ANEXO B

CÓDIGOS FUENTE DE LA APLICACIÓN CLIENTE DESARROLLADA PARA LOS CENTROS AMBULATORIOS

Clase CMA.java

```
/**
 * Esta clase es la principal y ejecutable; contiene los métodos a
 * llamar desde el exterior
 * @autora Mª José Aguilar Porro
 * @version 1.0
 */

package cma;

import javax.swing.*;
import cma.menus.MenuPpal;
import cma.paneles.PanelPpal;
import cma.nucleo.FuncionesGUI;
import cma.escritores.escritorError;

public class CMA
    extends javax.swing.JFrame {
  /** Cuerpo de la aplicación gráfica */
  public PanelPpal PanelPrincipal = null;

  /** menús de la aplicación gráfica */
  public MenuPpal MenuPrincipal = null;

  /**
   *  MAIN PRINCIPAL.
   */

  public static void main(String args[]) {
    try {
      CMA _cma = new CMA(args); // creamos una instancia
      if (_cma.MenuPrincipal == null) { // si no se lanzó la GUI
        new FuncionesGUI().informar("Saliendo...\n");
        System.exit(0);
      }
    }
    catch (Throwable e) {
      new FuncionesGUI().informar(
          "main(): " + new escritorError().escribirError(e));
    }

  }

  /**
   * Constructor sin parámetros. Ajusta su referencia en funcionesGUI.
   */

  private CMA() {
    try {
      new FuncionesGUI().fijarFrameRaiz(this); // así la encontraremos
    }                                          // si lo necesitamos
    catch (Throwable e) {
      new FuncionesGUI().informar(
          "CMA(): " + new escritorError().escribirError(e));
    }
  }

  /**
   * Constructor con argumentos, es quien lanza la interfaz de usuario
   */
```

```
public CMA(String[] args) {
  this();
  try {
    if (args != null && args.length != 0) {
      System.out.println("saliendo...");
    }
    else {
      this.interfazGrafica(); // lanzamos la interfaz de usuario
    }
  }
  catch (Throwable e) {
    new FuncionesGUI().informar(
        "CMA(String[]): " + new escritorError().escribirError(e));
  }
}

/**
 *  Este método es el que inicializa toda la interfaz gráfica
 */

private void interfazGrafica() {
  try {
    //título de la ventana
    setTitle("Formulario de consulta para CmA en acto único");

    // gestión del cierre de la ventana por la "x"

    addWindowListener(new java.awt.event.WindowAdapter() {
      public void WindowClosing(java.awt.event.WindowEvent evt) {
        System.exit(0);
      }
    });

    // fijamos la apariencia como la del sistema donde se ejecuta
    try {
      UIManager.setLookAndFeel(UIManager.
        getSystemLookAndFeelClassName());
      SwingUtilities.updateComponentTreeUI(this);
    }
          catch (Throwable e) {
      new FuncionesGUI().informar(new
         escritorError().escribirError(e));
      System.exit(0);
    }
    // la barra de menús
    MenuPrincipal = new MenuPpal();
    setJMenuBar(MenuPrincipal);
    // el contenido
    PanelPrincipal = new PanelPpal();

    getContentPane().add(PanelPrincipal);

    //ajustamos sitio y tamaño de los componentes
    pack();
    setResizable(false);
    //mostramos
    show();
  }
  catch (Throwable e) {
    System.exit(0);
  }}}
```

Clase paneles.PanelPpal.java

```
/**
 *    Esta clase define el cuerpo de la aplicación gráfica
 */

package cma.paneles;

import javax.swing.*;
import java .awt.*;

public class PanelPpal
    extends JPanel {

// Ponemos los subpaneles como variables públicas para su acceso
// externo

  /** panel de datos del paciente */
  public PanelDatosPersonales _PanelDatosPersonales;

  /** panel de datos médicos */
  public PanelDatosClinicos _PanelDatosClinicos;

  /** panel para patologías intercurrentes */
  public PanelPatologias _PanelPatologias;

  /** panel para incluir la fotografía*/
  public PanelFotografia _PanelFotografia;

  /** panel para el médico solicitante*/
  public PanelOtrosDatos _PanelOtrosDatos;

  /** panel para albergar algunos paneles mediante pestañas*/
  public JTabbedPane panelTabbed;

  /**
   * Constructor
   */

  public PanelPpal() {
    // definimos cada uno de los subpaneles

    _PanelDatosPersonales = new PanelDatosPersonales();

    _PanelDatosClinicos = new PanelDatosClinicos();

    _PanelPatologias = new PanelPatologias();

    _PanelFotografia = new PanelFotografia();

    _PanelOtrosDatos = new PanelOtrosDatos();

    // ubicamos cada componente en su sitio

    setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

    // aquí añadiremos el campo de datos del paciente
    JPanel panel_intermedio1 = new JPanel();
    panel_intermedio1.add(_PanelDatosPersonales);
    add(panel_intermedio1);
```

```
    panelTabbed = new JTabbedPane();

    //aquí irán los datos médicos, la foto y los datos del médico que
envía el formulario
    panelTabbed.addTab("Otros Datos", _PanelOtrosDatos);
    panelTabbed.addTab("Datos clínicos", _PanelDatosClinicos);
    panelTabbed.addTab("Patologías intercurrentes", _PanelPatologias);
    panelTabbed.addTab("Fotografía de la lesión", _PanelFotografia);

    panelTabbed.setEnabledAt(0, true);
    panelTabbed.setEnabledAt(1, true);
    panelTabbed.setEnabledAt(2, true);
    panelTabbed.setEnabledAt(3, true);
    panelTabbed.setSelectedIndex(0);

    add(panelTabbed);
    //repaint();

  }
}
```

Clase paneles.PanelDatosPersonales.java

```
/**
 * Panel de datos personales del paciente
 */

package cma.paneles;

import javax.swing.*;
import java.awt.*;

public class PanelDatosPersonales
    extends JPanel {

// Ponemos los componentes como vbles públicas para su acceso exterior
// Como todo son campos de texto, agrupamos campo + etiqueta

  /** Nombre del paciente */
  public static JTextField nombreCampo;
  public JLabel nombre;

  /** Primer apellido */
  public static JTextField apell1Campo;
  public JLabel apell1;

  /** Segundo apellido */
  public static JTextField apell2Campo;
  public JLabel apell2;

  /** Dirección */
  public static JTextField domicilioCampo;
  public JLabel domicilio;

  /** Localidad */
  public static JTextField localidadCampo;
  public JLabel localidad;

  /** Número de teléfono */
  public static JTextField tfnoCampo;
  public JLabel numTfno;
```

```
/** Fecha de nacimiento */
public static JTextField fecNac1Campo;
public static JTextField fecNac2Campo;
public static JTextField fecNac3Campo;
public JLabel fecNac;

/** Número de afiliación a la Seguridad Social */
public static JTextField afiliSSCampo;
public JLabel afiliSSocial;

/** Número de tarjeta sanitaria */
public static JTextField taSSCampo;
public JLabel taSSocial;

/** CheckboxGroup para ver si es la primera vez que acude a esta
 * consulta */
public CheckboxGroup vecesCons;
public Checkbox si;
public Checkbox no;
public JLabel veces;

/**
 *  Constructor
 */
public PanelDatosPersonales() {

  nombreCampo = new JTextField("");
  nombreCampo.setEnabled(true);
  nombreCampo.setToolTipText("Inserte el nombre del paciente");

  nombre = new JLabel("Nombre");
  nombre.setToolTipText("Inserte el nombre del paciente");
  nombre.setLabelFor(nombreCampo);

  apell1Campo = new JTextField("");
  apell1Campo.setEnabled(true);
  apell1Campo.setToolTipText("Inserte el primer apellido");

  apell1 = new JLabel("Primer Apellido");
  apell1.setToolTipText("Inserte el primer apellido");
  apell1.setLabelFor(apell1Campo);

  apell2Campo = new JTextField("");
  apell2Campo.setEnabled(true);
  apell2Campo.setToolTipText("Inserte el segundo apellido");

  apell2 = new JLabel("Segundo Apellido");
  apell2.setToolTipText("Inserte el segundo apellido");
  apell2.setLabelFor(apell2Campo);

  domicilioCampo = new JTextField("");
  domicilioCampo.setEnabled(true);
  domicilioCampo.setToolTipText("Inserte el domicilio");

  domicilio = new JLabel("Domicilio");
  domicilio.setToolTipText("Inserte el domicilio");
  domicilio.setLabelFor(domicilioCampo);

  localidadCampo = new JTextField("");
  localidadCampo.setEnabled(true);
```

```
localidadCampo.setToolTipText("Inserte la localidad");

localidad = new JLabel("Localidad");
localidad.setToolTipText("Inserte la localidad");
localidad.setLabelFor(localidadCampo);

tfnoCampo = new JTextField(9);
tfnoCampo.setEnabled(true);
tfnoCampo.setToolTipText("Inserte el teléfono");

numTfno = new JLabel("Teléfono");
numTfno.setToolTipText("Inserte el teléfono");
numTfno.setLabelFor(tfnoCampo);

fecNac1Campo = new JTextField(2);
fecNac1Campo.setEnabled(true);
fecNac1Campo.setToolTipText(
    "Insértela de la siguiente forma: p.ej: 7 de mayo de 1977:
     07/05/1977");

fecNac2Campo = new JTextField(2);
fecNac2Campo.setEnabled(true);
fecNac2Campo.setToolTipText(
    "Insértela de la siguiente forma: p.ej: 7 de mayo de 1977:
     07/05/1977");

fecNac3Campo = new JTextField(4);
fecNac3Campo.setEnabled(true);
fecNac3Campo.setToolTipText(
    "Insértela de la siguiente forma: p.ej: 7 de mayo de 1977:
     07/05/1977");

fecNac = new JLabel("Fecha de Nacimiento");
fecNac.setToolTipText(
    "Insértela de la siguiente forma: p.ej: 7 de mayo de 1977:
      07051977");
fecNac.setLabelFor(fecNac1Campo);

afiliSSCampo = new JTextField(12);
afiliSSCampo.setEnabled(true);
afiliSSCampo.setToolTipText("Inserte el número sin barras ni
     espacios");

afiliSSocial = new JLabel("N° de la S. Social");
afiliSSocial.setToolTipText("Inserte el número sin barras ni
     espacios");
afiliSSocial.setLabelFor(afiliSSCampo);

taSSCampo = new JTextField(12);
taSSCampo.setEnabled(true);
taSSCampo.setToolTipText(
    "N° sin barras ni espacios, si el paciente no tiene, inserte
     el n° de la SS");

taSSocial = new JLabel("N° de Tarjeta Sanitaria");
taSSocial.setToolTipText(
    "N° sin barras ni espacios, si el paciente no tiene, inserte
     el n° de la SS");
taSSocial.setLabelFor(taSSCampo);

vecesCons = new CheckboxGroup();
```

```
veces = new JLabel(
    "¿Es la primera vez que acude a esta consulta de hospital?");
si = new Checkbox("Sí", false, vecesCons);
no = new Checkbox("No", false, vecesCons);

// A continuación, colocamos los componentes en la interfaz

JPanel panel_intermedio = new JPanel();
panel_intermedio.setLayout(new BoxLayout(panel_intermedio,
    BoxLayout.Y_AXIS));

JPanel subpanel_1 = new JPanel();
subpanel_1.setLayout(new BoxLayout(subpanel_1,
    BoxLayout.X_AXIS));
subpanel_1.add(nombre);
subpanel_1.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_1.add(nombreCampo);
subpanel_1.add(Box.createRigidArea(new Dimension(10, 0)));
subpanel_1.add(apell1);
subpanel_1.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_1.add(apell1Campo);
subpanel_1.add(Box.createRigidArea(new Dimension(10, 0)));
subpanel_1.add(apell2);
subpanel_1.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_1.add(apell2Campo);
panel_intermedio.add(subpanel_1);

JPanel subpanel_2 = new JPanel();
subpanel_2.setLayout(new BoxLayout(subpanel_2,
    BoxLayout.X_AXIS));
subpanel_2.add(domicilio);
subpanel_2.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_2.add(domicilioCampo);
subpanel_2.add(Box.createRigidArea(new Dimension(10, 0)));
subpanel_2.add(localidad);
subpanel_2.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_2.add(localidadCampo);
panel_intermedio.add(subpanel_2);

JPanel subpanel_3 = new JPanel();
subpanel_3.setLayout(new BoxLayout(subpanel_3,
    BoxLayout.X_AXIS));
subpanel_3.add(numTfno);
subpanel_3.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_3.add(tfnoCampo);
subpanel_3.add(Box.createRigidArea(new Dimension(10, 0)));
subpanel_3.add(afiliSSocial);
subpanel_3.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_3.add(afiliSSCampo);
subpanel_3.add(Box.createRigidArea(new Dimension(10, 0)));
subpanel_3.add(taSSocial);
subpanel_3.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_3.add(taSSCampo);

panel_intermedio.add(subpanel_3);

JPanel subpanel_5 = new JPanel();
subpanel_5.add(fecNac);
subpanel_5.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_5.add(fecNac1Campo);
JLabel etiq1 = new JLabel("/");
```

```
    subpanel_5.add(etiq1);
    subpanel_5.add(fecNac2Campo);
    JLabel etiq2 = new JLabel("/");
    subpanel_5.add(etiq2);
    subpanel_5.add(fecNac3Campo);

    subpanel_5.add(veces);
    subpanel_5.add(Box.createRigidArea(new Dimension(10, 0)));
    subpanel_5.add(si);
    subpanel_5.add(Box.createRigidArea(new Dimension(10, 0)));
    subpanel_5.add(no);
    panel_intermedio.add(subpanel_5);

    panel_intermedio.setPreferredSize(new Dimension(750, 125));
    panel_intermedio.setBorder(BorderFactory.createTitledBorder(
        "Datos personales del paciente"));
    add(panel_intermedio);

  }

}
```

Clase paneles.PanelDatosClinicos.java

```
package cma.paneles;

/**
 *   Panel para los datos clínicos del paciente
 */

import javax.swing.*;
import java.awt.*;

public class PanelDatosClinicos
    extends JPanel {
  // ponemos los componentes como variables públicas para su acceso
  // exterior

  /** área de texto para resumen de la enfermedad */
  public static JTextArea resumenEnfermedad;

  /** para escrollar el área de texto */
  public JScrollPane resEnfScrollPane;

  /**área de texto para resumen de la exploración */
  public static JTextArea resumenExploracion;
  public JScrollPane resExpScrollPane;

  /** área para mostrar exploraciones complementarias */
  public static JTextArea exploracionComplem;
  public JScrollPane exploCompScrollPane;

  /** área para diagnóstico provisional */
  public static JTextArea diagnosticoProv;
  public JScrollPane diagProvScrollPane;

  /** aquí irían tratamientos anteriores */
  public static JTextArea tratamientoAnterior;
  public JScrollPane tratamAntScrollPane;

  /** aquí van los fundamentos clínicos de la petición */
```

```
public static JTextArea fundamentoClinico;
public JScrollPane fundClinScrollPane;

/**
 *  Constructor
 */
public PanelDatosClinicos() {
  resumenEnfermedad = new JTextArea();
  resumenEnfermedad.setToolTipText("Escriba aquí un resumen de la
    enfermedad");
  resumenEnfermedad.setEnabled(true);
  // dejamos editar
  resumenEnfermedad.setEditable(true);
  // configuramos que salte de línea si no cabe, sin cortar palabras
  resumenEnfermedad.setLineWrap(true);
  resumenEnfermedad.setWrapStyleWord(true);
  resumenEnfermedad.setBorder(BorderFactory.createLineBorder(
      Color.gray));
  // incrustamos en un panel de scroll
  resEnfScrollPane = new JScrollPane(resumenEnfermedad);
  // configuramos barra de desplazamiento vertical
  resEnfScrollPane.setVerticalScrollBarPolicy(
      JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

  resumenExploracion = new JTextArea();
  resumenEnfermedad.setToolTipText(
      "Escriba aquí un resumen de la exploración clínica");
  resumenExploracion.setEnabled(true);
  resumenExploracion.setEditable(true);
  resumenExploracion.setLineWrap(true);
  resumenExploracion.setWrapStyleWord(true);
  resumenExploracion.setBorder(BorderFactory.createLineBorder(
      Color.gray));
  resExpScrollPane = new JScrollPane(resumenExploracion);
  resExpScrollPane.setVerticalScrollBarPolicy(
      JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

  exploracionComplem = new JTextArea();
  exploracionComplem.setToolTipText(
      "Escriba aquí exploraciones complementarias realizadas (si se
       han hecho)");
  exploracionComplem.setEnabled(true);
  exploracionComplem.setEditable(true);
  exploracionComplem.setLineWrap(true);
  exploracionComplem.setWrapStyleWord(true);
  exploracionComplem.setBorder(BorderFactory.createLineBorder(
      Color.gray));
      exploCompScrollPane = new JScrollPane(exploracionComplem);
  exploCompScrollPane.setVerticalScrollBarPolicy(
      JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

  diagnosticoProv = new javax.swing.JTextArea();
  diagnosticoProv.setToolTipText("Escriba aquí el diagnóstico
      provisional");
  diagnosticoProv.setEnabled(true);
  diagnosticoProv.setEditable(true);
  diagnosticoProv.setLineWrap(true);
  diagnosticoProv.setWrapStyleWord(true);
  diagnosticoProv.setBorder(BorderFactory.createLineBorder(
      Color.gray));
  diagProvScrollPane = new JScrollPane(diagnosticoProv);
```

```
    diagProvScrollPane.setVerticalScrollBarPolicy(
        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

    tratamientoAnterior = new JTextArea();
    tratamientoAnterior.setToolTipText("Escriba aquí tratamientos
        anteriores");
    tratamientoAnterior.setEnabled(true);
    tratamientoAnterior.setEditable(true);
    tratamientoAnterior.setLineWrap(true);
    tratamientoAnterior.setWrapStyleWord(true);
    tratamientoAnterior.setBorder(BorderFactory.createLineBorder(
        Color.gray));
    tratamAntScrollPane = new JScrollPane(tratamientoAnterior);
    tratamAntScrollPane.setVerticalScrollBarPolicy(
        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

    fundamentoClinico = new javax.swing.JTextArea();
    fundamentoClinico.setToolTipText(
        "Escriba aquí el fundamento clínico de la petición");
    fundamentoClinico.setEnabled(true);
    fundamentoClinico.setEditable(true);
    fundamentoClinico.setLineWrap(true);
    fundamentoClinico.setWrapStyleWord(true);
    fundamentoClinico.setBorder(BorderFactory.createLineBorder(
        Color.gray));
    fundClinScrollPane = new JScrollPane(fundamentoClinico);
    fundClinScrollPane.setVerticalScrollBarPolicy(
        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

    // finalmente, ponemos cada componente en su sitio

    setLayout(new GridLayout(3, 2));
    resEnfScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Resumen de la enfermedad"));
    add(resEnfScrollPane);
    resExpScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Resumen de la exploración clínica"));
    add(resExpScrollPane);
    exploCompScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Exploraciones complementarias realizadas"));
    add(exploCompScrollPane);
    diagProvScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Diagnóstico provisional"));
    add(diagProvScrollPane);
    tratamAntScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Tratamiento anterior/actual"));
    add(tratamAntScrollPane);
    fundClinScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Fundamento Clínico de la petición"));
    add(fundClinScrollPane);

  }
}
```

Clase paneles.PanelOtrosDatos.java

```
package cma.paneles;

/**
* Panel para Otros Datos del Paciente
*/
```

```
import javax.swing.*;
import java.awt.*;

public class PanelOtrosDatos
    extends JPanel {

  /** campo para el nombre del médico + etiqueta correspondiente */
  public static JTextField doctorCampo;
  public JLabel doctor;

  /** campo para la especialidad + etiqueta */
  public static JTextField especialidadCampo;
  public JLabel especialidad;

  /** campo para el código del médico + etiqueta correspondiente */
  public static JTextField codigoCampo;
  public JLabel codigo;

  /** campo para el ambulatorio + etiqueta */
  public static JTextField ambulatorioCampo;
  public JLabel ambulatorio;

  /** campo para la dirección + etiqueta correspondiente */
  public static JTextField direccionCampo;
  public JLabel direccion;

  /** campo para la localidad + etiqueta */
  public static JTextField localidadCampo;
  public JLabel localidad;

  public CheckboxGroup hospitales;
  public Checkbox universitario;
  public Checkbox valme;
  public Checkbox infantil;
  public Checkbox centroDiagn;
  public Checkbox general;
  public Checkbox maternal;
  public Checkbox traumatolog;
  public static JLabel hospitalElegido;

  // Esta última etiqueta es auxiliar

  // A continuación, campos y casillas para la forma de aviso

  public ButtonGroup formaAviso;
  public JRadioButton porEscrito;
  public JRadioButton porPersonal;
  public JRadioButton porTfno;
  public JTextField porTfnoCampo;
  public JLabel alTfno;
  public JTextField horaCampo;
  public JLabel hora;

  OyenteEventoHospitales oyente = new OyenteEventoHospitales();

  /**
   *  Constructor
   */

  public PanelOtrosDatos() {
```

```
// Campo de texto para introducir el nombre del médico
doctorCampo = new JTextField("");
doctorCampo.setEnabled(true);
doctorCampo.setToolTipText("Inserte nombre del médico");
doctor = new JLabel("Doctor/a: ");
doctor.setToolTipText("Inserte nombre del médico");
doctor.setLabelFor(doctorCampo);

// Campo de texto para introducir la especialidad
especialidadCampo = new JTextField("");
especialidadCampo.setEnabled(true);
especialidadCampo.setToolTipText("Inserte especialidad");
especialidad = new JLabel("Especialidad: ");
especialidad.setToolTipText("Inserte especialidad");
especialidad.setLabelFor(especialidadCampo);

// Campo de texto para introducir el código del médico
codigoCampo = new JTextField("", 12);
codigoCampo.setEnabled(true);
codigoCampo.setToolTipText("Inserte Número Oficial de Colegiado o
  Código Numérico Personal sin barras ni espacios");
codigo = new JLabel("Nº de Colegiado/Cód. Numérico Personal: ");
codigo.setToolTipText("Inserte Número Oficial de Colegiado o
  Código Numérico Personal sin barras ni espacios");
codigo.setLabelFor(codigoCampo);

// Campo de texto para introducir el ambulatorio
ambulatorioCampo = new javax.swing.JTextField("");
ambulatorioCampo.setEnabled(true);
ambulatorioCampo.setToolTipText("Inserte ambulatorio");
ambulatorio = new JLabel("Ambulatorio: ");
ambulatorio.setToolTipText("Inserte ambulatorio");
ambulatorio.setLabelFor(ambulatorioCampo);

// Campo de texto para introducir la dirección
direccionCampo = new JTextField("");
direccionCampo.setEnabled(true);
direccionCampo.setToolTipText("Inserte dirección");
direccion = new JLabel("Dirección: ");
direccion.setToolTipText("Inserte dirección");
direccion.setLabelFor(doctorCampo);

// Campo de texto para introducir la localidad
localidadCampo = new JTextField("");
localidadCampo.setEnabled(true);
localidadCampo.setToolTipText("Inserte localidad");
localidad = new JLabel("Localidad: ");
localidad.setToolTipText("Inserte especialidad");
localidad.setLabelFor(especialidadCampo);

hospitales = new CheckboxGroup();
universitario = new Checkbox("Hospital Universitario", false,
                                    hospitales);
universitario.addItemListener(oyente);
valme = new Checkbox("Hospital de Valme", false,
  hospitales);
valme.addItemListener(oyente);
infantil = new Checkbox("Hospital Infantil", false, hospitales);
infantil.addItemListener(oyente);

centroDiagn = new Checkbox("Centro Diagnóstico", false,
```

```
  hospitales);
centroDiagn.addItemListener(oyente);
general = new Checkbox("Hospital General", false, hospitales);
general.addItemListener(oyente);
maternal = new Checkbox("Hospital Maternal", false, hospitales);
maternal.addItemListener(oyente);
traumatolog = new Checkbox("CR Traumatología", false, hospitales);
traumatolog.addItemListener(oyente);

// Campo de texto para introducir el número de teléfono
porTfnoCampo = new JTextField("");
porTfnoCampo.setEnabled(false);
porTfnoCampo.setEditable(false);
porTfnoCampo.setToolTipText(
    "Inserte aquí el teléfono al que desea ser llamado");
// etiqueta para el teléfono
alTfno = new JLabel("Al teléfono: ");
alTfno.setToolTipText("Inserte aquí el teléfono al que desea ser
     llamado");
alTfno.setLabelFor(porTfnoCampo);
// Campo de texto para introducir la hora de llamada
horaCampo = new JTextField("");
horaCampo.setEditable(false);
horaCampo.setToolTipText("Inserte aquí la hora que prefiere el
     paciente");
// etiqueta para la hora de llamada
hora = new JLabel("Preferentemente a las: ");
hora.setToolTipText("Inserte aquí la hora que prefiere el
     paciente");
hora.setLabelFor(horaCampo);
porEscrito = new JRadioButton("Por escrito", false);
porEscrito.setEnabled(true);
porEscrito.addItemListener(new event.ItemListener(){
      public void itemStateChanged(event.ItemEvent e){
        if(porEscrito.getState() == true){
          porTfnoCampo.setEditable(false);
          porTfnoCampo.setEnabled(false);
          horaCampo.setEditable(false);
          horaCampo.setEnabled(false);
        }
      }
      });

porPersonal = new JRadioButton("Personal del Hospital", false);
porPersonal.setEnabled(true);
porPersonal.addItemListener(new event.ItemListener(){
    public void itemStateChanged(event.ItemEvent e){
      if(porEscrito.getState() == true){
        porTfnoCampo.setEditable(false);
        porTfnoCampo.setEnabled(false);
        horaCampo.setEditable(false);
        horaCampo.setEnabled(false);
      }
    }
    });
porTfno = new JRadioButton("Por Teléfono", false);
porTfno.setEnabled(true);
porTfno.addActionListener(new event.ActionListener() {
  public void actionPerformed(event.ActionEvent e) {
    if (porTfno.isSelected() == true) {
      porTfnoCampo.setEditable(true);
```

```
          porTfnoCampo.setEnabled(true);
          horaCampo.setEditable(true);
          horaCampo.setEnabled(true);
        }
        else {
          porTfnoCampo.setEnabled(false);
          horaCampo.setEnabled(false);
        }
      }
    });
    porTfno.addItemListener(new event.ItemListener(){
      public void itemStateChanged(event.ItemEvent e){
        if (porTfno.isSelected() == true){
          porTfnoCampo.setEnabled(true);
          horaCampo.setEnabled(true);
        }
        else{
          porTfnoCampo.setEnabled(false);
          horaCampo.setEnabled(false);
        }
      }
         }
         );
    formaAviso = new ButtonGroup();
    formaAviso.add(porEscrito);
    formaAviso.add(porPersonal);
    formaAviso.add(porTfno);

    // finalmente, ponemos cada componente en su sitio

    setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
    JPanel panel_intermedio1 = new JPanel();
    panel_intermedio1.setLayout(new GridLayout(1, 2));
    JPanel panel_intermedio11 = new JPanel();
    panel_intermedio11.setLayout(new BoxLayout(panel_intermedio11,
        BoxLayout.Y_AXIS));
    JPanel subpanel_1 = new JPanel();
    subpanel_1.setLayout(new BoxLayout(subpanel_1,
        BoxLayout.X_AXIS));
    subpanel_1.add(doctor);
    subpanel_1.add(Box.createRigidArea(new Dimension(5, 0)));
    subpanel_1.add(doctorCampo);
    panel_intermedio11.add(subpanel_1);
    JPanel subpanel_2 = new JPanel();
    subpanel_2.setLayout(new BoxLayout(subpanel_2,
        BoxLayout.X_AXIS));
    subpanel_2.add(especialidad);
    subpanel_2.add(Box.createRigidArea(new Dimension(5, 0)));
    subpanel_2.add(especialidadCampo);
    panel_intermedio11.add(subpanel_2);
    JPanel subpanel_3 = new JPanel();
    subpanel_3.setLayout(new BoxLayout(subpanel_3,
        BoxLayout.X_AXIS));
    subpanel_3.add(codigo);
    subpanel_3.add(Box.createRigidArea(new Dimension(5, 0)));
    subpanel_3.add(codigoCampo);
    panel_intermedio11.add(subpanel_3);
    JPanel subpanel_4 = new JPanel();
    subpanel_4.setLayout(new BoxLayout(subpanel_4,
        BoxLayout.X_AXIS));
    subpanel_4.add(ambulatorio);
```

```
subpanel_4.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_4.add(ambulatorioCampo);
panel_intermedio11.add(subpanel_4);
JPanel subpanel_5 = new JPanel();
subpanel_5.setLayout(new BoxLayout(subpanel_5,
    BoxLayout.X_AXIS));
subpanel_5.add(direccion);
subpanel_5.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_5.add(direccionCampo);
panel_intermedio11.add(subpanel_5);
JPanel subpanel_6 = new JPanel();
subpanel_6.setLayout(new BoxLayout(subpanel_6,
    BoxLayout.X_AXIS));
subpanel_6.add(localidad);
subpanel_6.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel_6.add(localidadCampo);
panel_intermedio11.add(subpanel_6);

panel_intermedio11.setBorder(
    BorderFactory.createTitledBorder("Médico solicitante"));
panel_intermedio1.add(panel_intermedio11);

JPanel panel_intermedio12 = new JPanel();
panel_intermedio12.setLayout(new GridLayout(4, 2));
panel_intermedio12.add(universitario);
panel_intermedio12.add(valme);
panel_intermedio12.add(infantil);
panel_intermedio12.add(centroDiagn);
panel_intermedio12.add(general);
panel_intermedio12.add(maternal);
panel_intermedio12.add(traumatolog);
panel_intermedio12.setBorder(
    BorderFactory.createTitledBorder("Se envía a"));
panel_intermedio12.setToolTipText(
    "Seleccione aquí el hospital de destino de la consulta");
panel_intermedio1.add(panel_intermedio12);
add(panel_intermedio1);

JPanel panel_intermedio2 = new JPanel();
panel_intermedio2.setLayout(new BoxLayout(panel_intermedio2,
    BoxLayout.Y_AXIS));

JPanel subpanel21 = new JPanel();
subpanel21.setLayout(new GridLayout(1, 3));
subpanel21.add(porEscrito);
subpanel21.add(porPersonal);
subpanel21.add(porTfno);
panel_intermedio2.add(subpanel21);

JPanel subpanel22 = new JPanel();
subpanel22.setLayout(new BoxLayout(subpanel22,
    BoxLayout.X_AXIS));
subpanel22.add(alTfno);
subpanel22.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel22.add(porTfnoCampo);
subpanel22.add(Box.createRigidArea(new Dimension(30, 0)));
subpanel22.add(hora);
subpanel22.add(Box.createRigidArea(new Dimension(5, 0)));
subpanel22.add(horaCampo);
panel_intermedio2.add(subpanel22);
```

```
    panel_intermedio2.setToolTipText(
        "Seleccione la forma en que desea ser avisado el paciente");
    panel_intermedio2.setBorder(BorderFactory.createTitledBorder(
        "Desea ser avisado:"));
    add(panel_intermedio2);

    JPanel panel_intermedio3 = new JPanel();
    panel_intermedio3.setPreferredSize(new Dimension(700, 200));
    add(panel_intermedio3);

  }

  class OyenteEventoHospitales
      implements event.ItemListener {
    public void itemStateChanged(event.ItemEvent ev) {
      hospitalElegido = new JLabel("");
      if (ev.getSource() == universitario) {
        hospitalElegido.setText("Hospital Universitario");
      }
      else if (ev.getSource() == valme) {
        hospitalElegido.setText("Hospital de Valme");
      }
      else if (ev.getSource() == infantil) {
        hospitalElegido.setText("Hospital Infantil");
      }
      else if (ev.getSource() == centroDiagn) {
        hospitalElegido.setText("Centro Diagnóstico");
      }
      else if (ev.getSource() == general) {
        hospitalElegido.setText("Hospital General");
      }
      else if (ev.getSource() == maternal) {
        hospitalElegido.setText("Hospital Maternal");
      }
      else if (ev.getSource() == traumatolog) {
        hospitalElegido.setText("CR Traumatología");
      }
    }
  }

}
```

Clase paneles.PanelPatologias.java

```
package cma.paneles;

import java.awt.*;
import javax.swing.*;

/**
 * Panel para patologías intercurrentes
 * @author Mª José Aguilar Porro
 * @version 1.0
 */

public class PanelPatologias
    extends JPanel {

  public static CheckboxGroup diabetico;
  public static Checkbox diabetSi;
  public static Checkbox diabetNo;
```

```
public JLabel etiqDiabet;

public static CheckboxGroup hipertenso;
public static Checkbox hipertSi;
public static Checkbox hipertNo;
public JLabel etiqHipert;

public static CheckboxGroup anticoagulantes;
public static Checkbox anticoagSi;
public static Checkbox anticoagNo;
public JLabel etiqAnticoag;

/** área de texto para intervenciones previas */
public static JTextArea intervPrevias;
public JScrollPane intervPreviasScrollPane;

/** área de texto para alergias a medicamentos */
public static JTextArea alergiasMedic;
public JScrollPane alergiasMedicScrollPane;

/** área de texto para enfermedades anteriores */
public static JTextArea enfAnteriores;
public JScrollPane enfAnterioresScrollPane;

/** área de texto para otros comentarios */
public static JTextArea otros;
public JScrollPane otrosScrollPane;

public PanelPatologias() {

  /**
   * En primer lugar, inicializamos las clases que vamos a usar
   */
  diabetico = new CheckboxGroup();
  diabetSi = new Checkbox("Sí", false, diabetico);
  diabetNo = new Checkbox("No", false, diabetico);
  etiqDiabet = new JLabel("¿Es diabético/a?");
  etiqDiabet.setToolTipText("Marque la opción adecuada");

  hipertenso = new CheckboxGroup();
  hipertSi = new Checkbox("Sí", false, hipertenso);
  hipertNo = new Checkbox("No", false, hipertenso);
  etiqHipert = new JLabel("¿Padece hipertensión?");
  etiqHipert.setToolTipText("Marque la opción adecuada");

  anticoagulantes = new CheckboxGroup();
  anticoagSi = new Checkbox("Sí", false, anticoagulantes);
  anticoagNo = new Checkbox("No", false, anticoagulantes);
  etiqAnticoag = new JLabel("¿Sigue algún tratamiento con
    anticoagulantes?");
  etiqAnticoag.setToolTipText("Marque la opción adecuada");

  intervPrevias = new JTextArea();
  intervPrevias.setToolTipText(
      "Escriba aquí intervenciones previas realizadas al paciente");
  intervPrevias.setEnabled(true);
  // dejamos editar
  intervPrevias.setEditable(true);
  // configuramos que salte de línea si no cabe, sin cortar palabras
  intervPrevias.setLineWrap(true);
  intervPrevias.setWrapStyleWord(true);
```

```
intervPrevias.setBorder(BorderFactory.
  createLineBorder(Color.gray));
// incrustamos en un panel de scroll
intervPreviasScrollPane = new JScrollPane(intervPrevias);
// configuramos barra de desplazamiento vertical
intervPreviasScrollPane.setVerticalScrollBarPolicy(
    JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

alergiasMedic = new JTextArea();
alergiasMedic.setToolTipText(
    "Escriba aquí alergias a medicamentos del paciente");
alergiasMedic.setEnabled(true);
alergiasMedic.setEditable(true);
alergiasMedic.setLineWrap(true);
alergiasMedic.setWrapStyleWord(true);
alergiasMedic.setBorder(BorderFactory.
  createLineBorder(Color.gray));
alergiasMedicScrollPane = new JScrollPane(alergiasMedic);
alergiasMedicScrollPane.setVerticalScrollBarPolicy(
    JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

enfAnteriores = new JTextArea();
enfAnteriores.setToolTipText(
    "Escriba aquí enfermedades anteriores padecidas por el
  paciente");
enfAnteriores.setEnabled(true);
enfAnteriores.setEditable(true);
enfAnteriores.setLineWrap(true);
enfAnteriores.setWrapStyleWord(true);
enfAnteriores.setBorder(BorderFactory.
  createLineBorder(Color.gray));
enfAnterioresScrollPane = new JScrollPane(enfAnteriores);
enfAnterioresScrollPane.setVerticalScrollBarPolicy(JScrollPane.
    VERTICAL_SCROLLBAR_AS_NEEDED);

otros = new JTextArea();
otros.setToolTipText("Escriba aquí otros comentarios que quisiera
  hacer");
otros.setEnabled(true);
otros.setEditable(true);
otros.setLineWrap(true);
otros.setWrapStyleWord(true);
otros.setBorder(BorderFactory.createLineBorder(Color.gray));
otrosScrollPane = new JScrollPane(otros);
otrosScrollPane.setVerticalScrollBarPolicy(
    JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

/**
 * Ahora tendremos que colocarlos en el panel
 */

JPanel panelIntermedio = new JPanel();
panelIntermedio.setLayout(new BoxLayout(panelIntermedio,
  BoxLayout.Y_AXIS));

JPanel subpanel_1 = new JPanel();
subpanel_1.setLayout(new GridLayout(1, 2));

JPanel subpanel_11 = new JPanel();
subpanel_11.setLayout(new BoxLayout(subpanel_11,
  BoxLayout.Y_AXIS));
```

```
    JPanel subpanel3 = new JPanel();
    subpanel3.add(etiqDiabet);
    subpanel3.add(Box.createRigidArea(new Dimension(155, 0)));
    subpanel3.add(diabetNo);
    subpanel3.add(Box.createRigidArea(new Dimension(10, 0)));
    subpanel3.add(diabetSi);
    subpanel_11.add(subpanel3);

    JPanel subpanel4 = new JPanel();
    subpanel4.add(etiqHipert);
    subpanel4.add(Box.createRigidArea(new Dimension(125, 0)));
    subpanel4.add(hipertNo);
    subpanel4.add(Box.createRigidArea(new Dimension(10, 0)));
    subpanel4.add(hipertSi);
    subpanel_11.add(subpanel4);

    JPanel subpanel5 = new JPanel();
    subpanel5.add(etiqAnticoag);
    subpanel5.add(Box.createRigidArea(new Dimension(10, 0)));
    subpanel5.add(anticoagNo);
    subpanel5.add(Box.createRigidArea(new Dimension(10, 0)));
    subpanel5.add(anticoagSi);
    subpanel_11.add(subpanel5);

    subpanel_1.add(subpanel_11);

    JPanel subpanel_12 = new JPanel();

    subpanel_1.add(subpanel_12);

    panelIntermedio.add(subpanel_1);

    JPanel subpanel_2 = new JPanel();
    subpanel_2.setLayout(new GridLayout(2, 2));

    intervPreviasScrollPane.setBorder(BorderFactory.
      createTitledBorder(
        "Intervenciones Previas"));
    subpanel_2.add(intervPreviasScrollPane);

    alergiasMedicScrollPane.setBorder(BorderFactory.
      createTitledBorder(
        "Alergias a Medicamentos"));
    subpanel_2.add(alergiasMedicScrollPane);

    enfAnterioresScrollPane.setBorder(BorderFactory.
      createTitledBorder(
        "Enfermedades Previas"));
    subpanel_2.add(enfAnterioresScrollPane);

    otrosScrollPane.setBorder(BorderFactory.createTitledBorder(
        "Otros comentarios"));
    subpanel_2.add(otrosScrollPane);

    subpanel_2.setPreferredSize(new Dimension(700, 275));
    panelIntermedio.add(subpanel_2);

    add(panelIntermedio);
  }
}
```

Clase paneles.PanelFotografia.java

```
package cma.paneles;

import javax.swing.*;
import java.io.*;
import java.awt.*;
import java.net.*;
import java.lang.reflect.*;
import java.util.*;
import com.archimed.dicom.*;
import com.archimed.dicom.network.*;
import cma.dicoms.*;
import cma.paneles.*;
import cma.escritores.*;

public class PanelFotografia
    extends JPanel {

  public JButton adquirirFoto;
  public JButton compruebaConex;
  public JButton mandaDicom;
  public JButton cancelar;
  public JButton limpiar;
  public static ImageIcon foto;
  public JLabel marcoFoto;
  public JLabel nombreFoto;
  public JOptionPane parent = new JOptionPane();
  OyenteEventoBoton oyente1 = new OyenteEventoBoton();
  OyenteEventoCompConexion oyente3 = new OyenteEventoCompConexion();
  OyenteEventoEnviar oyente4 = new OyenteEventoEnviar();
  OyenteEventoCancelar oyente5 = new OyenteEventoCancelar();
  OyenteEventoLimpiar oyente6 = new OyenteEventoLimpiar();
  public static String nuevaFoto;
  public static ImageIcon fotodic;

//  public static ObjetoDicom dcm;
  public static FileOutputStream file;
  public static com.archimed.dicom.image.DicomImage dcm;
  public boolean verbose = false; // Para manejar las salidas de
                             // información por pantalla

  /**
   *  Constructor
   */

  public PanelFotografia() {

    foto = new ImageIcon("");
    marcoFoto = new JLabel(foto);
    marcoFoto.setPreferredSize(new Dimension(320, 240));
    adquirirFoto = new JButton("Seleccionar imagen");
    adquirirFoto.setMnemonic(event.KeyEvent.VK_S);
    compruebaConex = new JButton("Comprueba conexión con servidor");
    compruebaConex.setMnemonic(event.KeyEvent.VK_O);
    mandaDicom = new JButton("Enviar solicitud");
    mandaDicom.setMnemonic(event.KeyEvent.VK_E);
    cancelar = new JButton("Cancelar");
    cancelar.setMnemonic(event.KeyEvent.VK_C);
    limpiar = new JButton("Limpiar datos");
    limpiar.setMnemonic(event.KeyEvent.VK_L);
```

```
    nombreFoto = new JLabel("");

    setLayout(new GridLayout(1, 2));

    JPanel panel_intermedio1 = new JPanel();
    panel_intermedio1.add(marcoFoto);
    panel_intermedio1.setBorder(BorderFactory.createTitledBorder(
        "Fotografía de la lesión"));
    add(panel_intermedio1);
    JPanel panel_intermedio2 = new JPanel();
    panel_intermedio2.setLayout(new BoxLayout(panel_intermedio2,
        BoxLayout.Y_AXIS));
    JPanel subpanel_21 = new JPanel();
    subpanel_21.add(adquirirFoto);
    JPanel subpanel_22 = new JPanel();
    subpanel_22.add(compruebaConex);
    subpanel_22.add(mandaDicom);
    panel_intermedio2.add(subpanel_21);
    panel_intermedio2.add(subpanel_22);
    JPanel subpanel_23 = new JPanel();
    subpanel_23.add(limpiar);
    subpanel_23.add(cancelar);
    panel_intermedio2.add(subpanel_23);
    panel_intermedio2.add(nombreFoto);
    add(panel_intermedio2);

    adquirirFoto.addActionListener(oyente1);
    compruebaConex.addActionListener(oyente3);
    mandaDicom.addActionListener(oyente4);
    cancelar.addActionListener(oyente5);
    limpiar.addActionListener(oyente6);
  }

  /**
   *  El siguiente método devuelve un objeto de la clase ImageIcon
   *  a partir del path de la imagen seleccionada
   */

  ImageIcon dameFoto(String cadena) {
    ImageIcon f = new ImageIcon(cadena);
    return f;
  }
  /**
   * La siguiente clase implementa el modo en que se selecciona una
   * imagen
   */
  class OyenteEventoBoton
      implements event.ActionListener {
    public void actionPerformed(event.ActionEvent evento) {
      JFileChooser chooser = new JFileChooser();
      int returnVal = chooser.showOpenDialog(parent);
      if (returnVal == JFileChooser.CANCEL_OPTION) {
        parent.getParent();
      }
      if (returnVal == JFileChooser.APPROVE_OPTION) {
        nuevaFoto = chooser.getSelectedFile().getPath();
        System.out.println(nuevaFoto);
        marcoFoto.setIcon(dameFoto(nuevaFoto));
        fotodic = new ImageIcon(dameFoto(nuevaFoto).getImage());

        nombreFoto.setText("Se ha seleccionado la imagen " +
```

```
                            chooser.getSelectedFile().getName());
        System.out.println("Se ha seleccionado la imagen " +
                            chooser.getSelectedFile().getName());

      }

    }
  }

/**
 *  Mediante la siguiente clase se comprueba la conexión con el
 *  servidor
 */
   class OyenteEventoCompConexion
      implements event.ActionListener {
    public void actionPerformed(event.ActionEvent e) {
      CompConexion comprueba = new CompConexion();
    }
  }

 /**
  *  La siguiente clase es para enviar el objeto DICOM una vez
  *  que se pulsa el botón correspondiente
  */

  class OyenteEventoEnviar
      implements event.ActionListener {
    public void actionPerformed(event.ActionEvent e) {
      dcm = new com.archimed.dicom.image.DicomImage();
      boolean f = ObjetoDicom().creaDicom(dcm);
      if (f == true) {
        try {
          boolean g = enviaFich().associate();
          if (g == true) {
            JOptionPane.showMessageDialog(null,
                 "Asociación establecida con el servidor DICOM");
            boolean h = enviaFich().exchangeInfo(dcm);
            if (h == true) {
              JOptionPane.showMessageDialog(null,
                   "El archivo se ha enviado con éxito al servidor");
            }
            else {
              JOptionPane.showMessageDialog(null,
                   "El archivo no pudo ser enviado con éxito al
                   servidor");
            }
          }
          else {
            JOptionPane.showMessageDialog(null,
                  "No se pudo establecer la asociación con el
                  servidor");
          }
          boolean i = enviaFich().release();
          if (i == true) {
            JOptionPane.showMessageDialog(null,
                   "La asociación se ha liberado con éxito");
          }
          else {
            JOptionPane.showMessageDialog(null,
                   "No se pudo liberar la conexión con éxito");
          }
```

```
        }
        catch (Throwable t) {
          System.out.println(new escritorError().escribirError(t));
        }
      }
    }
  }
class OyenteEventoCancelar
      implements event.ActionListener {
    public void actionPerformed(event.ActionEvent e) {
      System.exit(0);
    }
  }

  class OyenteEventoLimpiar
      implements event.ActionListener {
    public void actionPerformed(event.ActionEvent e) {
      /**
       * Lo único que hace este método es limpiar los campos de datos
       *  del formulario
       */
      PanelDatosPersonales.afiliSSCampo.setText("");
      PanelDatosPersonales.apell1Campo.setText("");
      PanelDatosPersonales.apell2Campo.setText("");
      PanelDatosPersonales.domicilioCampo.setText("");
      PanelDatosPersonales.fecNac1Campo.setText("");
      PanelDatosPersonales.fecNac2Campo.setText("");
      PanelDatosPersonales.fecNac3Campo.setText("");
      PanelDatosPersonales.localidadCampo.setText("");
      PanelDatosPersonales.nombreCampo.setText("");
      PanelDatosPersonales.taSSCampo.setText("");
      PanelDatosPersonales.tfnoCampo.setText("");

      PanelDatosClinicos.diagnosticoProv.setText("");
      PanelDatosClinicos.exploracionComplem.setText("");
      PanelDatosClinicos.fundamentoClinico.setText("");
      PanelDatosClinicos.resumenEnfermedad.setText("");
      PanelDatosClinicos.resumenExploracion.setText("");
      PanelDatosClinicos.tratamientoAnterior.setText("");

      PanelOtrosDatos.ambulatorioCampo.setText("");
      PanelOtrosDatos.codigoCampo.setText("");
      PanelOtrosDatos.direccionCampo.setText("");
      PanelOtrosDatos.doctorCampo.setText("");
      PanelOtrosDatos.especialidadCampo.setText("");
      PanelOtrosDatos.localidadCampo.setText("");

    }
  }
}
```

Clase nucleo.FuncionesGUI.java

```
package cma.nucleo;

import cma.CMA;
import cma.escritores.escritorError;

/**
 * Esta clase maneja la comunicación entre el código en sí y el
 * entorno gráfico */
```

```
public class FuncionesGUI {
  /** desde aquí se localizan todos los componentes */
  public static CMA frameRaiz = null;

  /**
   * Asigna un valor a la variable frameRaiz. Se usa al inicializar
   *  @param pframeRaiz: el valor a guardar en frameRaiz.
   */

  public void fijarFrameRaiz(CMA pframeRaiz) {
    try {
      frameRaiz = pframeRaiz;
    }
    catch (Throwable e) {
      System.out.println("fijarFrameRaiz(): "
                         + new escritorError().escribirError(e));
    }
  }

  /**
   * Recuperar frameRaiz, para tener acceso a los componentes gráficos
   * @return la instancia de la clase CMA que actúa como raíz
   */

  public CMA obtenerFrameRaiz() {
    try {
      return frameRaiz;
    }
    catch (Throwable e) {
      System.out.println("obtenerFrameRaiz(): "
                         + new escritorError().escribirError(e));
      return null;
    }
  }

  /**
   * Saca textualmente a la ventana de información o a System.out la
   * cadena dada. Se puede deshabilitar el código si no se quiere
   * información de depuración
   * @param mensaje: un String con lo que se quiera informar.
   */

  public void informar(String mensaje) {
    try {
      if (mensaje == null) {
        return;      }
      // si no hay ventana, escribimos en la consola
      if (frameRaiz == null || !frameRaiz.isVisible()) {
        System.out.print(mensaje);
        return;
      }

    }
    catch (Throwable e) {
      System.out.println("informar(): "
                         + new escritorError().escribirError(e));
    }
  }

}
```

Clase nucleo.FuncionesFichero.java

```
package cma.nucleo;

import cma.escritores.*;

public class FuncionesFichero {

  /**
    * Se llama a este método para salir de la aplicación
    */

   public void salir()
   {
     try{
       new FuncionesGUI().informar("Cerrando la aplicación...\n");
       new FuncionesGUI().obtenerFrameRaiz().PanelPrincipal = null;
       new FuncionesGUI().obtenerFrameRaiz().MenuPrincipal = null;
       new FuncionesGUI().obtenerFrameRaiz().dispose();
       System.exit (0);
     }
     catch (Throwable e){
       new FuncionesGUI().informar("salir(): "
                   + new escritorError().escribirError(e));
     }
   }

}
```

Clase nucleo.FuncionesAyuda.java

```
package cma.nucleo;

import cma.escritores.*;

public class FuncionesAyuda {
  /**
   * Información acerca de esta aplicación
   */

  public void acercaDe() {
    try {
      // mostrar una ventana encima

      javax.swing.JOptionPane.showMessageDialog(
          new FuncionesGUI().obtenerFrameRaiz(),
          new String[] {
        "Herramienta de Telemedicina para clasificación de pacientes
            para CmA en acto único",
            "Por: Mª José Aguilar Porro",
            "Versión 1.0"
      }

      ,
          "Acerca de...",
          javax.swing.JOptionPane.INFORMATION_MESSAGE);
    }
    catch (Throwable e) {
      new FuncionesGUI().informar("acercaDe(): "
       + new escritorError().escribirError(e));
```

```
    }
  }
}
```

Clase dicoms.ObjetoDicom.java

```
package cma.dicoms;

import com.archimed.dicom.*;
import cma.paneles.*;
import cma.escritores.escritorError;
import javax.swing.*;
import java.io.*;
import java.awt.*;

/**
 * La clase ObjetoDicom crea un fichero con formato compatible al
 * estándar a partir de la imagen de la lesión y los datos del
 * paciente
 * @author Mª José Aguilar Porro
 * @version 1.0
 */
public class ObjetoDicom {

  public static FileOutputStream salvar;
  public image.PixelGrabber pg;

  public boolean creaDicom(com.archimed.dicom.image.DicomImage
      dcmNuevo) {
    try {
      java.awt.image.PixelGrabber grabber;
      /**
       * A partir de aquí, iremos rellenando los campos del objeto
       * DICOM que queremos construir
       */

      java.awt.Image imagenJPG = PanelFotografia.fotodic.getImage();
      grabber = new java.awt.image.PixelGrabber(imagenJPG, 0, 0,
        imagenJPG.getHeight(null),imagenJPG.getWidth(null), true);
      try {
        grabber.grabPixels();
      }
      catch (Throwable e) {
        System.out.println("error al grabar los píxeles" +
                           new escritorError().escribirError(e));
        return false;
      }
      int[] pix = imageToIntArray(imagenJPG);
      dcmNuevo.imagePixelData(imagenJPG.getHeight(null),
            imagenJPG.getWidth(null),0, pix);
      String paciente = PanelDatosPersonales.apell1Campo.getText() + "
         " + PanelDatosPersonales.apell2Campo.getText() + " " +
          PanelDatosPersonales.nombreCampo.getText();
      dcmNuevo.set(DDict.dPatientName, new Person(paciente));
      String direccion =
            PanelDatosPersonales.domicilioCampo.getText();
      dcmNuevo.set(DDict.dPatientAddress, direccion);
      String region = PanelDatosPersonales.localidadCampo.getText();
      dcmNuevo.set(DDict.dRegionOfResidence, region);
      String telefono = PanelDatosPersonales.tfnoCampo.getText();
      dcmNuevo.set(DDict.dPatientTelephoneNumber, telefono);
```

```
String fecha = PanelDatosPersonales.fecNac3Campo.getText() +
    PanelDatosPersonales.fecNac2Campo.getText() +
    PanelDatosPersonales.fecNac1Campo.getText();
DDate fecNac = new DDate(fecha);
dcmNuevo.set(DDict.dPatientBirthDate, fecNac);
String doctor = PanelOtrosDatos.doctorCampo.getText();
dcmNuevo.set(DDict.dReferringPhysiciansName,
             new Person(doctor));
String address = PanelOtrosDatos.direccionCampo.getText();
dcmNuevo.set(DDict.dReferringPhysiciansAddress,
             address);

/**
 *  Si falta alguno de los siguientes campos por rellenar,
 * lo indicará y no continuará la ejecución
 */

if (PanelDatosPersonales.nombreCampo.getText().length() < 1) {
  JOptionPane.showMessageDialog(null,
     "No insertó nombre del paciente");
  return false;
}
if (PanelDatosPersonales.apell1Campo.getText().length() < 1) {
  JOptionPane.showMessageDialog(null,
     "No insertó el primer apellido del paciente");
  return false;
}
if (PanelDatosPersonales.apell2Campo.getText().length() < 1) {
  JOptionPane.showMessageDialog(null,
     "No insertó el segundo apellido del paciente");
  return false;
}
if (direccion.length() < 1) {
  JOptionPane.showMessageDialog(null,
     "Falta la dirección del paciente");
  return false;
}
if (region.length() < 1) {
  JOptionPane.showMessageDialog(null,
     "Falta la localidad donde vive el paciente");
  return false;
}

/**
 *  El resto de los campos no van a ser obligatorios, ya que se
 *  supone que con los datos personales más representativos
 *  es suficiente
 */

else {
/**
 * Los campos que vienen a continuación, los creamos para poder
 * enviar información adicional en el archivo DICOM. Los tenemos
 * que crear porque no están recogidos en el diccinario de datos
 * del estandar (parte 3 de DICOM). Los tenemos que meter en el
 * else del bucle if porque si no, se repetirían las entradas si
 * dejásemos alguno de los campos anteriores sin rellenar y da
 * error
 */

  // En primer lugar, creamos una instancia DDict (diccionario
```

```
//de datos DICOM)
DDict diccionario = new dicom.DDict();
// A continuación, una entrada de datos de paciente, por eso
// grupo = 0010, tipo cadena de caracteres
DDictEntry entrada1 = new DDictEntry(0010, 4001,
   DDict.tST,"Resumen enfermedad", "1");
diccionario.addEntry(entrada1); // la añadimos al diccionario
String resumen =
    PanelDatosClinicos.resumenEnfermedad.getText();
if (resumen.length() < 1) {
  resumen = "No se introdujo en el formulario de petición de
    consulta";
// Y finalmente, la añadimos a nuestro objeto de imagen DICOM
}
dcmNuevo.set_ge(0010, 4001, resumen);
// A partir de aquí, procedemos de igual forma con todos los
// campos nuevos que insertamos

// Campo para el resumen de la exploración clínica:
// (grupo,elto) = (0010,4002)
DDictEntry entrada2 = new DDictEntry(0010, 4002, DDict.tST,
               "Resumen exploración", "1");
diccionario.addEntry(entrada2);
String exploracion =
    PanelDatosClinicos.resumenExploracion.getText();
if (exploracion.length() < 1) {
  exploracion =
  "No se introdujo en el formulario de petición de consulta";
}
dcmNuevo.set_ge(0010, 4002, exploracion);

// Campo para el las exploraciones complementarias:
// (grupo,elto) = (0010,4003)
DDictEntry entrada3 = new DDictEntry(0010, 4003, DDict.tST,
               "Exploraciones complementarias", "1");
diccionario.addEntry(entrada3);
String expcomplem =
    PanelDatosClinicos.exploracionComplem.getText();
if (expcomplem.length() < 1) {
  expcomplem =
  "No se introdujo en el formulario de petición de consulta";
}
dcmNuevo.set_ge(0010, 4003, expcomplem);

// Campo para el diagnóstico provisional:
// (grupo,elto) = (0010,4004)
DDictEntry entrada4 = new DDictEntry(0010, 4004, DDict.tST,
               "Diagnóstico provisional", "1");
diccionario.addEntry(entrada4);
String diagnostico =
    PanelDatosClinicos.diagnosticoProv.getText();
if (diagnostico.length() < 1) {
  diagnostico =
  "No se introdujo en el formulario de petición de consulta";
}
dcmNuevo.set_ge(0010, 4004, diagnostico);

// Campo para el tratamiento anterior/actual:
// (grupo,elto) = (0010,4005)
DDictEntry entrada5 = new DDictEntry(0010, 4005, DDict.tST,
               "Tratamiento anterior o actual", "1");
```

```
diccionario.addEntry(entrada5);
String tratamiento =
    PanelDatosClinicos.tratamientoAnterior.getText();
if (tratamiento.length() < 1) {
  tratamiento =
  "No se introdujo en el formulario de petición de consulta";
}
dcmNuevo.set_ge(0010, 4005, tratamiento);

// Campo para el fundamento clínico de la petición:
// (grupo,elto) = (0010,4006)
DDictEntry entrada6 = new DDictEntry(0010, 4006,DDict.tST,
               "Fundamento clínico de petición", "1");
diccionario.addEntry(entrada6);
String fundamento =
    PanelDatosClinicos.fundamentoClinico.getText();
if (fundamento.length() < 1) {
  fundamento =
 "No se introdujo en el formulario de petición de consulta";
}
dcmNuevo.set_ge(0010, 4006, fundamento);

// Campo para el Código Numérico Personal del médico
// responsable: (grupo,elto) = (0010,4007)
DDictEntry entrada7 = new DDictEntry(0010, 4007, DDict.tST,
               "Fundamento clínico de petición", "1");
diccionario.addEntry(entrada7);
String codigo = PanelOtrosDatos.codigoCampo.getText();
if (codigo.length() < 1) {
  codigo = "No se introdujo en el formulario de petición de
    consulta";
}
dcmNuevo.set_ge(0010, 4007, codigo);

// Campo para la especialidad del médico responsable:
// (grupo,elto) = (0010,4008)
DDictEntry entrada8 = new DDictEntry(0010, 4008, DDict.tST,
               "Especialidad del médico", "1");
diccionario.addEntry(entrada8);
String especialidad =
    PanelOtrosDatos.especialidadCampo.getText();
if (especialidad.length() < 1) {
  especialidad =
  "No se introdujo en el formulario de petición de consulta";
}
dcmNuevo.set_ge(0010, 4008, especialidad);

// Campo para la localidad del médico responsable:
// (grupo,elto) = (0010,4009)
DDictEntry entrada9 = new DDictEntry(0010, 4009, DDict.tST,
               "Localidad del médico", "1");
diccionario.addEntry(entrada9);
String localid = PanelOtrosDatos.localidadCampo.getText();
if (localid.length() < 1) {
  localid = "No se introdujo en el formulario de petición de
    consulta";
}
dcmNuevo.set_ge(0010, 4009, localid);

DDictEntry entrada10 = new DDictEntry(0010, 4010, DDict.tST,
               "Número de afiliación a la SS", "1");
```

```
diccionario.addEntry(entrada10);
String numAfiliac =
    PanelDatosPersonales.afiliSSCampo.getText();
if (numAfiliac.length() < 1) {
  numAfiliac = PanelDatosPersonales.taSSCampo.getText();
}
dcmNuevo.set_ge(0010, 4010, numAfiliac);

DDictEntry entrada11 = new DDictEntry(0010, 4011, DDict.tST,
    "Diabetes", "1");
diccionario.addEntry(entrada11);
String diabet;
if (PanelPatologias.diabetSi.getState()) {
  diabet = "Sí";
}
else if (PanelPatologias.diabetNo.getState()) {
  diabet = "No";
}
else {
  diabet = "No se señaló en el formulario";
}
dcmNuevo.set_ge(0010, 4011, diabet);

DDictEntry entrada12 = new DDictEntry(0010, 4012, DDict.tST,
    "Hipertensión", "1");
diccionario.addEntry(entrada12);
String hipert;
if (PanelPatologias.hipertSi.getState()) {
  hipert = "Sí";
}
else if (PanelPatologias.hipertNo.getState()) {
  hipert = "No";
}
else {
  hipert = "No se señaló en el formulario";
}
dcmNuevo.set_ge(0010, 4012, hipert);

DDictEntry entrada13 = new DDictEntry(0010, 4013, DDict.tST,
               "Tratamientos con anticoagulantes", "1");
diccionario.addEntry(entrada13);
String anticoag;
if (PanelPatologias.anticoagSi.getState()) {
  anticoag = "Sí";
}
else if (PanelPatologias.anticoagNo.getState()) {
  anticoag = "No";
}
else {
  anticoag = "No se señaló en el formulario";
}
dcmNuevo.set_ge(0010, 4013, anticoag);

DDictEntry entrada14 = new DDictEntry(0010, 4014, DDict.tST,
               "Intervenciones Previas", "1");
diccionario.addEntry(entrada14);
String intervprev = PanelPatologias.intervPrevias.getText();
if (intervprev.length() < 1) {
  intervprev =
"No se introdujeron en el formulario de petición de consulta";
}
```

```
  dcmNuevo.set_ge(0010, 4014, intervprev);

  DDictEntry entrada15 = new DDictEntry(0010, 4015, DDict.tST,
                "Enfermedades anteriores", "1");
  diccionario.addEntry(entrada15);
  String enfants = PanelPatologias.enfAnteriores.getText();
  if (enfants.length() < 1) {
    enfants =
  "No se introdujeron en el formulario de petición de consulta";
  }
  dcmNuevo.set_ge(0010, 4015, enfants);

  DDictEntry entrada16 = new DDictEntry(0010, 4016, DDict.tST,
                "Alergias a medicamentos", "1");
  diccionario.addEntry(entrada16);
  String alermedic = PanelPatologias.alergiasMedic.getText();
  if (alermedic.length() < 1) {
    alermedic =
  "No se introdujeron en el formulario de petición de consulta";
  }
  dcmNuevo.set_ge(0010, 4016, alermedic);

  DDictEntry entrada17 = new DDictEntry(0010, 4017, DDict.tST,
                "Otros comentarios", "1");
  diccionario.addEntry(entrada17);
  String otros = PanelPatologias.otros.getText();
  if (otros.length() < 1) {
    otros = "No se introdujeron en el formulario de petición de
      consulta";
  }
  dcmNuevo.set_ge(0010, 4017, otros);

}

/**
 * No nos podemos olvidar de la SOPClassUID ni de la
 * SOPInstanceUID que será "1.2.840.10008.5.1.4.1.1.7" para
 * ambas pq estamos tratando con "Secondary Capture Image
 * Storage"
 */

dcmNuevo.set(com.archimed.dicom.DDict.dSOPClassUID,
             "1.2.840.10008.5.1.4.1.1.7");
dcmNuevo.set(com.archimed.dicom.DDict.dSOPInstanceUID,
             "1.2.840.10008.5.1.4.1.1.7");

/**
 * Lo último seria construir un archivo dicom con la imagen que
 * hemos creado y los datos correspondientes
 */
try {
  String cadena;
  // El archivo se creará con el nombre que el usuario prefiera
  Object o = JOptionPane.showInputDialog(null,
    "Escriba el nombre del archivo DICOM que va a ser creado");
  cadena = o.toString();
  salvar = new FileOutputStream("c:/dicoms/" + cadena + ".dcm");
  dcmNuevo.write(salvar, true);
  return true;
}
catch (Throwable e) {
```

```
        System.out.println("Error al escribir fichero DICOM" +
                           new escritorError().escribirError(e));
        return false;
      }

    }

    catch (Throwable ex) {
      System.out.println("error dicom " + new
            escritorError().escribirError(ex));
      return false;
    }

  }

 /**
  * El siguiente método devuelve la información de la imagen que se
  * le pasa como parámetro en forma de array de enteros
  */

  public int[] imageToIntArray(java.awt.Image im) {
    int[] p;
    pg = new java.awt.image.PixelGrabber(im, 0, 0, im.getWidth(null),
                                         im.getHeight(null), true);
    try {
      pg.grabPixels();
    }
    catch (Throwable e) {
      System.out.println("error al grabar los píxeles" +
                         new escritorError().escribirError(e));
    }

    p = (int[])pg.getPixels();

    return p;
  }

}
```

Clase dicoms.enviaFich.java

```
package cma.dicoms;

/**
 * Esta clase contiene los métodos necesarios para implementar el
 * servicio DIMSE C-STORE
 */

import cma.paneles.*;
import cma.escritores.*;
import java.net.*;
import java.io.*;
import java.lang.reflect.*;
import java.util.*;
import com.archimed.dicom.*;
import com.archimed.dicom.network.*;
import com.archimed.tool.*;

public class enviaFich {

  public String host = "huvr"; // nombre del servidor DICOM
```

```
public int port = 104; // puerto por el que se envía, es el
                       // reservado para DICOM
public String called; // Entidad llamada
public String calling = "HUVR"; // Entidad llamante, la que
                       // corresponda en el HUVR
public int maxpdu; // Vble para q las entidades se pongan de acuerdo
    // en el tamaño máximo de las PDU
public boolean verbose = false; // Para manejar las salidas de
         // información por pantalla
public int numpc = 0;
public int[] tsids = {TransferSyntax.ImplicitVRLittleEndian};
    // Para ponerse de acuerdo en la sintaxis de transferencia
public int preftsid = TransferSyntax.ImplicitVRLittleEndian;
public boolean combts = false;
public Vector usablePCIndices = new Vector();
Association as;
Request request;
Acknowledge ack;
com.archimed.dicom.image.DicomImage dcmNuevo = new
    com.archimed.dicom.image.DicomImage();

boolean f = true;

/**
 *  Lo primero que hay que hacer para utilizar el servicio C-STORE
 * es crear una asociación entre las dos entidades en la que ambas
 * se pongan de acuerdo en los contextos de presentación que
 * van a utilizar y en el tipo de información que van a intercambiar
 */
boolean associate() throws Exception {
  Object p = JOptionPane.showInputDialog(
      "Por favor, inserte el nombre de la entidad DICOM llamada");
  called = p.toString();

  // Conectamos y creamos un objeto de asociación
  Socket s = new Socket(host, port); // tengo que inicializar el
                           // servidor ; port = 104
  as = new Association(s.getInputStream(), s.getOutputStream());

  // Preparamos el comando de petición con el título de entidad
  // llamante, entidad llamada, sintaxis abstracta y sintaxis de
  // transferencia
  request = new Request();
  calling = request.getCallingTitle();
  request.setCalledTitle(called);
  request.setCallingTitle(calling);
  request.setMaxPduSize(maxpdu);

  if (combts) {
    request.addPresentationContext(SOPClass.
          SecondaryCaptureImageStorage, tsids);
  }
  else {
    for (int i = 0; i < tsids.length; i++) {
      request.addPresentationContext(SOPClass.
          SecondaryCaptureImageStorage, new int[] {tsids[i]});
    }
  }

  Class cl = SOPClass.class;
  Field[] fields = cl.getDeclaredFields();
```

```
int n = 0;
int i = 0;
int j = 0;
while (n < numpc) {
  if (((Integer)fields[i % fields.length].get(null)).intValue() ==
      SOPClass.SecondaryCaptureImageStorage) {
    i++;
    continue;
  }
  if (combts) {
    request.addPresentationContext(((Integer)fields[i %
      fields.length].get(null)).intValue(),tsids);
    i++;
  }
  else {
    request.addPresentationContext(((Integer)fields[i %
      fields.length].get(null)).intValue(),new int[]
      {tsids[j++]});
    if (j == tsids.length) {
      i++;
      j = 0;
    }
  }
  n++;
}

// Imprime información
if (verbose) {
  System.out.println();
  System.out.println(request);
}

// Envía la petición y recibe la respuesta
as.sendAssociateRequest(request);
Response response = as.receiveAssociateResponse();

if (verbose) {
  System.out.println();
  System.out.println(response);
}

// Analiza la respuesta, que será siempre asentimiento, rechazo o
// "Abort"
if (response instanceof Acknowledge) {
  ack = (Acknowledge) response;
  for (i = 0; i < request.getPresentationContexts(); i++) {
    if (request.getAbstractSyntax(i).getConstant() !=
        SOPClass.SecondaryCaptureImageStorage) {
      continue;
    }

    for (j = 0; j < ack.getPresentationContexts(); j++) {
      if (request.getID(i) == ack.getID(j)) {
        if (ack.getResult(j) == Acknowledge.ACCEPTANCE) {
          usablePCIndices.addElement(new Integer(i));
        }
      }
    }
  }
  return true;
}
```

```
    else if (response instanceof Reject) {
      System.out.println("Asociación rechazada: " + response);
      JOptionPane.showMessageDialog(null,
          "Petición de asociación para almacenamiento rechazada: " +
            response);
      return false;
    }
    else if (response instanceof Abort) {
      System.out.println("Asociación abortada: " + response);
      JOptionPane.showMessageDialog(null,
          "Asociación abortada: " + response);
      return false;
    }
    return false;
  }

  /**
   *  El método siguiente implementa el servicio de almacenamiento
   *  C-STORE en sí, es decir, el intercambio de información para
   *  almacenar entre el cliente y el servidor propiamente dicho
   */

  boolean exchangeInfo(DicomObject dcmNuevo) throws Exception {
    /**
     * En primer lugar, se inicializa el contexto de presentación el
     * que se envía el C-STORE
     */
    int pcid = request.getID(((Integer)
      usablePCIndices.firstElement()).intValue());

    // Determinamos el contexto de presentación teniendo en cuenta la
    // sintaxis de transferencia preferida
    loop:
        for (int i = 0; i < usablePCIndices.size(); i++) {
      for (int j = 0; j < ack.getPresentationContexts(); j++) {
        if (ack.getID(j) ==
            request.getID(((Integer)
              usablePCIndices.elementAt(i)).intValue())) {
          if (ack.getTransferSyntax(j).getConstant() == preftsid) {
            pcid = ack.getID(j);
            break loop;
          }
        }
      }
    }
    if (verbose) {
      System.out.println("Utilizando contexto de presentación con ID:
          " + pcid);
      /**
       * Creamos una petición de almacenamiento C-STORE , la enviamos
       * y recibimos la respuesta
       */

    }
    DicomObject storereq = DimseUtil.createStoreRequest(new nteger(0),
        new Integer(SOPClass.SecondaryCaptureImageStorage), new
        Integer("0002H"), "1.2.840.10008.5.1.4.1.1.7", calling,
        called);
    if (verbose) {
      System.out.println();
      System.out.println("********** C-STORE COMMAND REQUEST
```

```
            ***********");
      storereq.dumpVRs(System.out);

      System.out.println("*****************************************
            ***");
      System.out.println();
    }

    /**
     * Enviamos el objeto Dicom que generamos antes en el contexto de
     * presentación seleccionado
     */

    as.sendInPresentationContext(pcid, storereq, dcmNuevo);
    DicomObject storeresp = as.receiveCommand();
    if (verbose) {
      System.out.println();
      System.out.println("*********** C-STORE COMMAND RESPONSE
            **********");
      storereq.dumpVRs(System.out);
      System.out.println("*****************************************
      ***");
      System.out.println();
    }

    // Comprobamos que la respuesta es de éxito
    if (DimseUtil.getCommandType(storeresp) !=
      DimseUtil.C_STORE_RESPONSE) {
      if (verbose) {
        System.out.println("No es un comando de respuesta válido de C-
            STORE: " + DimseUtil.getCommandType(storeresp));
      }
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
            Abort.REASON_NOT_SPECIFIED);
      return false;
    }

    String sopClass = storeresp.getS(DDict.dAffectedSOPClassUID);
    if (DimseUtil.getAffectedSOPClass(storeresp) !=
        SOPClass.SecondaryCaptureImageStorage) {
      if (verbose) {
        System.out.println(
            "UID de clase SOP no válida en el mensaje de respuesta C-
              STORE: " + DimseUtil.getAffectedSOPClass(storeresp));
      }
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
        Abort.REASON_NOT_SPECIFIED);
      return false;
    }

    if (DimseUtil.getMessageIDBeingRespondedTo(storeresp) != 0) {
      if (verbose) {
        System.out.println("ID del Mensaje de respuesta C-STORE no
            válido: " + DimseUtil.getMessageID(storeresp));
      }
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
         Abort.REASON_NOT_SPECIFIED);
      return false;
    }

    int status = storeresp.getI(DDict.dStatus);
```

```
    /**
     * Si este campo vale 0000, es que el almacenamiento se realizó
     * con éxito, si no, hubo algún fallo
     */
    if (status != 0) {
      if (verbose) {
        System.out.println("El estado en la respuesta C-STORE no es de
            éxito: " + status);
      }
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
            Abort.REASON_NOT_SPECIFIED);
      return false;
    }
    /**
     * Si no se produjo ninguna de las condiciones anteriores, el
     * envío del archivo DICOM y su correspondiente almacenamiento se
     * realizaron con éxito
     */
    JOptionPane.showMessageDialog(null,
        "El formulario ha sido enviado y almacenado correctamente");
    return true;
  }

  /**
   * Finalmente, solo nos queda el método que libera la asociación
   */
  boolean release() throws Exception {
    // Liberamos la asociación
    as.sendReleaseRequest();
    as.receiveReleaseResponse();
    if (verbose) {
      System.out.println("Recibido asentimiento de liberación de
        asociación");
    }
    return true;
  }
}
```

Clase dicoms.CompConexion.java

```
package cma.dicoms;

/**
 * Con esta clase se implementa el servicio C-ECHO de DICOM
 */

import cma.dicoms.ObjetoDicom;
import java.net.*;
import java.io.*;
import java.lang.reflect.*;
import java.util.*;
import com.archimed.dicom.*;
import com.archimed.dicom.network.*;
import com.archimed.tool.*;
import cma.escritores.*;

public class CompConexion {
  public String host; // servidor
  public int port = 104; // puerto por el que se envía
  public String called; // Entidad llamante
  public String calling // Entidad llamada, la correspondiente en HUVR
```

```
  public int numpc = 0;
  public int[] tsids = {TransferSyntax.ImplicitVRLittleEndian};
 // Para ponerse de acuerdo en la sintaxis de transf
  public int preftsid = TransferSyntax.ImplicitVRLittleEndian;  public
  boolean combts = false;
  public boolean verbose = false;
  public Vector usablePCIndices = new Vector();
  Association as;
  Request request;
  Acknowledge ack;

  public CompConexion() {
    verbose = isVerbose();
    boolean f = true;
    try {
      if (f = associate()) {
        exchangeEcho();
        release();
      }
    }
    catch (Throwable t) {
      JOptionPane.showMessageDialog(null,
          "La conexión con el servidor es defectuosa: " +
             new escritorError().escribirError(t));
    }
  }
  // Método para crear la asociación
  boolean associate() throws Exception {
      // conexión y creación de un objeto de asociación
    Socket s = new Socket(host, port);
    as = new Association(s.getInputStream(), s.getOutputStream());
    Object p = javax.swing.JOptionPane.showInputDialog(
        "Por favor, inserte el nombre de la entidad DICOM llamada");
    called = p.toString();

    // prepara un objeto de "request
    request = new Request();
    request.setCalledTitle(called);
    calling = request.getCallingTitle;
    request.setCallingTitle(calling);
    if (combts) {
      request.addPresentationContext(SOPClass.Verification, tsids);
    }
    else {
      for (int i = 0; i < tsids.length; i++) {
        request.addPresentationContext(SOPClass.Verification,
                                       new int[] {tsids[i]});
      }
    }

    Class cl = SOPClass.class;
    Field[] fields = cl.getDeclaredFields();
    int n = 0;
    int i = 0;
    int j = 0;
    while (n < numpc) {
      if (((Integer)fields[i % fields.length].get(null)).intValue() ==
          SOPClass.Verification) {
        i++;
        continue;
      }
```

```
      if (combts) {
        request.addPresentationContext(((Integer) fields[i %
         fields.length].get(null)).intValue(), tsids);
        i++;
      }
      else {
        request.addPresentationContext(((Integer) fields[i %
         fields.length].get(null)).intValue(),new int[] {tsids[j++]});
        if (j == tsids.length) {
          i++;
          j = 0;
        }
      }
      n++;
    }

    // Imprime la información
    if (verbose) {
      System.out.println();
      System.out.println(request);
    }
    // Envía la petición y recibe la respuesta
    as.sendAssociateRequest(request);
    Response response = as.receiveAssociateResponse();
    if (verbose) {
      System.out.println();
      System.out.println(response);
    }
    // Analiza la respuesta, que será siempre del tipo
    // Acknowledge, Reject o Abort
    if (response instanceof Acknowledge) {
      ack = (Acknowledge) response;
      for (i = 0; i < request.getPresentationContexts(); i++) {
        if (request.getAbstractSyntax(i).getConstant() !=
            SOPClass.Verification) {
          continue;
        }
        for (j = 0; i < ack.getPresentationContexts(); j++) {
          if (request.getID(i) == ack.getID(j)) {
            if (ack.getResult(j) == Acknowledge.ACCEPTANCE) {
              usablePCIndices.addElement(new Integer(i));
            }
          }
        }
      }
      return true;
    }
    else if (response instanceof Reject) {
      JOptionPane.showMessageDialog(null,
          "Petición de asociación rechazada: " + response);
      return false;
    }
    else if (response instanceof Abort) {
      JOptionPane.showMessageDialog(null,
           "Asociación abortada: " + response);
      return false;
    }
    return false;
  }
```

```
/**
  * El siguiente método implementa el servicio C-ECHO propiamente
  * dicho, es decir, el intercambio de las primitivas de verificación
  * de la conexión entre ambas entidades
  */
 boolean exchangeEcho() throws Exception {
   // Inicializa el contexto de presentación en el que se enviará la
     // petición de C-ECHO
   int pcid = request.getID(((Integer)
     usablePCIndices.firstElement()).intValue());
   loop:
       for (int i = 0; i < usablePCIndices.size(); i++) {
     for (int j = 0; j < ack.getPresentationContexts(); j++) {
       if (ack.getID(j) ==
           request.getID( ( (Integer)
           usablePCIndices.elementAt(i)).intValue())) {
         if (ack.getTransferSyntax(j).getConstant() == preftsid) {
           pcid = ack.getID(j);
           break loop;
         }
       }
     }
   }
   if (verbose) {
     System.out.println("Utilizando el contexto de presentación con
     ID: " + pcid);
   }

   // Creamos una petición C-ECHO , la enviamos y recibimos la
   // respuesta
   DicomObject echoreq = DimseUtil.createEchoRequest(new Integer(0),
       new Integer(SOPClass.Verification));
   if (verbose) {
     System.out.println();
     System.out.println("*********** C-ECHO COMMAND REQUEST
           ************");
     echoreq.dumpVRs(System.out);
     System.out.println("****************************************
           ***");
   }
   as.sendInPresentationContext(pcid, echoreq, null);
   DicomObject echores = as.receiveCommand();
   if (verbose) {
     System.out.println();
     System.out.println("*********** C-ECHO COMMAND RESPONSE
           ***********");
     echores.dumpVRs(System.out);
     System.out.println("****************************************
           ***");
     System.out.println();
   }

   // Comprobamos que echores es una respuesta C-ECHO correcta
   if (DimseUtil.getCommandType(echores) !=
     DimseUtil.C_ECHO_RESPONSE) {
     JOptionPane.showMessageDialog(null,
         "No C-ECHO RESPONSE command field : " +
          DimseUtil.getCommandType(echores));
     as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
           Abort.REASON_NOT_SPECIFIED);
     return false;
```

```
    }
    String sopclass = echores.getS(DDict.dAffectedSOPClassUID);
    if (DimseUtil.getAffectedSOPClass(echores) !=
      SOPClass.Verification) {
      JOptionPane.showMessageDialog(null,
          "Affected SOP Class UID in C-ECHO RESPONSE wrong : " +
           DimseUtil.getAffectedSOPClass(echores));
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
            Abort.REASON_NOT_SPECIFIED);
      return false;
    }
    if (DimseUtil.getMessageIDBeingRespondedTo(echores) != 0) {
      JOptionPane.showMessageDialog(null,
          "Message ID in C-ECHO RESPONSE wrong: " +
           DimseUtil.getMessageID(echores));
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
            Abort.REASON_NOT_SPECIFIED);
      return false;
    }
    int status = echores.getI(DDict.dStatus);
    if (status != 0) {
      JOptionPane.showMessageDialog(null,
          "Non successful status in C-ECHO RESPONSE: " + status);
      as.sendAbort(Abort.DICOM_UL_SERVICE_USER,
            Abort.REASON_NOT_SPECIFIED);
      return false;
    }
    return true;
  }

  boolean release() throws Exception {
    // Liberación de la asociación
    as.sendReleaseRequest();
    as.receiveReleaseResponse();
    JOptionPane.showMessageDialog(null,
        "Recibido asentimiento de liberación de conexión");
    return true;
  }
}
```

El resto de clases están contenidas en los paquetes *cma.escritores* y *cma.menus*; como son muy parecidas a las mismas clases en las otras dos aplicaciones, no incluiremos el código aquí, aunque sí que está en el CD correspondiente.