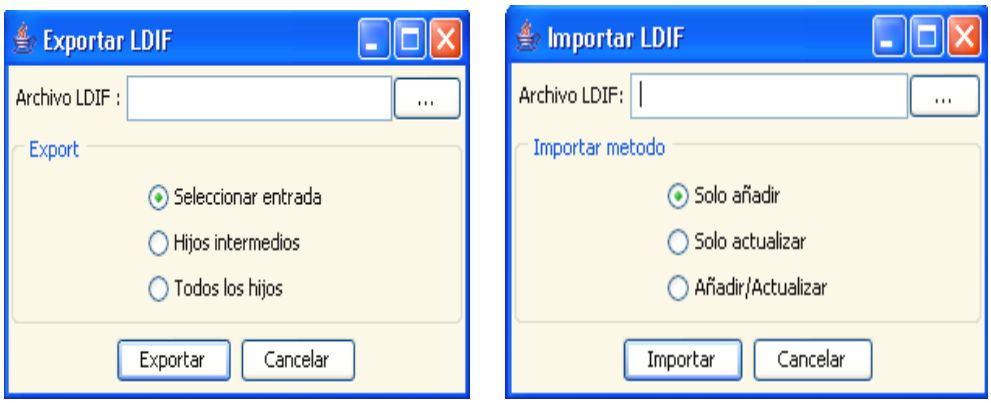


Gráfico -35-
Exportar e Importar LDIF

Ingresar Entrada: Después de revisar toda la visualización del directorio LDAP, continuaremos con la opción añadir entradas en el

directorio LDAP. Nuestra aplicación permitirá crear entradas de organizaciones (facultades, administración, rectorado, etc,), unidades organizacionales (departamentos de cada organización), personas (docentes, personal administrativo y de servicios, alumnos) y organizaciones de personas (anonimos, administradores).

Se accede a la opción Agregar Entrada del menú Editar, luego procedemos a elegir el tipo de entrada, los cuales se encuentran en un fichero templates, definiendo el tipo de entrada según la selección que realizó: Organization (Organización), Person (Persona), OrganizationalPerson , OrganizationalUnit(Unidad Organizativa)

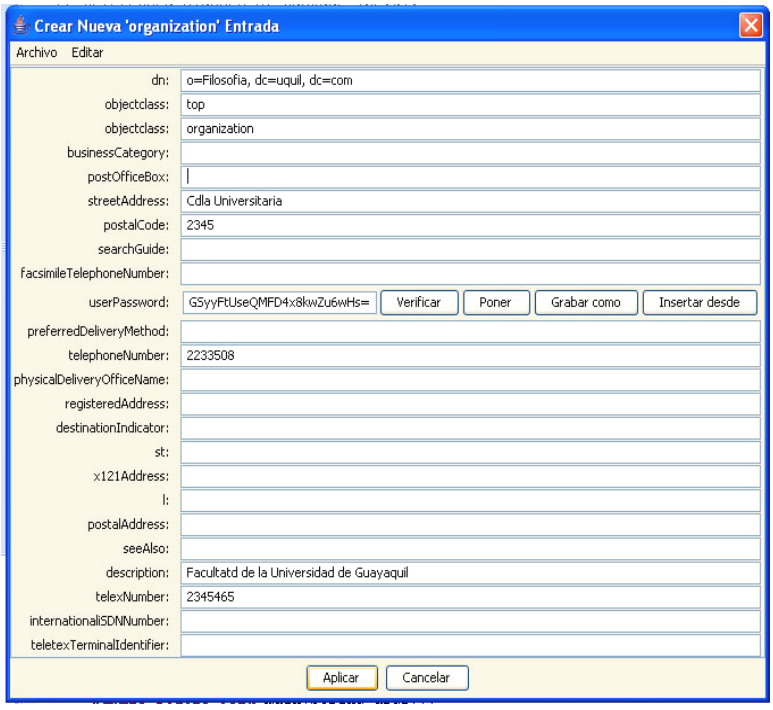


Gráfico -36-
Crear Nueva Entrada

Una vez seleccionado el tipo entrada que se desea hacer se deberá ingresar datos como nombre, número de teléfono, descripción, fax, en este caso la nueva entrada es la facultad de filosofía (Gráfico #36)

Después de presionar el botón aplicar se observará el nodo ingresado con sus respectivos atributos y valores.

Consulta o Búsqueda.- Si se desea realizar una búsqueda se lo puede hacer presionando el botón de buscar  o haciendo clic la opción Ver – Buscar. Aparecerá una ventana en la que se muestra automáticamente la raíz (DN: dc=uquil, dc=com) del árbol (Gráfico #37). En Filtrar se deberá indicar el criterio de búsqueda, por ejemplo objectclass=*, objectclass=top, objectclass=person,. objectclass=organization.

En atributos, se indicará los valores o atributos que se buscan de determinada entrada. Teniendo dos opciones de búsqueda:

- Un nivel: Muestra en la parte inferior la raíz del nodo buscado.

- Sub niveles de árbol: Muestra el nodo a buscar y los nodos del cual se deriva.

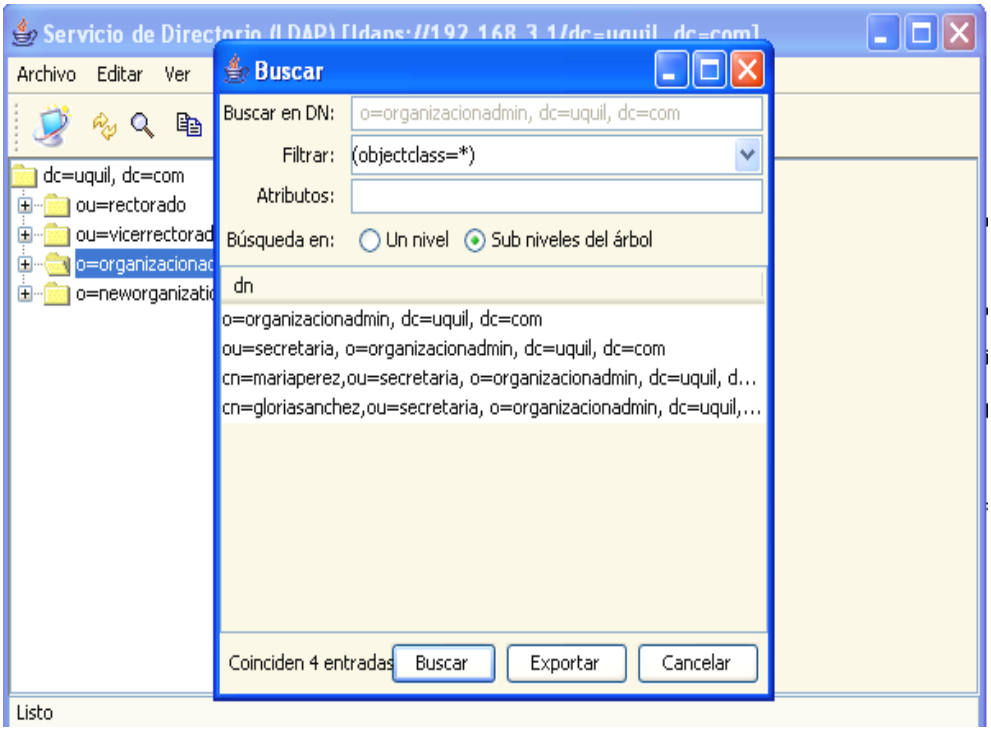


Gráfico -37-
Ventana de Búsqueda

Editar.- Para realizar modificaciones a una entrada o atributo se procede a dar clic en la opción Editar – Editar Atributo o Entrada, también presionando clic el botón editar que mostrará una interfaz que permitirá la edición de los nodos del árbol de directorio, se podrá modificar los valores de los atributos anteriormente ingresados (Gráfico #38).

Al presionar el botón aplicar se observará los valores de los atributos modificados.

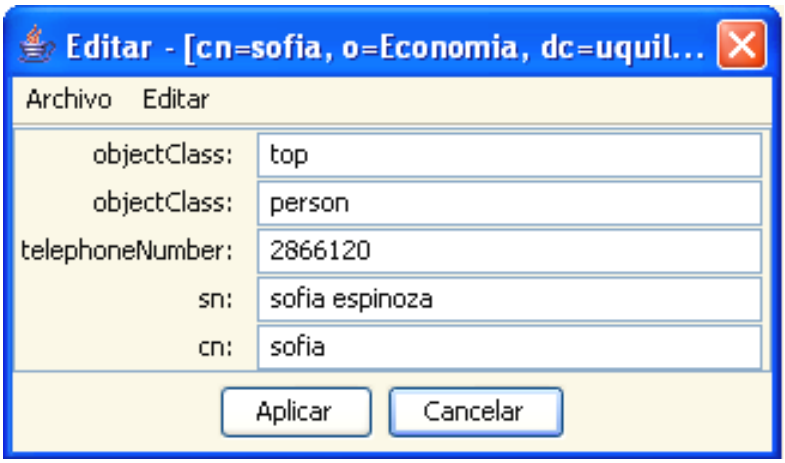


Gráfico -38-
Ventana de Editar

Eliminación: si se requiere hacer eliminaciones tanto de entradas como de atributos se procede haciendo clic en en el botón  o dando clic en la opción editar – Eliminar Atributo o Eliminar Entrada dependiendo de lo que se quiera eliminar. Esta pantalla permitirá al usuario eliminar el nodo. Si el nodo a eliminar tiene sub niveles tiene que estar seleccionada la opción con hijos. Caso contrario la aplicación mostrará un mensaje de error en la barra de estado y no permitirá la eliminación del nodo. (Gráfico #39)

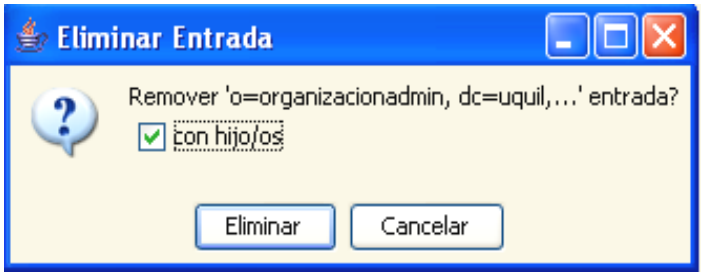


Gráfico -39-
Eliminar Entrada

Podemos realizar otras operaciones como copiar entradas, moverlas, crear plantillas de entradas estas opciones las podemos encontrar en el menú Editar, así como también podemos renombrar las entradas, agregar atributos a una entrada.

Mover entrada

Esta opción nos permite trasladar una entrada de un dn a otro. Denominamos dn la ruta de ubicación de la entrada dentro de la jerarquía del árbol del Servidor de Directorio LDAP.

Seleccionamos la entrada que deseamos mover y presionado clic derecho lo arrastramos hasta el dn destino de la entrada. Al realizar esta acción se presenta la ventana “Mover Entrada” en la cual nos presenta el dn donde se encuentra actualmente la entrada y el dn

destino a donde se colocará la entrada, antes de aceptar la acción debemos seleccionar si deseamos el traslado de la entrada con los nodos hijos y si deseamos que sea un nuevo dn (Gráfico # 40).

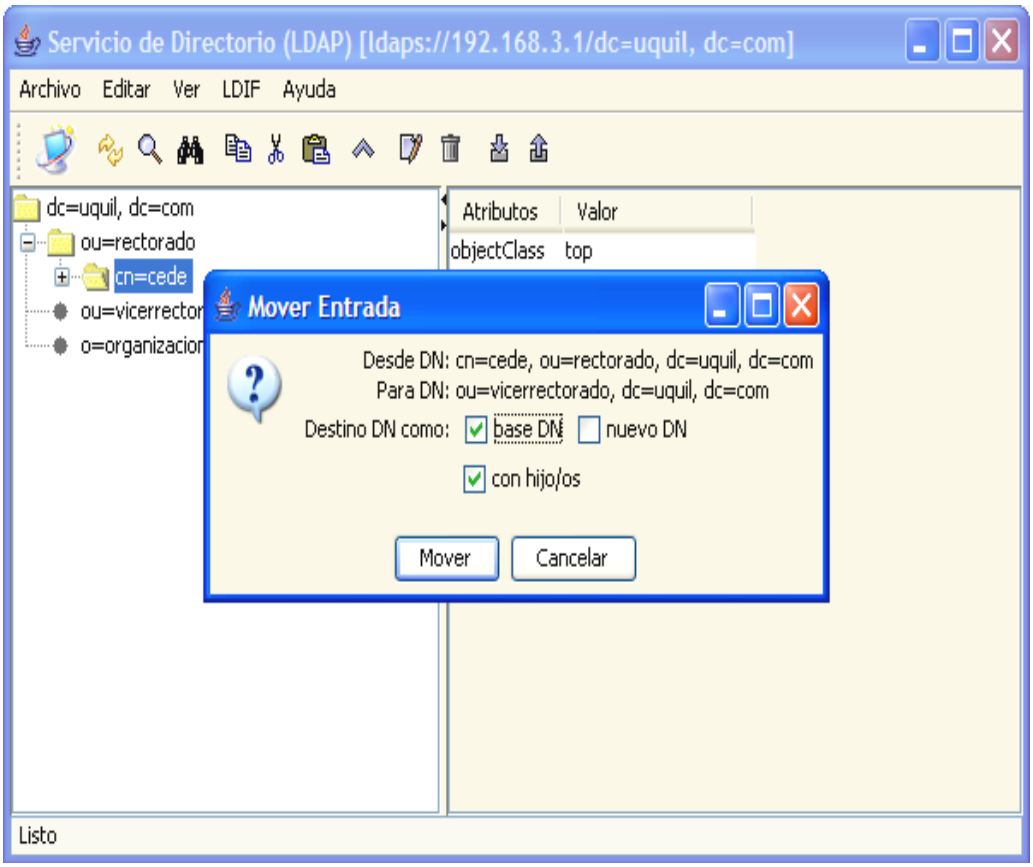


Gráfico -40-
Mover Entrada

Copiar entrada

Primero seleccionamos la entrada que deseamos copiar, luego en el menú Editar seleccionamos la opción Copiar Entrada aparecerá la ventana

de “Mover Entrada” pero esta contiene una diferencia de la anterior el destino dn se digita, es decir colocaremos la ubicación donde queremos copiar la entrada seleccionada. (Gráfico # 41)

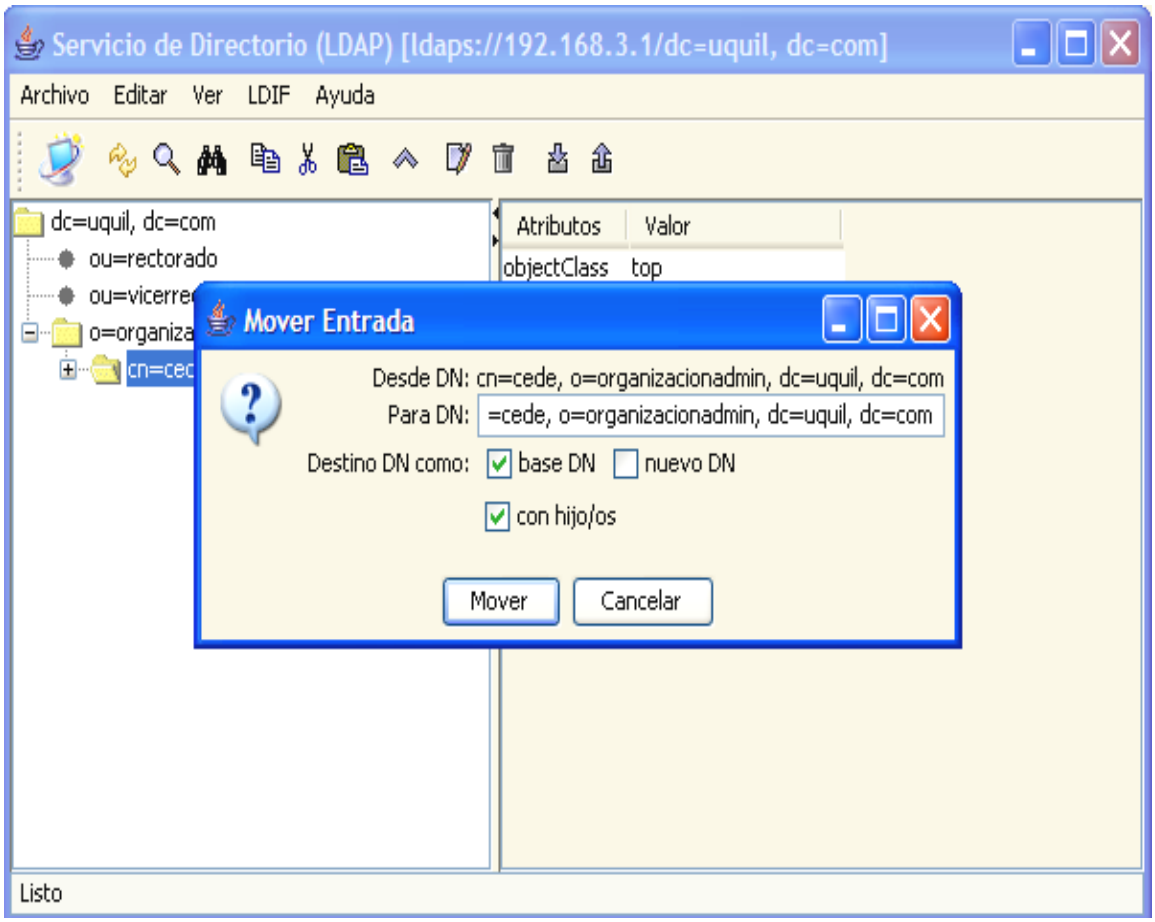


Gráfico -40A-
Copiar Entrada

Nota: Luego de Mover un atributo y/o copiar una entrada debemos refrescar el árbol desde el dn raiz

Crear Platillas

Crea un archivo de formato templates el cual contiene los atributos de la entrada que personalizamos.

Para llevar a cabo esta operación primero debemos seleccionar una entrada del directorio del cual deseamos copiar la plantilla para personalizarla colocando el nombre que deseemos. (Gráfico # 41)

Gráfico -41-
Crear Plantilla

Para comprobar que la plantilla fue creada, creamos una nueva entrada y al seleccionar el tipo de entrada observaremos el nombre de la plantilla que creamos. (Gráfico # 42)

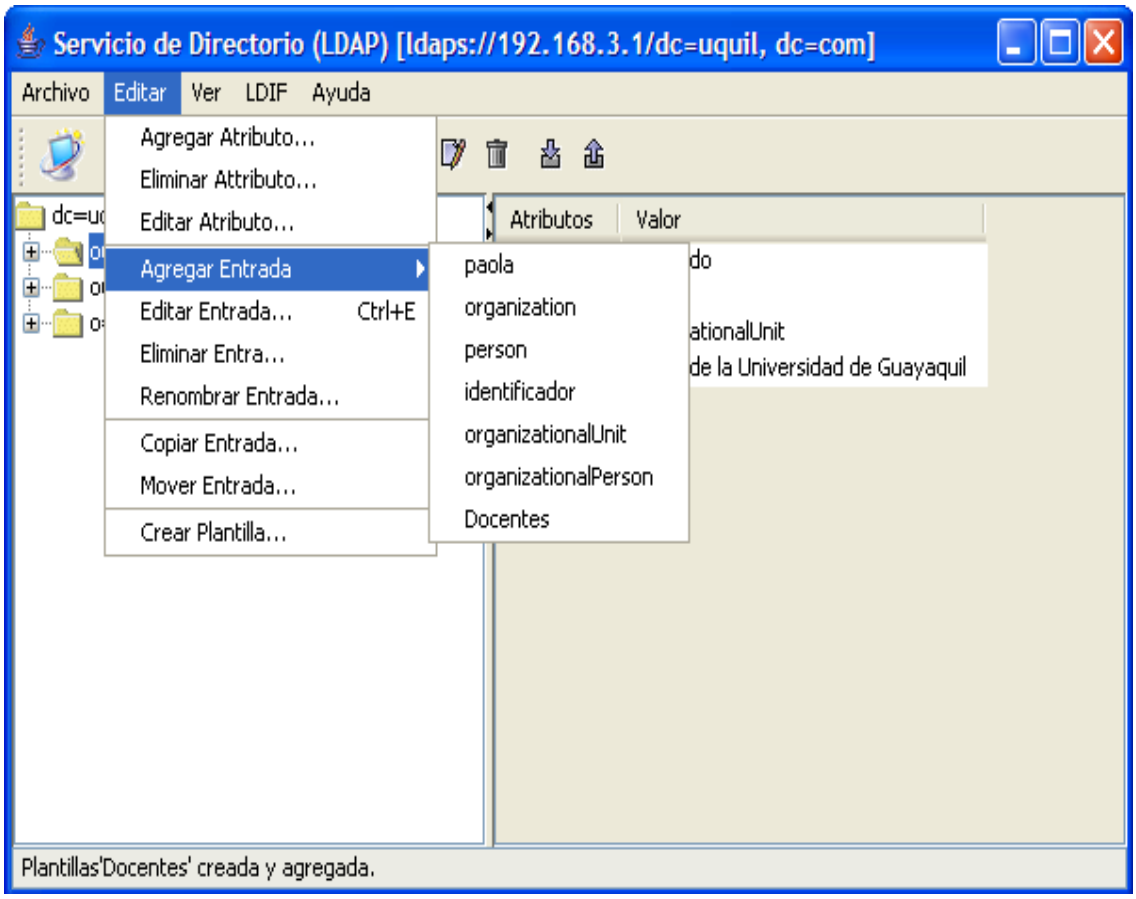


Gráfico -42-
Tipos de entradas - Plantillas creada

Agregrar un atributo

Seleccione la entrada a la cual desea agregarle el atributo, hacemos clic en menú Editar la opción “Agregar atributo” o clic derecho Administrador – “Agregar Atributo” y se presenta la siguiente pantalla (Gráfico #43)

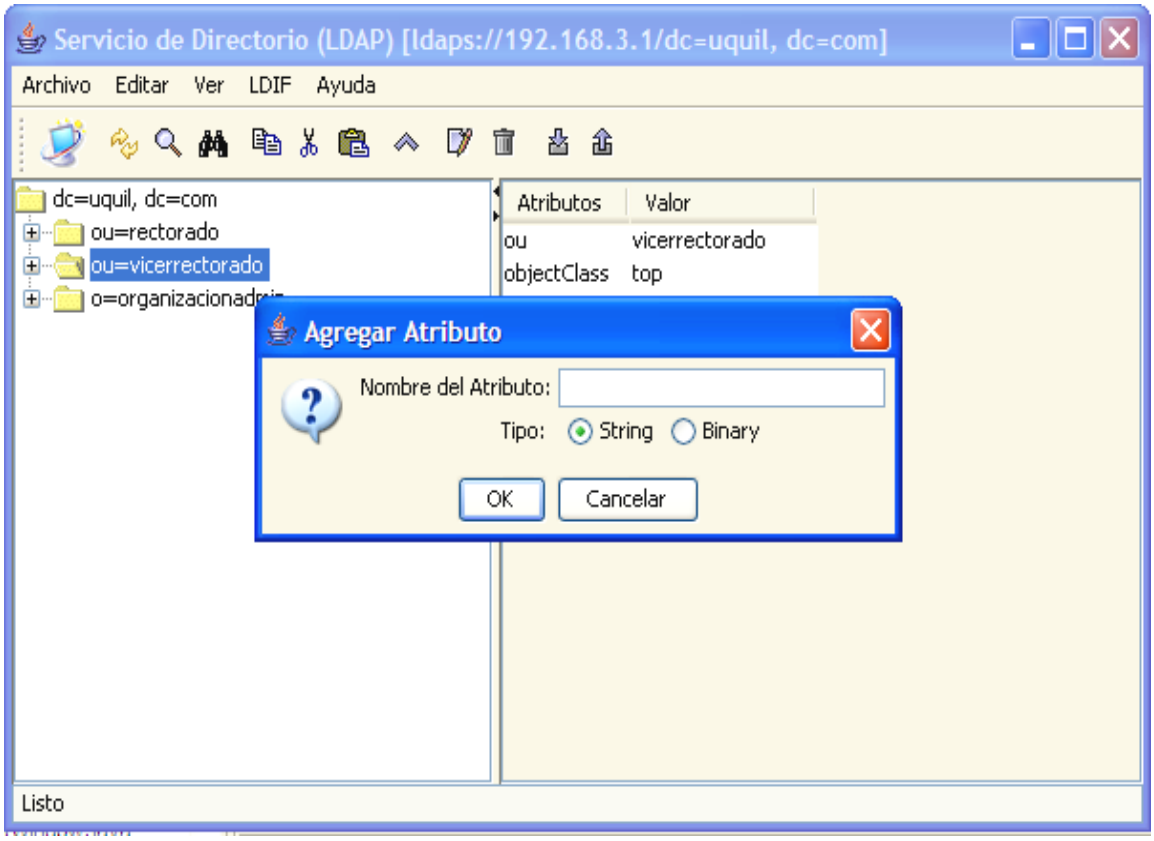


Gráfico -43-
Agregar Atributo

Debemos seleccionar el tipo de atributo si es string (soporta caracteres) o binario (soporta solo números), colocamos el nombre del atributo que deseamos agregar tal y cual lo soporta el servidor de directorio, por eso esta acción la debe realizar el administrador.

Error Log

En esta opción que ofrece el menú Ver, se muestran los mensajes de error que el servidor genera cuando existe un fallo en las operaciones que la aplicación realiza sobre él. (Gráfico # 44)

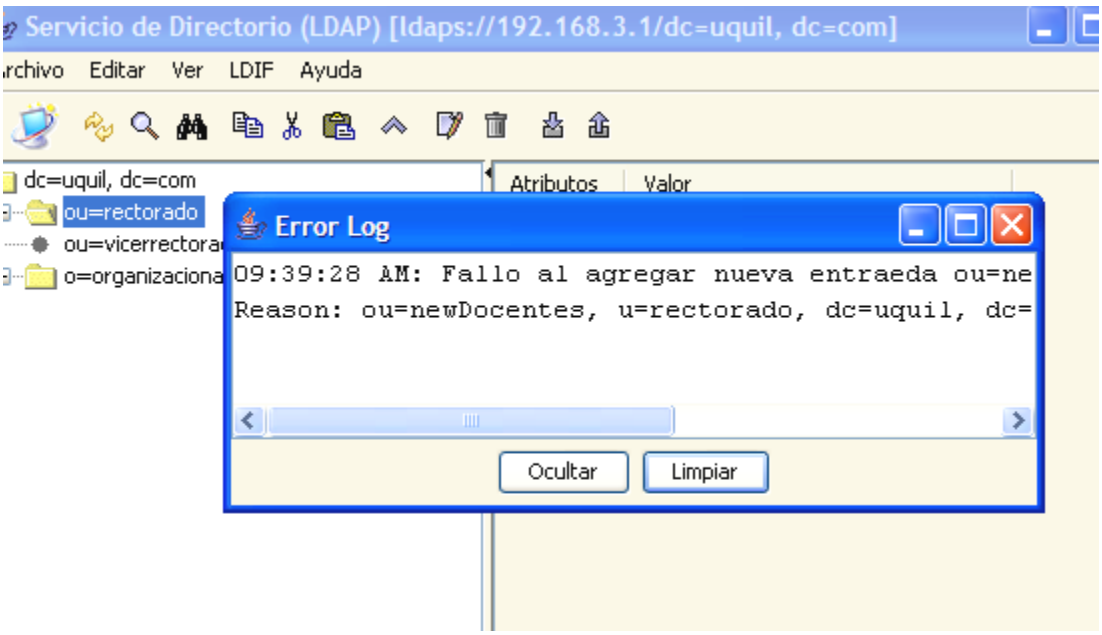


Gráfico -44-
Error Log

Adicionalmente en el menú Ayuda encontramos las opciones de General, Uso, Notas y About.

La opción General, como su nombre hace referencia ofrece información de las características generales de la aplicación para que usuario tenga una noción de la aplicación.(Gráfico # 45)

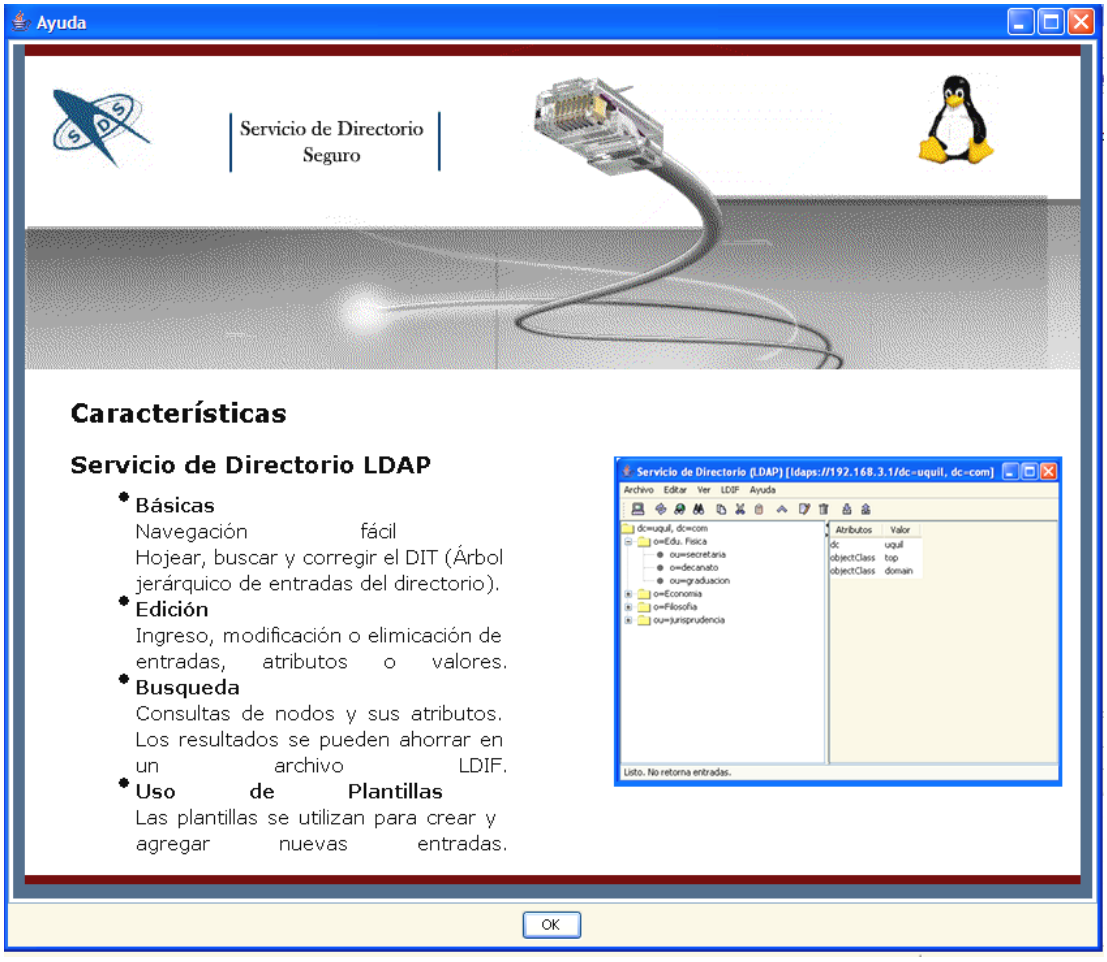


Gráfico -45 -
Ayuda - General

La opción Uso, presenta un pequeño manual para guiar al usuario de la aplicación (Gráfico # 46)

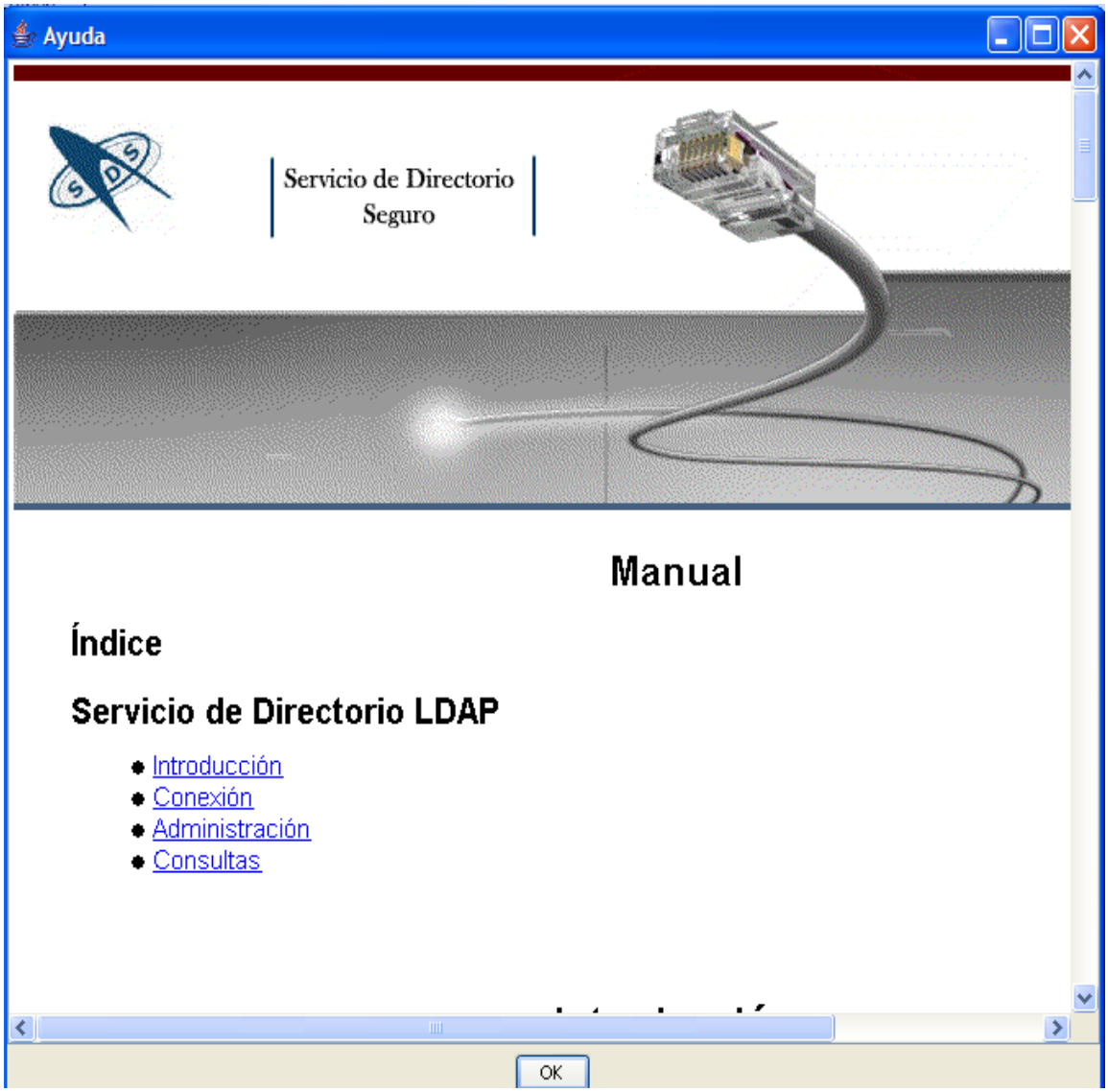


Gráfico -46-
Ayuda - Uso

La opción Notas, da una breve explicación de la importación y exportación de los archivos LDIF. (Gráfico # 47)

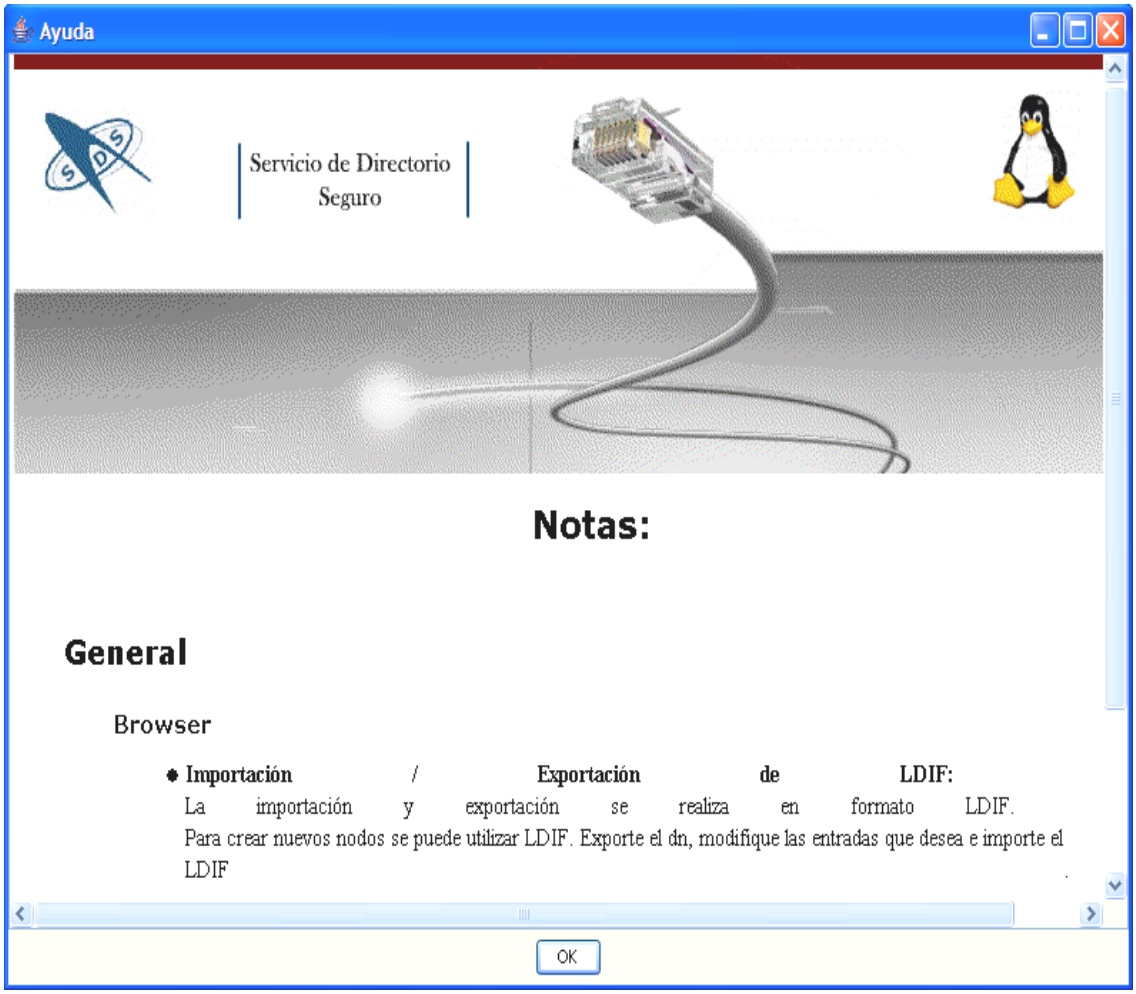


Gráfico -47-
Ayuda - Notas

Y finalmente encontramos la opción About, que hace referencia a donde se presenta el nombre de la aplicación, la versión y la dirección web donde nos pueden contactar.(Gráfico # 48)

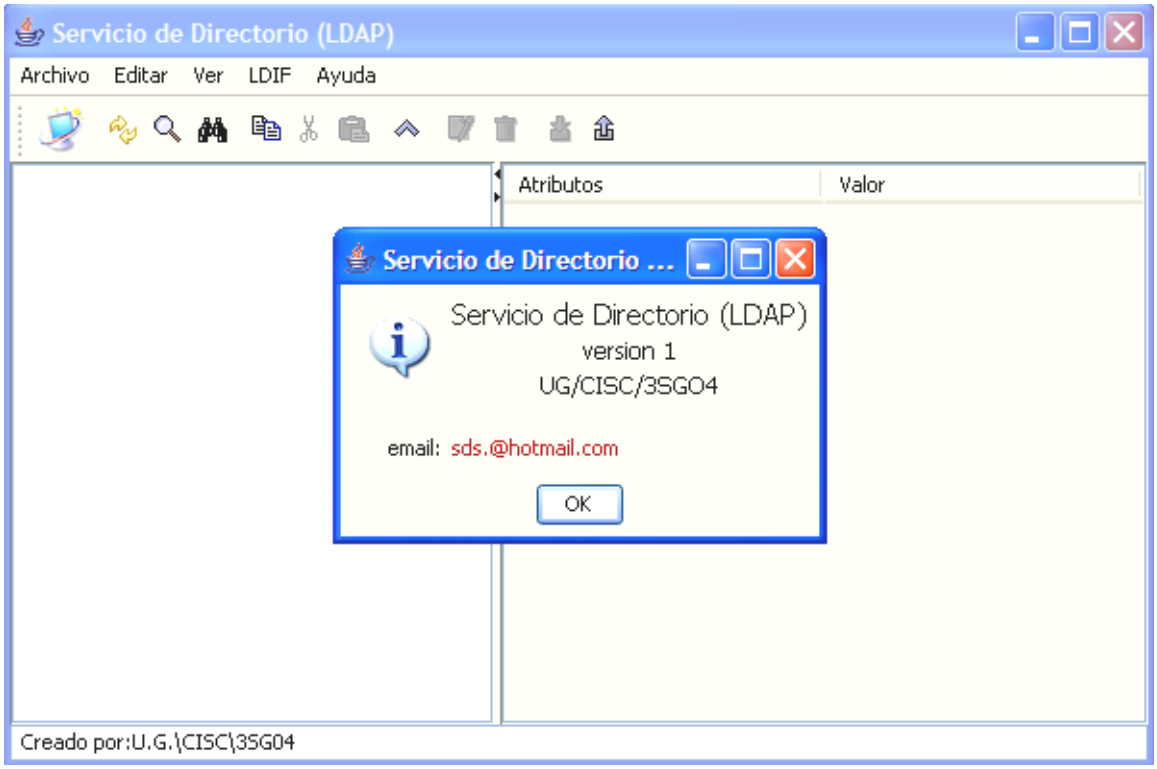


Gráfico -48-
Ayuda - About