MANUAL TÉCNICO 1

1. Introducción

El siguiente documento ha sido diseñado y orientado al personal técnico y analistas, los cuales estaran encargados del mantenimiento del sistema.

En este documento se describira todo lo relacionado con el diseño de , asi como tambiendescriben los principales componentes de la aplicación.

Es responsabilidad de los tecnicos y analistas hacer el mejor uso de este documento para de esta mantener siempre el sisteme operativo, asi como de darle un mantenimiento adecuado al mismo.

2. Diagramas

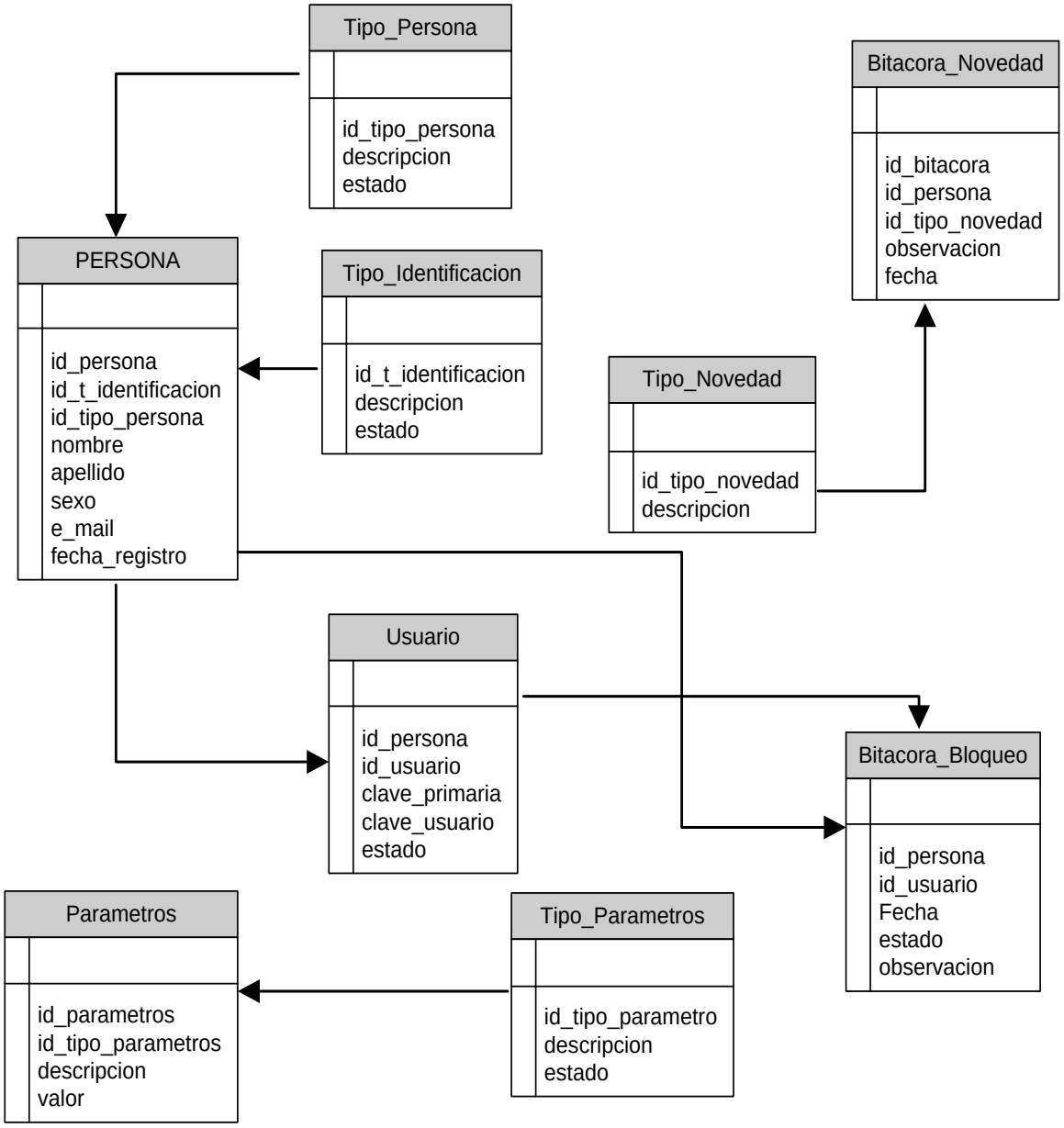
Detallamos a continuación cada uno de las representaciones gráficas utilizadas en el diseño de esta solución.

2.1 Diagramas de flujo de datos

El diagrama de flujo de datos (DFD), herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "conductos" y "tanques de almacenamiento" de datos. Siendo éste, una de las herramientas más comúnmente usadas, sobre todo por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejos que los datos que éste maneja.

2.2 Diagrama entidad relación

Modelo que representa a la realidad a través de un esquema gráfico empleando los terminología de ***entidades***, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas **atributos,** el enlace que rige la unión de las entidades esta representada por la **relación** del modelo.



D.E.R. SISTEMA CANCERBERO

3. Creacion de objetos de la base de datos

La base de datos cuenta con un sinnúmero de objetos entre los que podemos nombrar:

- Consultas complejas
- Foreign keys
- Triggers
- Vistas
- Integridad transaccional
- Control de concurrencia multiversión.
- Secuencias, entre otos.

Detallaremos a continuación los “scripts” o códigos SQL que permiten la creación de dichos objetos.

3.1 SCRIPT DE CREACIÓN DE LAS TABLAS

Script para la creación de la BDD en le cual el motor de la BDD PostGresql, se utiliza lenguaje sql con muy pocas variaciones en los comandos:

Tabla Usuarios

En esta tabla se crea la clase usuarios , que son todoa aquellos que interactuaran con el sistema directa o indirectamente. Consta la declaracion de su calve primaria, asi como de a que tabla o tablas esta relacionada directamente la misma.

```
mydb=# create table usuarios  (
mydb(# id_usuario varchar(15) not null,
mydb(# nombre1 varchar(30),
mydb(#  nombre2 varchar(30),
mydb(# apellidos varchar(60), tipo_identificacion char(1),
mydb(# identificacion varchar(13),
mydb(# ireccion varchar(60),
mydb(# password varchar(15),
mydb(# login_usr varchar(15),
mydb(# fecha_registro
mydb(# date, sexo char(1),
mydb(#  email varchar(60),
mydb(# constraint usuarios_pkey primary key (id_usuario));
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit
index'usuarios_pkey' for table 'usuarios'
CREATE TABLE
id_usuario | nombre1 | nombre2 | apellidos | tipo_identificacion |
identificacion | direccion | password | login_usr | fecha_registro |
sexo | email
------------+---------+---------+-----------+---------------------+----
```

Tabla Bitácora

Se guardaran los eventos a censar y gracias al cual se generaran los reportes.

```
mydb=# create table bitacora (
mydb(# id_bitacora int8 not null, id_usuario varchar(15),
mydb(# tipo_novedad char(1),
mydb(# observacion varchar(100),
mydb(# fecha date,
mydb(# constraint bitacora_pkey primary key (id_bitacora),
mydb(# constraint bitacoras_id_usuario_fk foreign key (id_usuario)
references usuarios (id_usuario) on update restrict on delete
restrict);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
'bitacora_pkey' for table 'bitacora'
NOTICE:  CREATE TABLE will create implicit trigger(s) for FOREIGN KEY
check(s)
```

```
CREATE TABLE
```

Tabla Bloqueos

Tabla en la cual se registraran todos aquellos accesos erroneos que se dieron sobre algún usuario.

```
mydb=# create table bloqueos (
mydb(# id_usuario varchar(15),
mydb(# fecha date, estado char(1),
mydb(# observacion varchar(100),
mydb(# constraint bloqueos_id_usuario_fk foreign key (id_usuario)
references usuarios (id_usuario) on update restrict on delete
restrict);
NOTICE:  CREATE TABLE will create implicit trigger(s) for FOREIGN KEY
check(s)
CREATE TABLE
```

Tabla Parámetros

Parametrizacion de la aplicación, ciertos valores que podrian ser necesarios en algun momento dado de la ejecución de la aplicación.

```
mydb=# create table parametros (
mydb(# id_parametros int not null,
mydb(# descripcion varchar(100),
mydb(# valor int(2),
mydb(# constraint _pkey primary key (id_parametros),
mydb(# constraint bitacoras_id_usuario_fk foreign key
(id_tipo_parametro) references usuarios (id_usuario) on update restrict
on delete restrict);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
'parametros_pkey' for table 'parametros'
NOTICE:  CREATE TABLE will create implicit trigger(s) for FOREIGN KEY
check(s)
CREATE TABLE
```

4. Modelo De Clases

Una clase es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase).

Bajo este esquema describimos cada una de las clases usadas para la implementación de este proyecto, cada una con sus respectivos atributos y metodos que me representan el tipo de información que maneja cada ente así como cada uno de los procesos que realizan cada uno de aquellos.

5. Codificación

Se muestra a continuación las principales clases implemetadas en la aplicación:

Conexión a la BDD

La clase BaseDB es la que permite la conexión a la BDD, mandando un requeriemiento a esta por medio de la siguiente instrucción:

```
private static final String url = jdbc:postgresql://localhost:5432/gupo2A";
```

Esta clase al lograr la conexión enviara un mensaje de conexión exitosa

```
System.out.println("Opening db connection");
```

mensaje de error

```
 System.err.println("Cannot connect to this database.");
```

BaseDB.java

```
package seminario.base;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import seminario.bean.UsuarioBean;

public class BaseDB {
    private static Connection conector = null;
    private Connection          connection;
    private Statement           statement;
    private ResultSet           resultSet;
    private static final String url = jdbc:postgresql://localhost:5432/gupo2A";
    //"jdbc:odbc:BASEPKI";
    private static String user;
    private static String paswd;
}
    public void conectaUsr(String user, String paswd) {
        this.user = user;
        this.paswd = paswd;
        //jdbc:postgresql://<HOST>:<PORT>/<DB>
        //org.postgresql.Driver
    }
```

Aqui se cierra la cconexion a la BDD

```
 public static void closeConector() {
        try {
            conector.close();
        } catch (SQLException e) {
            System.err.print(e);
        }
    }

    public static boolean getConector(String user, String paswd) {
        boolean conectado = false;
        try {
            if (conector == null || conector.isClosed()){
                Class.forName( "org.postgresql.Driver" );
                System.out.println("Opening db connection");
                conector = DriverManager.getConnection(url, user, paswd);
                conector.setAutoCommit(false);
                conectado = true;
            } else {
                conectado = true;
            }
        }
```

```
        catch (ClassNotFoundException ex) {
            System.err.println("Cannot find the database driver classes.");
            System.err.println(ex);
            conectado = false;
        }
        catch (SQLException ex) {
            System.err.println("Cannot connect to this database.");
            System.err.println(ex);
            conectado = false;
        }
        return conectado;
    }
```

CIFRADOR

Para esto hay que usar ciertas librerias propias de java a nivel de seguridad asy como las librerias de encriptación propias de JAVA.

```
import java.security.InvalidAlgorithmParameterException;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
```

Funcion de encriptación.-

Se recibe dos parámetros los cuales la función los retornará como una cadena de caracteres ya encriptada.

```
public static String encriptar(String passPhrase, String str) {
        Cipher ecipher = null;
        Cipher dcipher = null;
        try {
                // Crear la key
                KeySpec keySpec = new
PBEKeySpec(passPhrase.toCharArray(),
                                                SALT_BYTES,
ITERATION_COUNT);
                SecretKey key =
SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);
                ecipher = Cipher.getInstance(key.getAlgorithm());
                dcipher = Cipher.getInstance(key.getAlgorithm());
package seminario.util.seguridad;
import java.io.UnsupportedEncodingException;
import java.security.InvalidAlgorithmParameterException;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;
public class Cifrador{
    private static byte[] SALT_BYTES = {
                    (byte)0xA9,
                    (byte)0x9B,
                    (byte)0xC8,
                    (byte)0x32,
                    (byte)0x56,
                    (byte)0x35,
                    (byte)0xE3,
                    (byte)0x03      };
    private static int ITERATION_COUNT = 19;
        public static String encriptar(String passPhrase, String str) {
        Cipher ecipher = null;
        Cipher dcipher = null;
        try {
                // Crear la key
                KeySpec keySpec = new
PBEKeySpec(passPhrase.toCharArray(),
                                                SALT_BYTES,
ITERATION_COUNT);
                SecretKey key =
SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);
                ecipher = Cipher.getInstance(key.getAlgorithm());
                dcipher = Cipher.getInstance(key.getAlgorithm());

                // Preparar los parametros para los ciphers
                AlgorithmParameterSpec paramSpec = new
PBEParameterSpec(SALT_BYTES, ITERATION_COUNT);
```

```
                // Crear los ciphers
                ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
                dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
                            } catch (javax.crypto.NoSuchPaddingException
e) {
                e.printStackTrace();
            } catch (java.security.NoSuchAlgorithmException e) {
                e.printStackTrace();
            } catch (java.security.InvalidKeyException e) {
                e.printStackTrace();
            } catch (InvalidKeySpecException e) {
                e.printStackTrace();
            } catch (InvalidAlgorithmParameterException e) {
                e.printStackTrace();
            }

            try {
                // Encodear la cadena a bytes usando utf-8
                byte[] utf8 = str.getBytes("UTF8");
                // Encriptar
                byte[] enc = ecipher.doFinal(utf8);
                // Encodear bytes a base64 para obtener cadena
                return new sun.misc.BASE64Encoder().encode(enc);
            } catch (javax.crypto.BadPaddingException e) {   }
            catch (IllegalBlockSizeException e) {   }
            catch (UnsupportedEncodingException e) {   }
            return null;
    }
```

A continuacion la función que logra desecriptar el archive el cual recibe los parametros generados en la clase encriptar y devuelve una cadena desencriptada:

```
public static String desencriptar(String passPhrase, String str) {
        Cipher ecipher = null;
        Cipher dcipher = null;
        try {
            // Crear la key
            KeySpec keySpec = new PBEKeySpec(passPhrase.toCharArray(),
                                SALT_BYTES, ITERATION_COUNT);
            SecretKey key =
SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);
            ecipher = Cipher.getInstance(key.getAlgorithm());
            dcipher = Cipher.getInstance(key.getAlgorithm());
            // Preparar los parametros para los ciphers
            AlgorithmParameterSpec paramSpec = new
PBEParameterSpec(SALT_BYTES, ITERATION_COUNT);
            // Crear los ciphers
```

```
            ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);

            dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
            } catch (javax.crypto.NoSuchPaddingException e) {
                e.printStackTrace();
            } catch (java.security.NoSuchAlgorithmException e) {
                e.printStackTrace();
            } catch (java.security.InvalidKeyException e) {
                e.printStackTrace();
            } catch (InvalidKeySpecException e) {
                e.printStackTrace();
            } catch (InvalidAlgorithmParameterException e) {
                e.printStackTrace();
            }
                        try {
                // Decodear base64 y obtener bytes
                byte[] dec = new
sun.misc.BASE64Decoder().decodeBuffer(str);
                // Desencriptar
                byte[] utf8 = dcipher.doFinal(dec);
                // Decodear usando utf-8
                return new String(utf8, "UTF8");

            } catch (javax.crypto.BadPaddingException e) {
                e.printStackTrace();
            } catch (IllegalBlockSizeException e) {
                e.printStackTrace();
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }  return null;
        }
}
```

Encriptación del password:

```
Encrypt.java
package seminario.util.seguridad;
import java.security.*;

import seminario.util.seguridad.Cifrador;
public class Encrypt {
   private String password = null;
   private String password_encrypt = null;

   public Encrypt(){ }
      public String getPassword() {  return password;  }
   public void setPassword(String password){ this.password = password;}

   public String getPassword_encrypt() {  return password_encrypt; }
```

```
   public void setPassword_encrypt(String password_encrypt) {
this.password_encrypt = password_encrypt;}

   public String new2Password( String passwd ) {
      try {
        MessageDigest md = MessageDigest.getInstance("SHA-1");
        String clearPassword = passwd;
        md.update(clearPassword.getBytes());
        byte[] digestedPassword = md.digest();
        return new String(digestedPassword);
      }
      catch (java.security.NoSuchAlgorithmException e) {
         System.out.println("Rats, MD5 doesn't exist");
         System.out.println(e.toString());
         return null;
      }

   }
```

El password del usuario tambien se encripta y asi de esta forma la seguridad es mas robusta.

Encriptacion del Password

```
  public void encyptPassword() {
      try {
         MessageDigest sha = MessageDigest.getInstance("DES");
         byte[] tmp = this.password.getBytes();
         sha.update(tmp);
         this.password_encrypt = new String(sha.digest());

      }
      catch (java.security.NoSuchAlgorithmException e) {
         System.out.println("Rats, MD5 doesn't exist");
         System.out.println(e.toString());
      }
   }
```

Main principal para la encriptacion

```
public static void main( String [] args )
   {
      String clave = "jacquito";

      String clave_encriptada    = Cifrador.encriptar("password",clave);
      String                     clave_desencriptada                    =
Cifrador.desencriptar("password",clave_encriptada);
      System.out.println("Clave: "+clave);
      System.out.println("Clave encpritada: "+clave_encriptada);
      System.out.println("Clave desencriptada: "+clave_desencriptada);
    }
}
```

Generacion login del usuario

```
public class LoginUsuarioAction extends Action {
    /**This is the main action called from the Struts framework.
     * @param mapping The ActionMapping used to select this instance.
     * @param form The optional ActionForm bean for this request.
     * @param request The HTTP Request we are processing.
     * @param response The HTTP Response we are processing.
     */
    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response) throws
IOException,

ServletException {
        LoginBeanForm formaUSR = (LoginBeanForm)form;
        ActionErrors errores = new ActionErrors();
        ServerDES unserverDes = new ServerDES();
        if (form instanceof LoginBeanForm) {
            // mostramos los parametros del fichero
            FormFile theFile = formaUSR.getCertificado();
            String password  = formaUSR.getPassword();
            String contentType = theFile.getContentType();
            String fileName = theFile.getFileName();
            int fileSize = theFile.getFileSize();
            byte[] fileData = theFile.getFileData();
            System.out.println("Tipo: " + contentType);
            System.out.println("Nombre: " + fileName);
            System.out.println("Tamano: " + fileSize);
                        try {

               //guarda los datos del fichero
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                InputStream stream = theFile.getInputStream();
```

```
                // verificar el certificado
                if( unserverDes.verificaCertificado(password,stream)){
                    return mapping.findForward("aceptar");
                }else{
                    return mapping.findForward("error");
                }
                // solo si el archivo es de menos de 4MB
            }catch(Throwable e){
                System.out.println(e.getMessage());
            }
        }
        return mapping.findForward("error");
    }
}
```

Generación de Reportes

```
public class ReporteBloqueos extends DefaultInternalFrame {
    private JPanel jPanel1 = new JPanel();
    private BarraNavega barraNavega1 = new BarraNavega();
    private UsuarioBo unUsuarioBO = null;
    private UsuarioBean unUsuarioBean = null;
    private JTextField jTextField1 = new JTextField();
    private JLabel jLabel1 = new JLabel();
    private JLabel jLabel2 = new JLabel();
    private JTextField jTextField2 = new JTextField();
    private JButton jButton1 = new JButton();
    private JLabel jLabel3 = new JLabel();
    private JTextField jTextField3 = new JTextField();
    private JLabel jLabel4 = new JLabel();
    private JButton jButton2 = new JButton();
    private JPanel jPanel2 = new JPanel();
    private BorderLayout borderLayout1 = new BorderLayout();
    private JScrollPane jScrollPane1 = new JScrollPane();
    private JButton jButton3 = new JButton();

    public ReporteBloqueos() {
        try {
            jbInit();
            unUsuarioBO = new UsuarioBo();
        } catch (Exception e)

 {
            e.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        this.setSize(new Dimension(685, 383));
```

```
        this.setTitle("Reporte de Bloqueos");
        jPanel1.setBounds(new Rectangle(9, 12, 657, 93));
        jPanel1.setLayout(null);
```

Busqueda

Busqueda de usuario de acuerdo a ciertos parámetros a establecidos

```
String[] Head = { "codigo", "nombre" };
        Object[] fila = { "vacio" };
        String[] opciones = { "Codigo", "Nombre" };
```

```
    private void buscar(){
        UsuarioBean unXbean;
        String codigoCasa;
        int contador = 0;
        Iterator itList;
        ArrayList lista;
        Object[][] data;
        String[] Head = { "codigo", "nombre" };
        Object[] fila = { "vacio" };
        String[] opciones = { "Codigo", "Nombre" };
        try {
            lista = unUsuarioBO.buscarTodos();
            // TRANFORMAR EL ARRAYLIST DE DATOS EN UN ARREGLO DE DATOS
            itList = lista.iterator();
            data = new Object[lista.size()][2];
            while (itList.hasNext()) {
                unXbean = (UsuarioBean)itList.next();
                data[contador][0] = unXbean.getLoginUser();
                data[contador][1] = unXbean.getNombre1();
                contador++;
            }
            FrmBusqueda find =
                new FrmBusqueda(new Frame(),  opciones, Head,
                                data);
            find.setVisible(true);
            fila = find.getData();
            find.dispose();
            find = null;

            codigoCasa = (String)fila[0];
            this.unUsuarioBean = unUsuarioBO.buscar(codigoCasa);
            if (unUsuarioBean  != null ){
                jLabel4.setText(unUsuarioBean.getNombre1()
+unUsuarioBean.getNombre2() +unUsuarioBean.getApellido() );
                jTextField3.setText(unUsuarioBean.getLoginUser());
            }
            //mostrar();
        } catch (Throwable e) {
```

```
            System.out.println(e);
        }
    }
```

```
 private void jButton1_actionPerformed(ActionEvent e) {
        // proceso reporte
         JasperDesign jasperDesign;
         JasperReport jasperReport;
         JasperPrint jasperPrint;
         JRViewer jrv;
         Date fdesde = null;
         Date fhasta = null;
         try{
            fdesde = DataUtil.parseDate(jTextField1.getText());
            fhasta = DataUtil.parseDate(jTextField1.getText());
         }catch(ParseException er){
            er.printStackTrace();
         }

jScrollPane1.getViewport().removeAll();
 try{
 HashMap parametros = new HashMap();
 parametros.put("fdesde", fdesde);
 parametros.put("fhasta", fhasta);
 parametros.put("usuarioLog", jTextField3.getText());
 jasperPrint                                                    =
JasperFillManager.fillReport("C:\\reportes\\reporte_bloqueos.jasper",parametros,Ba
se.getConexion());
             jrv = new JRViewer(jasperPrint);
             jrv.setPreferredSize(new Dimension(600, 410));
             jScrollPane1.getViewport().add(jrv, null);
             jrv.setVisible(true);
             jScrollPane1.repaint();
         }catch(JRException j){
             System.out.println(j.getMessage());
         }
    }

    private void jButton3_actionPerformed(ActionEvent e) {
    quit();
    }
}
```

Mantenimiento

Edte codigo muestra como se da mantenimiento a cada uno de los usuarios del sistema

```
public class UsuarioBo  extends BO{
    UsuariosDao  unUsuarioDAO = null;

    public UsuarioBo() {
    }

    public UsuarioBean buscar(int idUsuario) {
        UsuarioBean valorRetornar = new UsuarioBean();
        try {
            conectarDao();
            valorRetornar = unUsuarioDAO.buscarRegistro(idUsuario);
        } catch (Throwable e) {
            System.out.println(e);
        }
        return valorRetornar;
    }

    public UsuarioBean buscar(String  identificacion) {
        UsuarioBean valorRetornar = new UsuarioBean();
        try {
            conectarDao();
            valorRetornar = unUsuarioDAO.buscarRegistro(identificacion);
        } catch (Throwable e) {
            System.out.println(e);
        }
        return valorRetornar;
    }
```

```
 public ArrayList buscarTodos(){
        ArrayList lista = new ArrayList();
        try{
            conectarDao();
            lista = unUsuarioDAO.buscarTodos();
        }catch(Throwable e){
            e.printStackTrace();
        }
        return lista;
       public boolean insertarUsuario(UsuarioBean unBean ){
        boolean retorna = false;
        try{
            conectarDao();
            retorna = unUsuarioDAO.InsertaRegistro(unBean);
        }catch(Throwable e){
            retorna = false;
            e.printStackTrace();
```

```
        } return retorna;    }
```

Realización del update del usuario una vez ya encontrado el mismo.

```
 public boolean actualizarUsuario(UsuarioBean unBean ){
        boolean retorna = false;
        try{
            conectarDao();
            retorna = unUsuarioDAO.ActualizaRegistro(unBean);
        }catch(Throwable e){
            retorna = false;
            e.printStackTrace();
        }
        return retorna;    }

    protected void conectarDao() {
        if (unUsuarioDAO == null){
            unaCon = Base.getConexion();
            unUsuarioDAO = new UsuariosDao(unaCon);
        }
  }}
```

5.7 Reporte Bloqueos

El bloque se genera una vez que se han efectuado tres intentos erroneos de ingreso en el mismo instante.

```
public class ReporteBloqueos extends DefaultInternalFrame {
    private JPanel jPanel1 = new JPanel();
    private BarraNavega barraNavega1 = new BarraNavega();
    private UsuarioBo unUsuarioBO = null;
    private UsuarioBean unUsuarioBean = null;
    private JTextField jTextField1 = new JTextField();
    private JLabel jLabel1 = new JLabel();
    private JLabel jLabel2 = new JLabel();
    private JTextField jTextField2 = new JTextField();
    private JButton jButton1 = new JButton();
    private JLabel jLabel3 = new JLabel();
    private JTextField jTextField3 = new JTextField();
    private JLabel jLabel4 = new JLabel();
    private JButton jButton2 = new JButton();
    private JPanel jPanel2 = new JPanel();
    private BorderLayout borderLayout1 = new BorderLayout();
    private JScrollPane jScrollPane1 = new JScrollPane();
    private JButton jButton3 = new JButton();
```

```
    public ReporteBloqueos() {
        try {
            jbInit();
            unUsuarioBO = new UsuarioBo();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

5.8 Captura ID Pen Drive.

Este procedimiento es aquel que permite obterner el ID del pen drive y de esta manera hacer practicamente inviolable el pen.

Librerias necesarias para lograr la captura del id:

```
package client.usb;
import ch.ntb.usb.LibusbJava;
import ch.ntb.usb.Usb_Bus;
import ch.ntb.usb.Usb_Config_Descriptor;
import ch.ntb.usb.Usb_Device;
import ch.ntb.usb.Usb_Device_Descriptor;
import ch.ntb.usb.Usb_Interface;
import ch.ntb.usb.Usb_Interface_Descriptor;
import ch.ntb.usb.usbView.UsbTreeModel;

import client.ClienteFrm;

import java.util.Enumeration;

import javax.swing.JTree;
import javax.swing.tree.TreeNode;
import javax.swing.tree.TreePath;
```

Class SensorUsb hilo que permite la lectura del usb como tal, el cual necesita librerias dll.

```
public class SensorUsb extends Thread {
    private boolean encontrado = false;
    private boolean conectado = false;
    String serialNumberProduct = "abb38e52152293";
    short idProduct = 0x163;
    ClienteFrm unaForma;
    private String operacion;
    Usb_Device deviceEncontrado;

    public SensorUsb(ClienteFrm forma) {
        this.unaForma = forma;
        operacion = "B";
    }
```

La siguiente funcion se obtiene gracias a una libreria el cual lo que hace es mostrar un árbol de dispositivos USB donde los busca y encuentra estos dispositivos:

```
    private Usb_Bus getBus() {
        Usb_Bus bus;
        LibusbJava.usb_init();
        LibusbJava.usb_find_busses();
        LibusbJava.usb_find_devices();
        bus = LibusbJava.usb_get_busses();
        return bus;
    }
```

De la función anterior se obtiene un valor que es enviado a la funcion escaneaNodo el cual verifica que el id sensado sea igual al que se encuentra almacenado .

```
public void scaneaNodo(Object nodo) {
        //Usb_Config_Descriptor verificaNodo;
        if (!encontrado) {
            Usb_Device_Descriptor verificaNodo;
            if (nodo instanceof Usb_Bus) {
                Usb_Device device = ((Usb_Bus)nodo).getDevices();
                scaneaNodo(device);
                return;
            }
            if (nodo instanceof Usb_Device) {
                //Usb_Device unnodo =;
                verificaNodo = ((Usb_Device)nodo).getDescriptor();
                deviceEncontrado = (Usb_Device)nodo;
                int handle = LibusbJava.usb_open((Usb_Device)nodo);
                String serialNumber =
                    LibusbJava.usb_get_string_simple(handle,
   verificaNodo.getISerialNumber());
                LibusbJava.usb_close(handle);
                System.out.println(serialNumber);
                //verificaNodo.
                System.out.println(((Usb_Device)nodo).getFilename());
                if (verificaNodo.getIdProduct() == idProduct &&
                    serialNumberProduct.equals(serialNumber)) {
```

```
                    // verificar el numero de serie del producto
                    encontrado = true;
                    System.out.println("Encontrado el Pen drive");
                    return;
                }
                //scaneaNodo(nodo);
                return;
            }
            if (nodo instanceof Usb_Config_Descriptor) {
                // verificar si es el Id o no
                return;

            }
            if (nodo instanceof Usb_Interface) {
                return;
            }
            if (nodo instanceof Usb_Interface_Descriptor) {
                //Usb_Interface_Descriptor otro;

                return;
            } else {
                //
                return;
            }

        }
        return;
    }
```

Funcion que corre con cada hilo

```
public void run() {
        // sensar los dispositovos USB en busca del penDrive
        //Usb_Device_Descriptor unDescriptor;
        while (true){
            try{
                sleep(100);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }

            if(operacion.equals("B")){  // buscar el dispositivo
                System.out.println("Buscando <---");
                try {
                    scaneaNodo(getBus());
                    if (encontrado) {
                        unaForma.recoger(encontrado);
                        this.operacion = " ";
                        sleep(1000);
                    } else {
                        sleep(1000);
```

```
                    }
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            if(operacion.equals("D")){ // desactivar  dispositivo
                //deviceEncontrado.
                 System.out.println("Matando");
                int handle2 = LibusbJava.usb_open(deviceEncontrado);
                //LibusbJava.usb_reset(handle2);

                LibusbJava.usb_release_interface(handle2,
deviceEncontrado.getDescriptor().getIdProduct());
                LibusbJava.usb_close(handle2);
                this.operacion = " ";
                try{
                    sleep(10);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        }
    }

    public void setEncontrado(boolean encontrado) {
        this.encontrado = encontrado;
    }

    public boolean isEncontrado() {
        return encontrado;
    }

    public void setOperacion(String operacion) {
        this.operacion = operacion;
    }

    public String getOperacion() {
        return operacion;
    }
}
```

DICCIONARIO DE DATOS

Un diccionario de datos es una lista de todos los elementos incluido en el conjunto de los diagramas de flujo de datos que describen un sistema. Los elementos principales en un sistema, el almacenamiento de datos y los procesos. El diccionario de datos almacena detalles y descripciones de estos elementos.

MANUAL DE USUARIO

1. Introducción

Este manual va dirigido a todas aquellas personas que sientan un gran interés en usar el sistema denominado “Cancerbero”. Se encuentra detallado de una forma clara todos los pasos a seguir para su instalación, y así el usuario lo aprenda a utilizar en el menor tiempo posible y sea del agrado del mismo.

El sistema simula un token para firma digital en un pen drive el cual alberga albergar una firma digital que impide la accesibilidad y copia de la información almacenada por personas no autorizadas. El usuario que vaya a emplear este documento es responsable del uso óptimo del sistema.

2. Instalación de “Cancerbero”

Para levantar el sistema es necesario tener instalado en la maquina donde residirá el mismo lo siguiente:

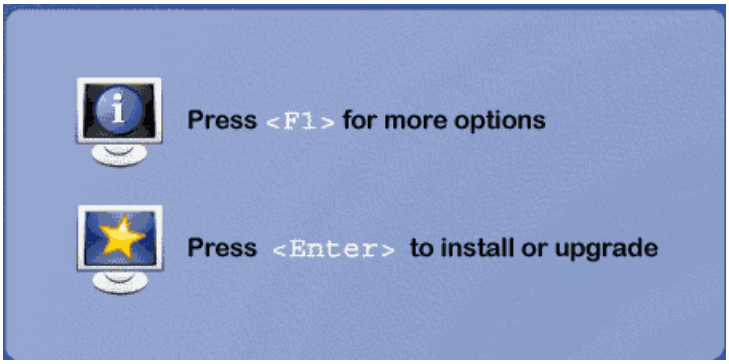
- Linux . Preferentemente en sus distribuciones Fedora Core, Mandrake
- JDK (JAVA)
- BDD (POSTGRESQL)

2.1 Proceso de Instalación de Linux Mandrake

Linux Mandrake / Mandriva no es más difícil de instalar que cualquier versión de Windows.

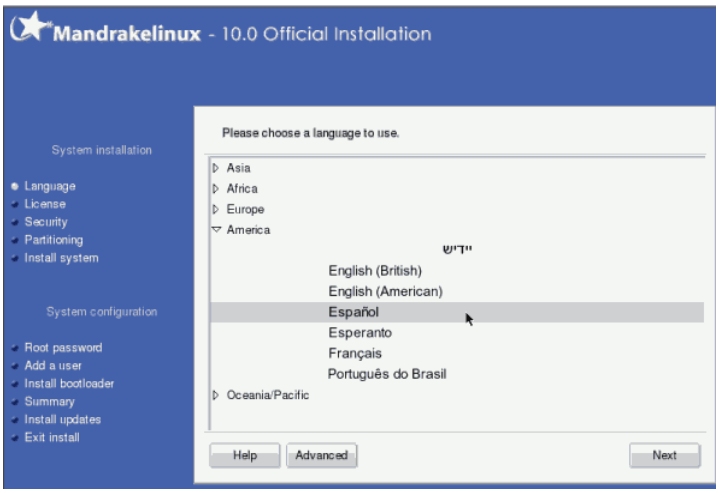
Comenzando la instalación

Al arrancar el CD se mostrará la siguiente ventana.



Presione ENTER para comenzar la instalación.

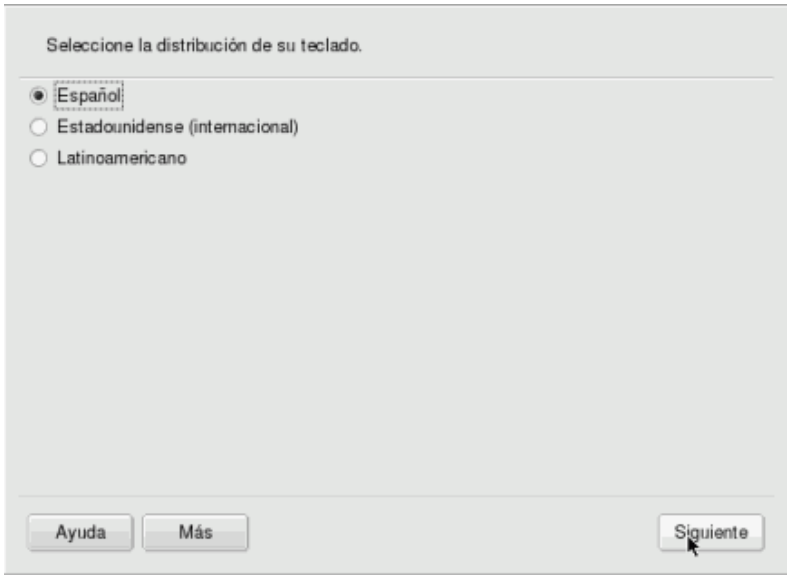
Elegir el idioma de la instalación.



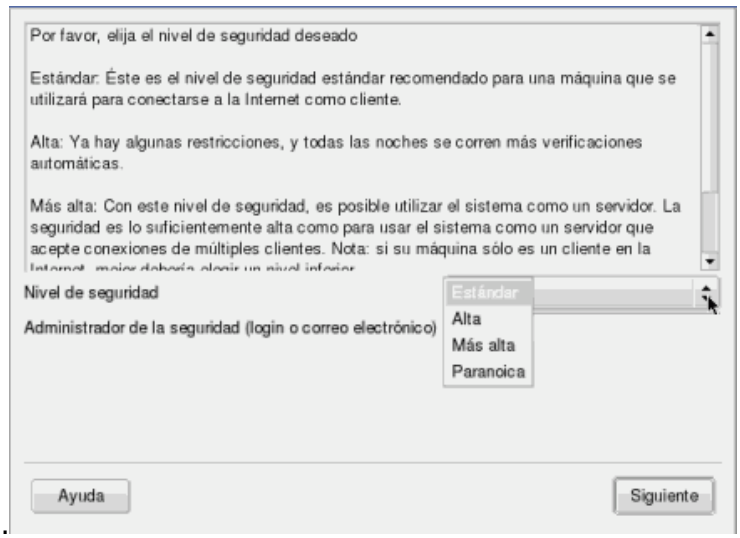
Contrato de la licencia, seleccione Aceptar y presione Siguiente.

En la lista de la derecha, se muestran todos los pasos, que quedan para acabar la instalación.

Elija la distribución del teclado.

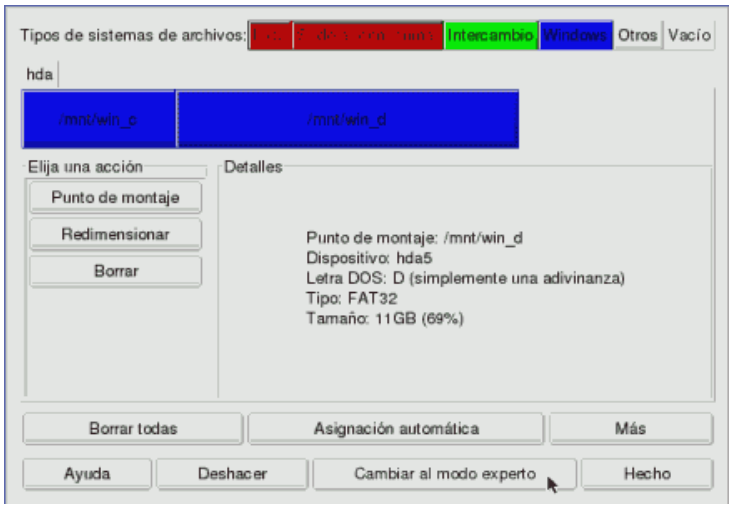


Selección del nivel de seguridad de acuerdo a las necesidades.

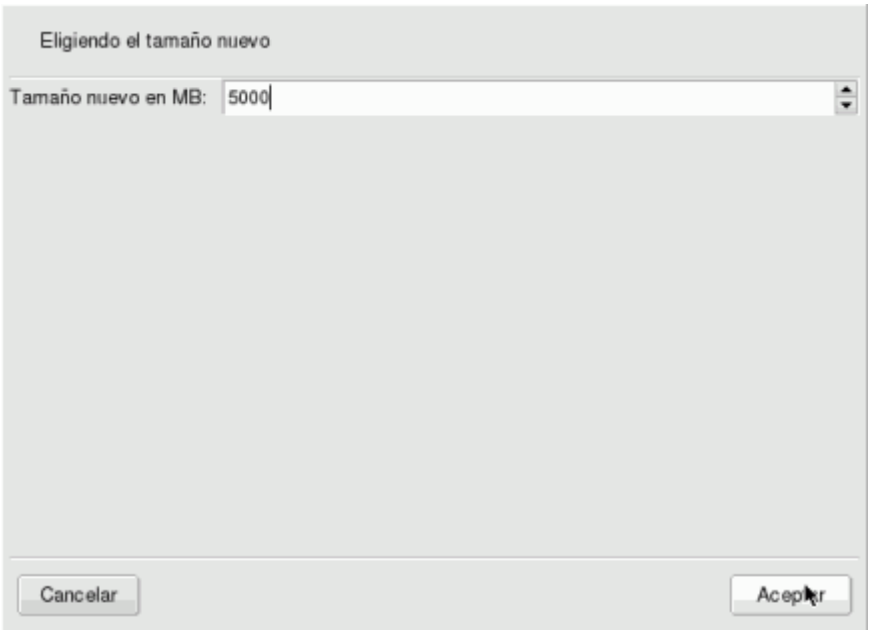


Particionamiento del disco.

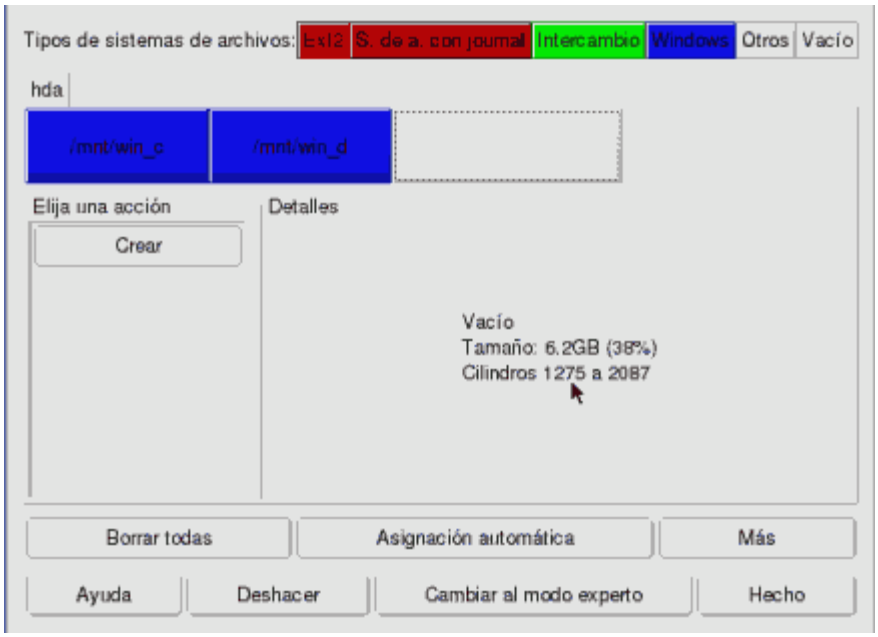
La instalación Mandrake/Mandriva pone a disposición un completo gestor de particiones



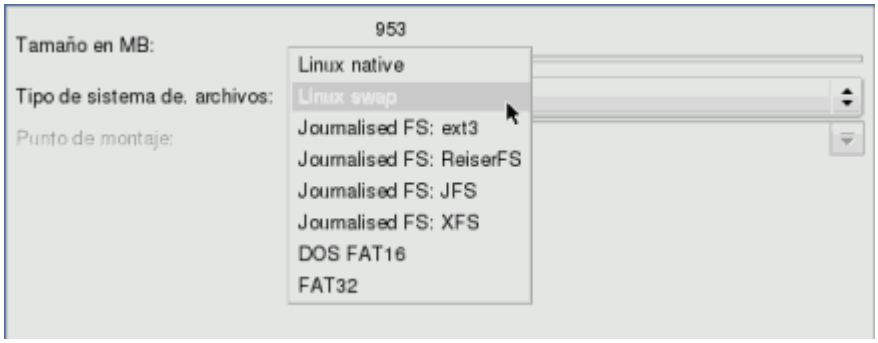
Aquí se podra ver el estado del disco disco duro. Las particiones nuevas se las crea a continuación.



Se crea cada particion



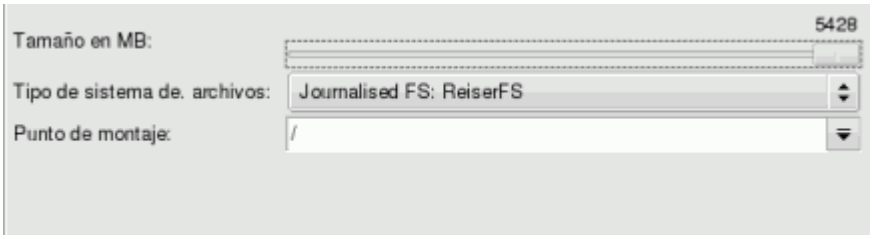
Espacio no asignado donde se instalara Linux. Se pulsa sobre el espacio no asignado y se procede a crear la partición SWAP.



De la lista Tipo de sistema de archivos seleccionar Linux swap y desplaza el botón del Tamaño para seleccionar el que se desea . Este espacio es utilizado como más memoria RAM.

Una vez creada la partición SWAP, se pulsa obre el espacio en blanco para crear el resto de particiones. Retorno a la opción Crear. En Tipo de sistema de

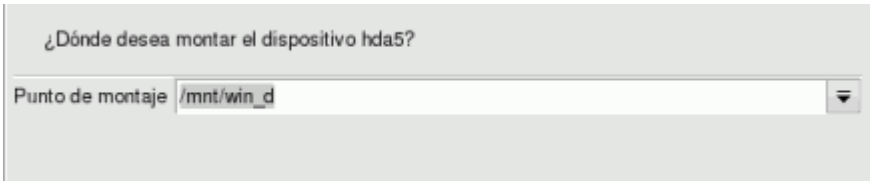
archivos elegimos el formato de la partición Ext3 o Journaling/ReiserFS Se elige un punto de montaje y elegir el tamaño.



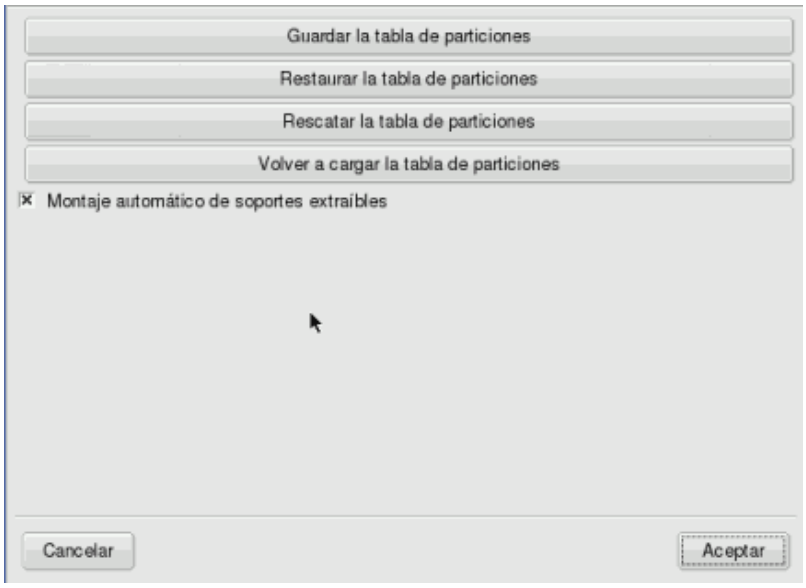
Si se desea rectificar alguno de los pasos se pulsa el botón "Deshacer".

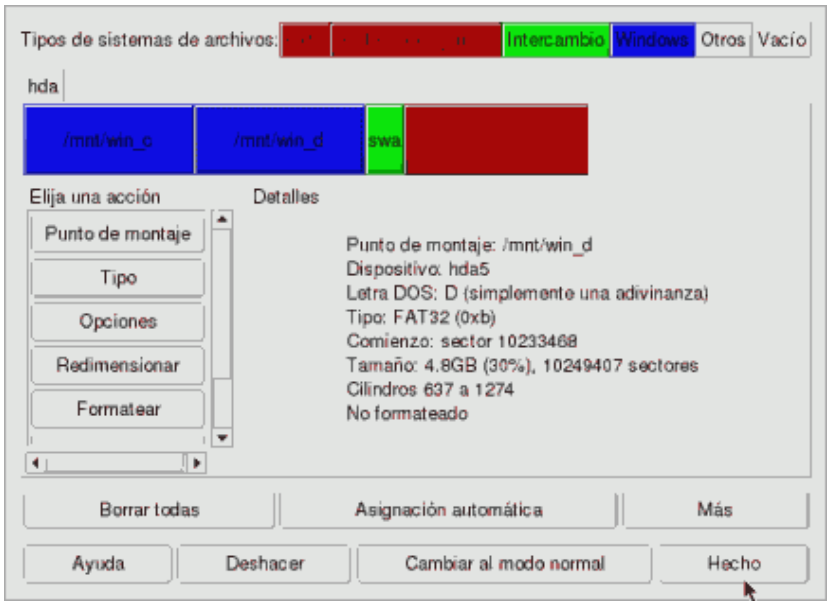
Punto de montaje.

A demás de las particiones de Linux, también se puede cambiar el punto de montaje del resto de las particiones.



El botón Más ofrece la posibilidad de guardar en un disquete el estado de la tabla de particiones (el MBR), para poder restaurarla en caso de querer dejarla como antes de hacer los cambios.





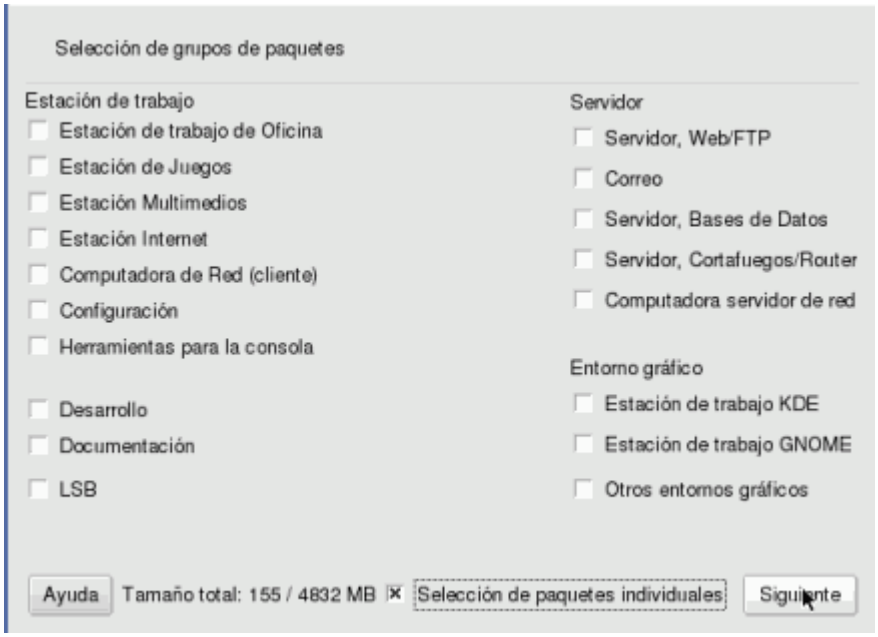
Click en hecho. Dependiendo de lo realizado en el gestor de particiones es posible que el programa muestre una lista de particiones donde elegir cuales queremos que se formateen y cuales no, marcándolas o desmarcándolas de la lista.

En este momento se procedea la instalación del sistema.

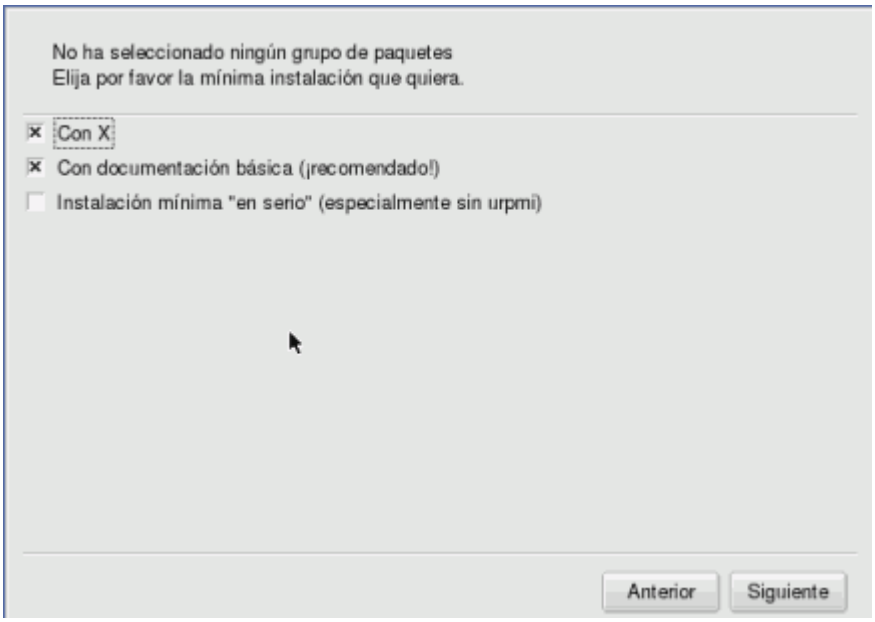
Instalar el sistema.

Selección de grupos de paquetes.

Selección de paquetes individuales de acuerdo a las necesidades. Se debe de manejar esto con mucho cuidado ya que al instalar mas paquetes, se requerira de mayor recursos asi como se dejaran muchos mas puertos abiertos. Solo se debe instalar los paquetes indispensables

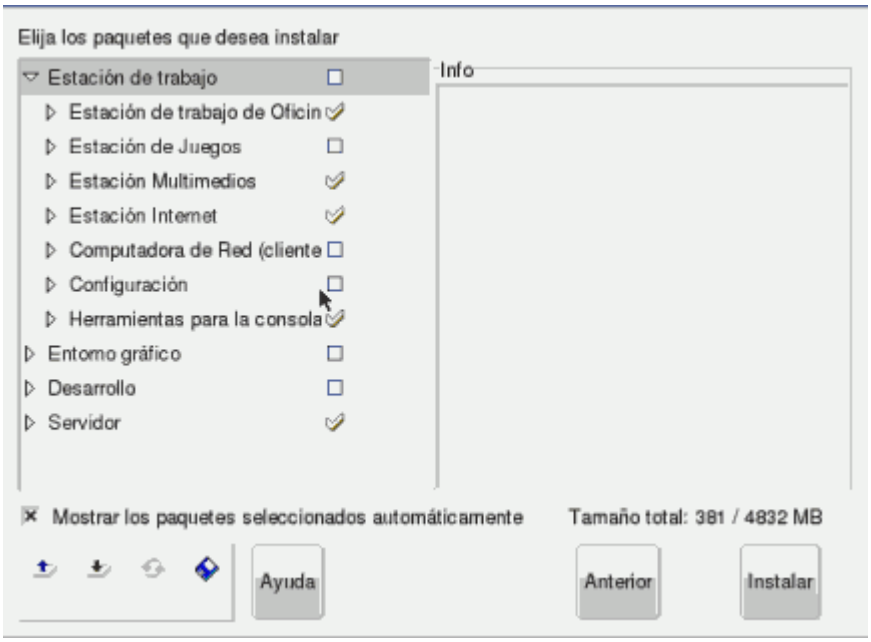


. Es recomendable seleccionar la instalación con X para no llevarse sorpresas desagradables porque faltó seleccionar algún paquete importante para el funcionamiento del servidor gráfico.



Selección Individual.

Seleccione solo los paquetes mínimos para usar KDE , dejando el resto de programas para el instalador del Centro de Control Mandrake, donde aparecen todos los paquetes mejor ordenados (separados por categorías mas descriptivas como, navegadores, correo, audio, video, etc.), una descripción mas detallada del paquete, así como un buscador para no tener que estar buscando por entre los paquetes a los que se quieren, e incluso se puede buscar entre sus descripciones.

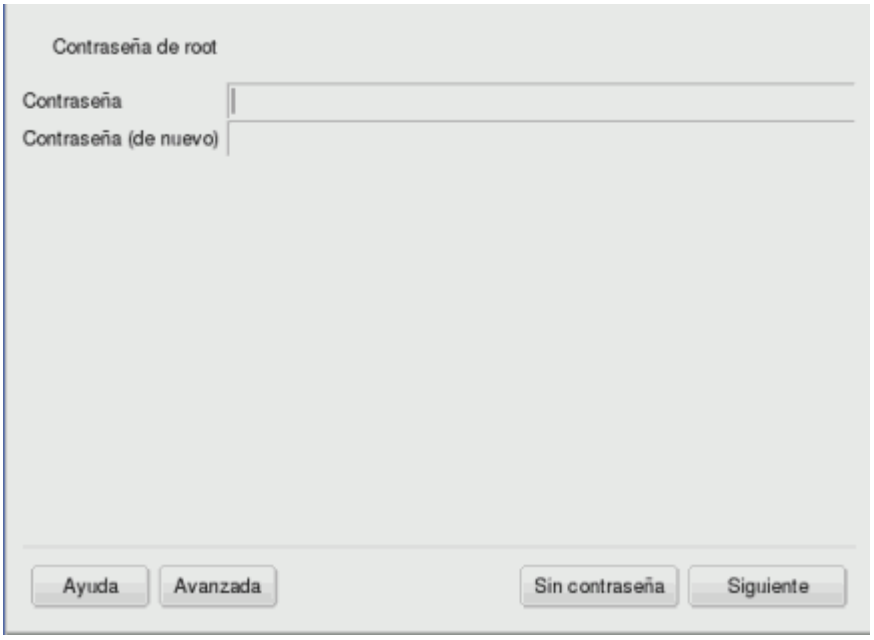


Para saber que se instalo todo lo necesario y no llevase sorpresas, se ayudara de que Mandrake/Mandriva añadirá los paquetes necesarios para el programa elegido, así por ejemplo para tener KDE funcionando "completamente" y Mandrake/Mandriva con todas sus herramientas de configuración, bastará con que seleccionemos el paquete del idioma Español de KDE kde-i18n-es y alguna aplicación importante de este entorno gráfico .

Contraseña de Root.

Una vez acabada la instalación de los paquetes seleccionados se mostrará la siguiente pantalla.

El root es un super –usuario es aquel que tiene todos los permisos. Se recomienda escribir una contraseña lo mas robusta posible.

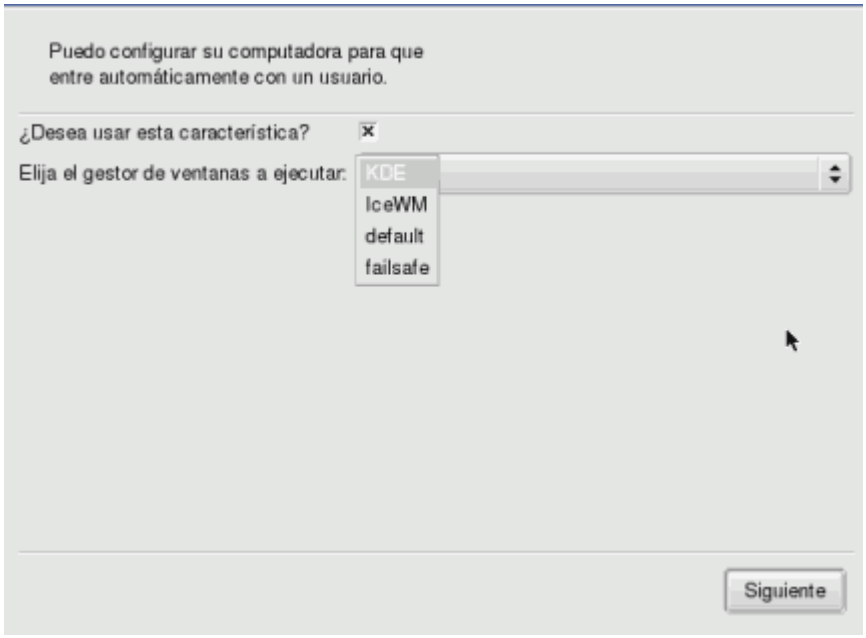


Añadir un usuario.

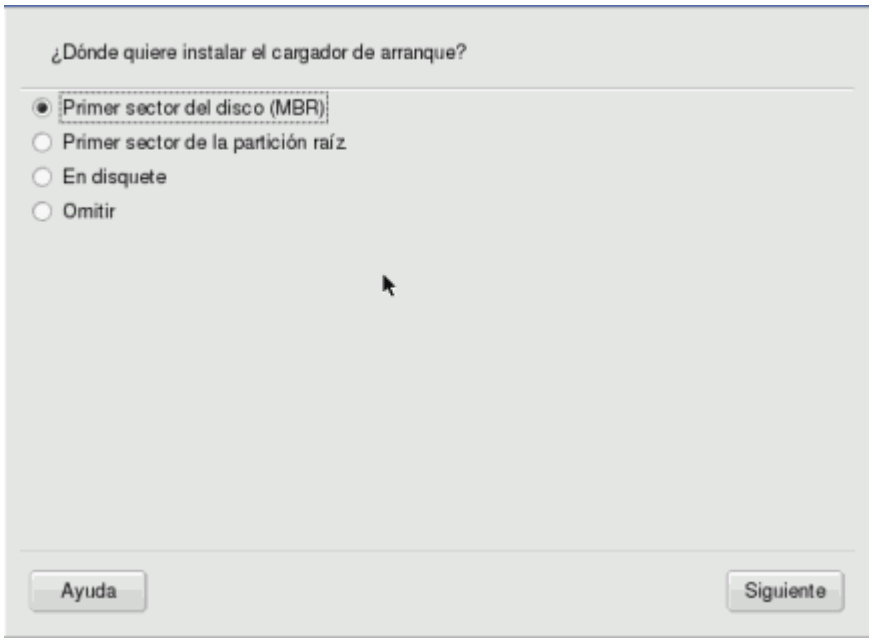
El usuario con el cual se trabajara

Se completar el formulario con los datos y pulsar "Aceptar el usuario" para introducir los datos del usuario y luego ingresar los datos para otro usuario, o "Siguiente" si se ha terminado.

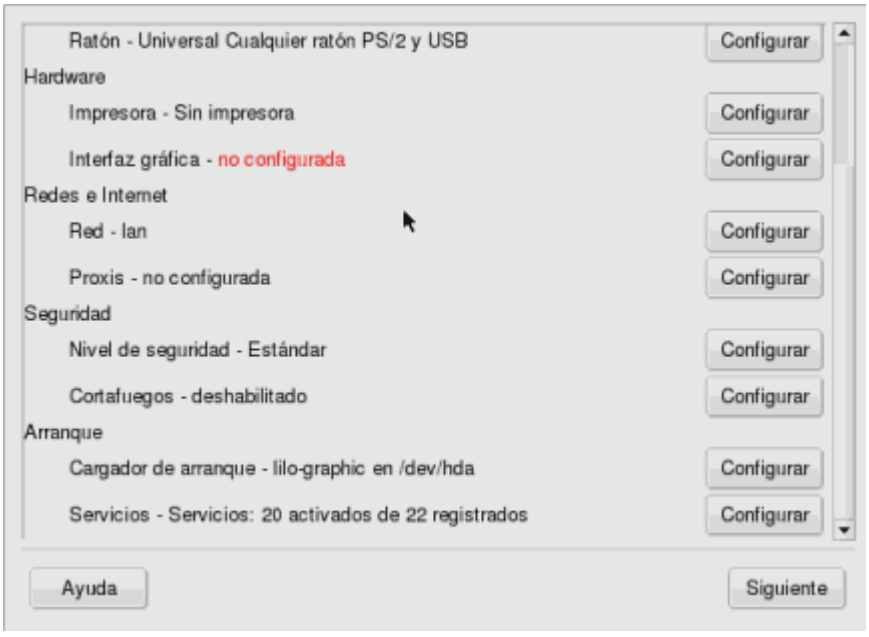
Mandrake/Mandriva nos ofrece la opción de que el sistema se inicie automáticamente con un usuario sin necesidad de identificarse y si se quiere, en un entorno directamente.



Cargador de arranque.


¿Dónde quiere instalar el cargador de arranque?
Primer sector del disco (MBR)
Primer sector de la partición raíz
En disquete
Omitir
Ayuda
Siguiente

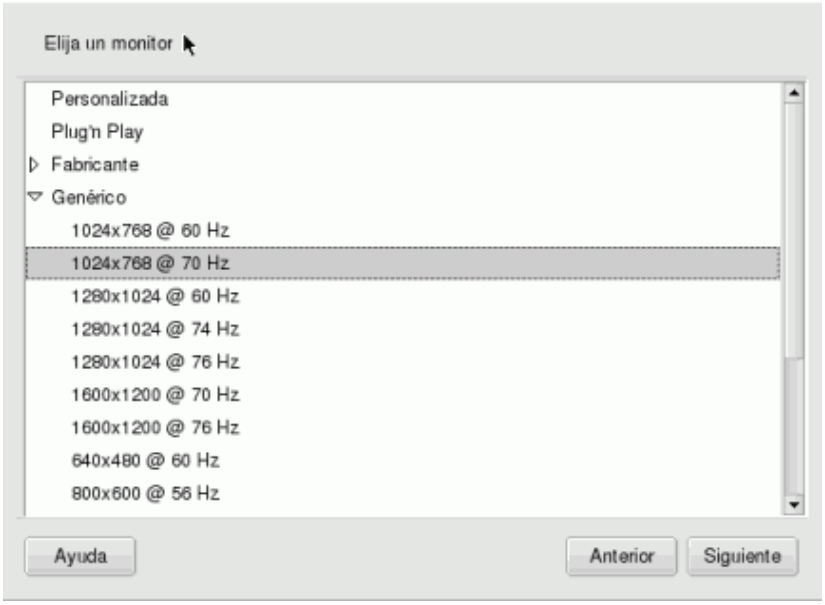
Resumen.


Ratón - Universal Cualquier ratón PS/2 y USB
Configurar
Hardware
Impresora - Sin impresora
Configurar
Interfaz gráfica - no configurada
Configurar
Redes e Internet
Red - lan
Configurar
Proxis - no configurada
Configurar
Seguridad
Nivel de seguridad - Estándar
Configurar
Cortafuegos - deshabilitado
Configurar
Arranque
Cargador de arranque - lilo-graphic en /dev/hda
Configurar
Servicios - Servicios: 20 activados de 22 registrados
Configurar
Ayuda
Siguiente

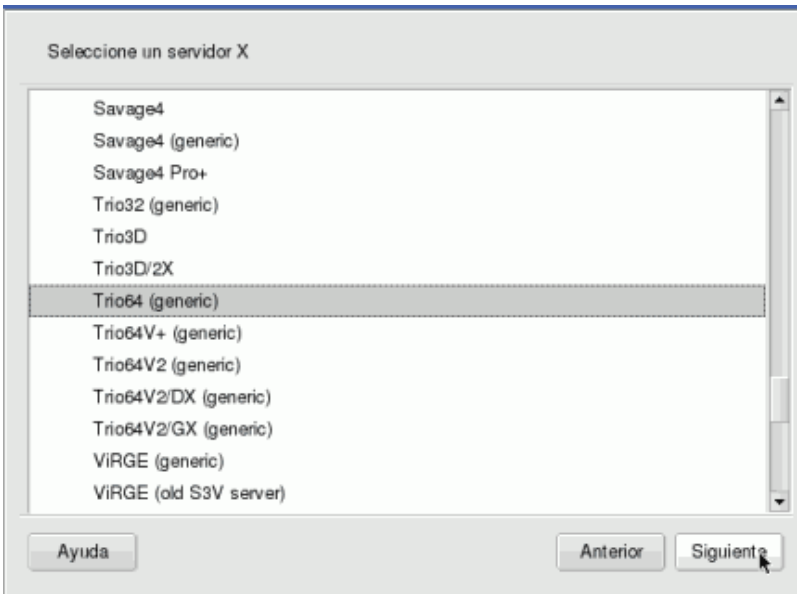
Interfaz Gráfica.

Una parte imprescindible para los principiantes, ya que sin ella se encontrarán perdidos, en la mayoría de los casos la detección automática funciona correctamente, pero por si acaso, porque queremos modificarla y por lo que es más importante PROBARLA. Aquí están los pasos a seguir.

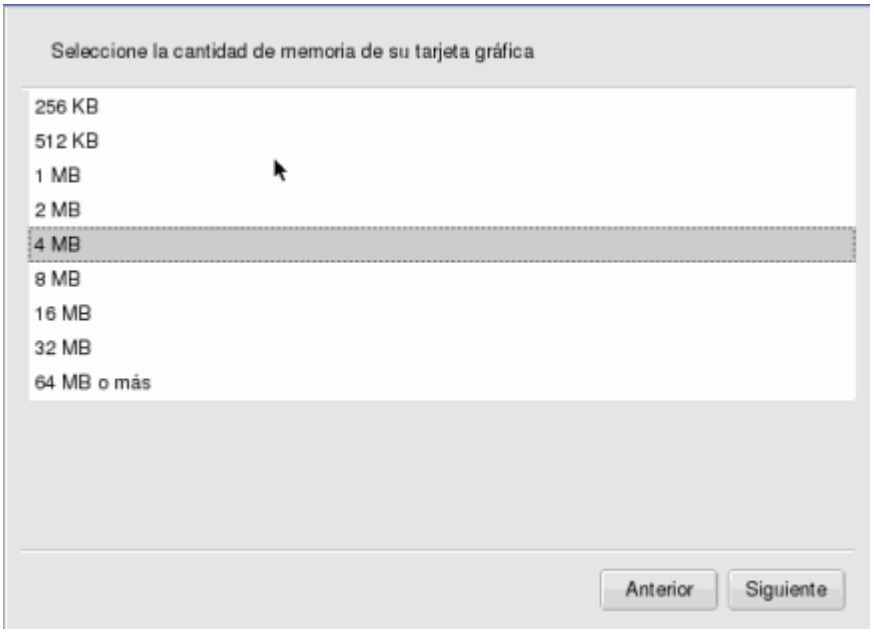
- Monitor



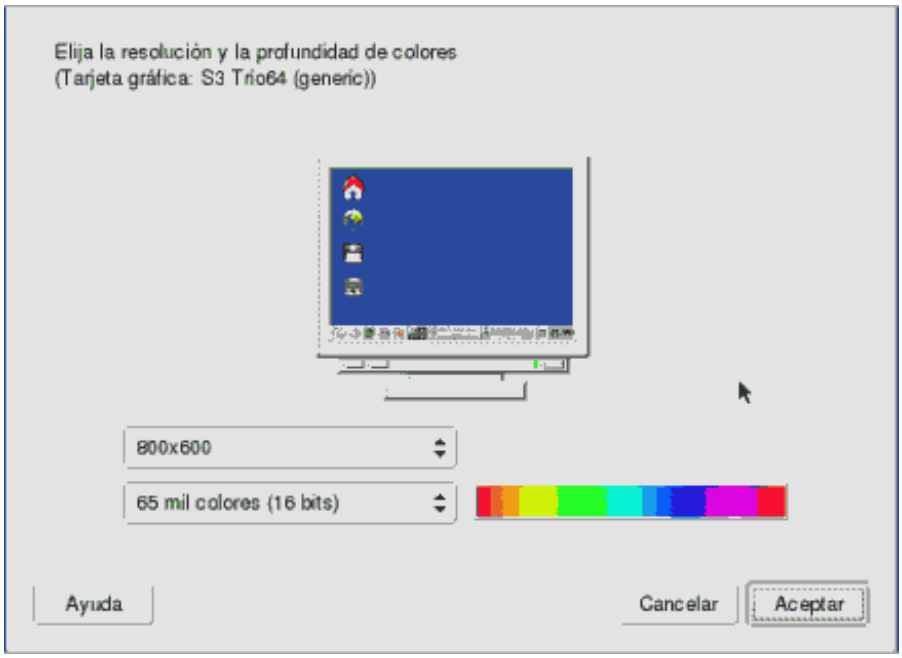
- Tarjeta gráfica



- Memoria de la tarjeta (depende del modelo seleccionado)



- Resolución

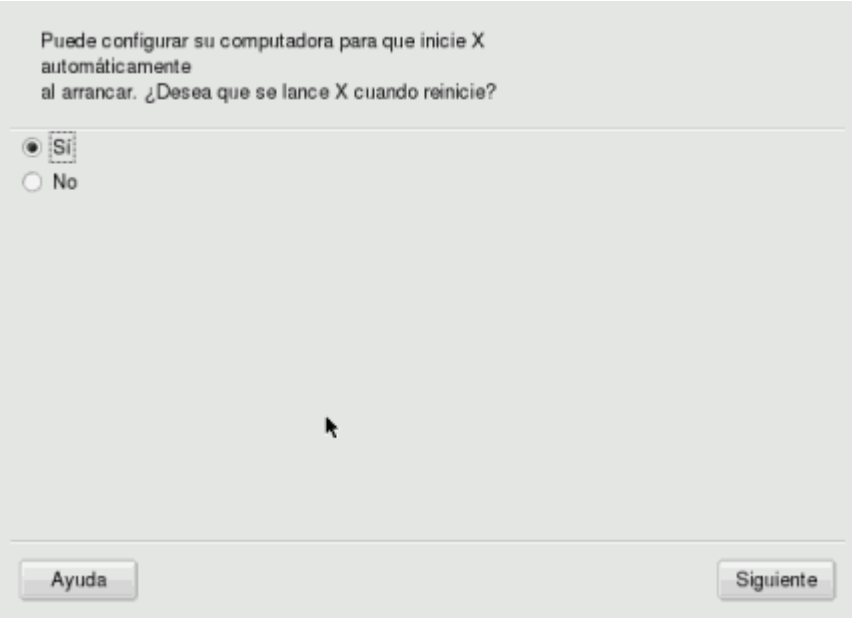


- Prueba

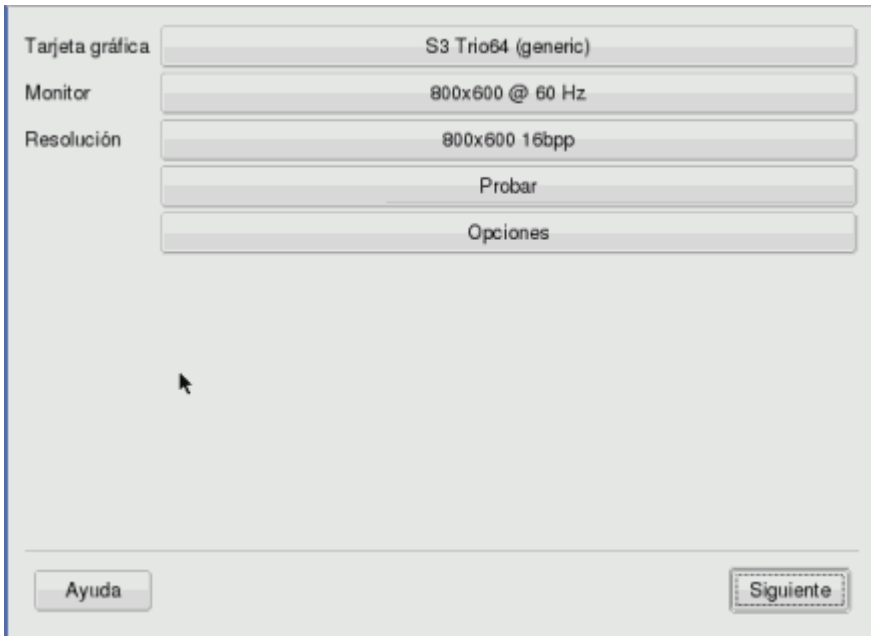
Esta es la pantalla que debemos ver si la configuración funciona correctamente.

- Opciones

Si se quiere que el sistema arranque en modo gráfico automáticamente, se marca SI

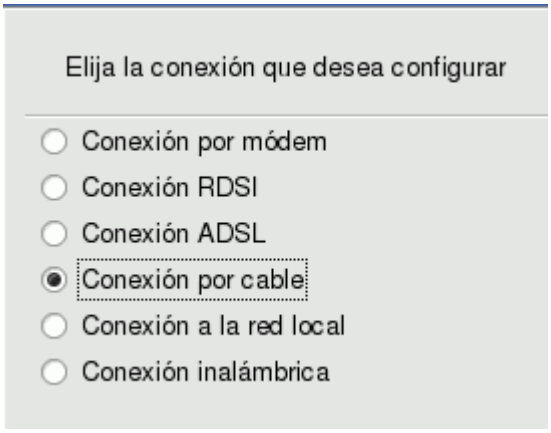


Si ya se tiene el servidor gráfico configurado, al presionar en Configurar, en lugar del proceso anterior, todos los pasos anteriores se muestran en una misma ventana donde se selecciona el que se desea configurar.

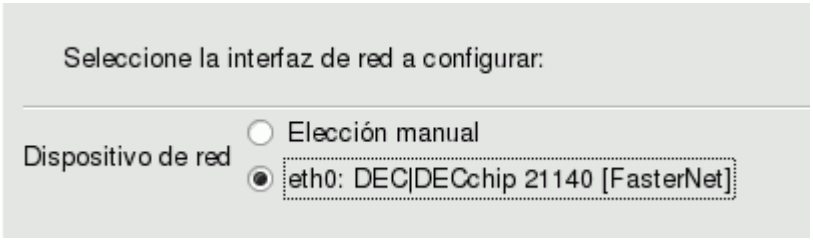


Red.

Tipo de conexión.

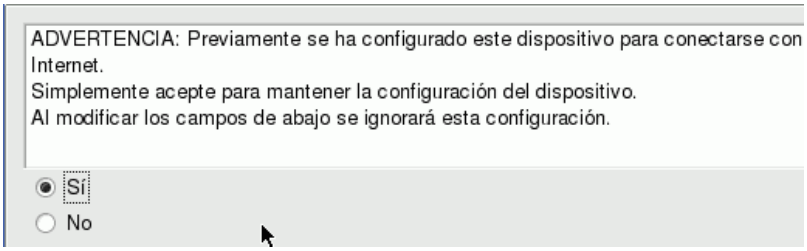


- Dispositivo

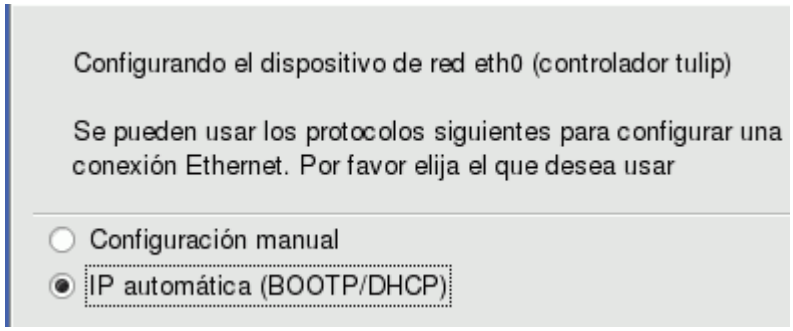


Modificar la configuración.

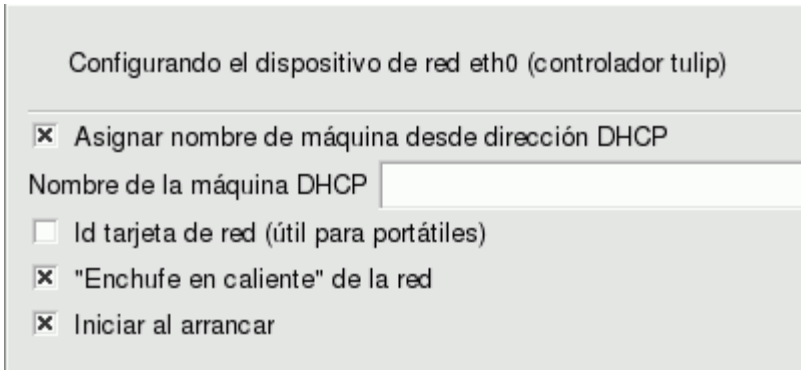
Si, para continuar, No para salir de la configuración.



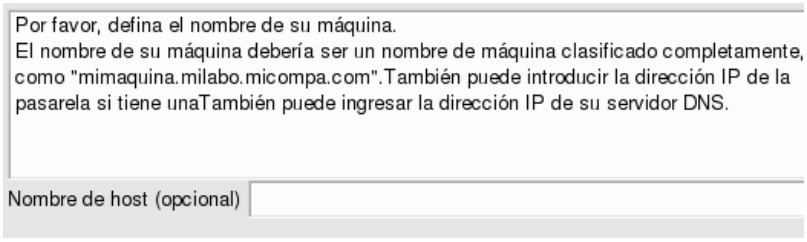
Tipo de configuración



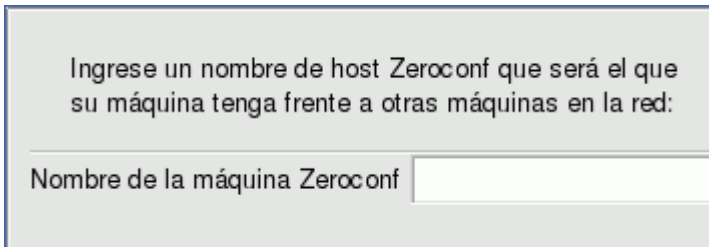
Configurar el dispositivo.



Nombre del host.

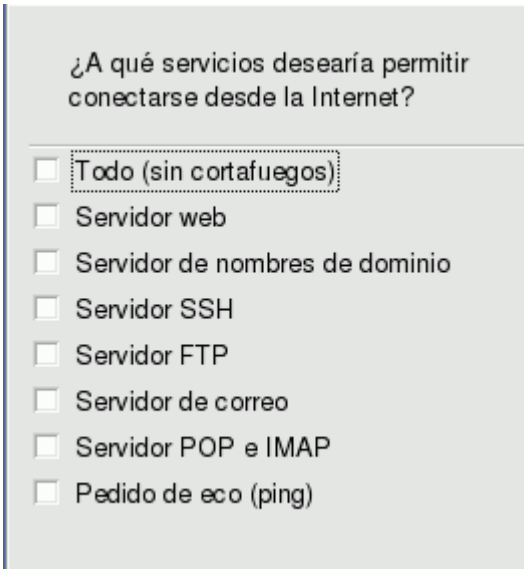


Nombre de la máquina.

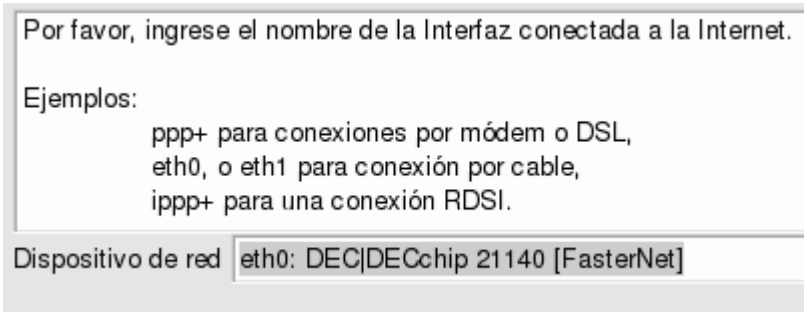


Cortafuegos. (Firewall)

Desde aquí se puede habilitar el cortafuegos (firewall) que se incluye en Mandrake/Mandriva. Solo seleccionaremos la casilla correspondiente si vamos a instalar un servidor de ese tipo en nuestra maquina.

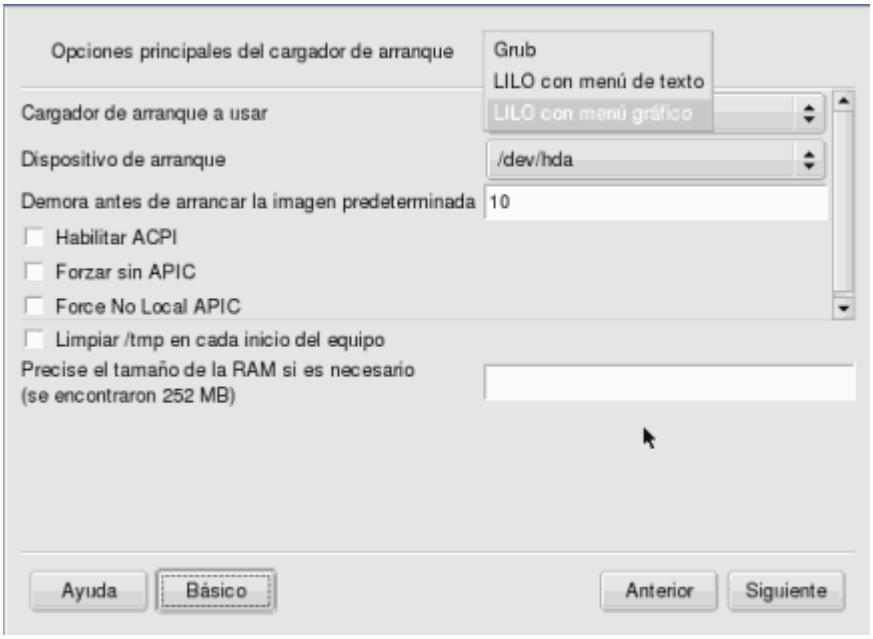


Se elige el dispositivo que controlará el cortafuegos.

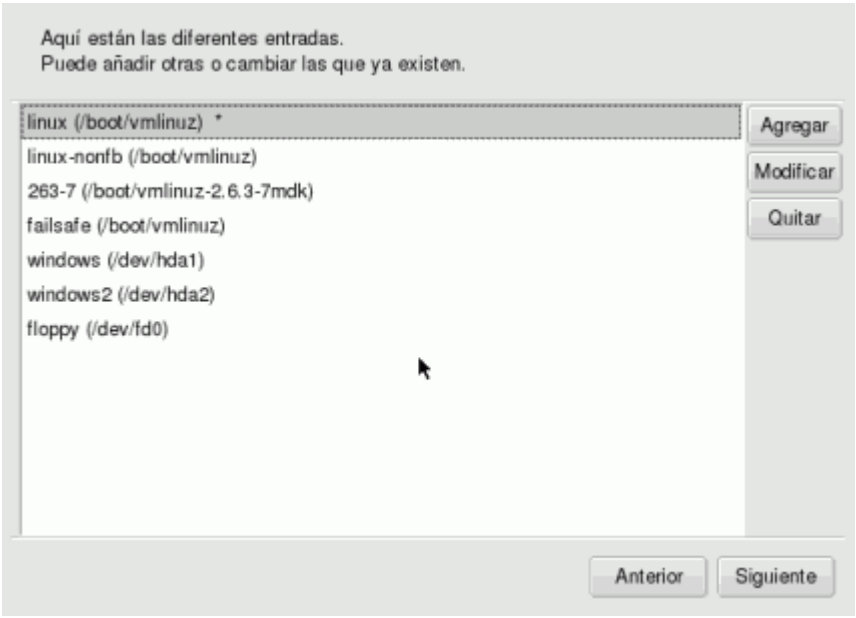


Cargador de arranque.

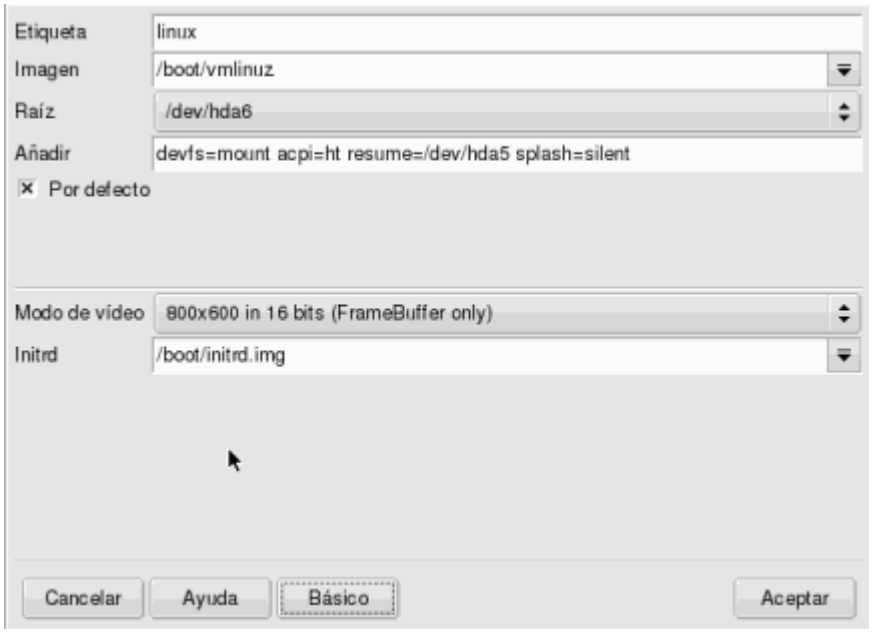
Se podra editar la configuración e incluso cambiar de cargador de arranque. Si no desea hacer cambios presionar Siguiente.



Siguiente para ver las entradas que contendrá el menú de arranque.



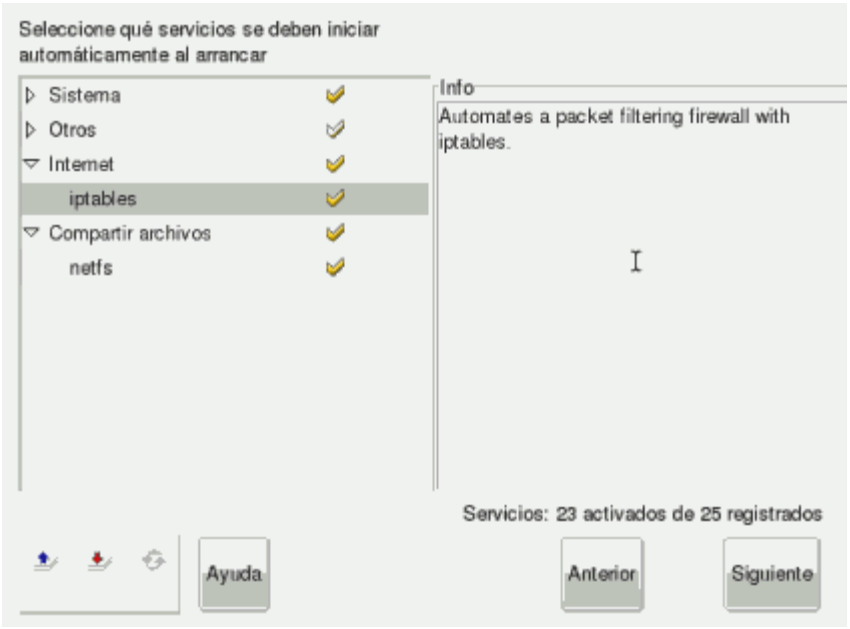
Se pueden eliminar las que no se desean presionando Quitar, o editarlas presionando el botón Modificar.



O añadir/editar entradas

Servicios.

Servicios que se debe de habilitar/dehabilitar al inicio, y también dependerá de lo que cada uno instale.



Termina para que se reinicie el computador y puedas entrar a probar tu nueva instalación de Linux Mandrake/Mandriva.

2.2 Proceso de Instalación de JDK

Lo primero que debemos hacer es crear una carpeta llamada “java” y dentro de ella creamos dos carpetas “tomcat” y “jdk” todo esto en el directorio “C:\”. Una vez efectuadas las indicaciones antes mencionadas nos ubicamos en la ruta “\Instaladores\J2E\” y empezamos haciendo clic en el archivo **“jdk-1_5_0_03.exe”** para proceder a instalar el J2E Development.

1. S

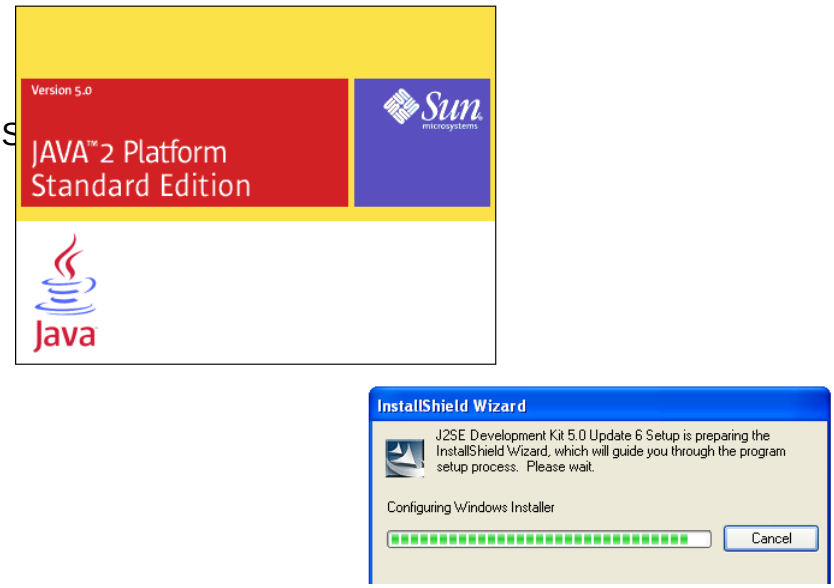


Figura 2.2.1 Preparando Instalación

2. Luego aparecerá la siguiente ventana donde debe seleccionar la primera opción y luego presionar “Next” para continuar con el proceso de instalación.

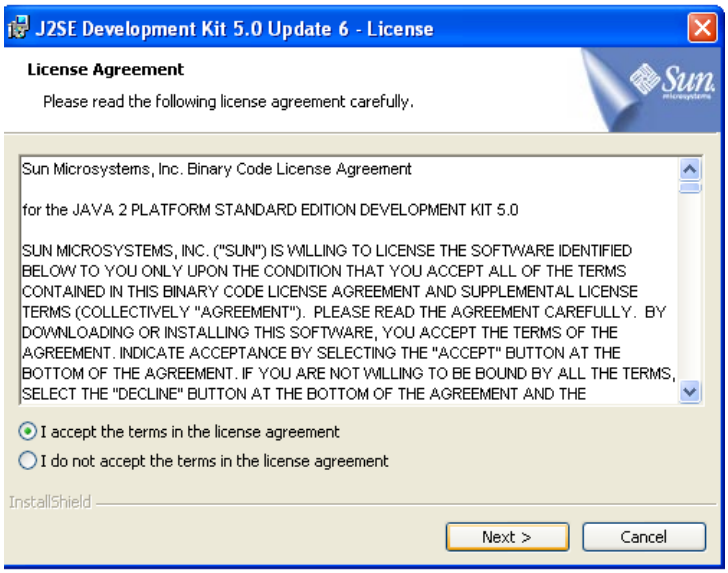


Figura 2.2.2 Aceptar Licencia

3. Ahora debemos ubicarnos en el directorio de instalación correcto que es “C:\java\jdk” y luego presionamos “Next”.

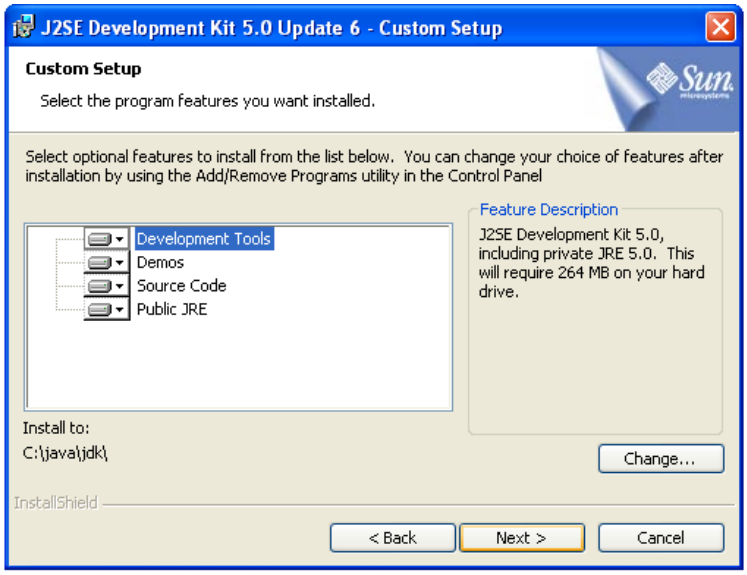


Figura 2.2.3 Ruta de instalación

4. Empieza la instalación del J2E y puede tardar unos minutos.

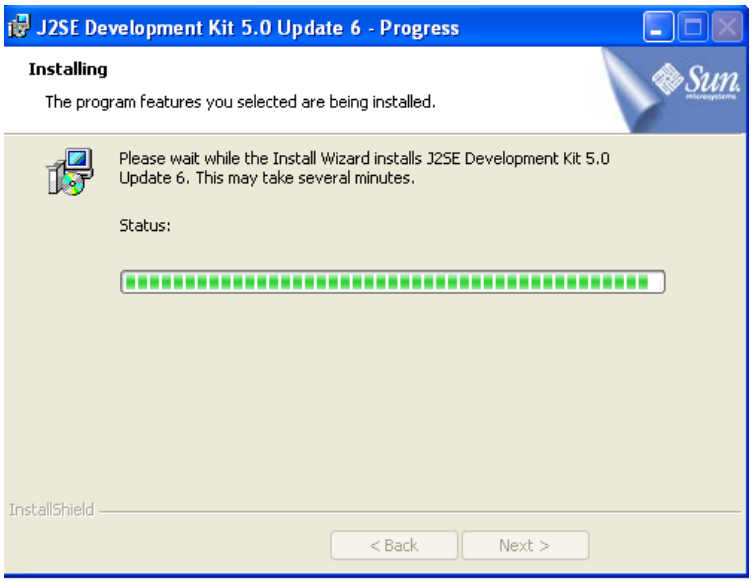


Figura 2.2.4 Proceso de Instalación

5. Se cambia el directorio para la instalación de los siguientes componentes al “C:\java\jdk\jre” y presionar “Next”.

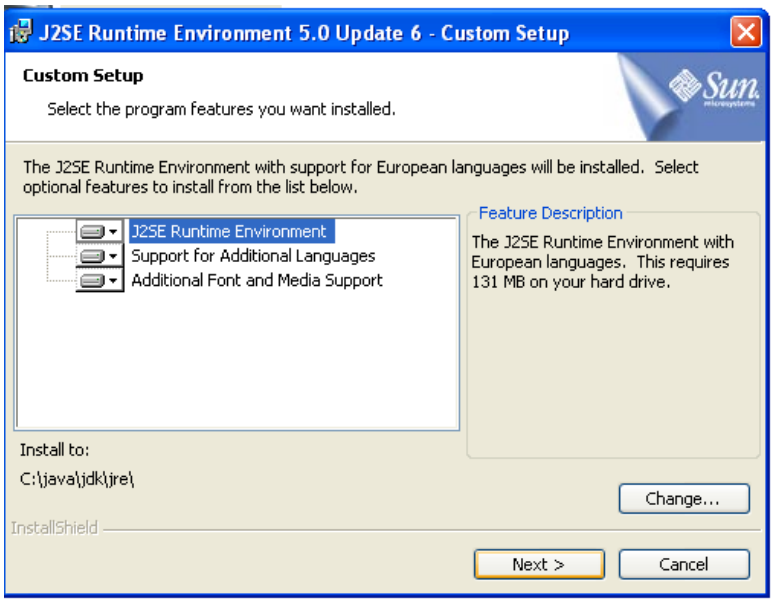


Figura 2.2.5 Selección de directorio

6. Seleccionar el browser (por defecto viene Microsoft Internet Explorer) y presionamos “Next”.

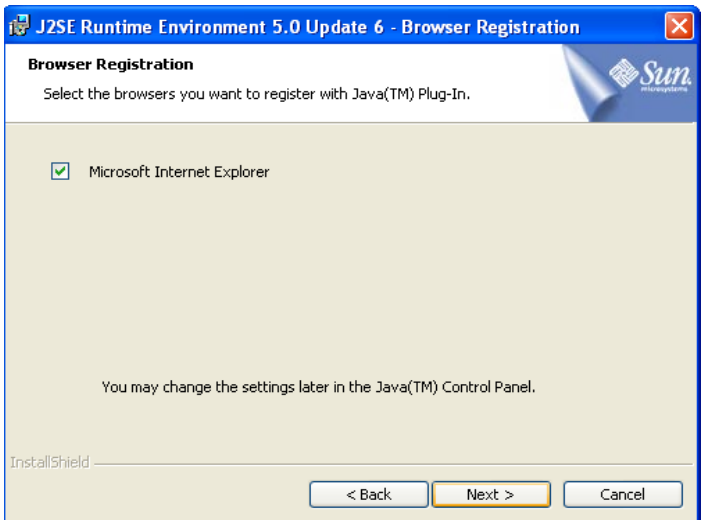


Figura 2.2.6 Selección de

7. Finalizando el proceso de instalación del J2E, esta operación puede tardar unos minutos.

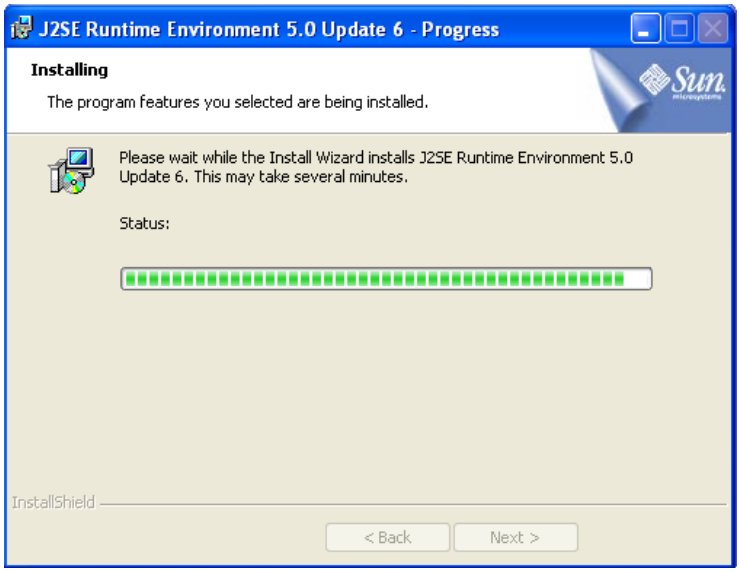
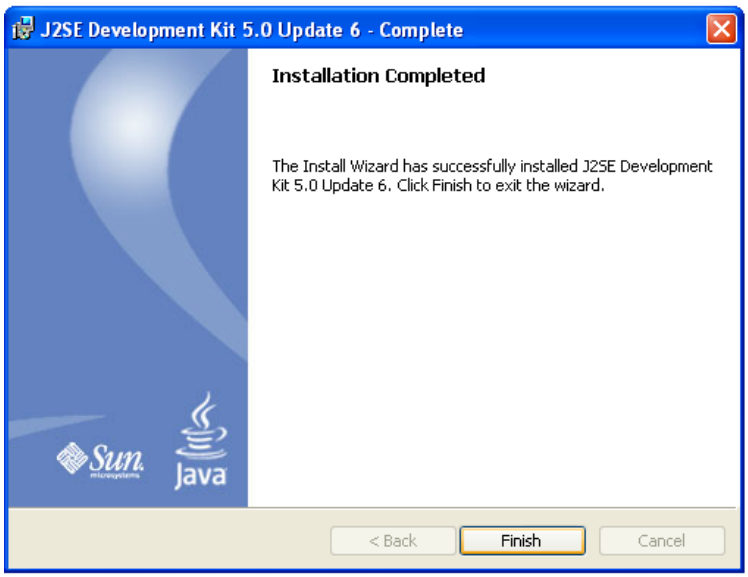


Figura 2.2.7 Finalizando la instalación

8. Instalación completada satisfactoriamente se ha finalizado el proceso.



A continuación, después de haber concluido el proceso de instalación del JDK, continuamos con la instalación de la base de datos que es PostgreSQL bajo Linux.

2.3 Proceso de Instalación de PostgreSQL

1. Crear grupo y usuario postgres.
El l "home directory" debe ser el mismo de la carpeta de instalación este caso será <</usr/local/pgsql>>

2. Ubicar en el directorio /usr/local/src el paqute de postgresql

mv postgresql-7.2.tar.gz /usr/local/src

3. Se crea un directorio llamado pgsql sobre el diecrtorio /usr/local.

mkdir /usr/local/pgsql

4. Situarse en el directorio /usr/local/src

cd /usr/local/src

5. Se descomprime el archivo .gz

gunzip postgresql-7.2.tar.gz

6. Desempaquetar el archivo .tar

tar -xvf postgresql-7.2.tar

7. Situarse en el directorio de intalación de PostgreSQL.

cd postgresql-7.2

8. Dar la ruta de la carpeta donde será instalado.
Utilizar el programa <<configure>>.

./configure --prefix=/usr/local/pgsql

9. Compilar el software para generar los archivos binarios.

make

10. Instalar el programa.

make install

11. Ahora cambiarle el dueño recursivamente a la carpeta y subcarpetas en las cuales se instaló el programa.

chown –R postgres:postgres /usr/local/pgsql

12. Editar el archivo rc.local para que mysql se inicie cuando se inicie el pc.

vi /etc/rc.d/rc.local

13. Agregar las siguientes lineas:

echo “starting PostgreSQL..”
rm /tmp/.s.PGSQL* 2> /dev/null
rm /usr/local/pgsql/data/postmaster.pid 2> /dev/null su – postgres –c

“/usr/local/pgsl/bin/postmaster –i –S –D /usr/local/pgsql/data > /usr/local/pgsql/data/postgreslog 2>&1”

14. Editar el archivo profile y asi crear la variable de ambiente de mysql y la configuración de las librerías y manuales..

vi /etc/profile

15. Agregar al Path después de /usr/local/bin:/usr/bin:/bin:_____(Creación de la variable de ambiente)

/usr/local/pgsql/bin:

16. Agregar a MANPATH después de /usr/man (Páginas de manuales)

/usr/local/pgsql/man:

17. Agregar después de MOZILLA_HOME 2 lineas (Librerías)

export PGLIB=/usr/local/pgsql/lib
export PGDATA=/usr/local/pgsql/data

18. Editar /etc/ld.so.conf y agregar

/usr/local/pgsql/lib

19. Configurar el encadenamiento dinámico en tiempo de ejecución.

ldconfig

20. Login como postgres (y crear las bases de datos)

su – postgres

initdb (Crea base de datos)

21. Crear la base de datos del sistema

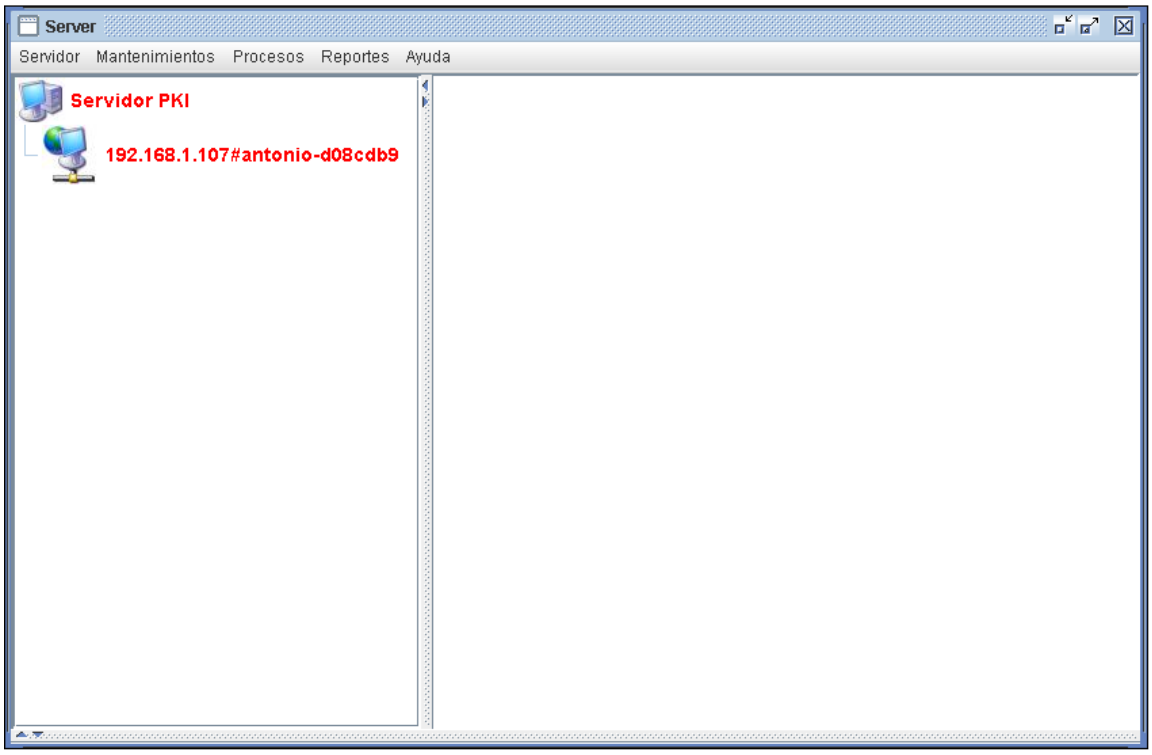
createdb postgres

22. Se reinicia el equipo.

2.4.- Aplicación “Cancerbero”

Pantallas principales

Pantalla de Inicio del ingreso del sistema.

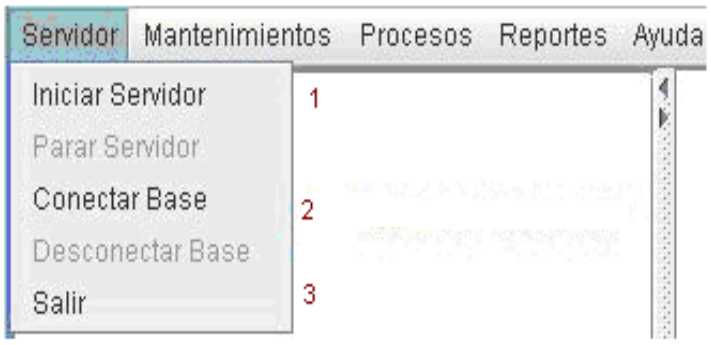


Al principio se muestra la pantalla en el cual, aparece en primera instancia la captura de la direccion ip desde la cual se esta conectando el cliente.

Gracias a la captura de esta Ip se lograr mayor seguridad.

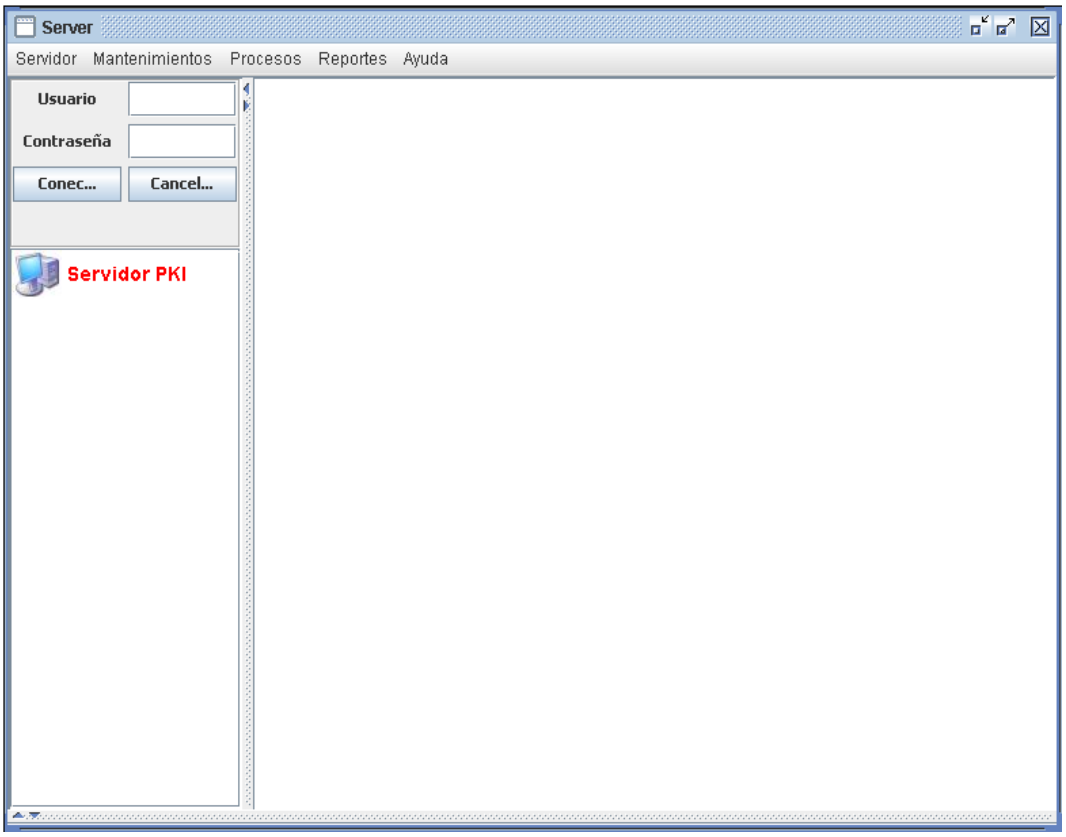
En la pantalla principal se encuentran diferentes menus como:

- Servidor
- Mantenimientos
- Procesos
- Reportes
- Ayuda



Para lograr la conexión se debe de inicializar el servidor . Todo esto es para obtener una mayor seguridad.

Para lograr la conexión se solicita una clave al administrador para lo cual aparece la siguiente pantalla.



1. Se debe ingresar el nombre de usuario o login que le fue asignado, como usuario del sistema.

2. Ingreso del password asignado. Habra políticas de password las cuales serán mínimo de 8 caracteres y usando combinación de los mismos letras, números y caracteres especiales claves asignadas automáticamente por el sistema a los usuarios

Inserción de usuarios

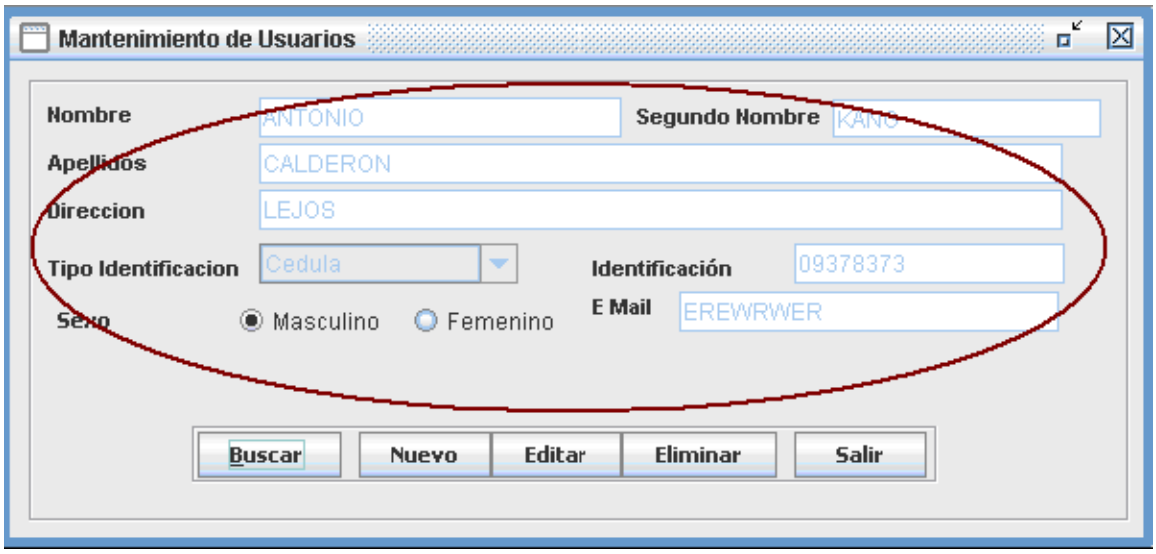
Al desear ingresar algún usuario nuevo sea este cliente u operador se ingresa al menú usuario el cual tendra un submenu con diferentes tipos de opciones como :

Ingreso

Actulizacion

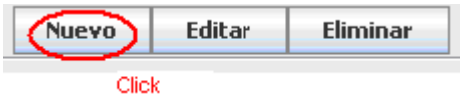
Eliminacion

Ingreso Usuario



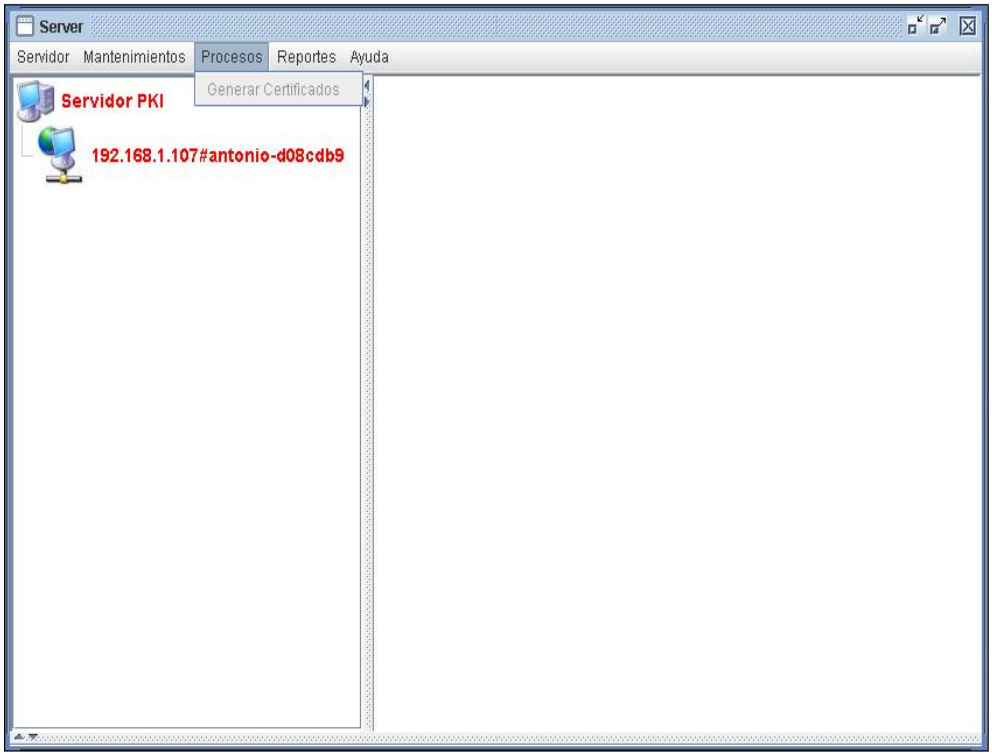
En la zona demarcada por el círculo se observa, donde se deben ingresar correctamente los datos solicitados ahí como: Nombre, apellido y otros.

Al momento de estar seguro de la información ingresada se procede a guardar la misma haciendo clic como se indica en la figura siguiente:

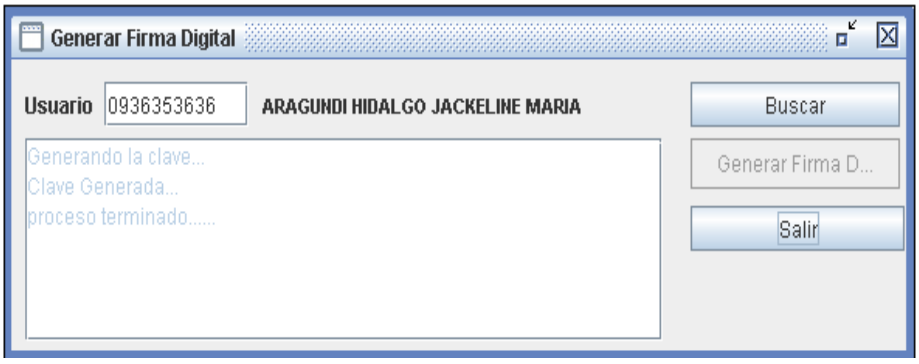


Generación de la Firma Digital

En el menú procesos se hace clic en la opcion que indica generar firma

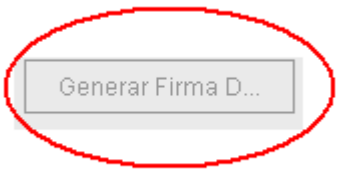


Un usuario al que previamente no se le han ingresado los datos personales no se lograra generar el archivo cifrado



Una vez que es ingresado el usuario a este se le asigna automáticamente una clave luego de haber llenado los campos

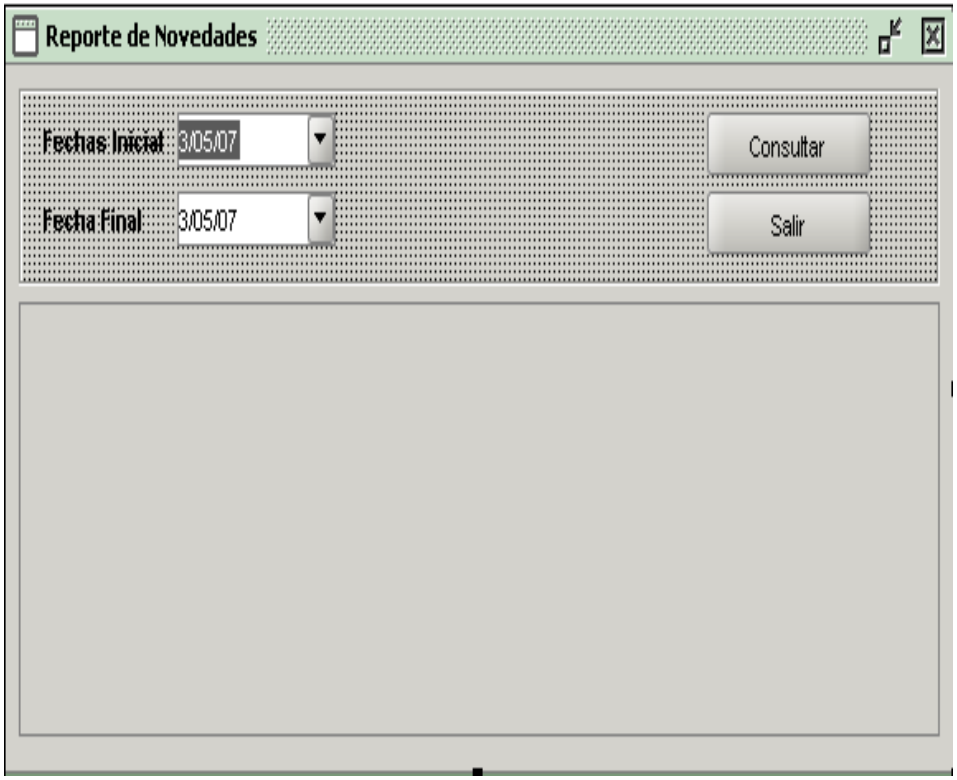
Una vez ingresados estos campos se pulsa clic de la siguiente forma:



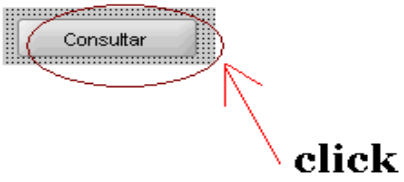
Luego aparecerán automáticamente un mensaje indicando que ya se ha realizado la encriptación del archivo.

Reporte de Actividades

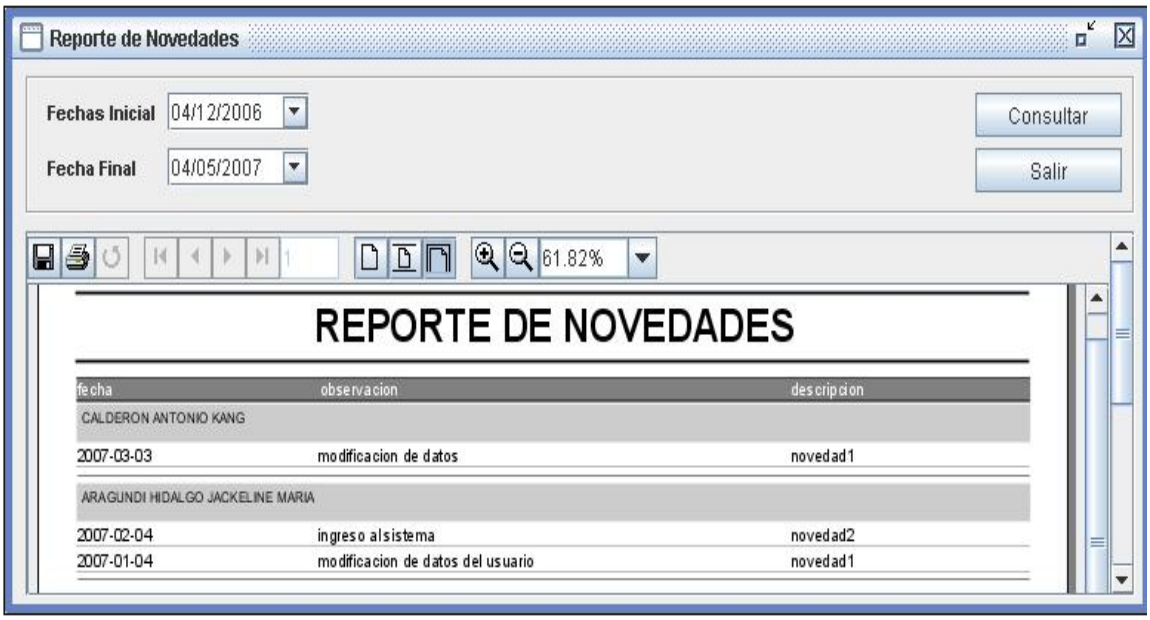
Los reportes de actividades se realizan por medio de rango de fechas, las cuales estan debidamente validadas.



Una vez que se ha ingresado el rango de fechas se hace click en el boton consultar y se genera automáticamente el reporte.



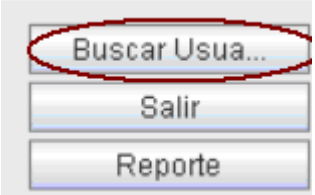
Se genera el reporte de novedades con todos los eventos acontecidos



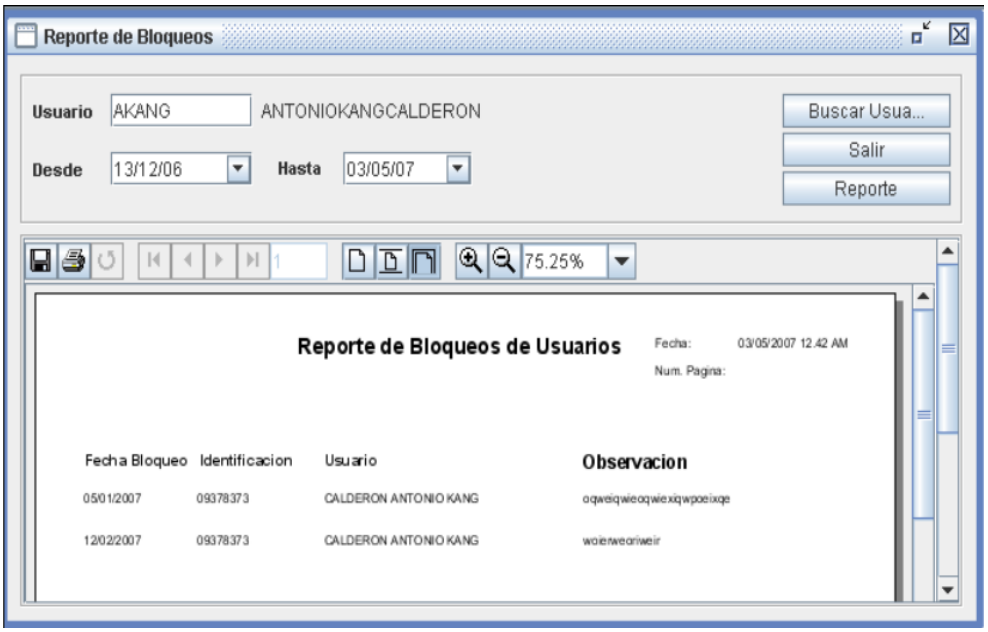
Reporte de Bloqueos

Para lograr generar el reporte de bloqueos primero se busca al usuario. Se ingresa el usuario, el rango de fechas por los cuales se quiere generar el reporte.

Busqueda del usuario



Se genera el reporte mostrando los bloqueos ocurrido en ese lapso de tiempo en el usuario seleccionado.

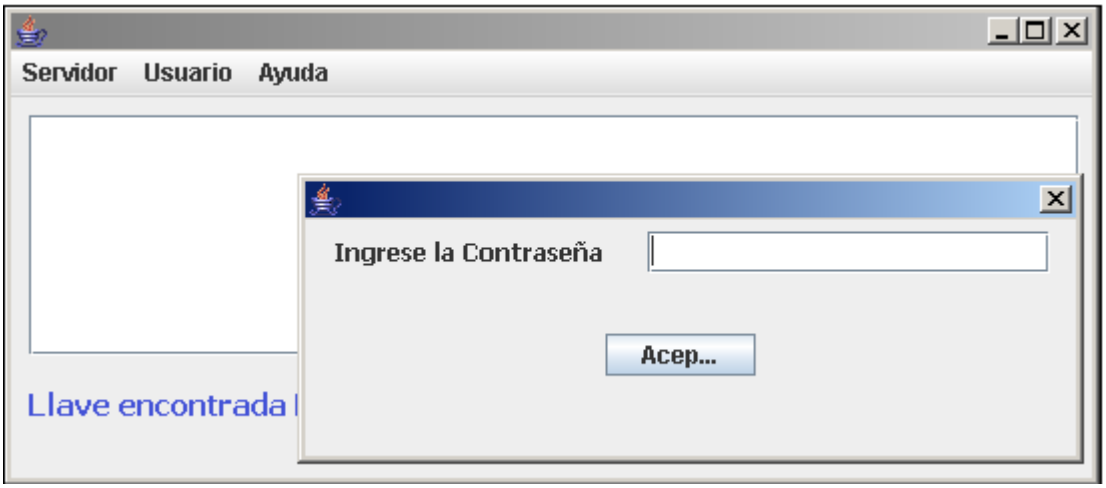

Reporte de Bloqueos
Usuario
AKANG
ANTONIOKANGCALDERON
Desde
13/12/06
Hasta
03/05/07
Buscar Usua...
Salir
Reporte
75.25%
Reporte de Bloqueos de Usuarios
Fecha:
03/05/2007 12:42 AM
Num. Pagina:
Fecha Bloqueo
Identificacion
Usuario
Observacion
05/01/2007
09378373
CALDERON ANTONIO KANG
oqweiqwieoqwiexiqwpoeixqe
12/02/2007
09378373
CALDERON ANTONIO KANG
woierweoriweir

Cliente:

El cliente debera tener instalado en su pc:

- Máquina Virtual de Java
- Cliente
- Librería para lectura del usb.

La pantalla a continuación es la qe se mostrara al cliente luego de que e dispositivo USB ha sido verificado por la aplicación.



.