6. Planos de código

En este capítulo se adjunta el código fuente de todas las clases, seis en total:

- `MenuSwing4.java`: es la cuarta versión de las desarrolladas para la ventana principal de la aplicación.

- `GenEnveloped.java`: esta clase permite firmar digitalmente un documento XML siguiendo el formato Enveloped. Toma un fichero de entrada y genera uno de salida ya firmado.

- `GenEnveloping.java`: esta clase permite firmar digitalmente un documento XML siguiendo el formato Enveloping. Toma un fichero de entrada y genera uno de salida ya firmado.

- `GenDetached.java`: esta clase permite firmar digitalmente un recurso externo definido en una URI siguiendo el formato Detached. Toma una URI de entrada y genera un fichero de salida con esa URI ya firmada.

- `Validar.java`: esta clase permite validar cualquiera de los documentos XML generado por las clases anteriores mostrando por pantalla todas las incidencias.

- `Ayuda.java`: información general sobre el funcionamiento y créditos de la aplicación.

```java
/**
 * @autor Fernando J. Pérez Borrero
 *
 * Clase MenuSwing4.java
 *
 * Ventana principal de la aplicación. Contiene el método principal main, y las clases
 * VigilaMenu, WindowHandler, DialogoModal y Dialogomodal2.
 *
 */

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

public class MenuSwing4 extends JFrame {

    Container contenedor;
    JTextArea areaTexto;

      // Para la construcción del menú hay que crear un objeto de la clase JMenuBar
    JMenuBar barraMenu = new JMenuBar();
      // Los objetos JMenu son aquellos con los que se despliegan los menús
    JMenu menuFirmar = new JMenu("Firmar");
    JMenu menuValidar = new JMenu("Validar");
    JMenu menuAyuda = new JMenu("Ayuda");
      // Los objetos JMenuItem representan las opciones del menú
    JMenuItem opcionEnveloped = new JMenuItem("Documento XML en formato Enveloped");
    JMenuItem opcionEnveloping = new JMenuItem("Documento XML en formato Enveloping");
    JMenuItem opcionDetached = new JMenuItem("Documento anexo - formato Detached");
    JMenuItem opcionSalirF = new JMenuItem("Salir");
    JMenuItem opcionValidarF = new JMenuItem("Validar XML firmado");
    JMenuItem opcionHowTo = new JMenuItem("How To...");
    JMenuItem opcionAbout = new JMenuItem("About");

      // Constructor de la clase MenuSwing4
    public MenuSwing4() {
```

```
        super("Firma digital de documentos");

            // Obtenemos una referencia al contenedor principal
        contenedor = getContentPane();
            // Añadimos los objetos JMenuItem (opciones)
        menuFirmar.add(opcionEnveloped);
        menuFirmar.add(opcionEnveloping);
        menuFirmar.add(opcionDetached);
        menuFirmar.add(opcionSalirF);
        menuValidar.add(opcionValidarF);
            menuAyuda.add(opcionHowTo);
            menuAyuda.add(opcionAbout);

            // Añadimos los objetos JMenu (menús)
        barraMenu.add(menuFirmar);
        barraMenu.add(menuValidar);
        barraMenu.add(menuAyuda);
            // Se asigna el marco de la aplicación
        setJMenuBar(barraMenu);
            // Sobre las opciones del menú se registran los escuchadores de eventos
            // de la clase VigilaMenu
        addWindowListener(new WindowHandler());
        opcionEnveloped.addActionListener(new VigilaMenu());
        opcionEnveloping.addActionListener(new VigilaMenu());
        opcionDetached.addActionListener(new VigilaMenu());
        opcionSalirF.addActionListener(new VigilaMenu());
        opcionValidarF.addActionListener(new VigilaMenu());
        opcionHowTo.addActionListener(new VigilaMenu());
        opcionAbout.addActionListener(new VigilaMenu());
        setSize(640,400);
            // Añadimos un objeto de texto para informar sobre los procesos de firma
            // y validación en lugar de usar la salida estándar que se deja para las
            // excepciones
        areaTexto = new JTextArea();
        contenedor.add("Center",areaTexto);
        setIconImage(Toolkit.getDefaultToolkit().createImage("Icono.gif"));
            // Centra la ventana en el escritorio y la presenta en pantalla
```

```
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String args[]) {
        MenuSwing4 app = new MenuSwing4();
    }

      /**
      * Manejador para eventos de ventana
      */

    public class WindowHandler extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    }

      /**
       * Nesitamos que nuestro código se pare en espera de la respuesta mientras
       * el usuario introduce ese dato en la ventana y pulsa <enter>.
       * Para estas situaciones tenemos los JDialog modales.
       * En el momento de hacerla visible llamando a setVisible(true),
       * el código se quedará parado en esa llamada hasta que la ventana se cierre.
       */

      public class DialogoModal extends JDialog {

            private JTextField textField;

            //Constructor de la ventana emergente para recoger el nombre del
            //fichero de salida y la hace modal.
            public DialogoModal(Frame padre) {

                  // Padre y modal
                  super(padre, true);
                  setTitle("Escribe un nombre para guardar el fichero XML firmado");
```

```
                textField = new JTextField(35);
                getContentPane().add(textField);

                // Ocultamos la ventana al pulsar <enter> sobre el textfield
                textField.addActionListener(new ActionListener() {

                      @Override
                      public void actionPerformed(ActionEvent arg0) {
                            setVisible(false);
                      }
                });
          }

          // Devuelve el texto en el jtextfield
          public String getText() {
                return textField.getText();
          }
    }

    public class DialogoModal2 extends JDialog {

          private JTextField textField;

          // Constructor de la ventana emergente para recoger la URI
          // necesaria en el formato Detached y la hace modal.
          public DialogoModal2(Frame padre) {

                super(padre, true);
                setTitle("Escribe una URI válida para firmar");
                textField = new JTextField(50);
                getContentPane().add(textField);

                textField.addActionListener(new ActionListener() {

                      @Override
                      public void actionPerformed(ActionEvent arg0) {
                            setVisible(false);
```

```
                        }
                  });
            }

            // Devuelve el texto en el jtextfield
            public String getText() {
                  return textField.getText();
            }
      }

      /**
      * Escuchadores de eventos de la barra de menús
      */

   public class VigilaMenu implements ActionListener {
       public void actionPerformed(ActionEvent e) {

            String accion = e.getActionCommand();
                  String fic_salida = "";
                  String URI = "";

                  try{

                        if(accion.equals("Documento XML en formato Enveloped")){
                              Frame ventana = new Frame();

                              // Solicitamos al sistema un fichero XML para firmar
                              FileDialog dialogo = new FileDialog(ventana,"Selecciona un fichero XML");
                              dialogo.setDirectory("c:\\");
                              dialogo.setVisible(true);
                              String nombreDir = (String)dialogo.getDirectory();
                              String nombreF = (String)dialogo.getFile();

                              // Usamos los JDialog para pausar la ejecución de código mientras
                              // el usuario escribe el nombre del fichero de salida
                              DialogoModal dialogoModal = new DialogoModal((Frame) ventana);
                              // Para darle tamaño automático a la ventana y centrarla
```

```
                dialogoModal.pack();
                dialogoModal.setLocationRelativeTo(null);
                // El código se detiene aquí a la espera de introducir el nombre del fichero
                dialogoModal.setVisible(true);

                fic_salida = dialogoModal.getText();

                areaTexto.setText("El fichero elegido para firmar es: "+nombreDir+nombreF);
                areaTexto.append("\nEl fichero firmado es: "+nombreDir+fic_salida);

                // Firmamos el documento XML en formato Enveloped
                GenEnveloped firma = new GenEnveloped();
                firma.firmar(nombreDir+nombreF,fic_salida);
        }

        if(accion.equals("Documento XML en formato Enveloping")){
                Frame ventana = new Frame();

                FileDialog dialogo = new FileDialog(ventana,"Selecciona un fichero XML");
                dialogo.setDirectory("c:\\");
                dialogo.setVisible(true);
                String nombreDir = (String)dialogo.getDirectory();
                String nombreF = (String)dialogo.getFile();

                DialogoModal dialogoModal = new DialogoModal(ventana);
                dialogoModal.pack();
                dialogoModal.setLocationRelativeTo(null);
                dialogoModal.setVisible(true);

                fic_salida = dialogoModal.getText();

                areaTexto.setText("El fichero elegido para firmar es: "+nombreDir+nombreF);
                areaTexto.append("\nEl fichero firmado es: "+nombreDir+fic_salida);

                // Firmamos el documento XML en formato Enveloping
                GenEnveloping firma = new GenEnveloping();
                firma.firmar(nombreDir+nombreF,fic_salida);
```

```
                }

                if(accion.equals("Documento anexo - formato Detached")){
                      Frame ventana = new Frame();

                      // Solicitamos URI por pantalla
                      DialogoModal2 dialogoModal2 = new DialogoModal2(ventana);
                      dialogoModal2.pack();
                      dialogoModal2.setLocationRelativeTo(null);
                      dialogoModal2.setVisible(true);

                      // Por ejemplo URI = "http://www.w3.org/TR/xml-stylesheet"
                      URI = dialogoModal2.getText();

                      DialogoModal dialogoModal = new DialogoModal(ventana);
                      dialogoModal.pack();
                      dialogoModal.setLocationRelativeTo(null);
                      dialogoModal.setVisible(true);

                      fic_salida = dialogoModal.getText();

                      areaTexto.setText("La URI para firmar es: "+URI);
                      areaTexto.append("\nEl fichero firmado es: "+fic_salida);


                      // Firmamos la URI en formato Detached
                      GenDetached firma = new GenDetached();
                      firma.firmar(URI,fic_salida);
                }

                if(accion.equals("Validar XML firmado")){
                      Frame ventana = new Frame();

                      FileDialog dialogo = new FileDialog(ventana,"Selecciona un fichero XML para
                                                          validarlo");
                      dialogo.setDirectory("c:\\");
                      dialogo.setVisible(true);
```

```
                                    String nombreDir = (String)dialogo.getDirectory();
                                    String nombreF = (String)dialogo.getFile();
                                    areaTexto.setText("El fichero elegido es: "+nombreDir+nombreF);

                                    Validar firmado = new Validar();
                                    // Verificamos que el documento XML en cualquiera de los formatos es válido
                                    firmado.valida(nombreF,areaTexto);
                              }

                              if(accion.equals("How To...")){
                                    Ayuda a = new Ayuda();

                                    a.howto(areaTexto);
                              }

                              if(accion.equals("About")){
                                    Ayuda a = new Ayuda();

                                    a.about(areaTexto);
                              }

                        }catch (Exception err){

                              // Imprime un texto indicando el tipo de error producido
                              System.out.println(err.getMessage());

                              // Imprime las llamadas producidas hasta el método que generó el error
                              err.printStackTrace();
                        }

                  if(accion.equals("Salir"))
                    System.exit(0);
            }
      }
}
```

```java
/**
 * @autor Fernando J. Pérez Borrero
 *
 * Clase GenEnveloped.java
 *
 * Crea una firma XML en el formato Enveloped usando
 * la JSR 105 API. La firma en este caso hace referencia
 * a una URI local que apunta a un elemento Object.
 *
 */

import javax.xml.crypto.*;
import javax.xml.crypto.dsig.*;
import javax.xml.crypto.dom.*;
import javax.xml.crypto.dsig.dom.DOMSignContext;
import javax.xml.crypto.dsig.keyinfo.*;
import javax.xml.crypto.dsig.spec.*;

import java.io.*;

import java.security.*;
import java.util.Collections;
import java.util.Iterator;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;

/*
 * El fichero XML firmado tendrá el siguiente formato:
 *
 *<Envelope xmlns="urn:envelope">
 * <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 *   <SignedInfo>
 *     <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n
```

```
*       -20010315"/>
*     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
*     <Reference URI="">
*       <Transforms>
*         <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
*       </Transforms>
*       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
*       <DigestValue>K8M/lPbKnuMDsO0Uzuj75lQtzQI=<DigestValue>
*     </Reference>
*   </SignedInfo>
*   <SignatureValue>
*     DpEylhQoiUKBoKWmYfajXO7LZxiDYgVtUtCNyTgwZgoChzorA2nhkQ==
*   </SignatureValue>
*   <KeyInfo>
*     <KeyValue>
*       <DSAKeyValue>
*         <P>
*           rFto8uPQM6y34FLPmDh40BLJ1rVrC8VeRquuhPZ6jYNFkQuwxnu/wCvIAMhukPBL
*           FET8bJf/b2ef+oqxZajEb+88zlZoyG8g/wMfDBHTxz+CnowLahnCCTYBp5kt7G8q
*           UobJuvjylwj1st7V9Lsu03iXMXtbiriUjFa5gURasN8=
*         </P>
*         <Q>
*           kEjAFpCe4lcUOdwphpzf+tBaUds=
*         </Q>
*         <G>
*           oe14R2OtyKx+s+60O5BRNMOYpIg2TU/f15N3bsDErKOWtKXeNK9FS7dWStreDxo2
*           SSgOonqAd4FuJ/4uva7GgNL4ULIqY7E+mW5iwJ7n/WTELh98mEocsLXkNh24HcH4
*           BZfSCTruuzmCyjdV1KSqX/Eux04HfCWYmdxN3SQ/qqw=
*         </G>
*         <Y>
*           pA5NnZvcd574WRXuOA7ZfC/7Lqt4cB0MRLWtHubtJoVOao9ib5ry4rTk0r6ddnOv
*           AIGKktutzK3ymvKleS3DOrwZQgJ+/BDWDW8kO9R66o6rdjiSobBi/0c2V1+dkqOg
*           jFmKz395mvCOZGhC7fqAVhHat2EjGPMfgSZyABa7+1k=
*         </Y>
*       </DSAKeyValue>
*     </KeyValue>
*   </KeyInfo>
```

```
 * </Signature>
 *</Envelope>
 */

public class GenEnveloped {

      // En esta clase no se define el constructor por defecto.

      public void firmar(String input, String output) throws Exception {

            // Creamos el objeto DOM XMLSignatureFactory que será usado para
            // generar el elemento XMLSignature
            String providerName = System.getProperty("jsr105Provider",
                        "org.jcp.xml.dsig.internal.dom.XMLDSigRI");

            XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM",
                  (Provider) Class.forName(providerName).newInstance());

            // Crea una Referencia del documento a firmar (en este caso firmamos
            // el documento completo por lo que se indica usando la URI ""). También
            // se especifica el algoritmo de resumen SHA1 y transformación ENVELOPED
            Reference ref = fac.newReference("",
                              fac.newDigestMethod(DigestMethod.SHA1, null),
                              Collections.singletonList
                              (fac.newTransform
                              (Transform.ENVELOPED, (TransformParameterSpec) null)),
                              null, null);

            // Crea el elemento SignedInfo
            SignedInfo si = fac.newSignedInfo(fac.newCanonicalizationMethod(
                              CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS,
                              (C14NMethodParameterSpec) null),
                              fac.newSignatureMethod(SignatureMethod.DSA_SHA1, null),
                              Collections.singletonList(ref));

            // Creamos un par de claves DSA
        KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
```

```
            kpg.initialize(512);
        KeyPair kp = kpg.generateKeyPair();

        // Crea un elemento KeyValue conteniendo la clave pública DSA que fue generada
            KeyInfoFactory kif = fac.getKeyInfoFactory();
        KeyValue kv = kif.newKeyValue(kp.getPublic());

            //Crea un elemento KeyInfo y añade el elemento KeyValue a él
        KeyInfo ki = kif.newKeyInfo(Collections.singletonList(kv));

            // Instanciamos el documento a firmar. Podíamos haber hecho este
            // paso al principio como en la clase GenEnveloping.java
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setNamespaceAware(true);
            Document doc = dbf.newDocumentBuilder().parse(new File(input));

            // Creamos un objeto DOMSignContext y especificamos la clave privada
            // DSA para firmar y la localización del elemento padre XMLSignature
            DOMSignContext dsc = new DOMSignContext
                  (kp.getPrivate(), doc.getDocumentElement());

            // Crea el elemento XMLSignature (pero no se firma todavía)
            XMLSignature signature = fac.newXMLSignature(si, ki);

            // Por último firmamos usando la clave privada
            signature.sign(dsc);

            // Generamos el documento
            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer trans = tf.newTransformer();
            trans.transform(new DOMSource(doc), new StreamResult(new FileOutputStream(output)));
    }
}
```

```java
/**
 * @autor Fernando J. Pérez Borrero
 *
 * Clase GenEnveloping.java
 *
 * Crea una firma XML en el formato Enveloping usando
 * la JSR 105 API. La firma en este caso hace referencia
 * a una URI local que apunta a un elemento Object.
 */

import javax.xml.crypto.*;
import javax.xml.crypto.dsig.*;
import javax.xml.crypto.dom.*;
import javax.xml.crypto.dsig.dom.DOMSignContext;
import javax.xml.crypto.dsig.keyinfo.*;
import javax.xml.crypto.dsig.spec.C14NMethodParameterSpec;

import java.io.*;

import java.security.*;
import java.util.Arrays;
import java.util.Collections;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Node;

/*
 * El fichero XML firmado tendrá el siguiente formato:
 *
 * <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 *   <SignedInfo>
 *     <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
 *     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
```

```
 *     <Reference URI="#object">
 *       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
 *       <DigestValue>7/XTsHaBSOnJ/jXD5v0zL6VKYsk=</DigestValue>
 *     </Reference>
 *   </SignedInfo>
 *   <SignatureValue>
 *     RpMRbtMHLa0siSS+BwUpLIEmTfh/0fsld2JYQWZzCzfa5kBTz25+XA==
 *   </SignatureValue>
 *   <KeyInfo>
 *     <KeyValue>
 *       <DSAKeyValue>
 *         <P>
 *           /KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0Imbz
 *           RMqzVDZkVG9xD7nN1kuFw==
 *         </P>
 *         <Q>
 *           li7dzDacuo67Jg7mtqEm2TRuOMU=
 *         </Q>
 *         <G>
 *           Z4RxsnqC9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4pNWMOH
 *           CBiNU0NogpsQW5QvnlMpA==
 *         </G>
 *         <Y>
 *           wbEUaCgHZXqK4qLvbdYrAc6+Do0XVcsziCJqxzn4cJJRxwc3E1xnEXHscVgr1Cql9
 *           i5fanOKQbFXzmb+bChqig==
 *         </Y>
 *       </DSAKeyValue>
 *     </KeyValue>
 *   </KeyInfo>
 *   <Object Id="object">Texto</Object>
 * </Signature>
 *
 */

public class GenEnveloping {

      // En esta clase no se define el constructor por defecto.
```

```
public void firmar(String input, String output) throws Exception {

      // Instanciamos un objeto de la factoría de documentos y fijamos el
      // espacio de nombres
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
      dbf.setNamespaceAware(true);

      // Creamos el objeto DOM XMLSignatureFactory que será usado para
      // generar el elemento XMLSignature
      String providerName = System.getProperty("jsr105Provider",
          "org.jcp.XML.dsig.internal.dom.XMLDSigRI");

      XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");

      // Creamos la URI que apunta a un elemento Reference dentro del
      // mismo documento que es un elemento Object y se especifica el
      // algoritmo de resumen SHA1
      Reference ref = fac.newReference("#pedido",
          fac.newDigestMethod(DigestMethod.SHA1, null));

      // Creamos el elemento referenciado Object
      Document XML = dbf.newDocumentBuilder().parse(new File(input));
      Node pedido = XML.getDocumentElement();
      XMLStructure content = new DOMStructure(pedido);
      XMLObject obj = fac.newXMLObject(Collections.singletonList(content),
          "pedido", null, null);

      // Crea el elemento SignedInfo
      SignedInfo si = fac.newSignedInfo(fac.newCanonicalizationMethod(
                CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS,
                (C14NMethodParameterSpec) null),
                fac.newSignatureMethod(SignatureMethod.DSA_SHA1, null),
                Collections.singletonList(ref));

      PrivateKey privateKey = null;
```

```
            // Creamos un par de claves DSA
            KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
            kpg.initialize(512);
            KeyPair kp = kpg.generateKeyPair();

            // Guardamos por separado la clave privada del par de claves
            privateKey = kp.getPrivate();

            KeyInfo ki = null;

            // Crea un elemento KeyValue conteniendo la clave pública DSA que fue generada
            KeyInfoFactory kif = fac.getKeyInfoFactory();
            KeyValue kv = kif.newKeyValue(kp.getPublic());

            // Crea un elemento KeyInfo y añade el elemento KeyValue a él
            ki = kif.newKeyInfo(Collections.singletonList(kv));

            // Crea el elemento XMLSignature (pero no se firma todavía)
            XMLSignature signature = fac.newXMLSignature(si, ki,
                Collections.singletonList(obj), null, null);

            // Creamos un objeto DOMSignContext y especificamos la clave
            // privada DSA para firmar y la localización del documento que
            // contendrá la firma XMLSignature
            Document doc = dbf.newDocumentBuilder().newDocument();
            DOMSignContext dsc = new DOMSignContext(privateKey, doc);

            // Por último firmamos usando la clave privada
            signature.sign(dsc);

            // Generamos el documento
            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer trans = tf.newTransformer();
            trans.transform(new DOMSource(doc), new StreamResult(new FileOutputStream(output)));
        }
}
```

```java
/**
 * @autor Fernando J. Pérez Borrero
 *
 * Clase GenDetached.java
 *
 * Crea una firma XML en el formato Detached usando
 * la JSR 105 API.
 *
 */

import javax.xml.crypto.*;
import javax.xml.crypto.dsig.*;
import javax.xml.crypto.dom.*;
import javax.xml.crypto.dsig.dom.DOMSignContext;
import javax.xml.crypto.dsig.keyinfo.*;
import javax.xml.crypto.dsig.spec.C14NMethodParameterSpec;

import java.io.*;

import java.security.*;
import java.util.Collections;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;

/*
 * El fichero XML firmado tendrá el siguiente formato:
 *
 * <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 *   <SignedInfo>
 *     <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 *     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
 *     <Reference URI="http://www.w3.org/TR/xml-stylesheet">
 *       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```
 *       <DigestValue>60NvZvtdTB+7UnlLp/H24p7h4bs=</DigestValue>
 *     </Reference>
 *   </SignedInfo>
 *   <SignatureValue>
 *     DpEylhQoiUKBoKWmYfajXO7LZxiDYgVtUtCNyTgwZgoChzorA2nhkQ==
 *   </SignatureValue>
 *   <KeyInfo>
 *     <KeyValue>
 *       <DSAKeyValue>
 *           <P>
 *           rFto8uPQM6y34FLPmDh40BLJ1rVrC8VeRquuhPZ6jYNFkQuwxnu/wCvIAMhukPBL
 *           FET8bJf/b2ef+oqxZajEb+88zlZoyG8g/wMfDBHTxz+CnowLahnCCTYBp5kt7G8q
 *           UobJuvjylwj1st7V9Lsu03iXMXtbiriUjFa5gURasN8=
 *         </P>
 *         <Q>
 *           kEjAFpCe4lcUOdwphpzf+tBaUds=
 *         </Q>
 *         <G>
 *           oe14R2OtyKx+s+60O5BRNMOYpIg2TU/f15N3bsDErKOWtKXeNK9FS7dWStreDxo2
 *           SSgOonqAd4FuJ/4uva7GgNL4ULIqY7E+mW5iwJ7n/WTELh98mEocsLXkNh24HcH4
 *           BZfSCTruuzmCyjdV1KSqX/Eux04HfCWYmdxN3SQ/qqw=
 *         </G>
 *         <Y>
 *           pA5NnZvcd574WRXuOA7ZfC/7Lqt4cB0MRLWtHubtJoVOao9ib5ry4rTk0r6ddnOv
 *           AIGKktutzK3ymvKleS3DOrwZQgJ+/BDWDW8kO9R66o6rdjiSobBi/0c2V1+dkqOg
 *           jFmKz395mvCOZGhC7fqAVhHat2EjGPMfgSZyABa7+1k=
 *         </Y>
 *       </DSAKeyValue>
 *     </KeyValue>
 *   </KeyInfo>
 * </Signature>
 */

public class GenDetached {

    // En esta clase no se define el constructor por defecto.
```

```
public void firmar(String URI, String output) throws Exception {

      // Creamos el objeto DOM XMLSignatureFactory que será usado para
      // generar el elemento XMLSignature
      String providerName = System.getProperty
                  ("jsr105Provider", "org.jcp.xml.dsig.internal.dom.XMLDSigRI");

      XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM",
            (Provider) Class.forName(providerName).newInstance());

      // Create a Reference to an external URI that will be digested
      // using the SHA1 digest algorithm
      Reference ref = fac.newReference(URI,
                        fac.newDigestMethod(DigestMethod.SHA1, null));

      // Crea el elemento SignedInfo
      SignedInfo si = fac.newSignedInfo(fac.newCanonicalizationMethod(
                              CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS,
                              (C14NMethodParameterSpec) null),
                              fac.newSignatureMethod(SignatureMethod.DSA_SHA1, null),
                              Collections.singletonList(ref));

      // Creamos un par de claves DSA
      KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
      kpg.initialize(512);
      KeyPair kp = kpg.generateKeyPair();

    // Crea un elemento KeyValue conteniendo la clave pública DSA que fue generada
      KeyInfoFactory kif = fac.getKeyInfoFactory();
      KeyValue kv = kif.newKeyValue(kp.getPublic());

      //Crea un elemento KeyInfo y añade el elemento KeyValue a él
      KeyInfo ki = kif.newKeyInfo(Collections.singletonList(kv));

      // Crea el elemento XMLSignature (pero no se firma todavía)
      // Se puede hacer después de DOMSignContext como en la
      // clase GenEnveloped.java
```

```
            XMLSignature signature = fac.newXMLSignature(si, ki);

            // Crea el documento que contendrá el resultado de la firma XMLSignature
            // El espacio de nombres es obligatorio
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setNamespaceAware(true);
            Document doc = dbf.newDocumentBuilder().newDocument();

            // Creamos un objeto DOMSignContext y fijamos la clave de firma privada
            // DSA. Especificamos el lugar donde se debe insertar la firma XMLSignature
        // en el documento resultante, en este caso, como elemento raíz.
            DOMSignContext dsc = new DOMSignContext(kp.getPrivate(), doc);

            // Por último firmamos usando la clave privada
            signature.sign(dsc);

            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer trans = tf.newTransformer();
            trans.transform(new DOMSource(doc), new StreamResult(new FileOutputStream(output)));
    }
}
```

```java
/**
 * @autor Fernando J. Pérez Borrero
 *
 * Clase Validar.java
 *
 * Se encarga de comprobar la veracidad e integridad de la firma
 * de un fichero XML en cualquiera de los tipos de formatos posibles.
 * Se usa la JSR 105 API.
 * Se asume que la clave necesaria está contenida en el elemento
 * KeyValue KeyInfo.
 *
 */

import javax.swing.*;

import javax.xml.crypto.*;
import javax.xml.crypto.dsig.*;
import javax.xml.crypto.dom.*;
import javax.xml.crypto.dsig.dom.DOMValidateContext;
import javax.xml.crypto.dsig.keyinfo.*;

import java.io.FileInputStream;

import java.security.*;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

public class Validar {

      JTextArea areaTexto;

      public void valida(String input, JTextArea areaTexto) throws Exception {
```

```
this.areaTexto = areaTexto;

// Instancia el documento para ser validado
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
Document doc =
      dbf.newDocumentBuilder().parse(new FileInputStream(input));

// Busca el elemento Signature
NodeList nl =
    doc.getElementsByTagNameNS(XMLSignature.XMLNS, "Signature");
if (nl.getLength() == 0) {
      areaTexto.setText("No se encuentra el elemento Signature");
    throw new Exception("No se encuentra el elemento Signature");
}

// Crea un objeto DOM XMLSignatureFactory que será usado para
// desorganizar (unmarshal) el documento que contiene XMLSignature
String providerName = System.getProperty
      ("jsr105Provider", "org.jcp.xml.dsig.internal.dom.XMLDSigRI");
XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM",
      (Provider) Class.forName(providerName).newInstance());

// Creamos un objeto DOMValidateContext y especificamos un elemento
// KeyValue KeySelector y un contexto de documento
DOMValidateContext valContext = new DOMValidateContext
    (new KeyValueKeySelector(), nl.item(0));

// Unmarshal del elemento XMLSignature
XMLSignature signature = fac.unmarshalXMLSignature(valContext);

// Valida el elemento XMLSignature (generado más arriba)
boolean coreValidity = signature.validate(valContext);

// Comprueba el estado de validación
if (coreValidity == false) {
```

```
            areaTexto.setText("La firma XMLSignature de "+input+" NO es válida\n");
          boolean sv = signature.getSignatureValue().validate(valContext);
            areaTexto.append("\nEstado de validación de la firma: " + sv);

          // Comprueba el estado de validación de cada elemento Reference
          Iterator i = signature.getSignedInfo().getReferences().iterator();
          for (int j=0; i.hasNext(); j++) {
            boolean refValid =
                ((Reference) i.next()).validate(valContext);
            areaTexto.append("\nEstado de validación elemento Ref["+j+"]: " + refValid);
          }
      } else {
                  areaTexto.setText("La firma XMLSignature de "+input+" es válida");
      }
    }

    /**
     * KeySelector localiza la clave pública dentro del elemento KeyValue
     * y la devuelve como resultado.
     * NOTA: si la clave del algoritmo no coincide con el algorimo de firma
     * entonces la clave pública se ignora.
     */

    private static class KeyValueKeySelector extends KeySelector {
      public KeySelectorResult select(KeyInfo keyInfo,
                                      KeySelector.Purpose purpose,
                                      AlgorithmMethod method,
                                      XMLCryptoContext context)
            throws KeySelectorException {
            if (keyInfo == null) {
                        throw new KeySelectorException("Objeto KeyInfo NULL");
            }
            SignatureMethod sm = (SignatureMethod) method;
            List list = keyInfo.getContent();

            for (int i = 0; i < list.size(); i++) {
            XMLStructure xmlStructure = (XMLStructure) list.get(i);
```

```
                if (xmlStructure instanceof KeyValue) {
                  PublicKey pk = null;
                  try {
                      pk = ((KeyValue)xmlStructure).getPublicKey();
                  } catch (KeyException ke) {
                      throw new KeySelectorException(ke);
                  }
                  // Hay que asegurarse que el algoritmo es compatible
                            // con el método
                  if (algEquals(sm.getAlgorithm(), pk.getAlgorithm())) {
                      return new SimpleKeySelectorResult(pk);
                  }
                      }
                }
            throw new KeySelectorException("No se encuentra ningun elemento KeyValue");
      }

    // Esto debería funcionar para claves distintas de DSA/RSA
      static boolean algEquals(String algURI, String algName) {
            if (algName.equalsIgnoreCase("DSA") &&
            algURI.equalsIgnoreCase(SignatureMethod.DSA_SHA1)) {
            return true;
            } else if (algName.equalsIgnoreCase("RSA") &&
                        algURI.equalsIgnoreCase(SignatureMethod.RSA_SHA1)) {
            return true;
            } else {
            return false;
            }
      }
    }

    private static class SimpleKeySelectorResult implements KeySelectorResult {
      private PublicKey pk;
      SimpleKeySelectorResult(PublicKey pk) {
          this.pk = pk;
      }
```

```
      public Key getKey() { return pk; }
    }
}
```

```java
/**
 * @autor Fernando J. Pérez Borrero
 *
 * Clase Ayuda.java
 *
 */

import javax.swing.*;

 public class Ayuda {

    // En esta clase no se define el constructor por defecto.

      public void howto(JTextArea areaTexto) {

      areaTexto.setText("En el menú firmar existen 3 posibles opciones para firmar un XML:\n");
      areaTexto.append("\n- En el formato Enveloped se solicita un documento XML de entrada cualquiera.");
      areaTexto.append("\n  En ese mismo directorio se creará el fichero de salida firmado.\n");
      areaTexto.append("\n- En el formato Enveloping se solicita un documento XML de entrada cualquiera.");
      areaTexto.append("\n  En ese mismo directorio se creará el fichero de salida firmado.\n");
      areaTexto.append("\n- En el formato Detached se solicita un recurso externo identificado por una URI.");
      areaTexto.append("\n  Por lo tanto no hay documento XML de entrada pero sí se crea un documento XML");
      areaTexto.append("\n  firmado en el mismo directorio que el intérprete de Java.\n");

      areaTexto.append("\nEl menú validar es común para comprobar la autenticidad de la firma de cualquier documento
                       XML firmado.\n");
      }

      public void about(JTextArea areaTexto) {

      areaTexto.setText("Firma digital de documentos XML v1.0\n");
      areaTexto.append("\n @autor: Fernando J. Pérez Borrero");
      }
}
```