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

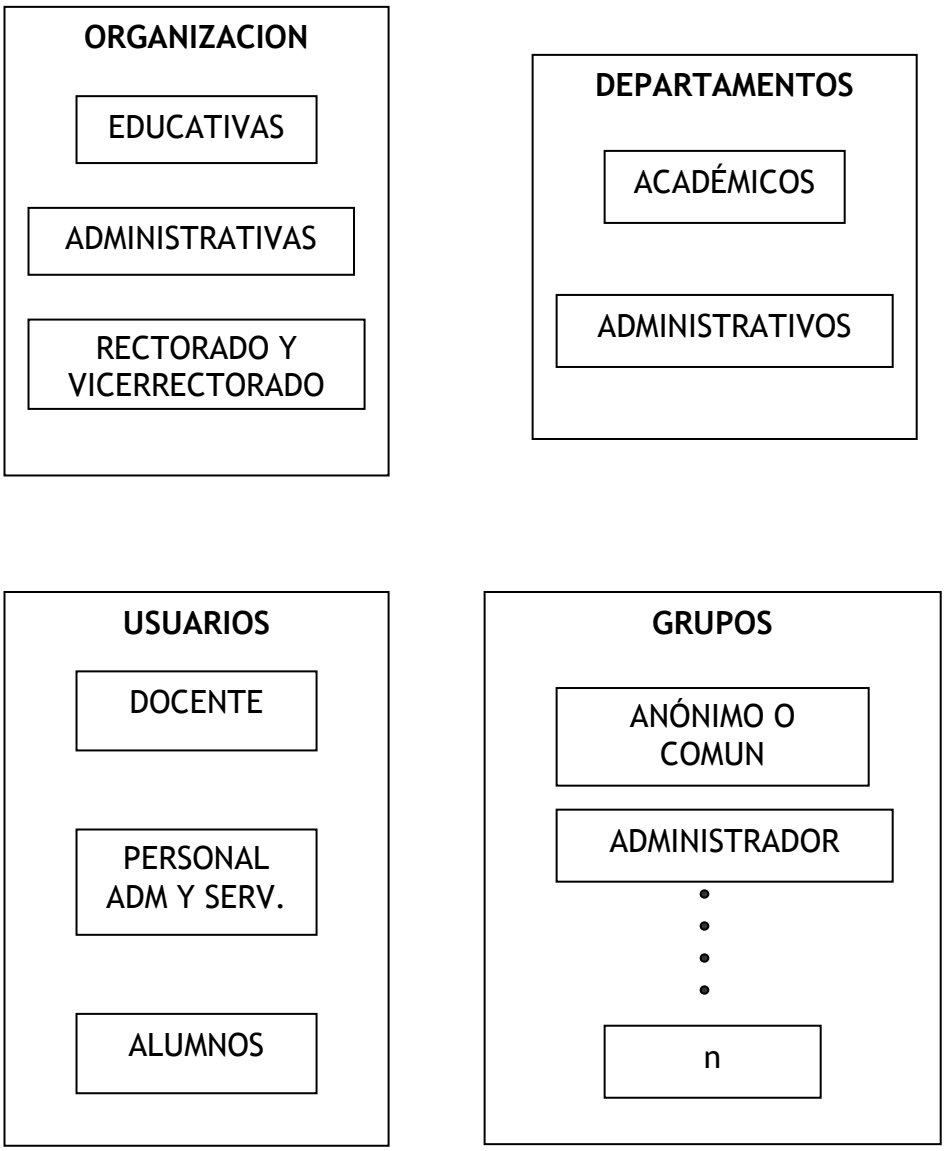


Gráfico -1-
Definición de Generalización

1.1.3 Diagrama de Relaciones de Objetos

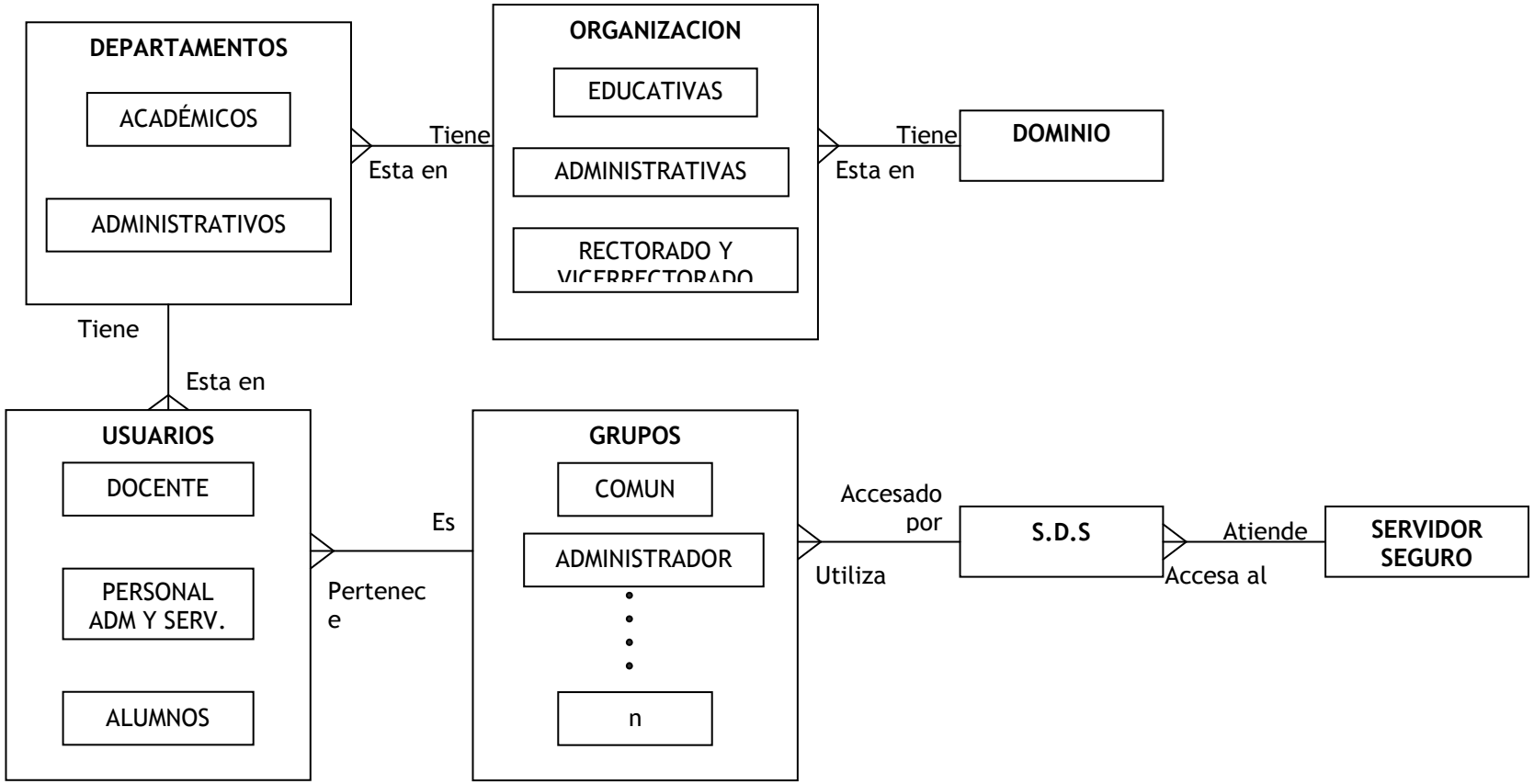
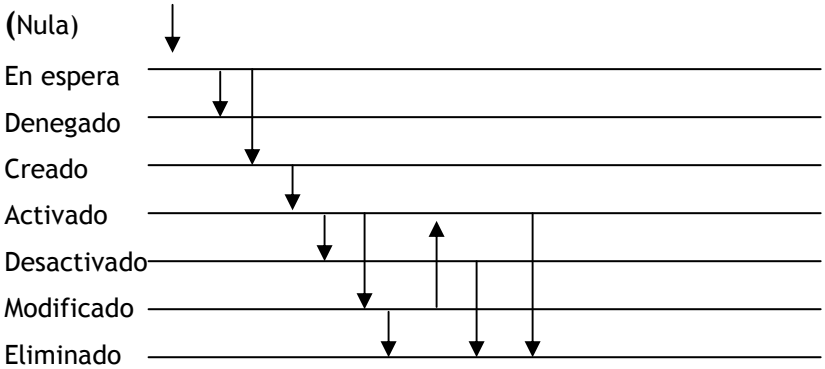


Gráfico -2-
Relación de Objetos

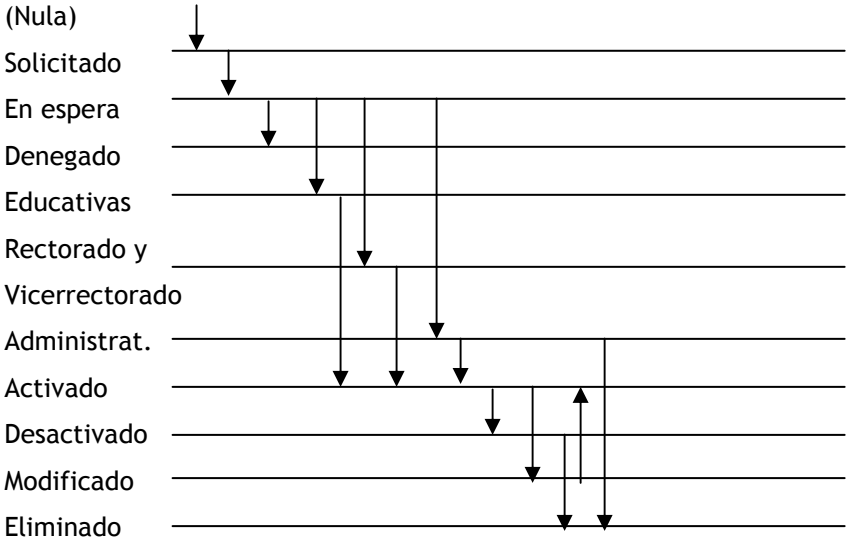
1.2 Análisis del Comportamiento de Objetos (ACO)

1.2.1 Estado de Objetos y cambio de los mismos

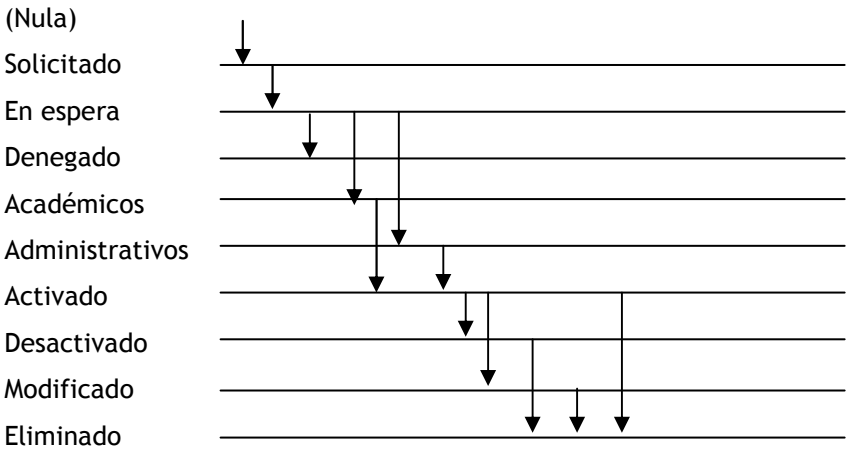
DOMINIO



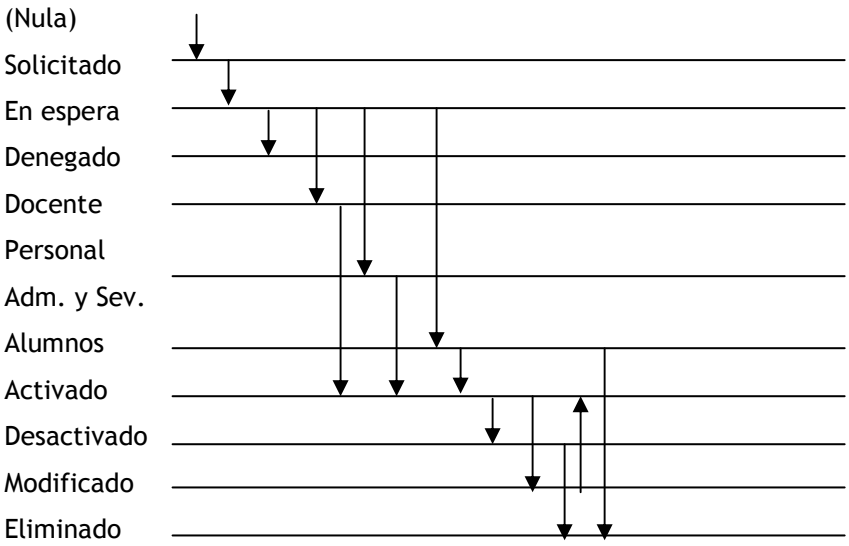
ORGANIZACION



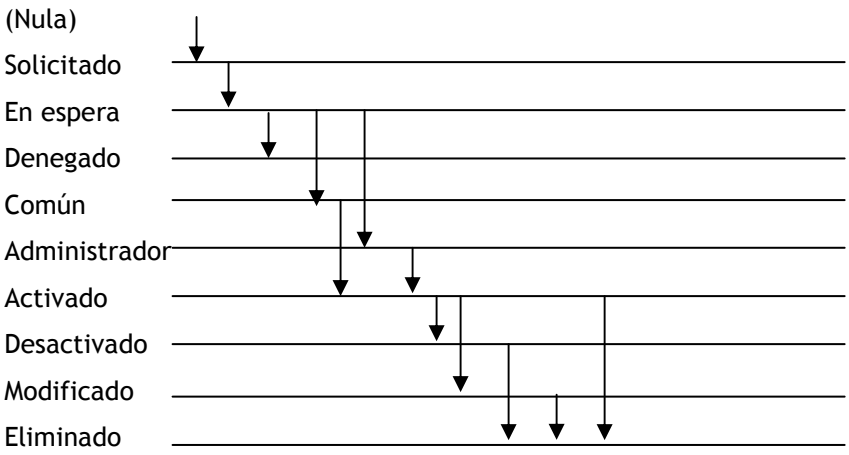
DEPARTAMENTOS



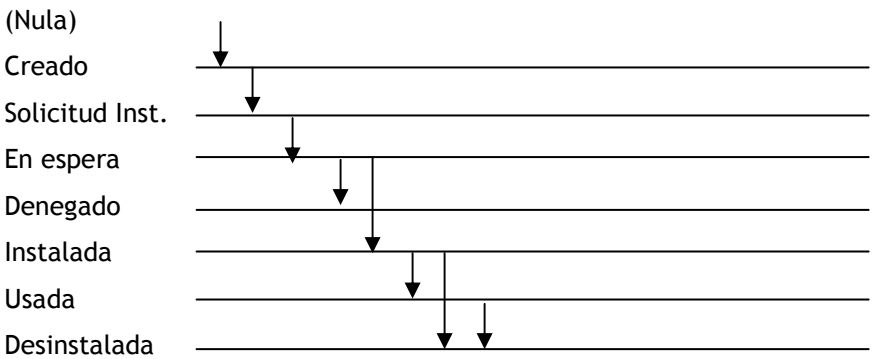
USUARIOS



GRUPOS



S.D.S.



SERVIDOR SEGURO

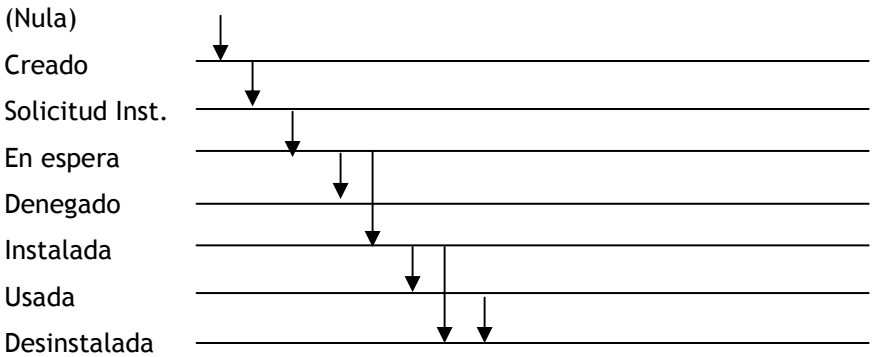


Gráfico -3-
Estados de Objetos

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

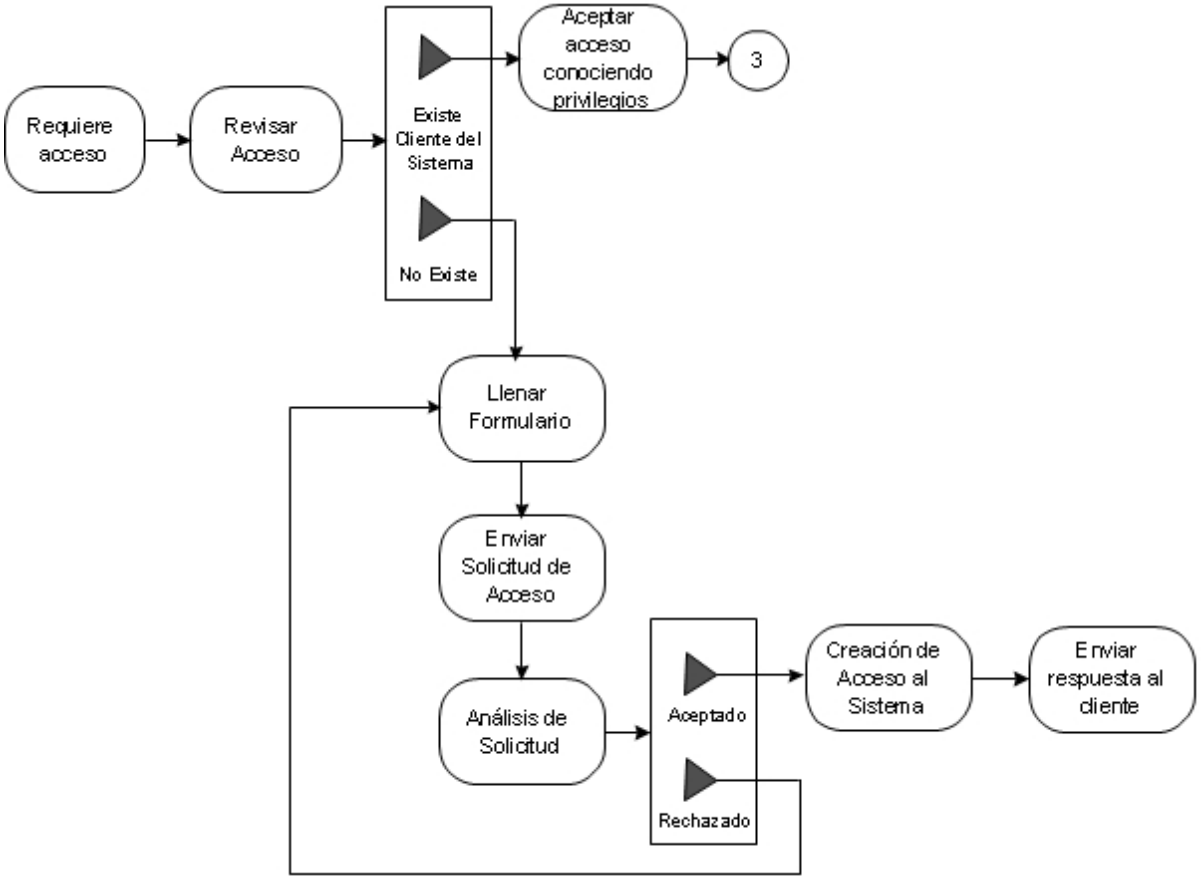
Nº	CONDICIONES DE ACTIVACIÓN
1	Si requiere acceso
2	Si no existe dominio, Usuario , Grupo y/o Solicitud Rechazada
3	Si existe formulario
4	Envio de solicitud existente
5	Análisis de solicitud aceptada
6	Si existe en el Servidor
7	Si objeto (dominio, organización, grupo, usuario) pertenece al Servidor 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



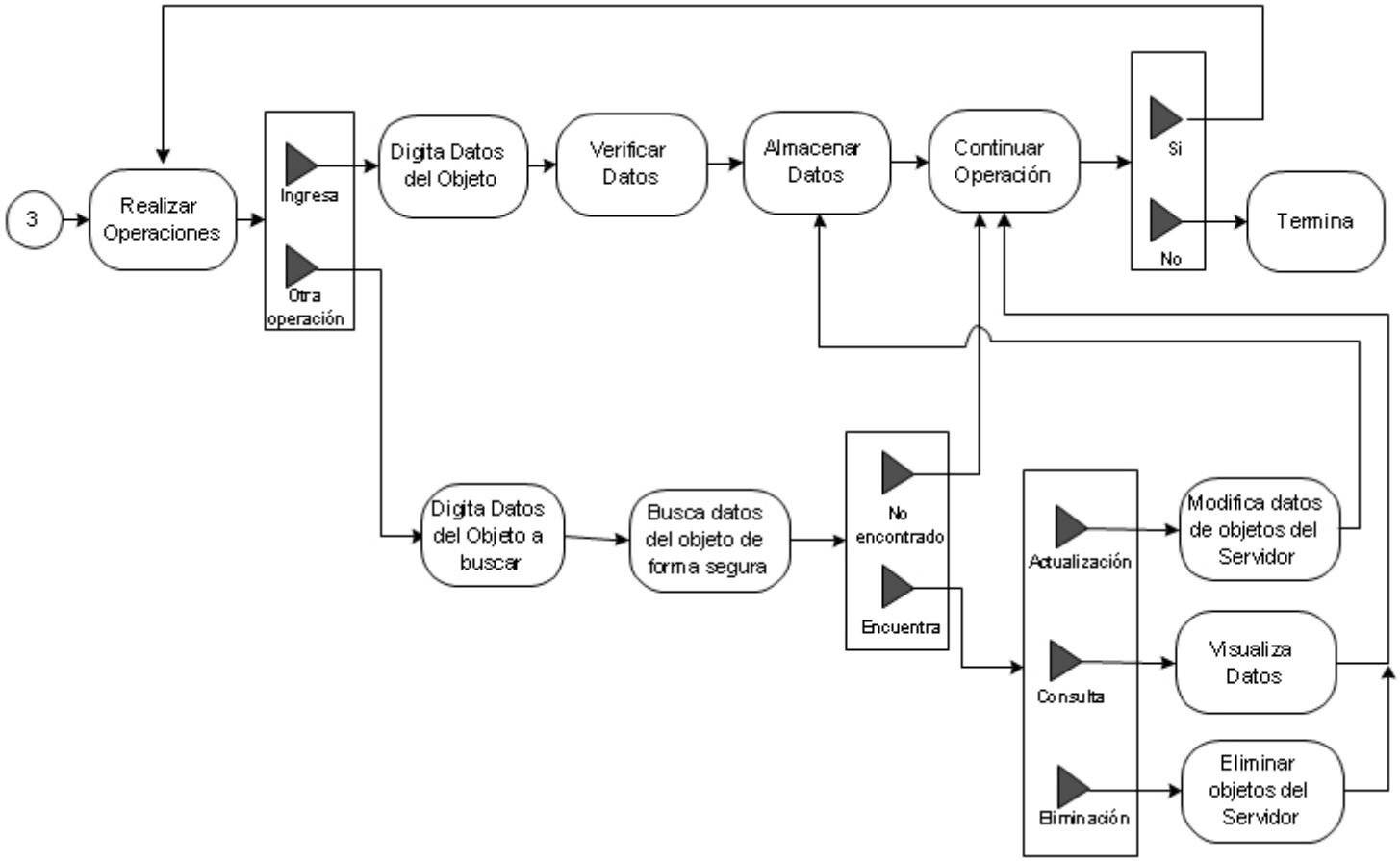
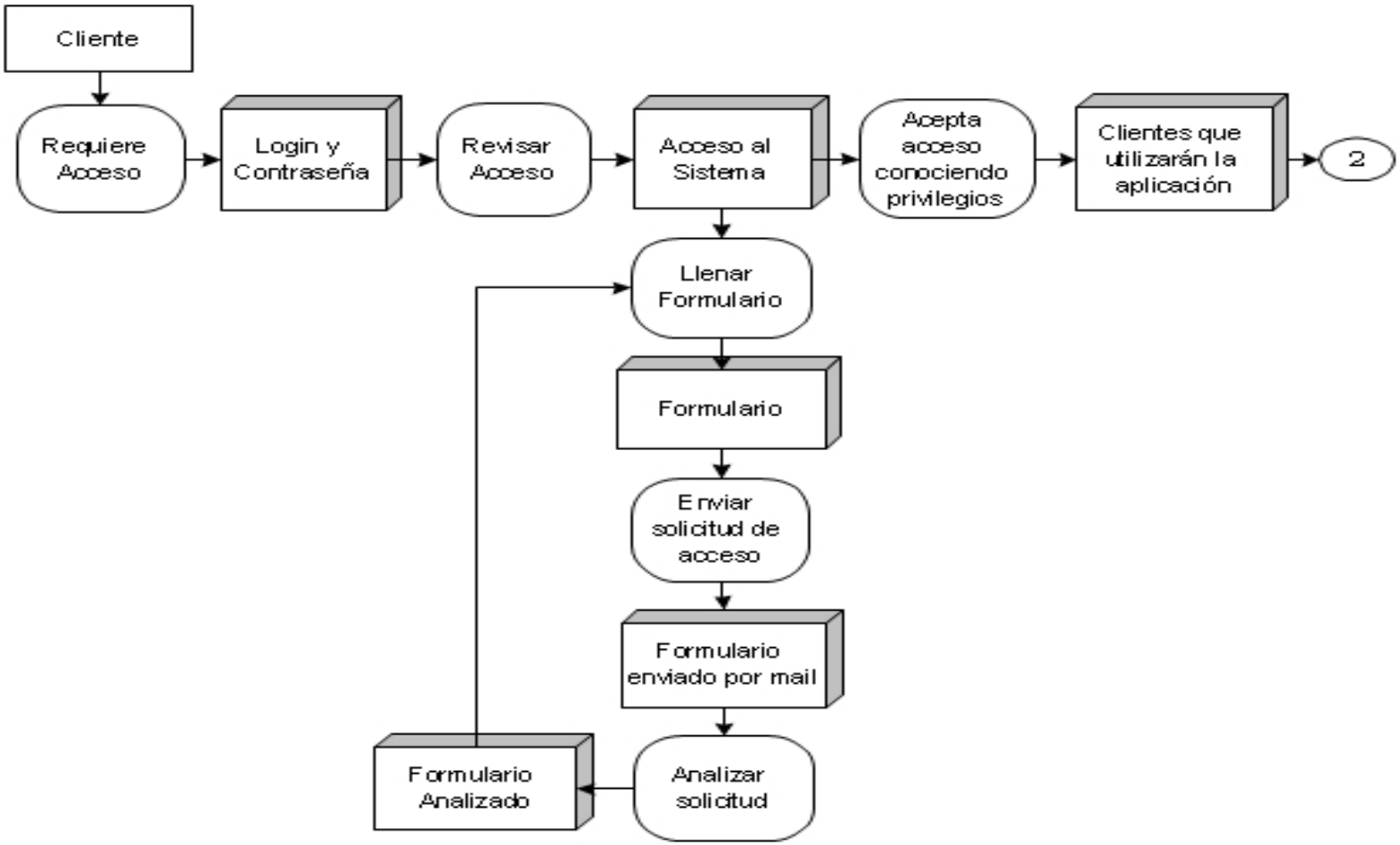


Gráfico -4-
Diagrama de Operaciones

1.2.4 Diagrama de flujo de Objetos



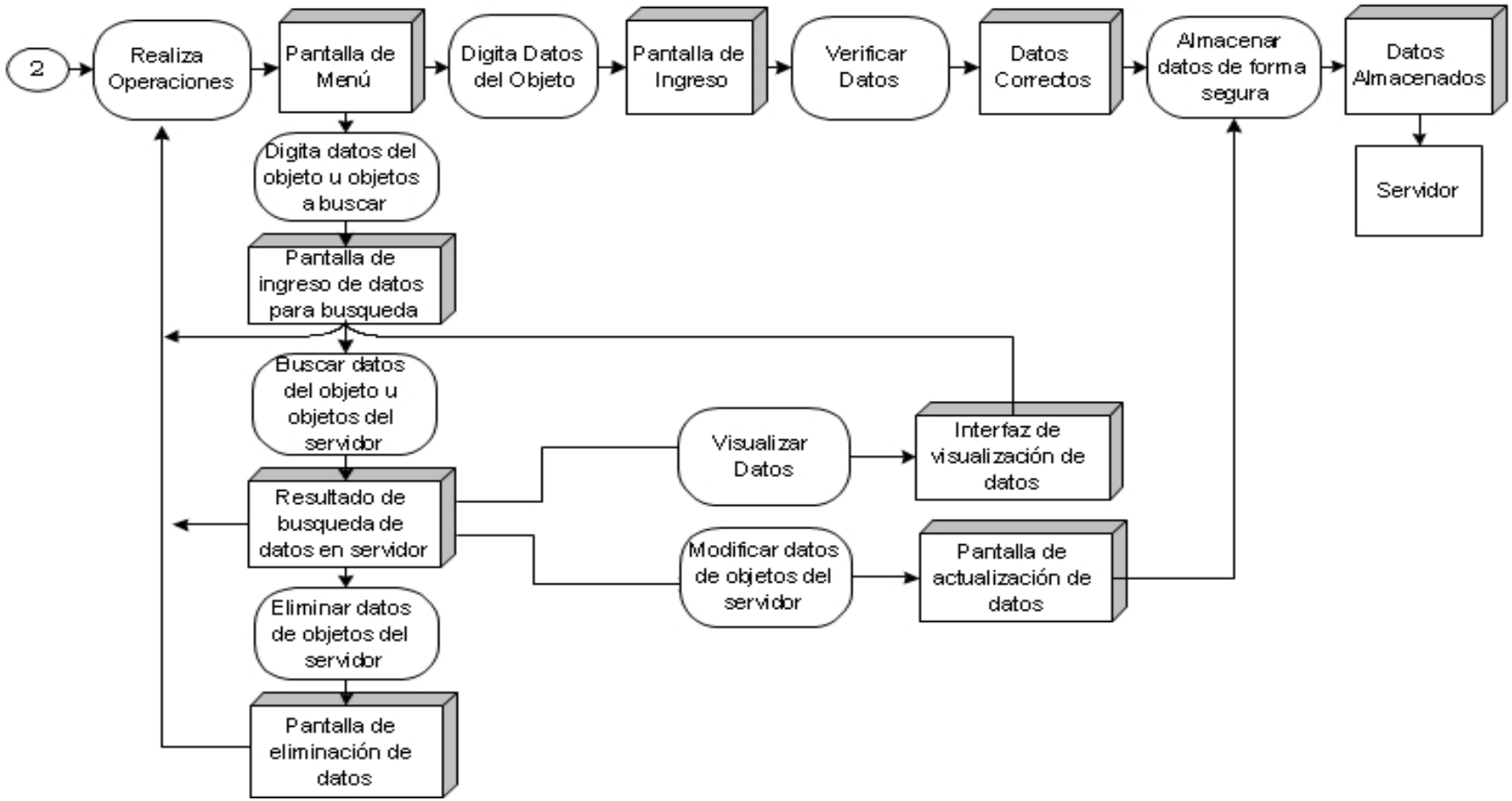
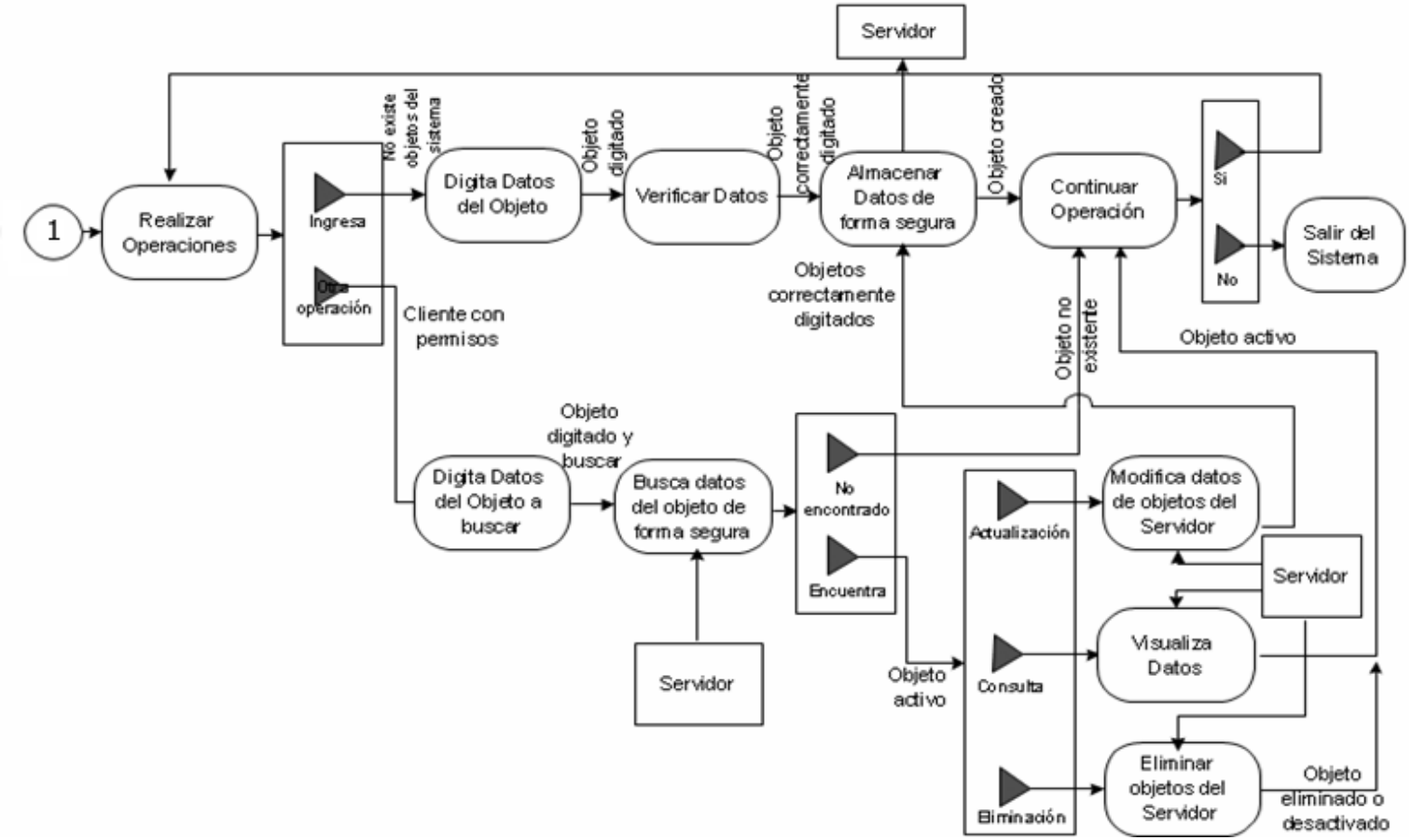


Gráfico -5-
Diagrama de Flujo de Objetos

1.2.5 Diagrama General



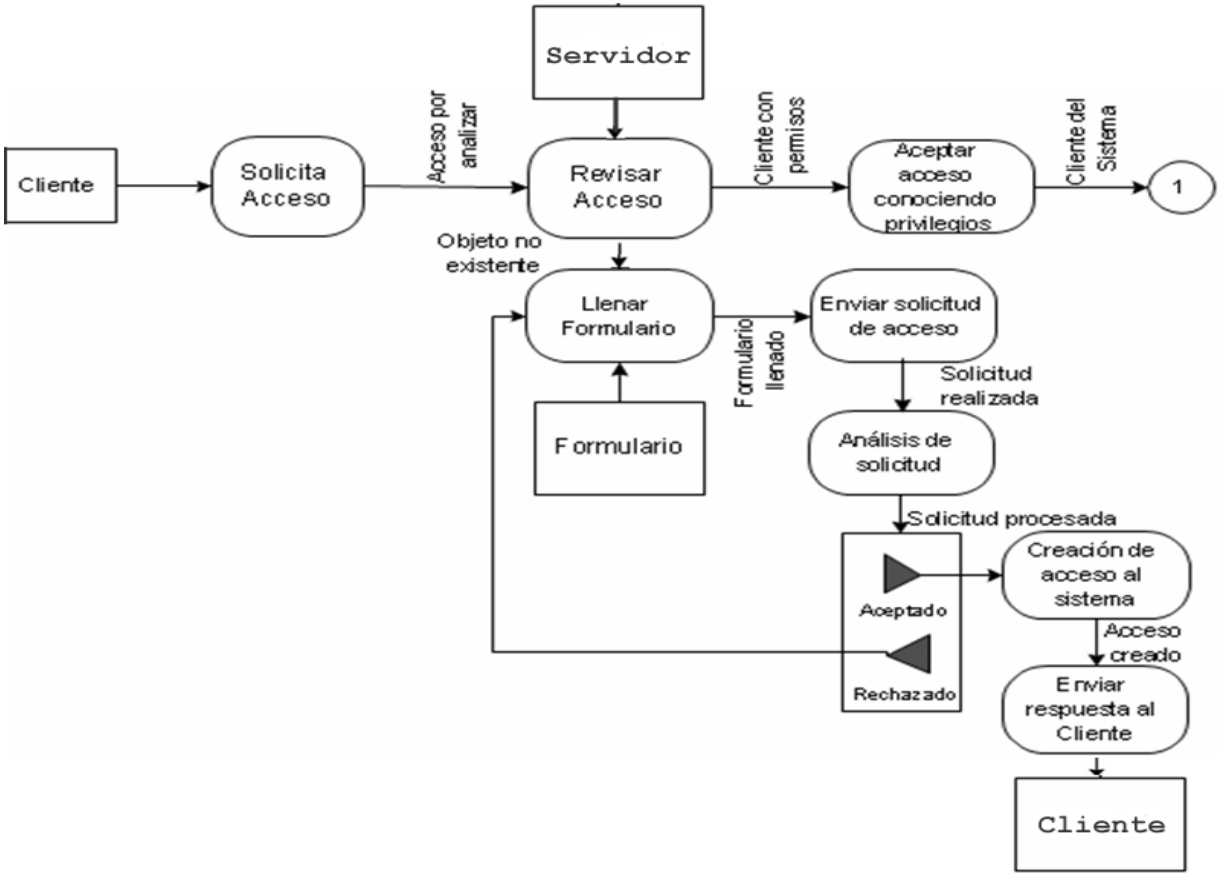


Gráfico -6-
Diagrama General

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).

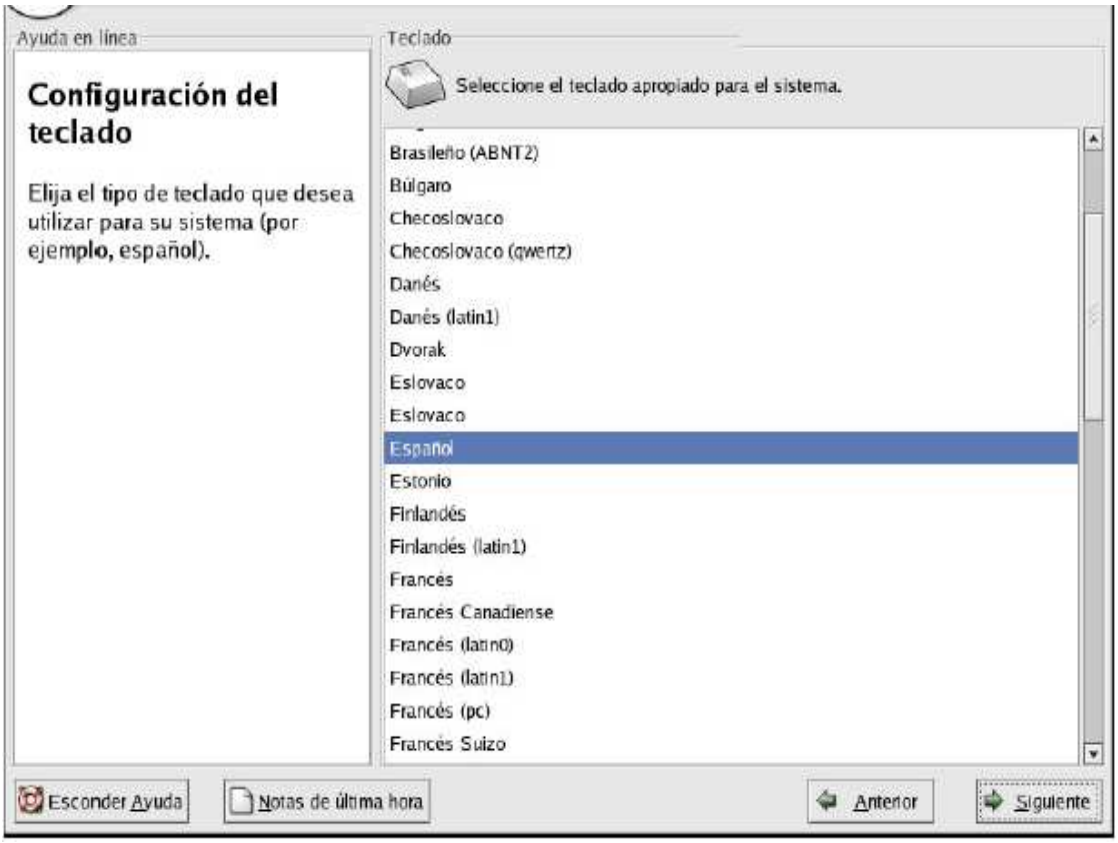


Gráfico -8-
Configuración del Teclado

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).

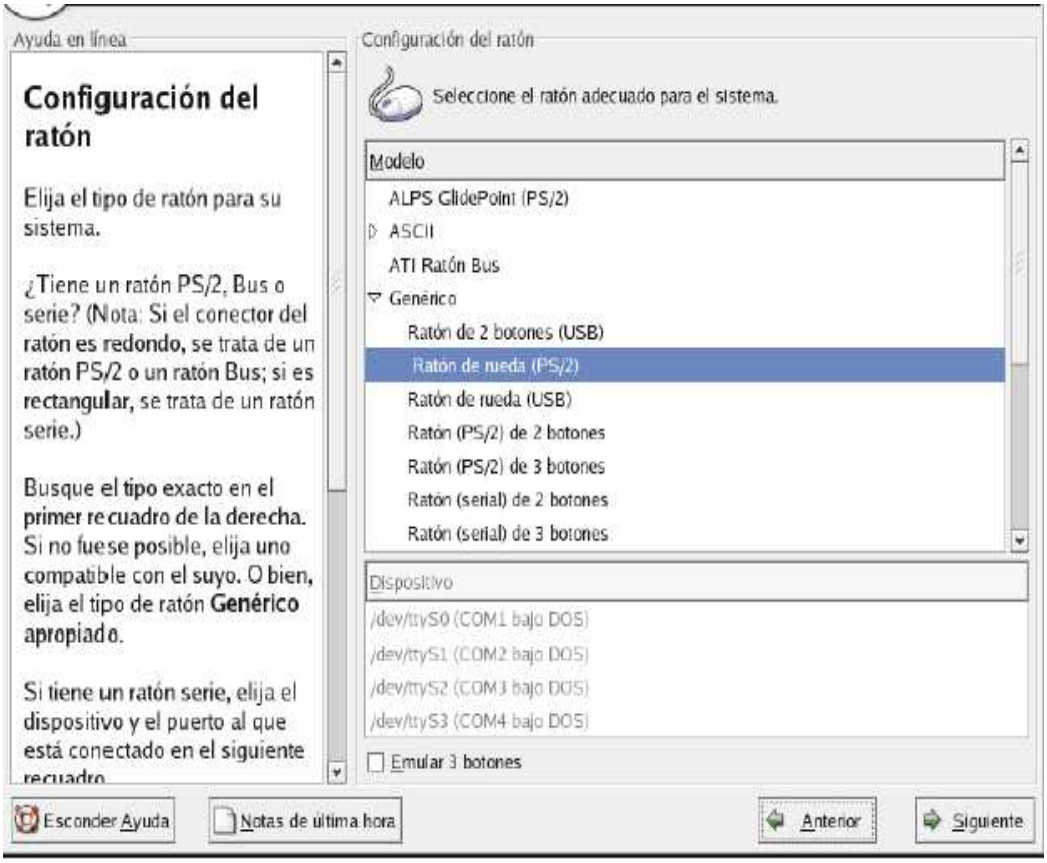


Gráfico -9-
Configuración del Ratón

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)

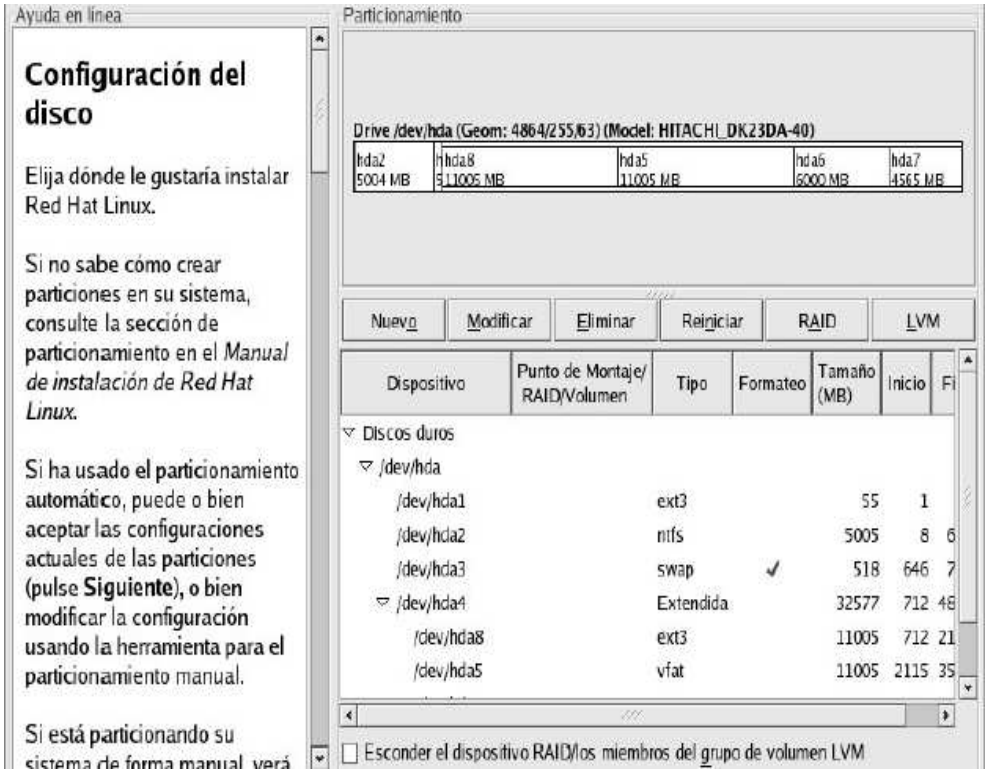


Gráfico -11-
Configuración de la Partición

¿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).

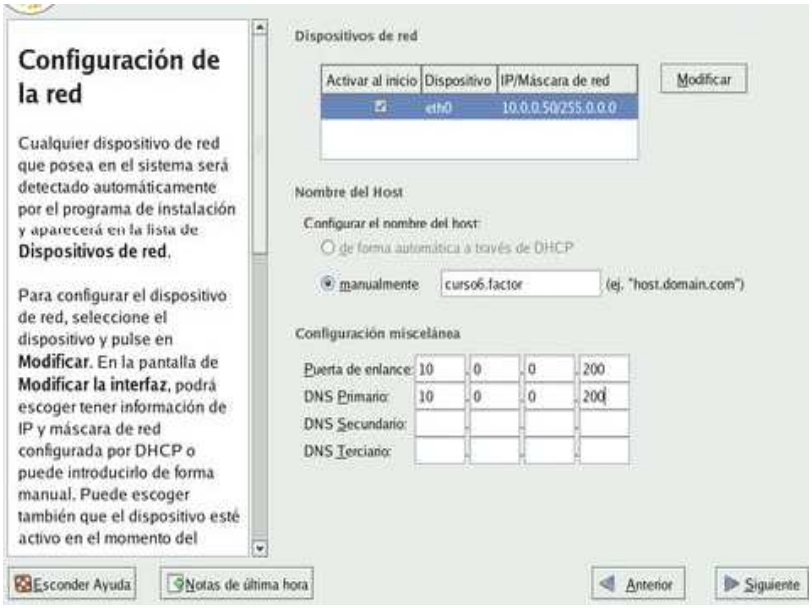


Gráfico -14-
Configuración de Red

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).

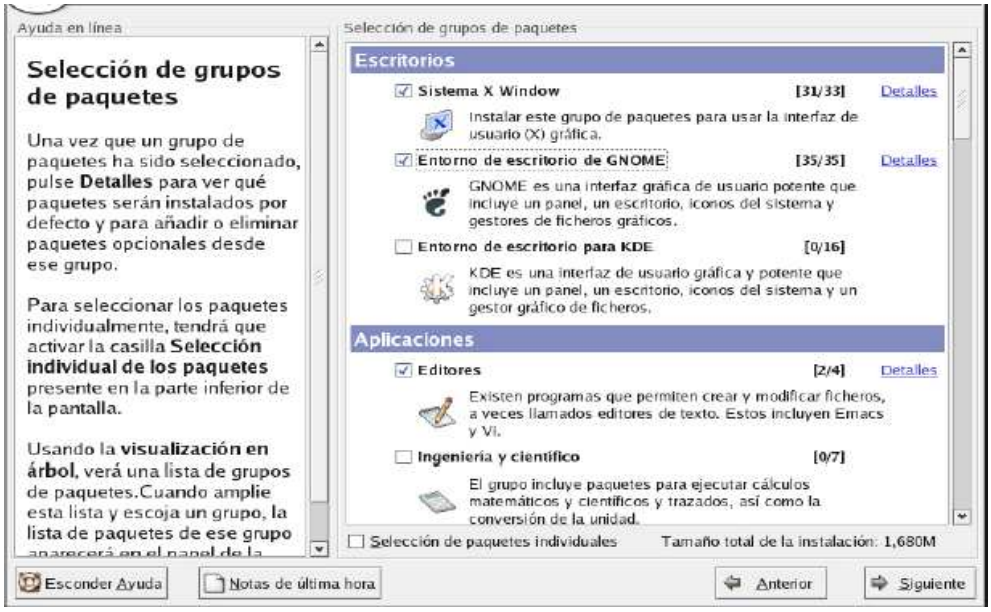


Gráfico -17-
Selección de Grupo de Paquetes

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

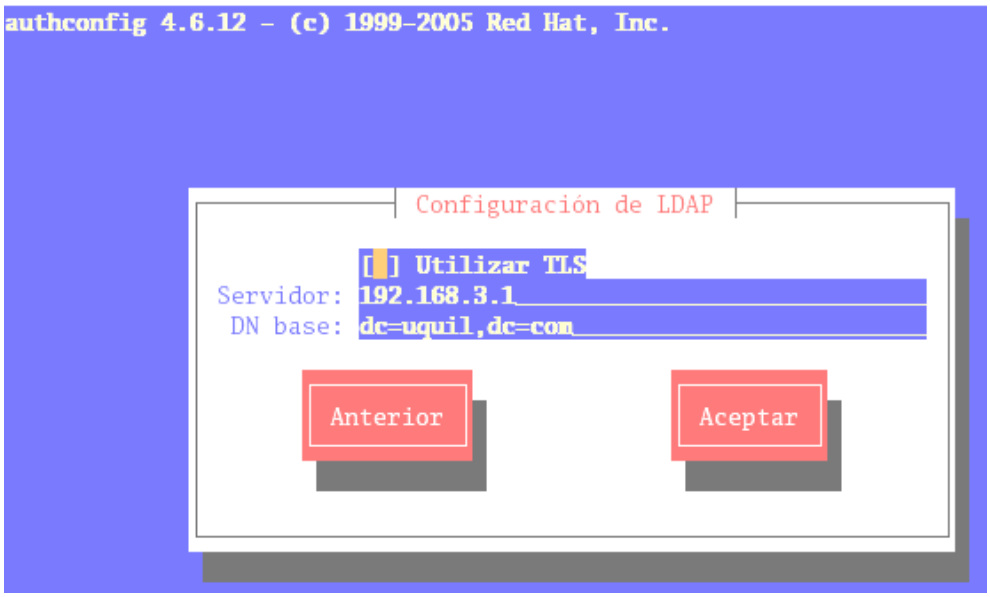


Gráfico -20-
Authconfig, Configuración LDAP

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).

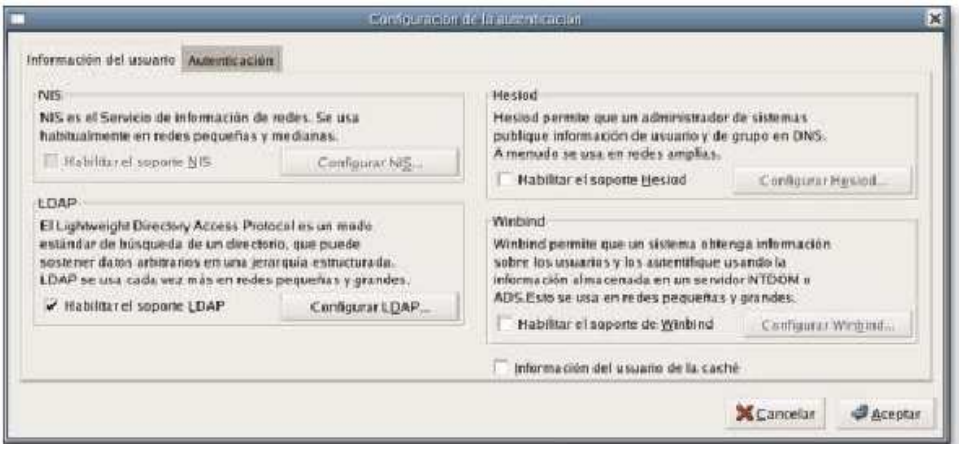


Gráfico -21-
Authconfig-gtk, Pestaña de Información del Usuario

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.

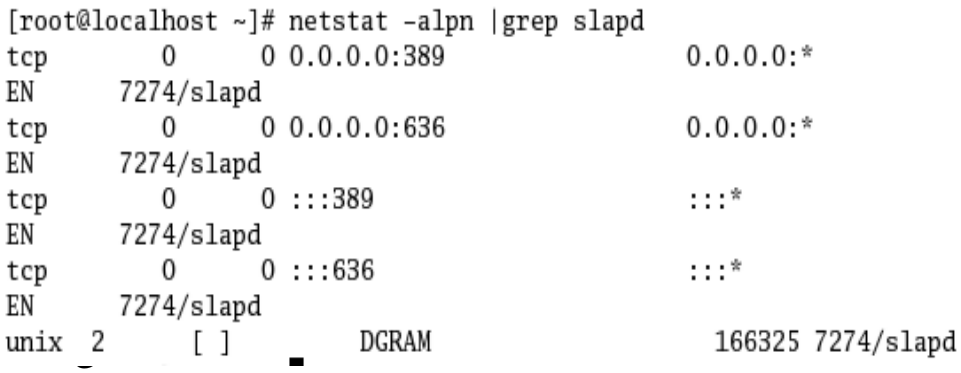


Gráfico -23-
Comprobación de Puertos abiertos

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).

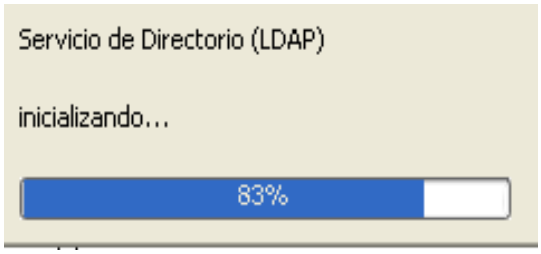


Gráfico -25-
Cargando la aplicación

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).

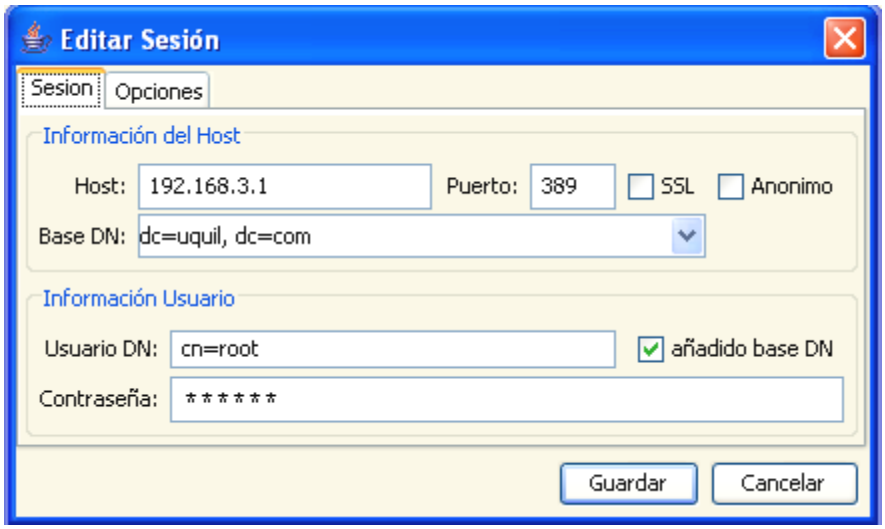


Gráfico -27-
Editar Sesión

- *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,

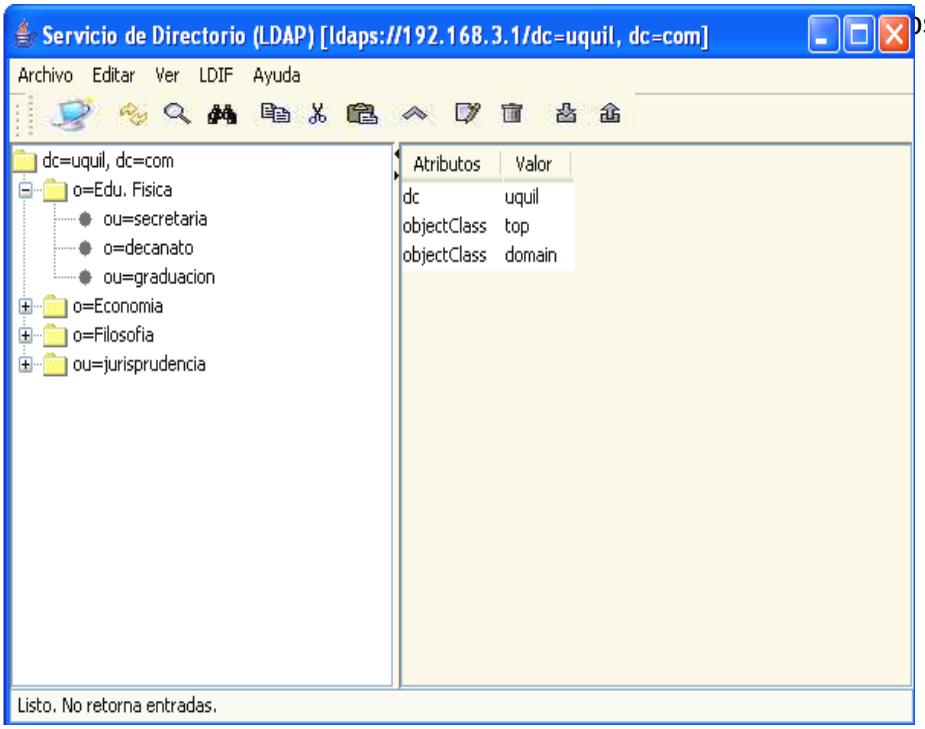


Gráfico -30-
Visualización del Árbol de Directorio

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).

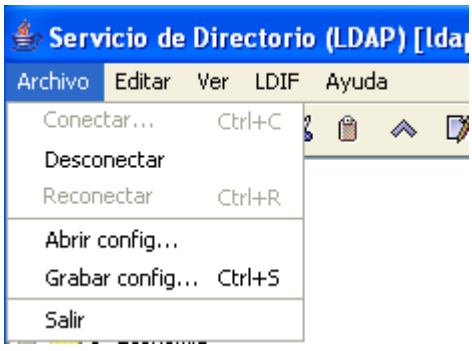


Gráfico -31-
Opciones de Menú Archivo

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).

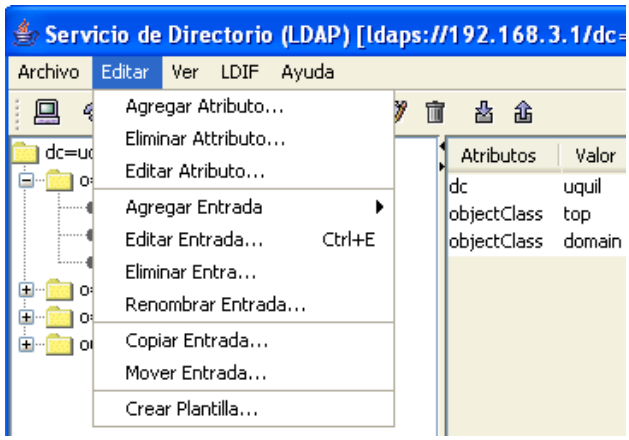


Gráfico -32-
Opciones de Menú Editar

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).

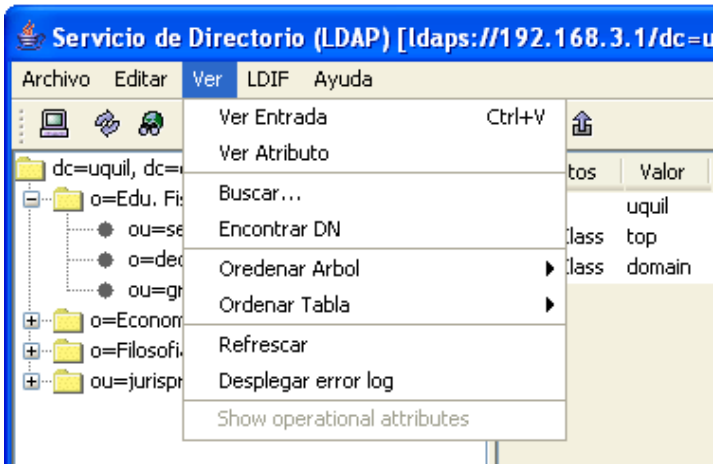


Gráfico -33-
Opciones de Menú Ver

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),

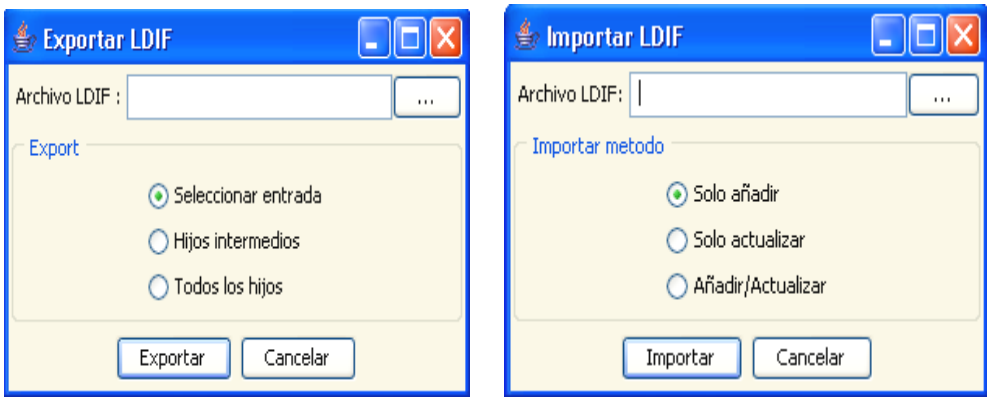


Gráfico -35-
Exportar e Importar LDIF

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)

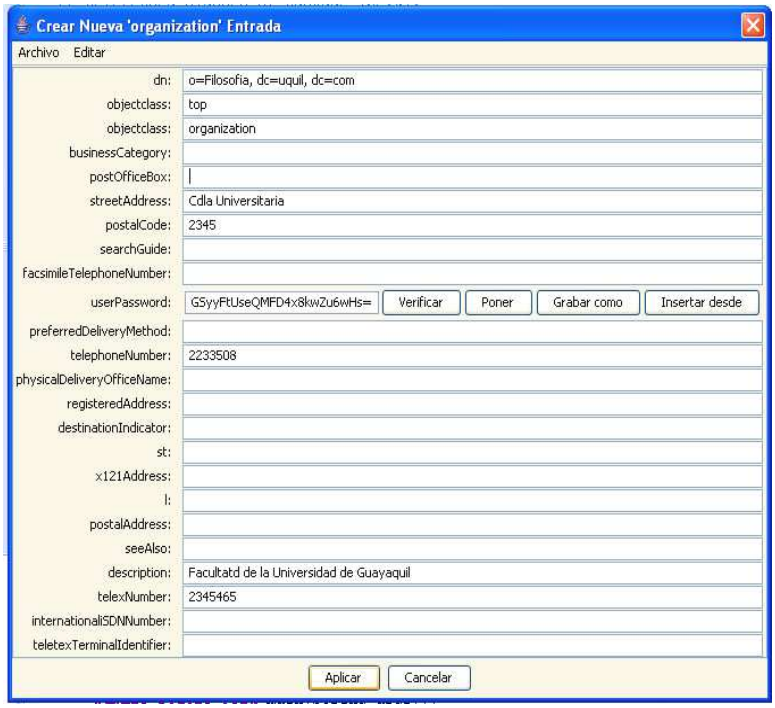


Gráfico -36-
Crear Nueva Entrada

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.

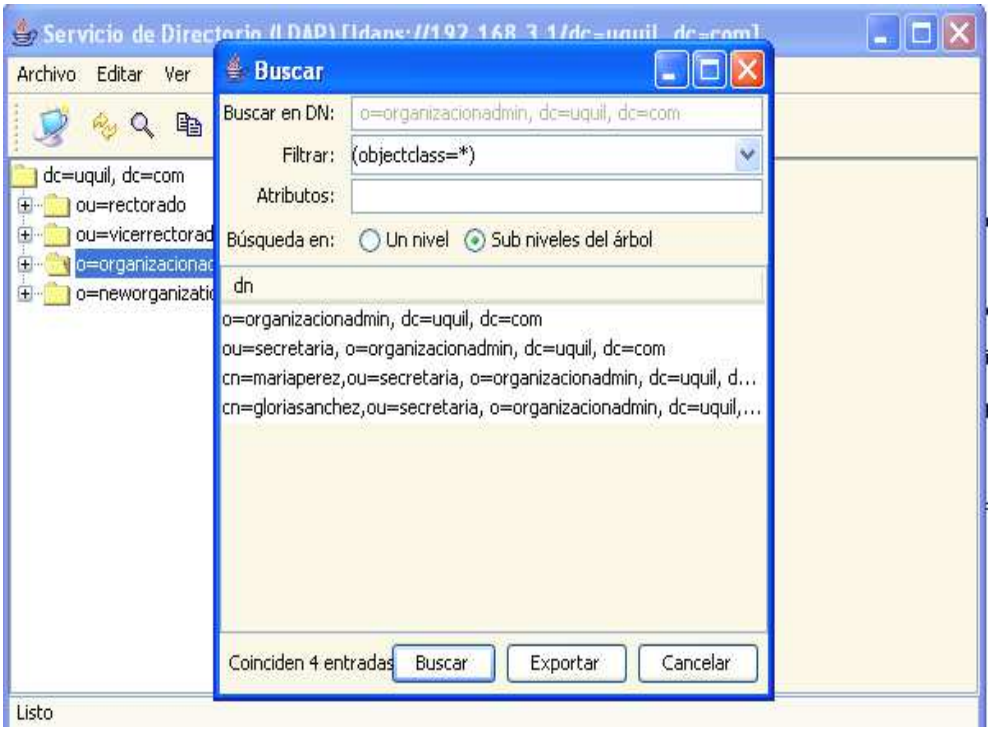


Gráfico -37-
Ventana de Búsqueda

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.

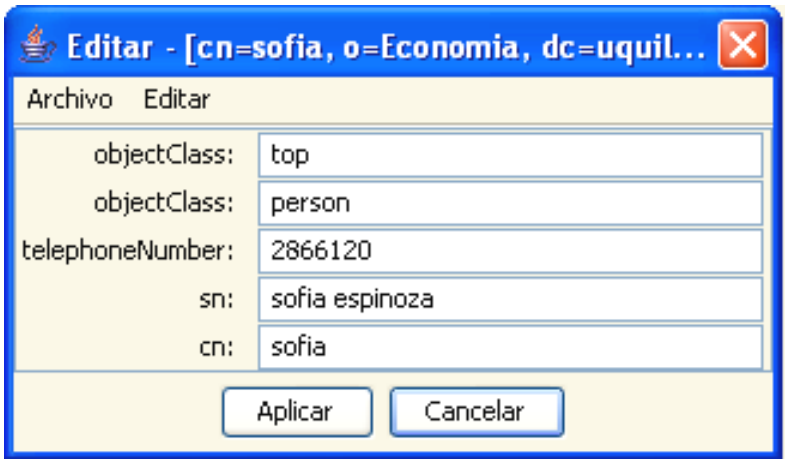


Gráfico -38-
Ventana de Editar

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)

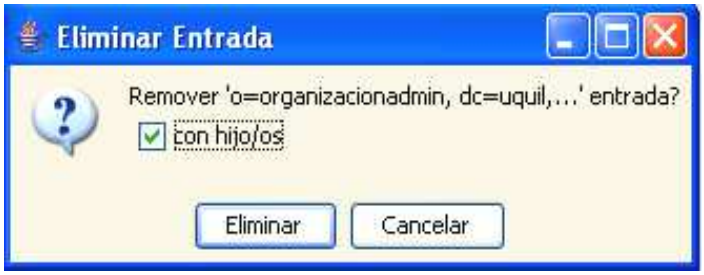


Gráfico -39-
Eliminar Entrada

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).

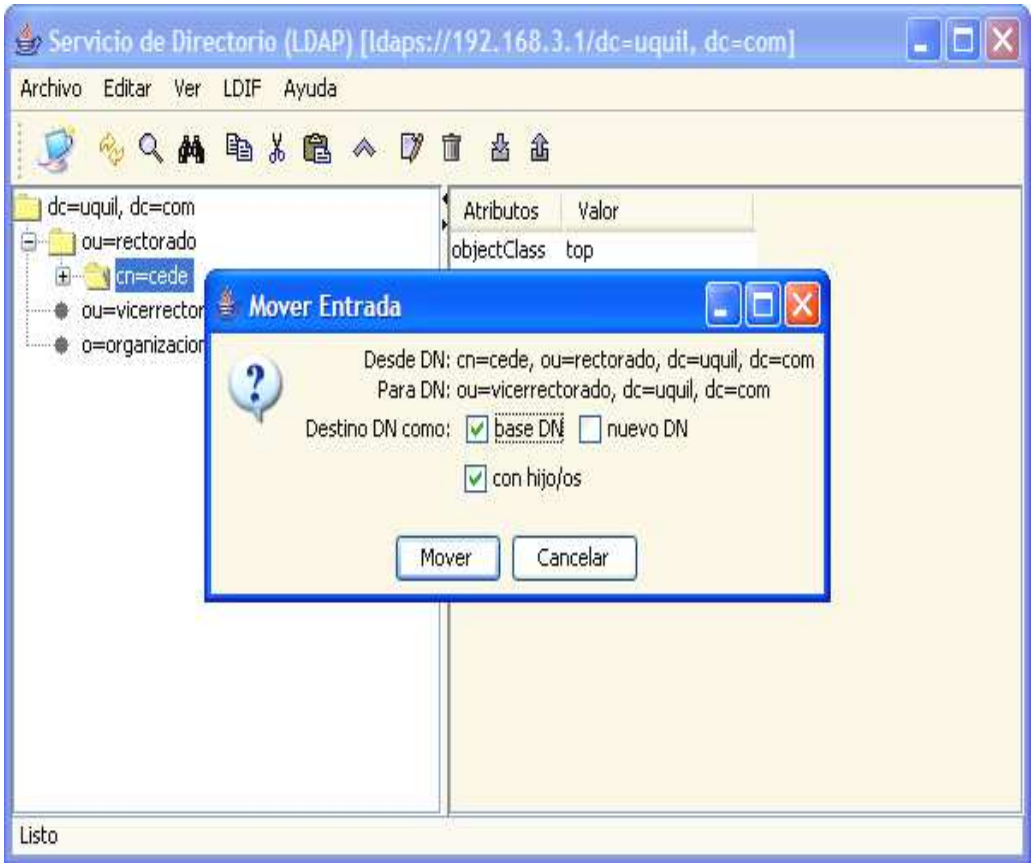


Gráfico -40-
Mover Entrada

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)

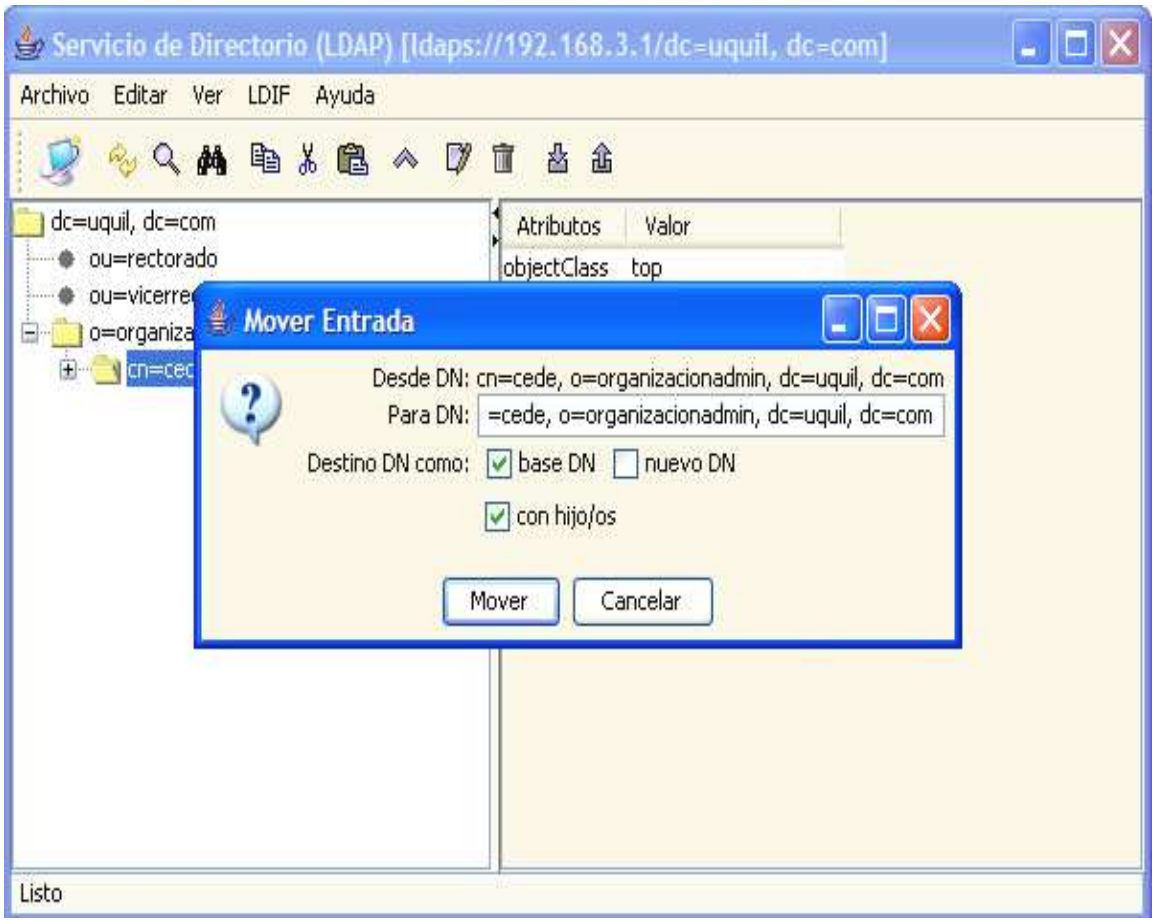


Gráfico -40A-
Copiar Entrada

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)

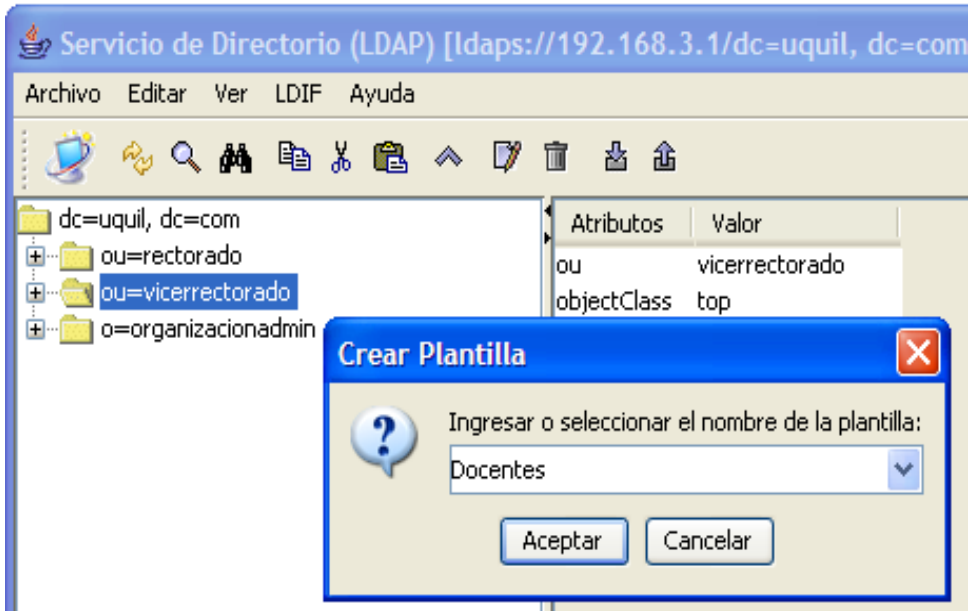


Gráfico -41-
Crear Plantilla

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)

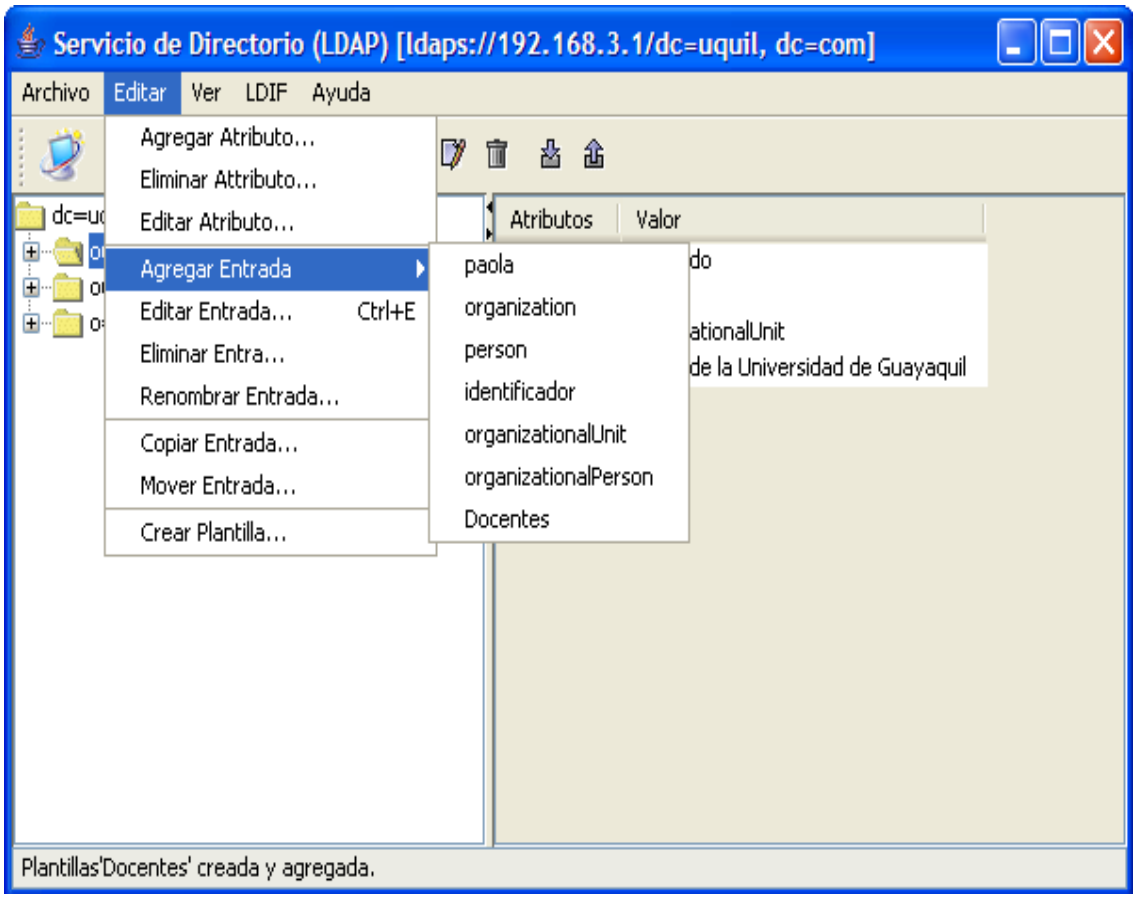


Gráfico -42-
Tipos de entradas - Plantillas creada

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)

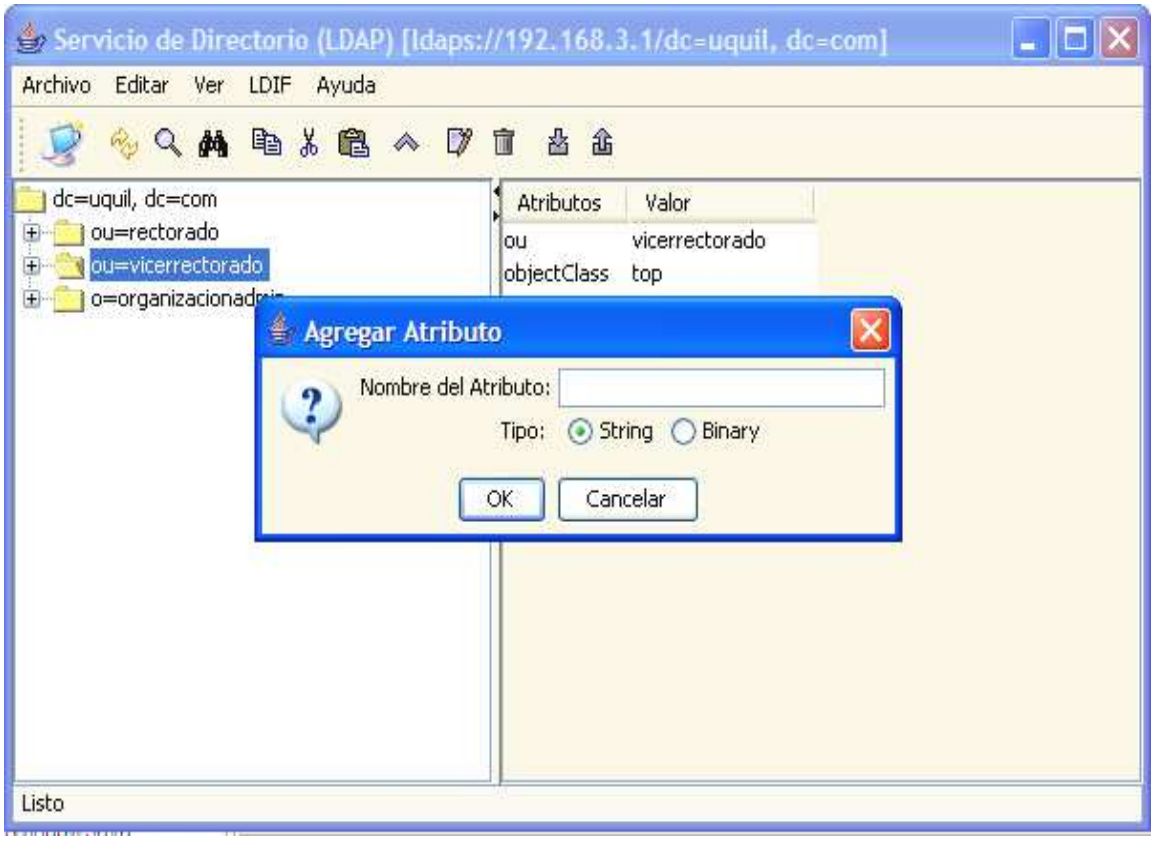


Gráfico -43-
Agregar Atributo

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)

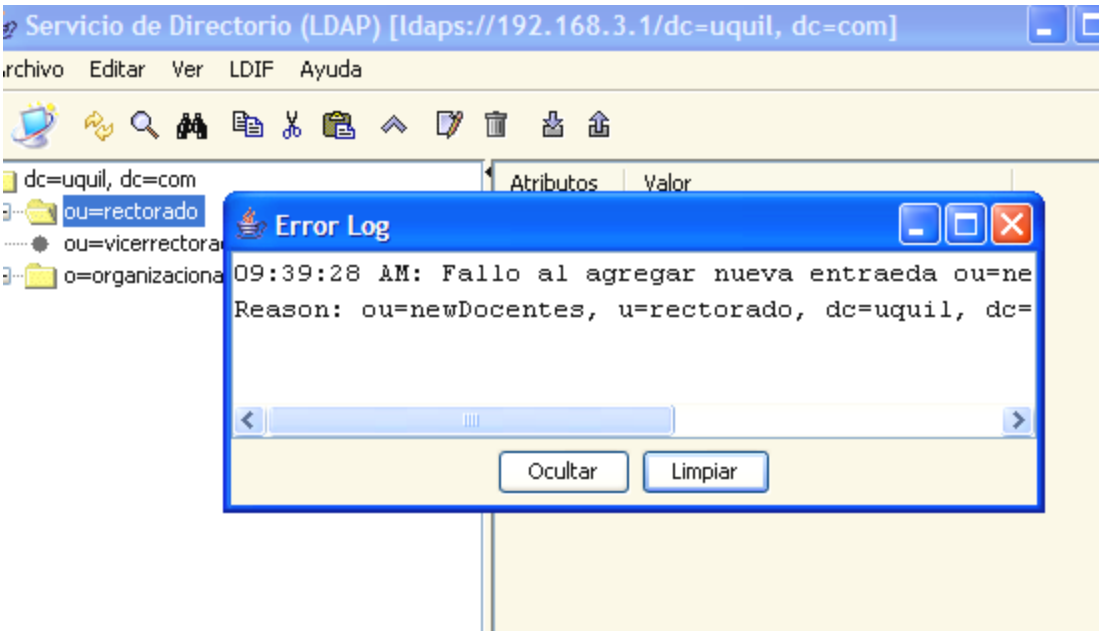


Gráfico -44-
Error Log

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)

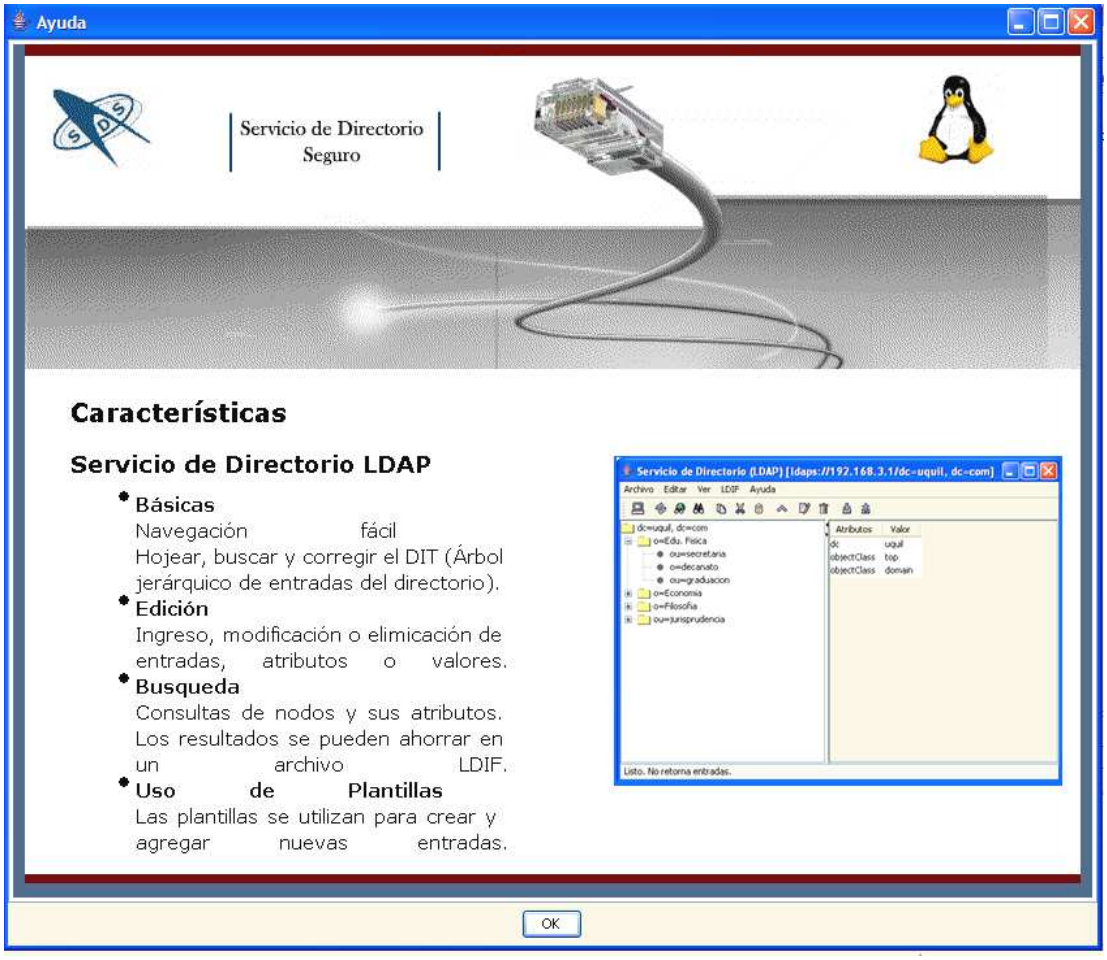


Gráfico -45 -
Ayuda - General

La opción Uso, presenta un pequeño manual para guiar al usuario de la aplicación (Gráfico # 46)

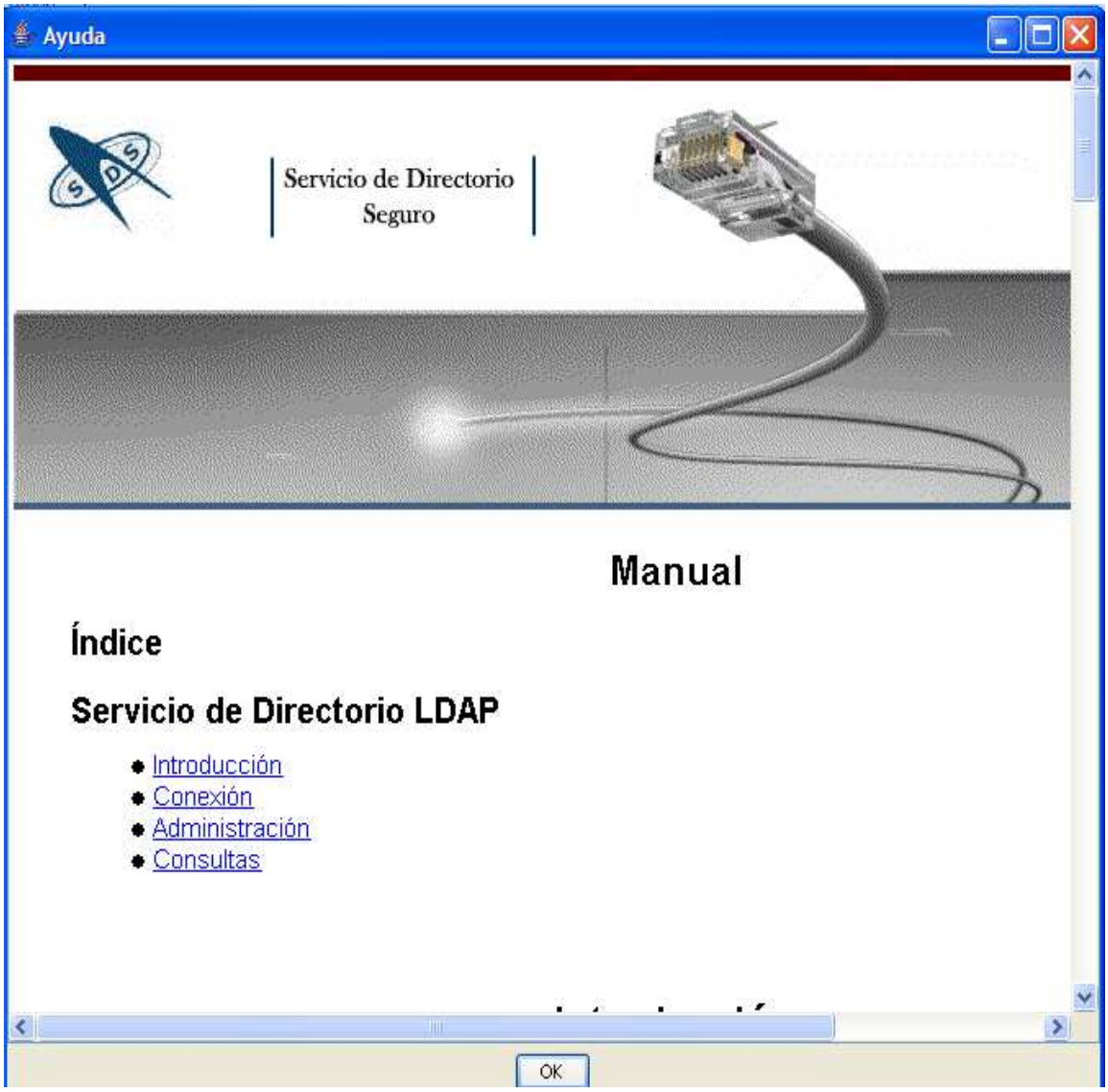


Gráfico -46-
Ayuda - Uso

La opción Notas, da una breve explicación de la importación y exportación de los archivos LDIF. (Gráfico # 47)

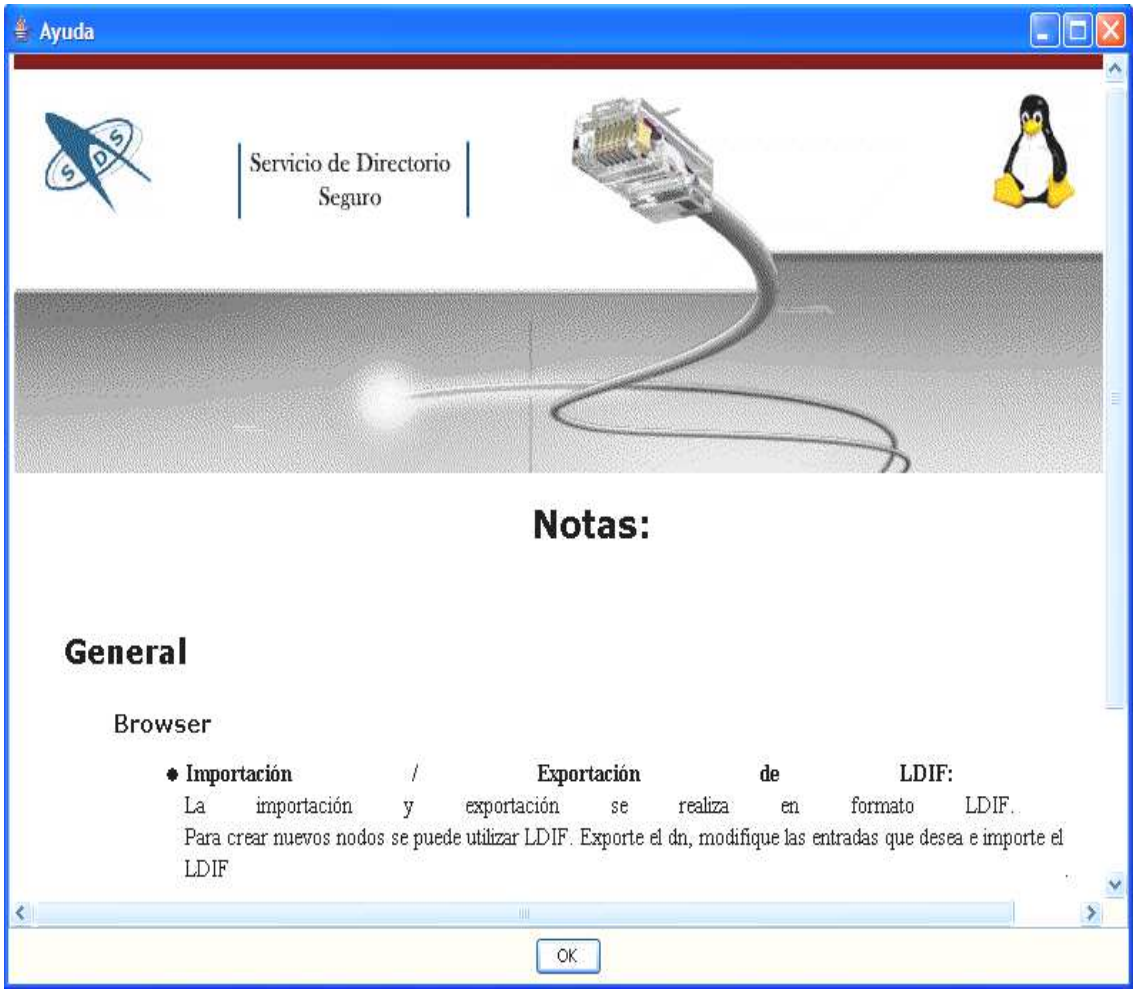


Gráfico -47-
Ayuda - Notas

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)

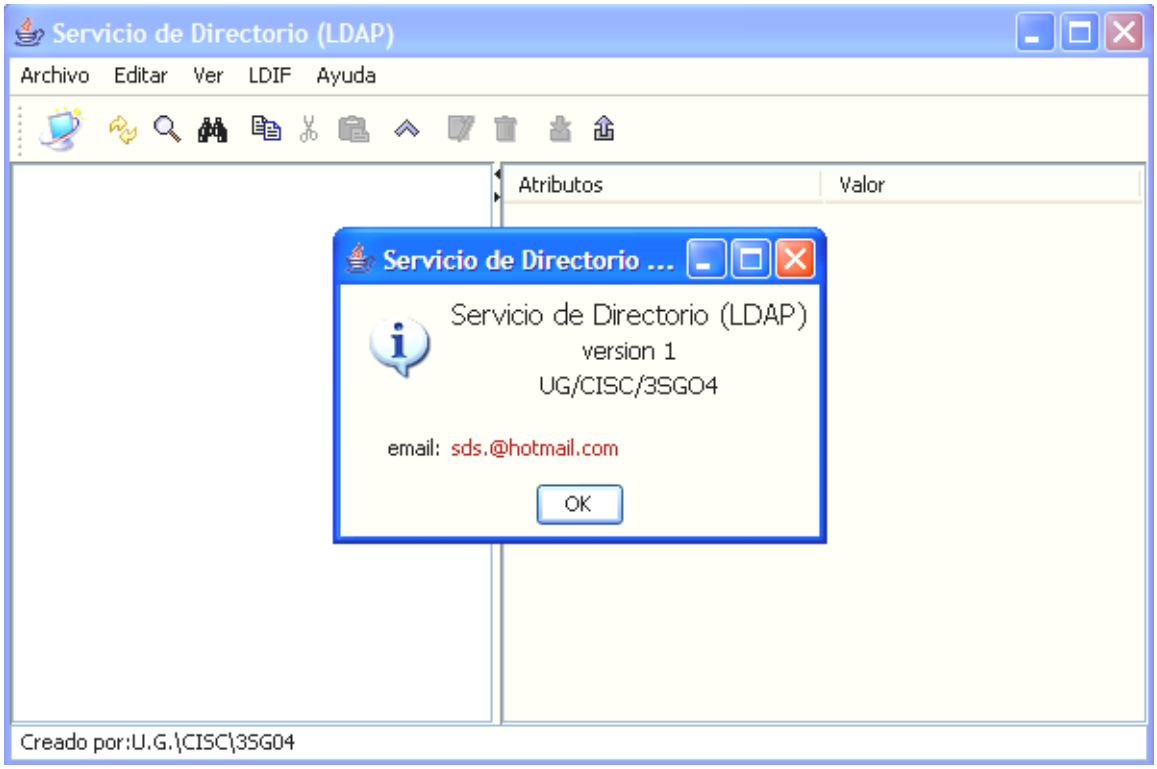


Gráfico -48-
Ayuda - About