5. Desarrollo de la aplicación de Firma Digital XML

Como trabajo de investigación y desarrollo en ingeniería que es este proyecto, no podríamos concluirlo sin desarrollar una idea. Así pues, este capítulo se dedica a explicar los pormenores de la tarea encomendada: la programación de una aplicación de firma digital que siga la recomendación analizada en el capítulo 3.

Después de ver algunas de las características para el diseño de la aplicación nos adentraremos de lleno en la explicación de los procesos de firma y validación, estructurándolos en pasos y detallando paralelamente las líneas de código que vamos usando en estas etapas.

En el cuarto apartado, nos centramos en el entorno de desarrollo y en probar la aplicación con una batería de ejemplos que cubre todos los casos posibles. Se escogerá un documento XML de prueba, y se firmará en los tres formatos posibles viendo en qué consiste cada uno y validándolos a posteriori.

En el quinto apartado encontramos una breve reseña de la Ley Española para la Firma Electrónica y los requisitos para que sea legalmente reconocida.

En la sexta sección describimos con detalle algunos de los proveedores de aplicaciones para la firma electrónica y otros servicios muy relacionados con ella. Es de esperar que después de estar siete años en vigor la ley española, hayan aparecido muchas iniciativas públicas y privadas que exploten las posibilidades que de la ley se derivan. Será de mucha utilidad el estudio de estas aplicaciones comerciales, ya que nos dará una idea de posibles mejoras para la nuestra gracias a que usan nuevas tecnologías y estándares recientes que serán de utilidad en un futuro.

5.1. Requisitos de diseño de la aplicación

El cliente que se va a programar es una herramienta elaborada en Java, con librerías de la API pertenecientes a la JSR 105 para la firma digital de documentos en formato XML. Dichas librerías ya forman parte del JDK estándar (kit de desarrollo que proporciona Java).

Tiene como característica principal que se ejecuta en el ordenador del cliente y no lanzado desde un entorno web, como podría ser posible si programásemos un *applet* de firma servido para el cliente en un entorno distribuido.

A modo de resumen sus características son:

- Firma documentos XML en formatos *enveloped* y *enveloping* y *detached*.

- Ejecución en el entorno del cliente.

- Sigue el formato XMLDSig (XML Signature Syntax and Processing propuesto por el W3C).

- Validación.

- Guardar el resultado de las operaciones a petición del usuario.

- Programado en Java siguiendo la especificación JSR 105 y W3C.

5.2. Proceso de creación de firmas con la API JSR 105

Al igual que hicimos en el apartado 3.7., ahora vamos a detallar los pasos que hay que seguir para firmar digitalmente con las clases de los paquetes de la API ya mencionada.

Algunos de estos pasos se pueden cambiar de orden. El único requisito es que no se haya procedido a realizar la firma.

Usaremos como ejemplo explicativo el código Java de la clase `GenEnveloped.java`, esto es, una firma en formato detached. Dicho código se encuentra disponible en el siguiente capítulo.

5.2.1. Instanciar el documento que se va firmar

Opcionalmente, y como hemos hecho en la clase `GenEveloping.java`, podríamos haber empezado creando en primer lugar un objeto `DocumentBuilderFactory` para manipular y tratar archivos XML. Es un objeto de JAXP (Java API for XML Processing), que no es más que una API de java para el procesado de XML.

En la clase `GenEnveloped.java` lo hemos hecho después de manipular los elementos de la firma (`KeyInfo`) para ver que es posible hacer pequeños cambios en el código.

Así pues, este paso puede ir a continuación del apartado 8.2.3. y empezar el largo proceso de ensamblaje de la firma XML antes que con la instancia del documento.

Las líneas de código pertenecientes a este paso son las siguientes:

```
DocumentBuilderFactory dbf =
  DocumentBuilderFactory.newInstance();
```

También advertimos del uso de espacios de nombres con:

```
dbf.setNamespaceAware(true);
```

A continuación usamos la factoría `dbf` para conseguir una instancia de `DocumentBuilder` que es usada para analizar (*parse*) el documento:

```
Document doc = dbf.newDocumentBuilder().parse(new File(input));
```

5.2.2. Ensamblar la firma XML

Ensamblamos las diferentes partes del elemento Signature dentro del objeto `XMLSignature`. Estos objetos se crean todos y se ensamblan usando un objeto `XMLSignatureFactory`.

Una aplicación obtiene una implementación DOM de `XMLSignatureFactory` declarando a la siguiente líneas de código:

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

A continuación invocamos varios métodos para crear las diferentes partes del objeto XMLSignature. Creamos un objetos Reference pasándole lo siguiente:

- La URI del objeto a firmar. Cuando especificamos "", implica el documento raíz.
- El método de resumen (`DigestMethod`) SHA-1.
- Una transformación, enveloped `Transform`, necesaria en este tipo de firma de tal forma que se elimine ella misma después de calcular la firma.

```
Reference ref = fac.newReference("",
  fac.newDigestMethod(DigestMethod.SHA1, null),
  Collections.singletonList(fac.newTransform
  (Transform.ENVELOPED, (TransformParameterSpec) null)),
  null, null);
```

A continuación creamos un objeto `SignedInfo`, que es el objeto que realmente se firma, como se muestra a continuación. Cuando creamos `SignedInfo` pasamos como parámetros:

- El método de canonización.
  `CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS`
- El algoritmo de firma. `SignatureMethod.DSA_SHA1`
- Y una lista de referencias (`Reference`), en este caso sólo una.

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

Lo siguiente es crear el objeto opcional `KeyInfo`, que contiene información que habilita el recipiente para encontrar la clave necesaria para luego validar la firma. En nuestro caso, añadimos un objeto `KeyValue` que contenga la clave pública. Para crear `KeyInfo` y sus varios subtipos, usamos un objeto `KeyInfoFactory`, que puede ser obtenido invocando el método `getKeyInfoFactory` de `XMLSignatureFactory` como sigue:

```
KeyInfoFactory kif = fac.getKeyInfoFactory();
```

Luego usamos `KeyInfoFactory` para crear el objeto `KeyValue` y añadirlo al objeto `KeyInfo`:

```
KeyValue kv = kif.newKeyValue(kp.getPublic());
KeyInfo ki = kif.newKeyInfo(Collections.singletonList(kv));
```

Finalmente, creamos el objeto `XMLSignature`, pasándole los parámetros de los objetos `SignedInfo` y `KeyInfo` que hemos creado con anterioridad:

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

Es importante volver a recordar en este punto todavía no se ha generado la firma, se hará luego.

Este paso se puede dar incluso en cuarto orden, justo antes de la firma digital, aunque en nuestro ejemplo de código `GenEnveloped.java` aparece en primer lugar.

5.2.3. Crear una pareja de claves pública-privada

Generamos la pareja de claves con `KeyPairGenerator`. Recordemos que se firma con la privada. En nuestro código creamos una pareja de claves DSA `KeyPair` con una longitud de 512 bits con:

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
kpg.initialize(512);
KeyPair kp = kpg.generateKeyPair();
```

Esta es una forma ficticia de crear claves que sólo vale a título ilustrativo. Realmente, las privadas se generan y guardan en un almacén de claves (un fichero del sistema, por ejemplo) asociadas a certificados de claves pública.

La herramienta keytool de Java permite generar claves privadas y guardarlas en su almacén de claves para su posterior uso en programas.

En el apéndice B damos un pequeño repaso a esta herramienta, que habría que usar para la aplicación del Servicio de Autenticación Web mediante DNIe. Como los documentos que firmamos aquí son meramente explicativos de la tecnología XMLDSig, nos basta con usar claves genéricas aunque no estén asociadas a información de certificados, simplemente que sean válidas para firmar y validar.

Puede aparecer en cualquier orden, siempre antes de firmar lógicamente, aunque lo natural sería que lo hiciese justo antes de crear el contexto de firma que vemos a continuación.

5.2.4. Crear un contexto de firma

Se suele dejar para el final, justo antes de proceder a la firma. Pero también podemos adelantar este paso incluso antes de ensamblar porque no tiene nada que ver con la firma en sí.

Lo único que se hace es crear un objeto `XMLSignContext` con parámetros de entrada para generar la firma.

Al usar DOM instanciamos un objeto `DOMSignContext` (subclase de `XMLSignContext`) pasándole dos parámetros, la clave privada que será usada para firmar el documento y el nodo raíz del documento que se firma:

```
DOMSignContext dsc = new DOMSignContext
  kp.getPrivate(), doc.getDocumentElement());
```

5.2.5. Crear la firma digital XML

Necesariamente esta operación va después de las cuatro anteriores, que como hemos visto se podían alternar en su orden, aunque se han enumerado de la forma más lógica posible.

Con todo lo anterior estamos preparados para generar la firma, que se consigue invocando el método `sign` del objeto XMLSignature pasándole el contexto de firma como sigue.

```
signature.sign(dsc);
```

El documento XML resultante contiene ahora la firma, que ha sido insertada como último nodo hijo del raíz.

5.2.6. Generar la salida del documento resultante XML

Creamos un fichero de salida con las siguientes órdenes:

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

5.3. Proceso de validación de firmas con la API JSR 105

En este apartado vamos a validar el documento XML firmado usando una implementación DOM de la JSR 105 para análisis.

El código lo encontramos en `Validar.java`.

5.3.1. Instanciar el documento que contiene la firma

Usamos un objeto JAXP `DocumentBuilderFactory` para analizar el documento que contiene la firma. La implementación por defecto de `DocumentBuilderFactory` se consigue con:

```
DocumentBuilderFactory dbf =
  DocumentBuilderFactory.newInstance();
```

También advertimos del uso de espacios de nombres con:

```
dbf.setNamespaceAware(true);
```

A continuación usamos la factoría `dbf` para conseguir una instancia de `DocumentBuilder` que es usada para analizar (*parse*) el documento:

```
Document doc = dbf.newDocumentBuilder().parse(new File(input));
```

5.3.2. Especificar el elemento `Signature` *para validar*

Necesitamos especificar el elemento Signature que queremos verificar porque podrí existir más de uno en el documento. Usamos el método DOM `getElementsByTagNameNS` pasándole la URI del espacio de nombres de la firma digital y el nombre de la etiqueta del elemento `Signature` como sigue:

```
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");

  }
```

Se devuelve una lista con todos los elementos `Signature` del documento. En nuestros ejemplos sólo hay un elemento `Signature`. Sino fuera así se lanzaría la excepción correspondiente, imprimiendo un mensaje indicador del error.

5.3.3. Crear un contexto de validación

Creamos una instancia que contenga los parámetros de entrada para validar la firma con el objeto `XMLValidateContext`. Como usamos DOM instanciamos una subclase de `XMLValidateContext`, `DOMValidateContext` a la que pasamos dos parámetros, `KeyValueKeyselector` y la referencia al elemento `Signature` que se va a validar (que es el primer elemento de `NodeList`).

```
DOMValidateContext valContext = new DOMValidateContext (new
  KeyValueKeySelector(), nl.item(0));
```

`KeyValueKeySelector` es explicado con más detalle en el apartado 8.3.7.

5.3.4. Unmarshalling de la firma XML

Se denomina *marshalling* al proceso de tomar una colección de items de datos y transformarlos en una forma adecuada para transmitirlos en un mensaje. Unmarshalling es el proceso contrario; revertir la transformación a la llegada del dato para producir una colección equivalente de items de datos en el receptor. Así, marshalling consiste en la traducción de items de datos estructurados y valores primitivos en una representación externa de datos. Del mismo modo, unmarshalling consiste en generar valores primitivos a partir de la representación externa de los datos y la reconstrucción de las estructuras de datos.

Se trata de extraer el contenido del elemento `Signature` en un objeto `XMLSignature`. Ese contenido de `Signature` se extrae usando un objeto `XMLSignatureFactory`. Una aplicación puede obtener una implementación DOM de `XMLSignatureFactory` con las líneas de código:

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

Luego llamamos al método `unmarshalXMLSignature` de la factoría para conseguir la nueva representación (unmarshal) del objeto `XMLSignature`, pasándole el contexto de validación creado anteriormente.

```
XMLSignature signature = fac.unmarshalXMLSignature(valContext);
```

5.3.5. Validar la firma digital XML

Ya estamos preparados para comprobar la autenticidad de la firma. Se realiza invocando el método `validate` del objeto `XMLSignature` con el parámetro del contexto de validación:

```
boolean coreValidity = signature.validate(valContext);
```

El método `validate` devuelve "true" si la firma se valida con éxito de acuerdo a las reglas del W3C, sección 3.2 (Core Validation) de la recomendación ya conocida, o "false" en cualquier otro caso.

5.3.6. ¿Qué ocurre si la firma no se valida?

Si el método `XMLSignature.validate` retorna "false", podemos intentar concretar la causa del fallo.

Hay dos fases en la validación de la firma XML (`core validation`):

- `Reference validation` (la verificación del resumen de cada referencia en la firma).
- `Signature validation` (la verificación criptográfica de la firma).

Para que la firma sea considerada válida se debe satisfacer cada fase. Para comprobar si la firma es válida criptográficamente podemos comprobar su estatus como sigue:

```
boolean sv = signature.getSignatureValue().validate(valContext);
areaTexto.append("\nEstado de validación de la firma: " + sv);
```

También podemos iterar sobre las referencias y comprobar el estatus de cada una:

```
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);

}
```

5.3.7. Uso de `KeySelectors`

`KeySelector` se usa para encontrar y seleccionar las claves necesarias para validar una firma XML.

Con anterioridad hemos creado el objeto `DOMValidateContext` al que le pasamos como primer argumento un objeto `KeyValueKeySelector`:

```
DOMValidateContext valContext = new DOMValidateContext (new
  KeyValueKeySelector(), nl.item(0));
```

Alternativamente podríamos haber pasado una clave pública `PublicKey` como primer argumento si ya sabíamos qué clave es necesaria para validar la firma. Sin embargo, todavía no lo sabemos.

`KeyValueKeySelector` es una implementación concreta de la clase abstracta `KeySelector`. La implementación `KeyValueKeySelector` trata de encontrar una clave para la validación de los datos contenidos en los elementos `KeyValue` del elemento `KeyInfo`. No determina si la clave es de confianza.

Esta es una implementación muy simple, diseñada para ilustrar más que para uso real. Un ejemplo más práctico de `KeySelector` es aquél que busca en el almacén de claves `KeyStore` por claves seguras pertenecientes a información `X509Data`. Por ejemplo, los elementos `X509SubjectName`, `X509IssuerSerial`, `X509SKI`, o `X509Certificate` (todos ellos contenidos en `KeyInfo`).

El código de `KeyValueKeySelector` es el siguiente:

```
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; }

          }

        }
```

5.4. Entorno de desarrollo y pruebas de la aplicación

En esta sección ejecutaremos el programa y realizaremos pruebas funcionales del mismo. Aprovecharemos para estudiar los resultados, que no son más que ficheros XML firmados en los distintos formatos.

5.4.1. JSR 105API

Para el desarrollo de la aplicación ha sido necesario hacer uso de esta especificación de Java, que actualmente ya se incluye en la versión estándar de Java 6 (con la antigua notación 1.6.x).

La librería consta de seis paquetes:

- `javax.xml.crypto`
- `javax.xml.crypto.dom`
- `javax.xml.crypto.dsig`
- `javax.xml.crypto.dsig.dom`
- `javax.xml.crypto.dsig.keyinfo`
- `javax.xml.crypto.dsig.spec`

El paquete `javax.xml.crypto` contiene clases comunes que se usan para implementar la operaciones criptográficas de XM, como generar una firma XML o cifrar los datos XML.

Hay una clase importante en el paquete, `KeySelector`, cuyo propósito es permitir a los desarrolladores proporcionar implementaciones que localizan y opcionalmente validan claves usando la información contenida en un objeto `KeyInfo`.

`javax.xml.crypto.dsig` incluye interfaces que representan los elementos de la especificación definidos en el estándar W3C XMLDSig. De suma importancia es la clase `XMLSignature`, que permite firmar y validar en formato XML. La mayoría de las estructuras o elementos de la firma XML son representados por su interfaz correspondiente (excepto las estructuras del elemento `KeyInfo`, que son incluidas en su propio paquete como veremos en el siguiente párrafo). Estas interfaces incluyen los elementos: `SignedInfo`, `CanonicalizationMethod`, `SignatureMethod`, `Reference`, `Transform`, `DigestMethod`, `XMLObject`, `Manifest`, `SignatureProperty`, y `SignatureProperties`. XMLSignatureFactory es una clase abstracta que se usa para crear objetos que implementan estas interfaces.

El paquete `javax.xml.crypto.dsig.keyinfo` contiene interfaces que representan la mayoría de las estructuras del elemento `KeyInfo` definidas en la recomendación XMLDSig, incluyendo `KeyInfo`, `KeyName`, `KeyValue`, `X509Data`, `X509IssuerSerial`, `RetrievalMethod`, and `PGPData`. The `KeyInfoFactory` class is an abstract factory that is used to create objects that implement these interfaces.

Por su parte `javax.xml.crypto.dsig.spec` contiene interfaces y clases que representan parámetros de entrada para las operaciones de resumen, firma, transformaciones, o algoritmos de canonización usados en el procesado de firmas XML.

Finalmente, los paquetes `javax.xml.crypto.dom` y `javax.xml.crypto.dsig.dom` contienen clases específicas de DOM para `javax.xml.crypto` y `javax.xml.crypto.dsig`. Los necesitamos porque usamos implementaciones de `XMLSignatureFactory` y `KeyInfoFactory` basados en DOM.

5.4.2. Compilación y ejecución de la aplicación

Para editar las clases de Java no hemos necesitado de ningún entorno de desarrollo integrado como Eclipse o NetBeans, nos ha bastado con un simple editor para lenguajes de programación como es Notepad++ y los javadocs de los paquetes citados anteriormente que están disponibles en la página de Oracle (apartado Java Security), que es quien distribuye los desarrollos de Sun Microsystems desde su compra el año pasado: http://download.oracle.com/javase/6/docs/.

Gracias a que los paquetes ya se encuentran disponibles en la actual versión de J2SE, no tenemos que registrar ningún Java Archive (.jar) adicional al classpath para advertirle al compilador de la existencia de las librerías de seguridad para la firma digital.

Para compilar el programa echamos mano de la línea de comandos de Windows.

El compilador `javac` y la máquina virtual `java` que incluye el JDK de Sun son los programas que necesitamos invocar. Están en el subdirectorio/bin. Las líneas necesarias para compilar y ejecutar el programa las vemos en la siguiente captura:

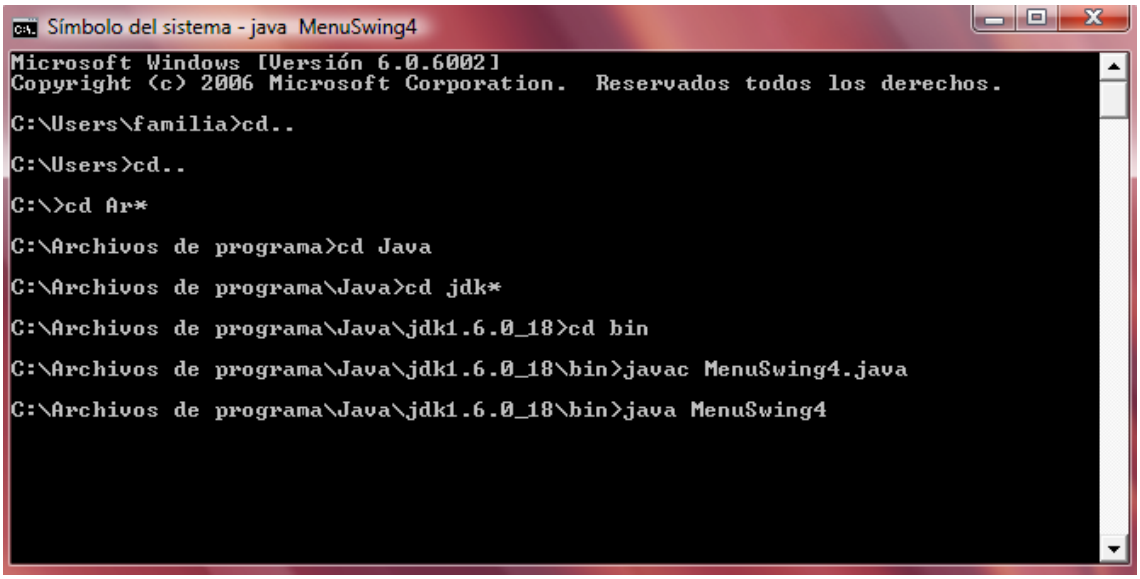

Figura 1: Compilación y ejecución del programa

En el momento de invocar la clase principal `MenuSwing4` aparece la siguiente ventana del programa:

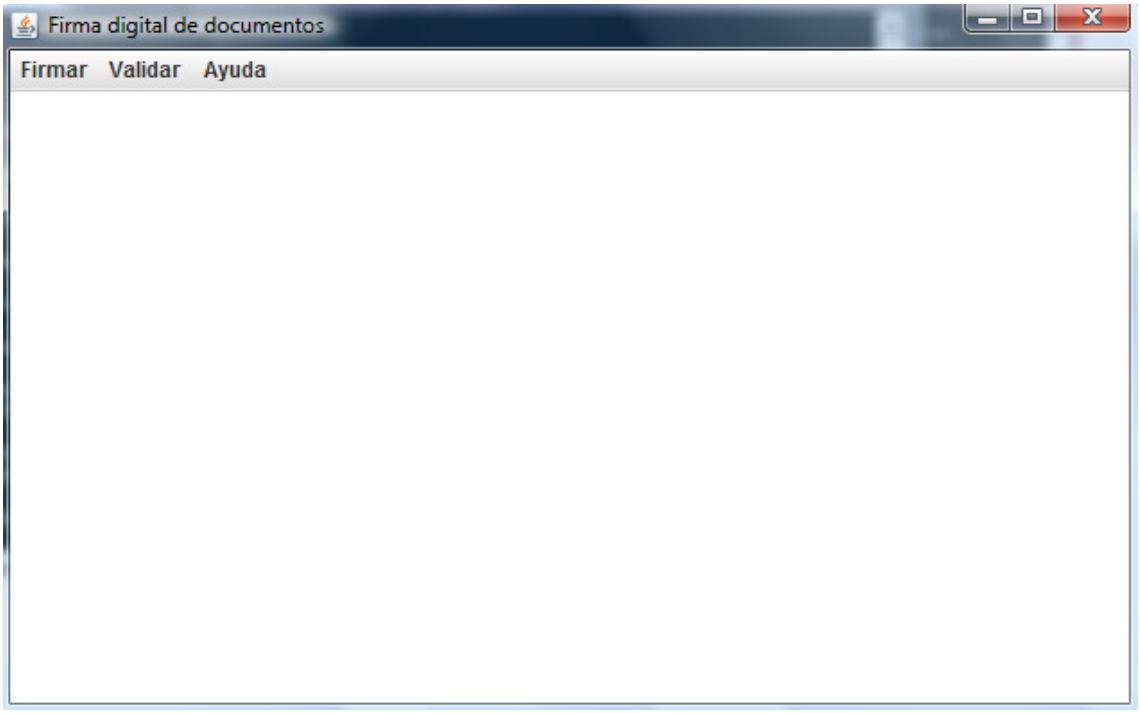

Figura 2: Ventana principal de la aplicación

En esta ventana distinguimos unos menús situados en la parte superior y un cuadro de texto en el que irán apareciendo los mensajes relativos a la firma y validación, así como la ayuda del programa.

En el menú Firmar se despliegan las opciones de firma en los tres formatos posibles:

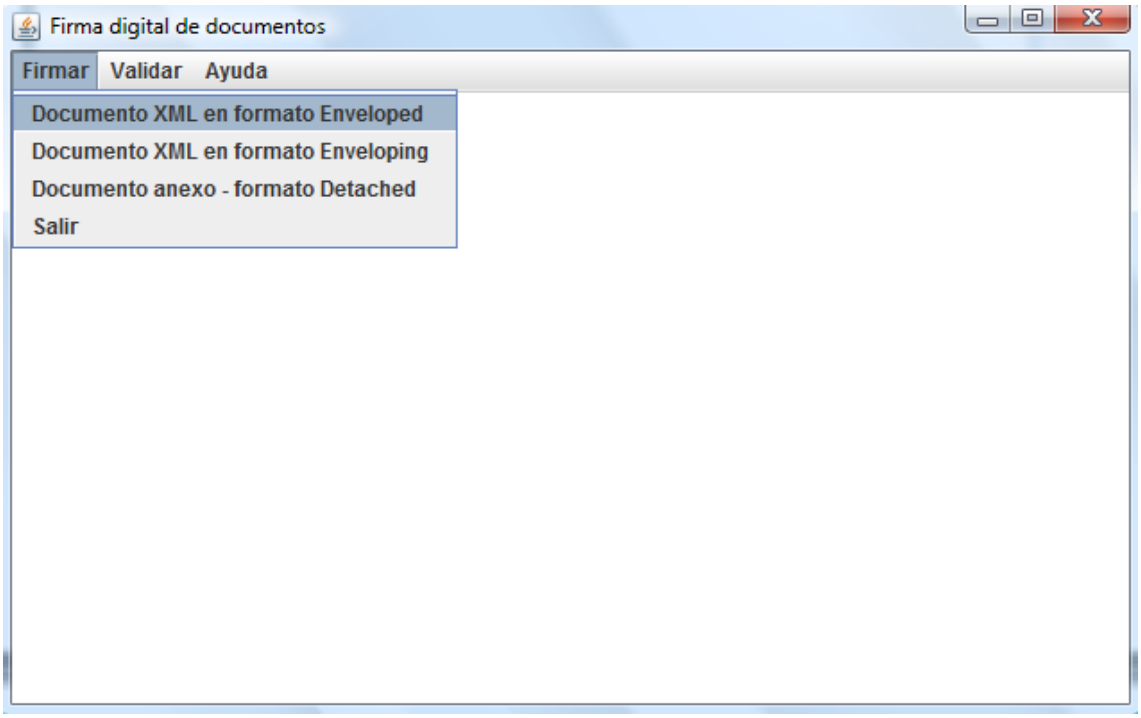

Figura 3: Menú Firmar

Con el menú Validar podemos ejecutar la clase que valida indistintamente los documentos XML en cualquiera de sus formatos:

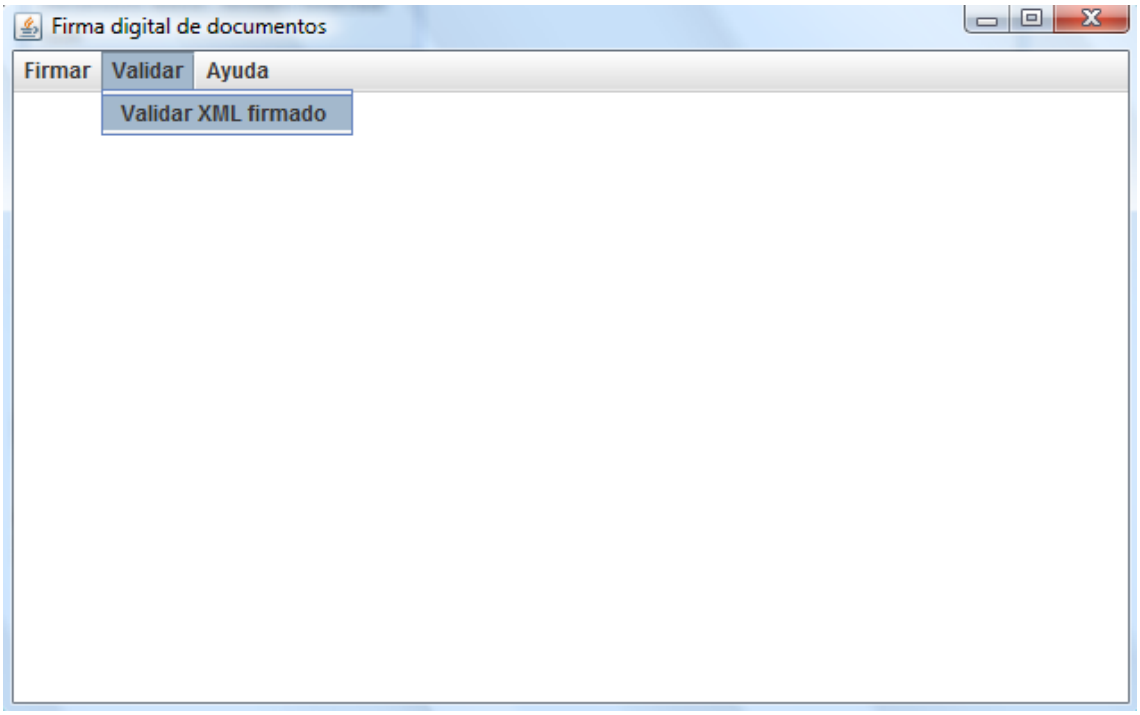

Figura 4: Menú Validar

En el menú Firmar existen 3 posibles opciones para firmar un XML:

- En el formato *Enveloped* se solicita un documento XML de entrada cualquiera. En ese mismo directorio se creará el fichero de salida firmado.

- En el formato *Enveloping* se solicita un documento XML de entrada cualquiera. En ese mismo directorio se creará el fichero de salida firmado.

- En el formato *Detached* se solicita un recurso externo identificado por una URI. Por lo tanto no hay documento XML de entrada pero sí se crea un documento XML firmado en el mismo directorio que el intérprete de Java.

El menú Validar es común para comprobar la autenticidad de la firma de cualquier documento XML firmado.

Esta misma información es la que encontramos en la Ayuda.

5.4.3. Ejemplos de prueba

Vamos a realizar la firma de un documento XML de ejemplo para ver qué salida genera y su posterior validación.

Si elegimos, por ejemplo, firmar en formato *Enveloped* o *Enveloping,* lo que nos parece por pantalla es un menú de diálogo del explorador de Windows con el que podemos buscar el XML a firmar.

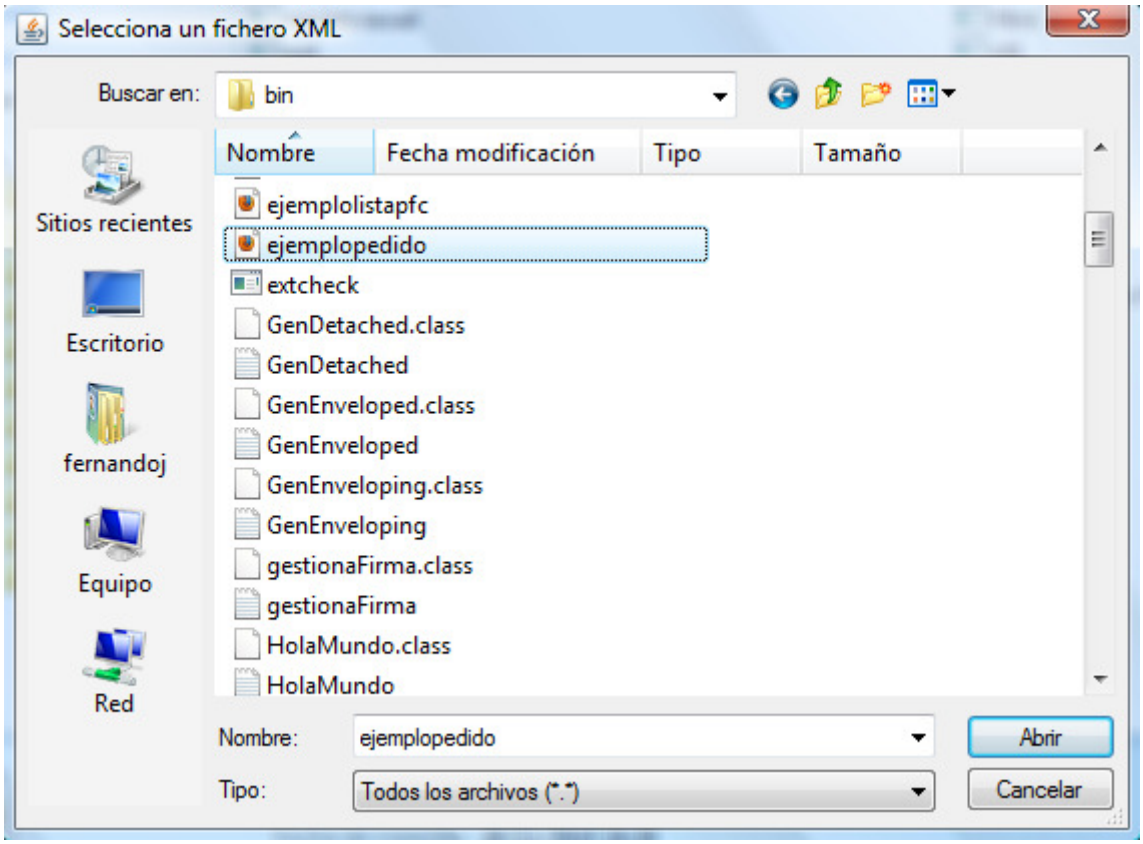

Figura 5: Ventana de diálogo con el Explorador de Windows

Una vez seleccionado el documento, en este caso `ejemplopedido.xml`, se nos pide un nombre para el fichero XML firmado. Escribimos, por ejemplo, `ejemplopedido_firmaenveloped.xml` y pulsamos enter.

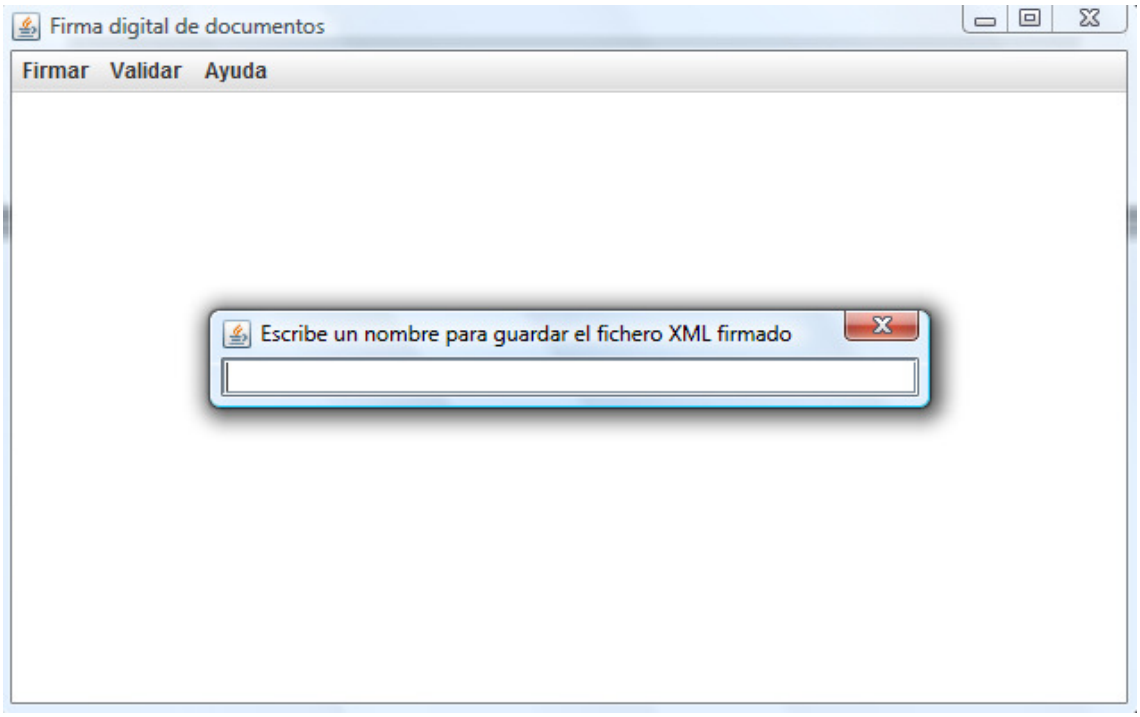

Figura 6: Ventana de diálogo que solicita un nombre para el XML firmado

Hecho esto, automáticamente se procede a firmar el mismo generando un documento con el nombre solicitado y mostrando información por pantalla. Por defecto se crean los archivos en el mismo directorio que los ficheros de entrada como vemos a continuación:

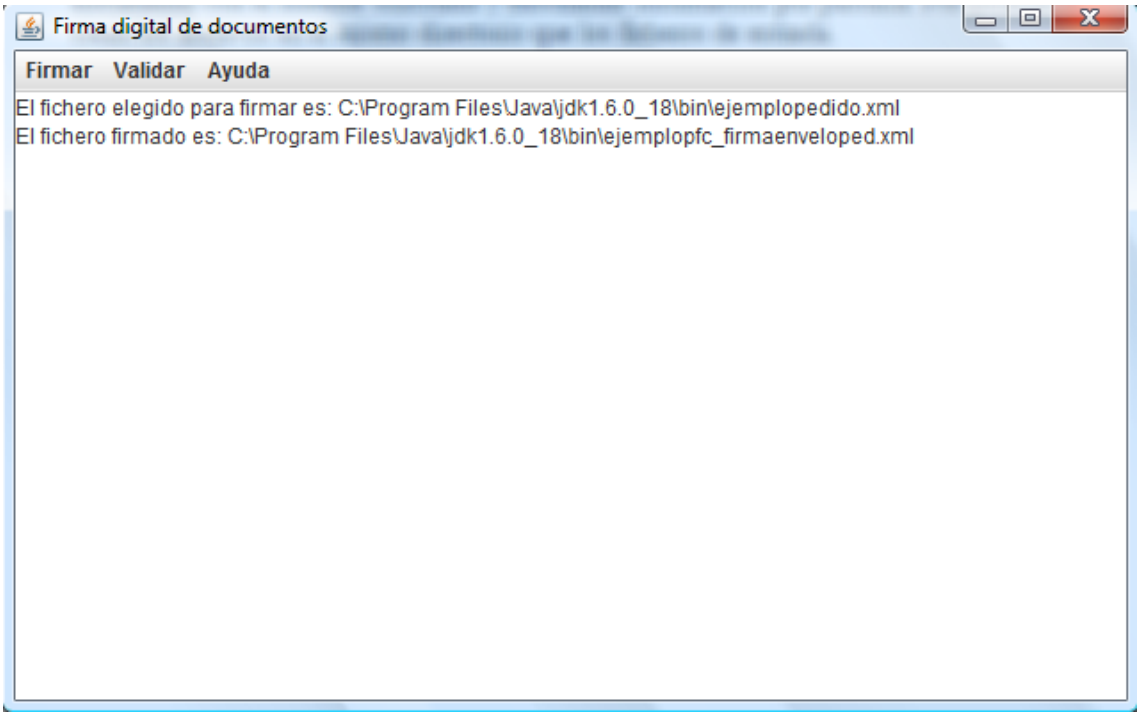

Figura 7: Ventana de diálogo que solicita un nombre para el XML firmado

Si hubiese habido algún tipo de error se genera el mensaje con la excepción correspondiente por pantalla y la línea de comandos.

A continuación mostramos el fichero de entrada `ejemplopedido.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<pedido XMLns="http://www.empresa.com/contabilidad">
    <items>
      <item>
        <desc>Java 2</desc>
        <tipo>libro</tipo>
        <precio>44.50</precio>
        <cantidad>1</cantidad>
      </item>
      <item>
        <desc>Interrail</desc>
        <tipo>guia-mapa</tipo>
        <precio>13.25</precio>
        <cantidad>2</cantidad>
      </item>
    </items>
    <tarjetacredito>
        <numero>1234567890</numero>
        <expira>03/03/2010</expira>
        <nombre>Sergio</nombre>
        <apellido>Prieto</apellido>
    </tarjetacredito>
```

Código 1: Documento XML *ejemplopedido.xml*

El resultado es el fichero firmado en el formato seleccionado, en este caso *Enveloped*.

```
<?xml version="1.0" encoding="UTF-8"?>
<pedido XMLns="http://www.empresa.com/contabilidad">
    <items>
      <item>
        <desc>Java 2</desc>
        <tipo>libro</tipo>
        <precio>44.50</precio>
        <cantidad>1</cantidad>
      </item>
      <item>
        <desc>Interrail</desc>
        <tipo>guia-mapa</tipo>
        <precio>13.25</precio>
        <cantidad>2</cantidad>
      </item>
    </items>
    <tarjetacredito>
        <numero>1234567890</numero>
        <expira>03/03/2010</expira>
        <nombre>Sergio</nombre>
        <apellido>Prieto</apellido>
    </tarjetacredito>
    <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="">
            <Transforms>
              <Transform
              Algorithm="http://www.w3.org/2000/09/XMLDSig#enveloped-
              signature"/>
              </Transforms>
            <DigestMethod
              Algorithm="http://www.w3.org/2000/09/XMLDSig#sha1"/>
            <DigestValue>OZo5NC4Ahx86VPk925SeoPOlGuE=</DigestValue>
          </Reference>
      </SignedInfo>
      <SignatureValue>
        A7EkjPoWIKSPF833iQjT6luWt+qQ9KZU0DAo8HwpmFwypmY2kKX+wg==
      </SignatureValue>
      <KeyInfo>
        <KeyValue>
          <DSAKeyValue>
            <P>
              /KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQT
              xeEu0ImbzRMqzVDZkVG9xD7nN1kuFw==
            </P>
            <Q>li7dzDacuo67Jg7mtqEm2TRuOMU=</Q>
            <G>
              Z4RxsnqC9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bp
              sy4pNWMOHCBiNU0NogpsQW5QvnlMpA==
            </G>
            <Y>
              LzksJt/2Opie5uXvfnb7DDztz9/8N2ahJZ0eOaOEH/vE/1plO5uUNrkC
              KIgCG0qUmrjV7Kh0Clib9rfkYanIQg==
            </Y>
          </DSAKeyValue>
        </KeyValue>
      </KeyInfo>
    </Signature>
</pedido>
```

Código 2: Documento XML *ejemplopedido_firmaenveloped.xml*

Vemos como efectivamente se añade un elemento `<Signature>` al último de los nodos del elemento raíz `<pedido>`, que es justamente en lo que consiste una firma *Enveloped*.

Si hubiésemos procedido igual (con el mismo fichero de entrada) para el caso del formato *Enveloping* tenemos como salida:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<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="#pedido">
      <DigestMethod
```

```
        Algorithm="http://www.w3.org/2000/09/XMLDSig#sha1"/>
      <DigestValue>Ra6TMSLhfMmV0I4uPU6SGqXWjCc=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>
    QnZ3Q/tQzWSKx75n4nkn+RS8W/8Li+Prh8XTBipZMyercxzaKR5Sog==
  </SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>
          /KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu
          0ImbzRMqzVDZkVG9xD7nN1kuFw==
        </P>
        <Q>li7dzDacuo67Jg7mtqEm2TRuOMU=</Q>
        <G>
          Z4RxsnqC9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4p
          NWMOHCBiNU0NogpsQW5QvnlMpA==
        </G>
        <Y>
          +xuPAKvxSbpTCYcCToYz1ww47UrOVDz4W14adR7wWqyinlUJw4kzCvdsqq34
          YM9dIg2gHVOUMJPdiEMVwG9abg==
        </Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
  <Object Id="pedido">
    <pedido XMLns="http://www.empresa.com/contabilidad" xmlns="">
      <items>
        <item>
          <desc>Java 2</desc>
          <tipo>libro</tipo>
          <precio>44.50</precio>
          <cantidad>1</cantidad>
        </item>
        <item>
          <desc>Interrail</desc>
          <tipo>guia-mapa</tipo>
          <precio>13.25</precio>
          <cantidad>2</cantidad>
        </item>
      </items>
      <tarjetacredito>
          <numero>1234567890</numero>
          <expira>03/03/2010</expira>
          <nombre>Sergio</nombre>
          <apellido>Prieto</apellido>
      </tarjetacredito>
    </pedido>
  </Object>
</Signature>
```

Código 3: Documento XML *ejemplopedido_firmaenveloping.xml*

El caso de las firmas en formato *Detached* es un poco distinto pues se necesita sólo una URI válida, ya lo que se firma es un recurso externo referenciado por esta URI como por ejemplo una URL:

```
http://www.w3.org/
```

La aplicación lo único que nos pide es la URI y el nombre para el documento firmado a la salida:

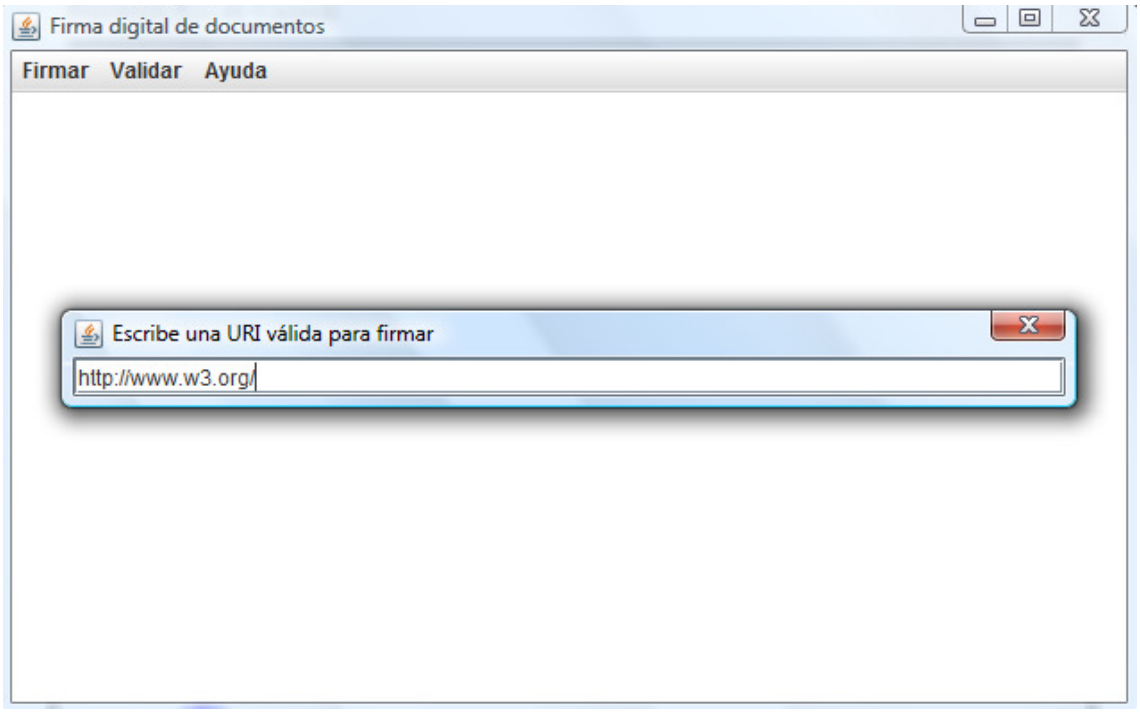

Figura 8: Ventana de diálogo que solicita una URI

En el caso del documento que se firma ahora, en el elemento `<Reference>` aparece como novedad la URI que hemos introducido:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<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="http://www.w3.org/">
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/XMLDSig#sha1"/>
      <DigestValue>kOvrgOAyfW5pP0dA5uNFAeXqZgM=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>
    Xzvat9TKUBxxwO5kcMwtrfCnf5mTG1KWeqIe5iVC7GuRdu7R6HggvA==
  </SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>
        /KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0I
        mbzRMqzVDZkVG9xD7nN1kuFw==
        </P>
        <Q>li7dzDacuo67Jg7mtqEm2TRuOMU=</Q>
        <G>
```

```
            Z4RxsnqC9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4p
            NWMOHCBiNU0NogpsQW5QvnlMpA==
          </G>
          <Y>
            N2Td2/MgPOTHx0bd116eBtH4rbTwpG5uuY/nLMoYWNseNbErxBuU8csolqAi
            R9s9BLLT0J44lvIydYZiKOYOew==
          </Y>
        </DSAKeyValue>
      </KeyValue>
    </KeyInfo>
</Signature>
```

Código 4: Documento XML *ejemplo_firmadetached.xml*

Para validar cualquiera de los documentos firmados hacemos uso del la única opción del menú Validar.

Nos sale de nuevo el Explorador de Windows y elegimos cualquiera de los tres documentos firmados, por ejemplo `ejemplopedido_firmaenveloped.xml`. Por pantalla, si la validación ha tenido éxito, saldrá un mensaje tal que así:

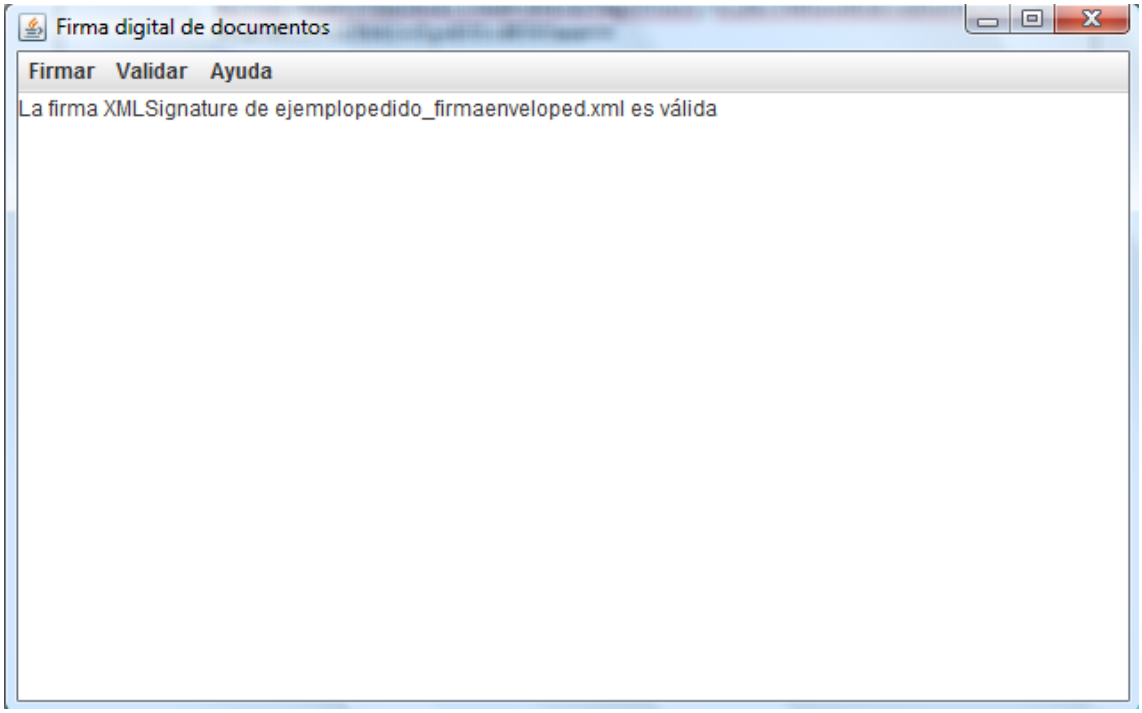

Figura 9: Validación exitosa de un documento XML firmado

En caso contrario, es decir, que la firma no se pueda validar por cualquier motivo, se indica el motivo de error.

La clase Validar, por tanto, no crea ningún fichero de salida.

Por último, sólo hacer notar que se puede firmar y validar (o volver a firmar) en otros formatos, sin necesidad de cerrar y reabrir de nuevo la aplicación. Sólo en el caso de error en una firma podría ser necesario volver a ejecutarla desde la línea de comandos.

5.5. Requisitos legales

No podíamos dejar sin estudiar algunos de los supuestos recogidos en la ley, ya que de ellos se derivan consecuencias técnicas. La Ley española 59/2003 de Firma Electrónica, define tres tipos de firma:

- Firma electrónica simple, que equivaldría a una firma manuscrita digitalizada: "la firma electrónica es el conjunto de datos en forma electrónica, consignados junto a otros o asociados con ellos, que pueden ser utilizados como medio de identificación del firmante".

- Firma electrónica avanzada, que sería la firma realizada con técnicas criptográficas de criptografía asimétrica: "(...) es la firma electrónica que permite identificar al firmante y detectar cualquier cambio ulterior de los datos firmados, que está vinculada al firmante de manera única y a los datos a que se refiere y que ha sido creada por medios que el firmante puede mantener bajo su exclusivo control."

- Firma electrónica reconocida, "se considera firma electrónica reconocida la firma electrónica avanzada basada en un certificado reconocido y generada mediante un dispositivo seguro de creación de firma. La firma electrónica reconocida tendrá respecto de los datos consignados en forma electrónica el mismo valor que la firma manuscrita en relación con los consignados en papel".

Por lo tanto, para que una firma electrónica tenga la misma validez que una firma manuscrita, son necesarios 3 elementos:

- Un formato de firma basada en tecnología de clave pública.

- Un certificado reconocido.

- Un dispositivo seguro de creación de firma.

La aplicación que hemos desarrollado nosotros sólo cumple el primero y tercero de los requisitos, ya que no usa certificados digitales para garantizar la identidad del firmante, sólo una pareja de claves pública-privada escogida al azar.

5.6. Proveedores españoles de tecnología PKI

Durante todos estos años en los que existe legislación específica, muchos organismos privados y públicos se han lanzado a explotar las tecnologías de clave pública. Han desarrollado aplicaciones de firma, con el mismo espíritu que la desarrollada por nosotros, pero con características y nuevos estándares más avanzados.

En Internet ya gozábamos de confidencialidad e integridad de los datos gracias a protocolos como SSL. Ya sólo quedaba terminar por dar autenticación para asegurar cualquier transacción.

Una entidad que use la infraestructura de clave pública está basada en cinco tipos de componentes:

1. Política de Seguridad. Define la dirección de máximo nivel de una organización sobre seguridad de información, así como los procesos y principios para el uso de la criptografía. Por lo general, incluye declaraciones sobre cómo gestionará la empresa las claves y la información valiosa, y establecerá el nivel de control requerido para afrontar los niveles de riesgo.

2. Autoridad de Certificación (CA). Algunos sistemas PKI se gestionan mediante terceras partes seguras, y por lo tanto, requieren un CPS. Éste es un documento en el que se detallan los procedimientos operativos sobre cómo ejecutar la política de seguridad y cómo aplicarla en la práctica. Por lo general, incluye definiciones sobre cómo se construyen y operan las Autoridades de Certificación, cómo se emiten, aceptan y revocan certificados, y cómo se generan, registran y certifican las claves, dónde se almacenan y cómo se ponen a disposición de los usuarios.

3. Autoridad de Registro (RA). Es la responsable de verificar el enlace entre los certificados (concretamente, entre la clave pública del certificado) y la identidad de sus titulares.

4. Sistema de Distribución de Certificados y Repositorios. Son los sistemas encargados de almacenar la información relativa a la infraestructura de clave pública. Los dos repositorios más importantes son el de certificados y el de listas de revocación. En una lista de revocación de certificados (en inglés CRL, *Certificate Revocation List*) se incluyen todos aquellos certificados que por algún motivo han dejado de ser válidos antes de la fecha establecida dentro del mismo certificado.

5. Aplicaciones basadas en clave pública. Cualquier servicio o aplicación como la desarrollada en este proyecto: firmas digitales, factura electrónica, etc.

El ejemplo más claro de organismo público que en nuestro país sostiene una infraestructura de clave pública es el Gobierno de España y Ministerio de Interior que hacen posible, como ya vimos en el capítulo 6, la implantación y uso del DNI electrónico. El trabajo de certificación corre a cuenta de CERES, que es un proyecto con página web de la FNMT (Fábrica Nacional de Moneda y Timbre) para la certificación española.

Luego, otros Ministerios como el de Industria, Turismo y Comercio desarrollan iniciativas paralelas, como la denominada Componentes de firma, que consiste en una serie de librerías desarrolladas en Java que proporcionan funcionalidad de firma digital.

También instituciones públicas como universidades están muy involucradas en el desarrollo de aplicaciones. Notorio es el caso de la *Universitat* Jaume I.

En el ámbito empresarial tenemos a grupos como Izenpe, C3PO, KSI Seguridad Digital, Viafirma, Notarline, Albalia Interactiva que proporcionan herramientas muy interesantes.

El Instituto Nacional de Tecnologías de la Comunicación ha desarrollado El Catálogo de Empresas y Soluciones de Seguridad TIC (http://www.inteco.es). Recoge de forma

sistemática y ordenada la oferta del mercado español de soluciones de seguridad TIC en España.

En él se anuncia la oferta de aplicaciones de seguridad del mercado en distintas categorías de productos y servicios. La consulta de las soluciones se puede realizar mediante la búsqueda directa, a través de las categorías, o utilizando la opción de "consulta asistida". Para cada producto o servicio se obtiene información descriptiva y técnica. El catálogo le ofrece además la posibilidad de realizar comparativas entre soluciones.

5.6.1. Componentes de firma (MITyC)

Componentes de firma es una iniciativa del Ministerio de Industria, Turismo y Comercio para potenciar el uso de la firma digital en la sociedad de la información. Consiste en una serie de librerías desarrolladas en Java que proporcionan funcionalidad de firma digital.

Actualmente las funcionalidades principales que están contempladas son:

- Acceso a almacenes de certificados.
- Consultas OCSP.
- Petición de sellos de tiempo.
- Validación de sellos de tiempo.
- Firma XAdES.
- Validaciones de firmas XAdES

Los componentes son las API siguientes:

- MITyCLibAPI. Esta librería contiene la funcionalidad base común al resto de componentes de firma del MITyC así como las definiciones de los interfaces que permite la comunicación, extensibilidad y adaptación de los componentes entre sí y con otros *frameworks*.
- MITyCLibCerts. Permite el acceso a almacenes de certificados. Estos accesos serán usados para realizar labores de criptografía como firma o comprobación de confianza.
- MITyCLibOCSP. Realiza la consulta del estado de los certificados mediante el protocolo OCSP (*Online Certifate Status Protocol*).
- MITyCLibTSA. Contiene la lógica necesaria para obtener y manejar sellos de tiempo de autoridades de sellado a través de una implementación que sigue la especificación RFC 2560 para tratar las peticiones y obtener dichos sellos de tiempo vía http teniendo en cuenta las normas descritas en el punto 3.4 de la especificación RFC 3161.

- MITyCLibTrust. Proporciona implementaciones concretas de validadores de confianza, de acuerdo con las interfaces definidas en MITyCLibAPI.

- MITyCLibPolicy. Implementa una serie de políticas XAdES. Actualmente se proporcionan las polícitas de FacturaE 3.0, de FacturaE 3.1, política de no transformadas y la política propia del MITyC.

- MITyCLibXAdES. Proporciona la funcionalidad necesaria para el manejo de ficheros con metadatos tipo XML, uso de certificados, y para la construcción de firmas electrónicas XAdES, así como para su validación. Todas las firmas generadas son de acuerdo a las especificaciones definidas por el ETSI (European Telecommunications Standards Institute).

Es especialmente interesante la librería MITyCLibXAdES puesto que cumple con los requisitos para la firma electrónica reconocida en la ley.

Figura 10: http://oficinavirtual.mityc.es/componentes/MITyCLibXADES/

Lo primero que nos llama la atención es que sigue el formato XAdES (recomendación *XML Advanced Electronic Signatures* del W3C), que ya especifica perfiles precisos de XMLDSig para ser usados con firma electrónica reconocida con el sentido de la directiva 1999/93/EC de la Unión Europea. Así pues todas las firmas generadas son de acuerdo a las especificaciones definidas por el ETSI.

XAdES define seis perfiles (formas) según el nivel de protección ofrecido. Cada perfil de firma extiende de los anteriores:

- XAdES-BES. Forma básica que simplemente cumple los requisitos legales de la Directiva para firma electrónica avanzada.

- XAdES-EPES. Forma básica a la que se la ha añadido información sobre la política de firma.

- XAdES-T (timestamp). Añade un campo de sellado de tiempo para proteger contra el repudio,

- XAdES-C (*complete*). Añade referencias a datos de verificación (certificados y listas de revocación) a los documentos firmados para permitir verificación y validación *off-line* en el futuro (pero no almacena los datos en sí mismos).

- XAdES-X (*extended*). Añade sellos de tiempo a las referencias introducidas por XAdES-C para evitar que pueda verse comprometida en el futuro una cadena de certificados.

- XAdES-X-L (*extended long-term*). Añade los propios certificados y listas de revocación a los documentos firmados para permitir la verificación en el futuro incluso si las fuentes originales (de consulta de certificados o de las listas de revocación) no estuvieran ya disponibles.

- XAdES-A (archivado). Añade la posibilidad de *timestamping* periódico (por ejemplo cada año) de documentos archivados para prevenir que puedan ser comprometidos debido a la debilidad de la firma durante un periodo largo de almacenamiento.

El Componente de firma se ajusta a los formatos *enveloped*, *enveloping*, *detached* y un tipo de firma mixta entre *enveloped* y *enveloping* puesto que la firma no contiene al nodo que se firma, ni tampoco es *enveloped*, puesto el nodo firmado no contiene a la firma.

Cada formato de firma se aplica sobre los demás siguiendo un orden jerárquico, y por lo tanto, para cada tipo de firma se aplicaría lo mencionado en los anteriores por este orden: BES- T – C .X – X-L. EPES sólo aporta información explícita a utilizar respecto a BES.

No ofrece soporte para XAdES-A. Existe abundante documentación con códigos de ejemplo tanto de firma como de validación de documentos.

La herramienta se distribuye bajo licencia *GNU Lesser General Public License* y la documentación bajo *Creative Commons*. Se instala muy fácil, incluyendo el JAR de la propia librería y todas sus dependencias en el *classpath* de la aplicación que va a usar esta librería.

5.6.2. @firma. Plataforma de validación y firma electrónica

@firma es la solución tecnológica en la que se basa la implementación de la Plataforma de validación y firma electrónica del Ministerio de Administraciones Públicas. Este proyecto promovido por el MAP se centra en facilitar a las aplicaciones los complementos de seguridad necesarios para implementar la autenticación y firma electrónica avanzada basada en certificados digitales. Se ofrecen así servicios que impulsan el uso de la certificación y firma electrónica en los sistemas de información de las diferentes administraciones públicas que así lo requieran.

Una de las medidas adoptadas de mayor trascendencia ha sido la sustitución gradual del Documento Nacional de Identidad por su versión electrónica.

De esta forma, la Plataforma cubre los siguientes objetivos:

- El objetivo primordial es impulsar la implantación de la firma-e y el DNIe mostrando una visión uniforme a los ciudadanos.

- Reducir los costes en licencias, soporte, infraestructuras, etc. por parte de los organismos usuarios.

- Tras la aprobación de la Ley 11/2007 de Acceso electrónico de los ciudadanos a los Servicios Públicos, con esta Plataforma se promueven y facilitan una serie de servicios destinados al cumplimiento de las obligaciones de las Administraciones para con los ciudadanos en materia de identificación y autenticación electrónica.

- Poder ofrecer los servicios a cualquier nivel de la Administración: local, autonómico o nacional.

- La solución técnica seleccionada es la más idónea para el ámbito de aplicación. Se basa en las siguientes premisas:

    - Se trata de una solución nada intrusiva a la hora de integrarse en arquitecturas y soluciones existentes en los organismos.
    - Además de gestionar la validación de los certificados del DNIe, está abierto a otros de diferentes Prestadores de Servicios de Certificación (PSC): MultiPSC, MultiPolítica, MultiCertificados, MultiFirma o MultiFormatos.
    - Igualmente, se establecerían redes de confianza paneuropeas: BRIDGE-CA. Ello estimula la creación de redes de confianza mutua entre los PSC acreditados por la Administración y los diferentes organismos públicos usuarios de estos servicios.
    - La administración y evolución del software de @firma será diseñada y articulada por los organismos usuarios.
    - La solución atesora las máximas garantías de seguridad y robustez: certificación de procesos y productos, óptimo rendimiento, alta disponibilidad, portabilidad,...

- El porfolio de servicios a ofrecer han de complementar las capacidades existentes en cada organización, no sustituirlas.

- El nivel de interoperabilidad y reusabilidad óptimo con sistemas.

@firma es una solución basada en software libre, estándares abiertos y en java: servidores web Apache, JBOSS, Sistema Operativo Solaris/Linux, AXIS, etc.

Además de determinar la validez de los certificados digitales, dispone de múltiples utilidades de valor añadido, entre las que se encuentran la generación y validación de firmas electrónicas en múltiples formatos, auditoría de las transacciones y documentos firmados, sellado de tiempos o la compatibilidad con certificados digitales generados por múltiples prestadores de servicios de certificación.

Para acceder a los servicios es necesario disponer de accesibilidad a la red SARA (Sistema de Aplicaciones y Redes para las Administraciones), que ofrece servicios de intranet entre las administraciones públicas. Estos sistemas que soportan los servicios de administración electrónica disponibles para los ciudadanos y empresas, accederán a la Plataforma a través de servicios web implementados en tecnología Microsoft® o Java.

5.6.2.1. Servicios ofrecidos por la Plataforma

Los servicios ofrecidos a los organismos se pueden catalogar en los bloques que se describen a continuación, basándose la mayoría en la publicación de un catálogo amplio de *web services* compatible con las tecnologías Java y .NET. Todos los *web services* se encuentran disponibles tanto en castellano como en inglés, facilitando así la integración e interoperabilidad con soluciones existentes en las implantaciones de los organismos usuarios.

Los bloques son los siguientes:

A) Servicios de validación.

Se corresponde con todos aquellos servicios orientados a obtener información de certificados y de la validación de certificados y firmas electrónicas. Entre los servicios de validación de certificados X.509 según la RFC 3280, se admite tanto el protocolo OCSP como la invocación de *web services* específicos.

Se permite realizar una validación completa de la firma electrónica proporcionada a la Plataforma. Por validación completa se entiende:

- Validación de la firma digital contenida en la firma electrónica frente a los datos proporcionados.

- Validación del certificado X.509 empleado y contenido en la firma electrónica. Se validará su integridad, periodo de validez y estado de revocación. Tanto el periodo de validez como el estado de revocación del certificado se comprueban frente a la fecha actual en caso que la firma electrónica no posea sello de tiempo o frente al mismo en caso contrario.

- Soporte del certificado. Se comprueba que el certificado y su emisor sean reconocidos y soportados por la plataforma. Este servicio puede ser empleado para validar tanto las firmas electrónicas generadas por la plataforma o el cliente de firma suministrado por el MAP como aquellas ajenas, siempre y cuando su formato sea soportado.

Los *web services* incluidos son:

- Validación de certificados.

- Obtención de información de certificados.

- Validación de firma electrónica en múltiples formatos: XMLDSig, XAdES, CMS...

- Validación de sellos de tiempo.

- Validar Firma Bloques completo.

- Validar Firma Bloques en documento.

En cuanto a la Autoridad de Sellado de Tiempo (TSA) se dispone del servicio web para verificar un sello de tiempo.

B) Servicios de firma electrónica.

Engloba todos los servicios relativos a la realización de firmas electrónicas por parte de la Plataforma. Dichas firmas pueden ser de la propia Plataforma o solicitudes de firmas de servidor de los organismos, a partir de certificados hospedados y firmas realizadas en dispositivos seguros de creación de firmas (eje. HSMs).

Este servicio permite a una aplicación cliente realizar, con el certificado de firma de servidor indicado, y siempre dentro del contexto de la plataforma @firma, una firma electrónica. Es decir, la generación de dicha firma se lleva a cabo en la plataforma, de ahí el nombre de firma electrónica servidor.

Es importante resaltar que hay 3 modos de realizar una firma electrónica en servidor:

- Firma electrónica servidor simple. Es el modo clásico. Se genera una firma electrónica con un formato determinado a partir de unos datos indicados.

- Firma electrónica servidor en paralelo (cosign). Consiste en, a partir de una firma electrónica existente, añadir un nuevo valor de firma al envoltorio que es la firma electrónica en sí.

- Firma electrónica servidor en cascada (countersign). En este tipo de firmas se realiza una firma digital sobre el valor de la firma digital de un firmante concreto, dentro del envoltorio de la firma electrónica. Es decir, es un proceso de aprobación de una firma existente por parte de otro firmante, el cual se incorpora a la misma firma electrónica.

A lo anterior hay que añadir que las firmas se pueden construir en varios formatos: PKCS#7, CMS (evolución de PKCS#7 realizada por el IETF que define un formato y una sintaxis de encapsulación de datos para firmar y cifrar documentos), XML Signature básico (XMLDSig), XML Signature avanzado (XAdES) y CMS avanzado (CAdES).

En cuanto a la Autoridad de sellado de tiempos (TSA) se dispone del servicio web para Solicitar sello de tiempo o un TimeStampReq si se realiza directamente en formato ASN.1.

C) Servicios auxiliares y de administración de transacciones.

Los servicios de este apartado se corresponden con aquellos considerados como subsidiarios de los de firma y validación. Permiten a los organismos disponer de una trazabilidad y auditabilidad completa de las operaciones realizadas en la Plataforma. Asimismo, se incluyen operaciones de custodia de transacciones y *hash* de documentos firmados.

5.6.2.2. Certificados reconocidos por la Plataforma

La plataforma @firma admite certificados digitales reconocidos conforme el estándar ITU-T X.509 v3, emitidos por múltiples prestadores de servicios de certificación. Todos los prestadores se encuentran inscritos en el registro de la Secretaria de Estado de Telecomunicaciones y para la Sociedad de Información del Ministerio de Industria, Turismo y Comercio de autoridades conforme a lo establecido en el artículo 30 de la Ley 59/2003, de 19 de diciembre, de firma electrónica.

5.2.2.3. Cliente de Firma

El cliente de firma es una utilidad para la firma digital de datos o ficheros. Tiene como característica principal que se ejecuta en el ordenador del cliente, no en el servidor web.

Permite la realización de las siguientes operaciones:

- Firma de formularios web (con y sin adjuntos).
- Firma de ficheros binarios.
- Firma masiva ficheros binarios.
- Firma multiformato: CMS, XMLDSig, XAdES, CadES.
- Co-firma (CoSign) o multifirma al mismo nivel.
- Contra-firma (CounterSign) o multifirma en cascada.
- Cifrado de datos y realización de sobre digital.

El cliente consta de un Cliente básico y Plugin XAdES que se descarga en el navegador del usuario (Internet Explorer y Mozilla Firefox) y se encarga de realizar la firma utilizando uno de los certificados dados de alta en dicho navegador. Este componente sólo se descarga una vez desde un servidor y se autoinstala en la máquina del usuario. Cualquier actualización se realizará sobre ya lo instalado.

5.6.3. CrytoAplett

CryptoApplet es una aplicación Java desarrollada en la *Universitat* Jaume I para la realización de firma electrónica avanzada. Su funcionamiento es muy simple. Dada una entrada de datos y una configuración definida en el servidor, un cliente web podrá realizar una firma digital sobre los datos, y devolverá como resultado una representación de la firma en el formato definido en la configuración.

Los formatos de representación de firma soportados por CryptoApplet son los siguientes:

- Firma "en bruto".
- CMS/PKCS#7.
- XAdES-X-L en formato DigiDoc.
- Firma PDF.

La gestión de los certificados con los que se va a firma se lleva a cabo de forma transparente para el usuario mediante el acceso directo al CryptoAPI si se utiliza Microsoft Internet Explorer o PKCS#11 si se usa Firefox (en Windows XP o GNU/Linux).

El applet también es capaz de ejecutarse en los sistemas operativos Microsoft Windows y GNU/Linux con el único requerimiento de tener instalada la Máquina Virtual Java de SUN (versión 1.5 o superior).

Algunas de las características:

- Lista de los certificados instalados en los navegadores de la familia Mozilla sin necesidad de solicitar la clave de acceso.

- Soporte para el DNI electrónico tanto en Intenet Explorer como en Mozilla.
- Menú de carga del controlador de dispositivo PKCS#11 que permite añadir nuevos dispositivos de firma instalados en el cliente.

- Archivo de configuración de firma en PDF que permite seleccionar los campos "Reason", "Location", "Contact" de la firma en PDF, así como los certificados raíz de las autoridades con las que se permitirá realizar la firma del documento PDF.

- Parametrización de la aplicación con JavaScript.

Una aplicación interesante es la firma de PDF con soporte para DNIe:

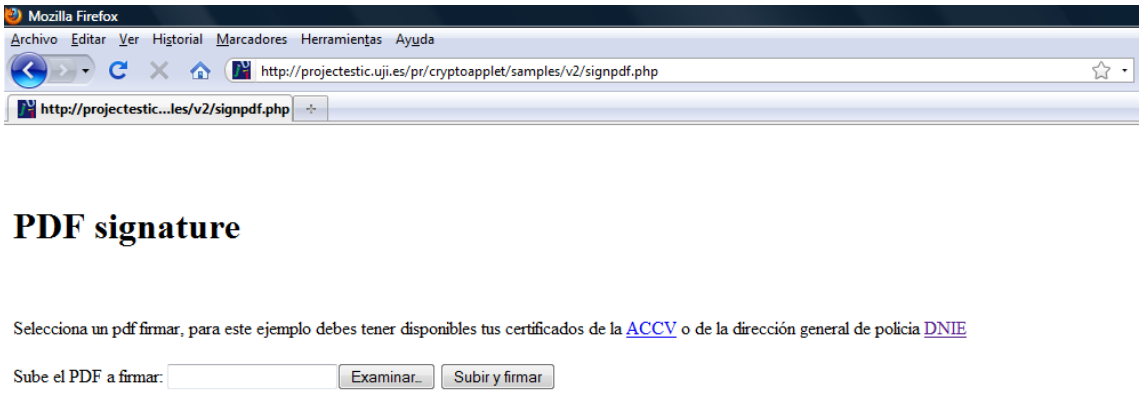

Figura 11: http://projectestic.uji.es/pr/cryptoapplet/samples/v2/signpdf.php

Se encuentra para libre descarga en http://forja.uji.es/frs/?group_id=24

5.6.4. Factura electrónica

La facturación electrónica es un equivalente funcional de la factura en papel y consiste en la transmisión de las facturas o documentos análogos entre emisor y receptor por medios electrónicos (ficheros informáticos) y telemáticos (de un ordenador a otro), firmados digitalmente con certificados reconocidos.

Las entidades de depósito (bancos, cajas de ahorros, cajas rurales y cooperativas de crédito), como sector de actividad económica, están interesadas en la facturación electrónica porque son emisores (por sus propias operaciones o en nombre de terceros) y receptores de facturas. Además, son el canal de gestión de cobro y financiación de gran número de transacciones comerciales, asociadas a instrumentos tales como cheques, efectos, recibos, etc.

Para aprovechar las ventajas que aporta la facturación electrónica, tales como, eliminación del papel, agilización de los trámites y disminución de los costes asociados al proceso de facturación, el sector financiero apoya la estandarización de este instrumento electrónico que todas las entidades de depósito puedan adoptar y utilizar en sus relaciones comerciales, tanto entre ellas como con sus clientes, inicialmente dentro del ámbito nacional, aunque sin perder de vista el entorno europeo.

El sector financiero, de común acuerdo con la AEAT, propone un formato estándar de factura electrónica dirigido a las entidades de depósito y, en general, a quienes estén interesados en adoptarlo y que reúne las siguientes características:

- Cumplimiento de los requisitos legales establecidos y de las condiciones establecidas por la A.E.A.T. para la facturación telemática. Las principales disposiciones normativas sobre la facturación electrónica se pueden consultar en esta página de la A.E.A.T.

- Universal: pueden utilizarlo todas las entidades de depósito y sus clientes.

- De libre uso: su utilización y posterior modificación no está sujeta al pago de cánones o a la autorización previa de una organización ajena al sector.

- Gratuito: se puede distribuir gratuitamente a las entidades de depósito y a sus clientes.

- Datos específicos de otros sectores económicos: Preparado para admitir información consensuada y relativa a otros sectores económicos.

- Información Factoring: Permite registrar, opcionalmente, datos específicos de Factoring.

- Formato flexible, de fácil implementación e interoprabilidad, por lo cual, se ha seleccionado el lenguaje XML.

Este formato de factura electrónica, nominado eFactura_AEAT-CCI, ha sido traspasado con el nombre de "Facturae" a la Administración Pública (Ministerio de Hacienda y Ministerio de Industria, Turismo y Comercio) que se han responsabilizado de su publicación, mantenimiento, evolución, divulgación y uso.

La Administración ha creado una web específica (www.facturae.es) donde se encuentra publicada toda la información referida al formato de factura electrónica facturae.

Se necesitan tres requisitos para la realización de la factura electrónica:

- Un formato electrónico de factura de mayor o menor complejidad (EDIFACT, XML, PDF, html, doc, xls, gif, jpeg o txt, entre otros).

- Es necesario una transmisión telemática (tiene que partir de un ordenador, y ser recogida por otro ordenador).

- Este formato electrónico y transmisión telemática, deben garantizar su integridad y autenticidad a través de una firma electrónica reconocida.

Dentro de la página, en el subapartado desarrollo de descargas, podemos encontrar una API que contiene la funcionalidad necesaria para gestionar facturas, dejando en manos de los usuarios la labor final de integración en sus aplicaciones propietarias.

En aplicaciones, tenemos el programa Gestión de Facturación Electrónica que permite de una manera cómoda y sencilla realizar facturas. Realizado con software abierto, se trata de un programa especialmente dirigido a PYMEs, microPYMEs y trabajadores autónomos.

5.6.5. Viafirma

Viafirma es una empresa dedicada el desarrollo de aplicaciones tecnológicas, centrándose especialmente en el desarrollo de soluciones de autenticación y firma electrónica.

En el margen inferior encontramos información sobre dos de los productos que ofrecen:

- Posibilidad de incorporar el DNIe en aplicaciones propias. Se podrá delegar en ella cuando se desee hacer que los usuarios firmen cualquier tipo de documento o comprobar la identidad mediante el DNIe.

- Soporte para firmar digitalmente facturas en cualquier formato, incluido facturae. Permite la custodia de las facturas realizadas, pudiendo ser recuperadas en cualquier momento desde una interfaz web y garantizando en todo momento que los documentos custodiados no han sido modificados.

5.6.6. Izenpe

Izenpe es una empresa prestadora de servicios de certificación, en el ámbito de las instituciones que integran el sector público vasco, de servicios de seguridad, técnicos y administrativos, en las comunicaciones a través de técnicas y medios electrónicos, informáticos y telemáticos.

Veamos algunos de los productos que desarrolla:

5.6.6.1. ZAIN

Ofrece un conjunto completo de servicios de confianza basados en infraestructuras de clave pública, los cuales pueden ser usados por cualquier tipo de consumidor, ya sea usuario final, aplicación u otro servicio.

- Autenticación y autorización. Intercambio de información de autenticación y autorización entre las aplicaciones corporativas y dominios de seguridad externos, haciendo posible el control de acceso único a nivel de web (SSO), mediante los estándares definidos por OASIS.

- Validación de certificados digitales. Reconocimiento de múltiples prestadores de servicios de certificación, uniformizando la información asociada a los certificados.

Soporta los mecanismos de validación de certificados estándares y admite la integración de cualquier otro mecanismo personalizado.

- Firma electrónica. Soporta la mayoría de los formatos de firma para documentos electrónicos, correo electrónico y mensajería web; incluyendo firmas múltiples, firmas con sello de tiempo y firmas longevas para poder validar la firma una vez transcurrido el periodo de vigencia de los certificados digitales.

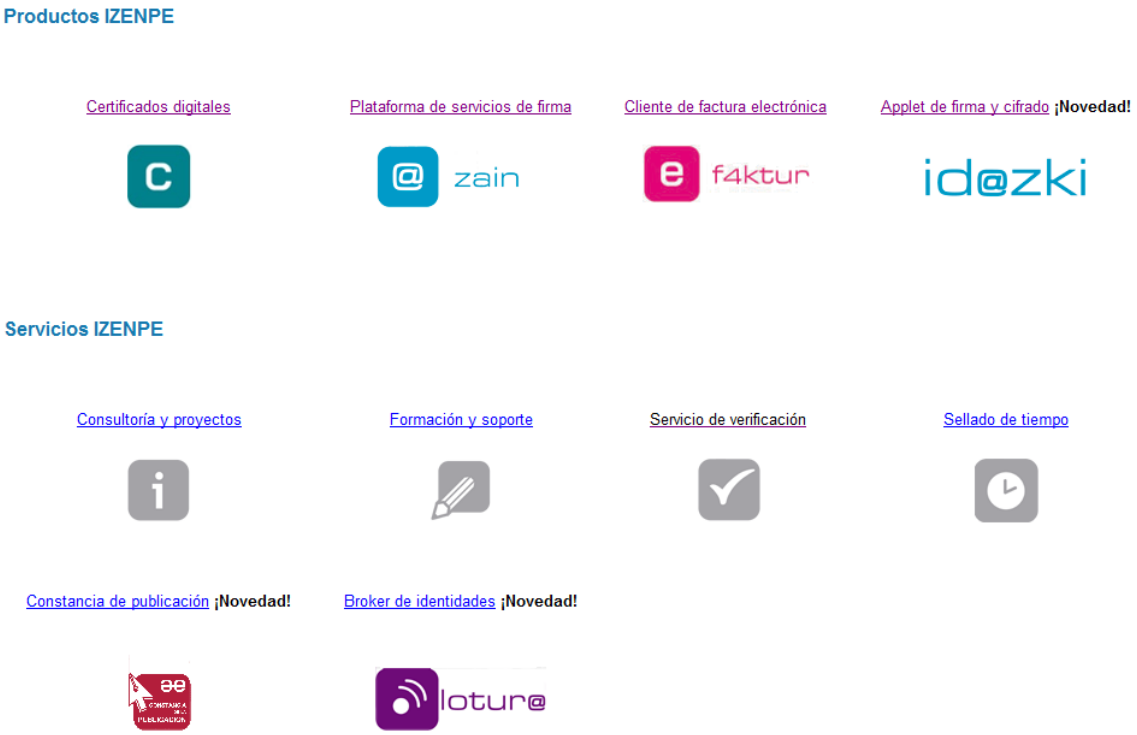

Figura 12: Catálogo de productos de Izenpe

- Cifrado de datos. Protección de la información mediante mecanismos de cifrado; ya sea documentos electrónicos, correo electrónico o mensajería web. En futuras versiones se incluirá el servicio de custodia de las claves de cifrado, controlando el acceso a los datos para los grupos de personas o sistemas de confianza.

- Pasarela de integración. Permite definir y conectar entre sí un conjunto de transformaciones sucesivas de datos XML, en interacción con los diferentes servicios de la plataforma, actuando de pasarela de confianza entre los procesos y aplicaciones o redes. Posibilita una integración de las aplicaciones con cero intrusiones.

- Auditoría y accounting. Servicio que gestiona de forma centralizada, uniforme y segura toda la información de traza generada por todos los componentes de servicio de la plataforma, así como la información de uso y/o consumo de los mismos.

- Gestión de objetos y entidades. Broker que ofrece una vista uniforme en XML de los objetos y entidades gestionados por la plataforma, enmascarando totalmente los formatos específicos de los datos (XML, ASN.1, Texto, etc.) y las diferentes fuentes

de información (LDAP, SQL, Ficheros, etc.) y permitiendo usarlos como servicio web.

5.6.6.2. ef4ktur

ef4ktur, es una iniciativa de las diputaciones forales vascas para dotar de forma gratuita a sus emisores de facturas, una solución de factura electrónica que permita de forma sencilla y práctica trabajar entre empresas y con las administraciones vascas.

El software de ef4ktur, ofrece autonomía de gestión en las operaciones telemáticas con garantías de seguridad y adicionalmente la posibilidad de adaptación a las necesidades funcionales del cliente.

La factura electrónica requiere de certificados de firma digital necesarios para poder emitir una factura para que esta tenga validez, así deberá contar con alguno de los certificados IZENPE admitidos bien en formato software o en tarjeta criptográfica.

ef4ktur es descargable desde su página web www.ef4ktur.com.

5.6.6.3. id@zki

Ofrece de forma gratuita el desarrollo de id@zki. Es una aplicación Java en forma de applet para poder ser integrada dentro de un navegador con funcionalidad de firma electrónica.

La versión de escritorio, sin embargo, tiene coste de licencia.

5.6.6.4. Servicio de verificación

Todo certificado de firma electrónica tiene caducidad. Además pueden darse casos de pérdida o robo en cuyo caso ese certificado habrá sido suspendido y/o revocado; tanto en el momento de la firma como a posteriori.

Aquí se ofrece un sistema on-line fácil y seguro por el que a través de OCSP (Online Certificate Status Protocol), y la conexión al servidor IZENPE se podrá consultar la validez de los certificados y el estado de los certificados usados en las aplicaciones.

Por cada certificado implicado en una transacción, la aplicación debe enviar al servidor de validación de certificados una petición de verificación de certificado recibiendo una respuesta firmada que indica su estado actual.

OCSP presenta múltiples ventajas sobre la validación basada en listas de revocación (CRL), entre las que cabe destacar:

- Evita la necesidad de distribuir periódicamente CRLs voluminosas y no totalmente actualizadas.

- Permite la verificación a posteriori de la validez de las transacciones si se almacenan las respuestas firmadas, sin necesidad de archivar todas las CRLs.

El uso de OCSP no excluye la utilización de CRL, sino que los dos sistemas se pueden complementar. Dado que la validación con OCSP requiere que el servidor de validación siempre esté disponible, es posible mantener la validación de los certificados contra CRL como un mecanismo alternativo.

5.6.6.5. Servicio de sellado de tiempo

Con frecuencia los trámites y procedimientos requieren constancia no sólo del estado de validez del certificado en el momento de la firma, sino también del momento en el que se han realizado. El servicio de sellado de tiempo es el elemento que le permite garantizar el día y la hora de cualquier operación. Este servicio de sello de tiempo ("time stamping" o "TSA") se ha diseñado siguiendo las recomendaciones y estándares internacionales:

Su uso es sencillo y se resume en 4 pasos básicos:

1. Normalmente se calcula un hash de los datos a sellar.

2. Se envía a la Autoridad de Sellado de Tiempo (TSA) de IZENPE.

3. La TSA genera un sello de tiempo con esta huella, la fecha y hora obtenida de una fuente fiable y la firma electrónica de la TSA.

4. El sello de tiempo se envía de vuelta al usuario.

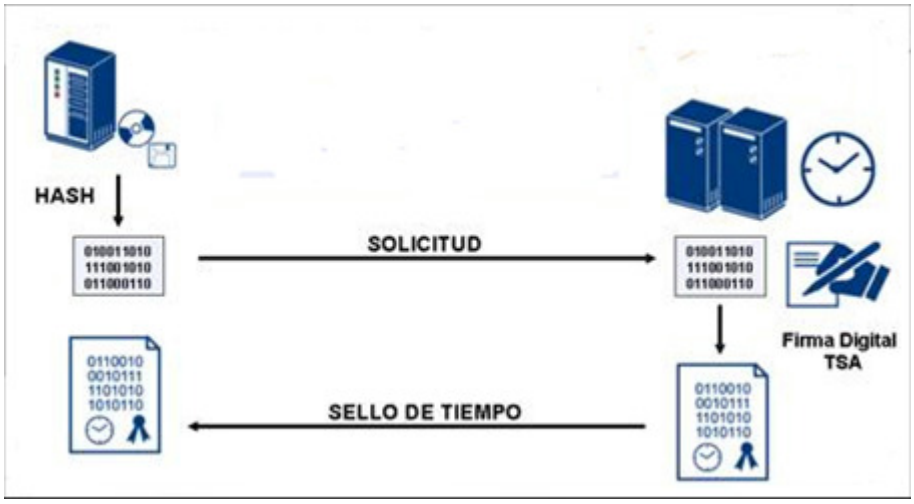

Figura 13: Sellado de tiempo

5.6.7. OpenDNI

Una iniciativa muy interesante que nos hemos encontrado disponible en la red es la promovida por la empresa Notarline denominada OpenDNI.

Ha puesto en funcionamiento un servicio gratuito de autenticación de usuarios con el DNI Electrónico (www.openDNI.com). El servicio ayudará a todas las empresas a mejorar la

seguridad de sus páginas web sustituyendo los habituales usuario y contraseña por métodos de autenticación más seguros. Tiene como principales objetivos:

1. Fomentar el uso del DNIe y otros métodos de autenticación fuerte para hacer de Internet un lugar más seguro tanto para los usuarios como para las empresas que ofrecen sus servicios o venden sus productos.

2. Facilitar a los *webmasters* que requieren autenticación de usuarios, un servicio de autenticación segura, listo para usar y probado que funciona con cualquier navegador y sistema operativo del mercado.

La solución OpenDNI se ofrece en modalidad SaaS (*Software as a Service*). Se trata por lo tanto de un servicio centralizado que no requiere de desarrollos por tu parte y que lo puedes integrar rápidamente en tu página web de forma inmediata, listo para usar.

El servicio verifica entonces:

- Que el usuario conozca el PIN de su DNIe.

- Que efectivamente su DNIe haya sido emitido por la policía.

- Que su DNIe está dentro del periodo de su validez y sigue vigente.

- Que su DNIe no haya sido revocado.

En caso de que la autenticación sea satisfactoria, devuelve a la aplicación web que llama al servicio información extraída del DNIe del usuario.

5.6.8. Otras empresas

Hasta ahora hemos dado un repaso profundo a las herramientas de instituciones públicas, que han sido de inspiración para las privadas. Éstas han tenido que adoptar los formatos promovidos por las administraciones públicas y mejorar e inventar servicios para captar clientes.

Son otras muchas la empresas que ofrecen servicios propios de firma digital en Internet. Entre ellas destacan:

- Albalia Interactiva

- KSI Seguridad Digital

- Isigma

- Camerfirma

5.7. Conclusiones

En el presente capítulo hemos desarrollado un prototipo de aplicación que sigue la recomendación estudiada en el capítulo 3.

La especificación de la Firma Digital XML (XMLDSig) elaborada por el W3C es posible implementarla para muchas plataformas y lenguajes, aunque sin duda el más atractivo para nosotros es Java porque proporciona códigos independientes de la plataforma y ya es conocido. Esto nos ha permitido disponer de un entorno de desarrollo libre y mucha documentación, con el ahorro que ello supone en costes y tiempo de diseño.

Que el estándar se base en XML nos proporciona datos también independientes de la plataforma, y si quisiéramos obtenerlos en otros formatos, las transformaciones XSL (XSLT) que forman parte de la hoja de estilo XML estándar resuelven el problema.

De esta forma, con un lenguaje y un modelo de datos independientes obtenemos un programa robusto, portable, de bajo coste y con abundante documentación para su desarrollo junto con información fácil de traducir a otros formatos y legible por humanos.

Para finalizar, hemos echado un vistazo a las soluciones y aplicaciones de algunos de los proveedores de la firma electrónica y otros servicios muy relacionados con ella tales como:

- Firma con autenticidad asegurada y sin repudio de documentos. En este apartado tenemos aplicaciones que cubren varios formatos como XMLDSig, XAdES, CadES, CMS, PDF, PKCS #7, etc. y varios entornos, desde aplicaciones de escritorio a aplicaciones en el servidor como los *applets* de firma.

- Aplicaciones que requieren sellado de tiempo:

    - Factura electrónica (formatos Facturae, EDIFACT, UBL, etc.

    - Voto electrónico.

    - Protección propiedad intelectual.

    - Dinero electrónico.

    - Gestión de procesos de administración pública: visados electrónicos, notificaciones judiciales electrónicas, registro electrónico, etc.

- Servicios pertenecientes a la infraestructura de clave pública: repositorios de certificados digitales, autoridades de certificación, autoridad de registro, autoridad de validación y autoridad de sellado de tiempo.

La integración en las empresas de estos servicios y aplicaciones relacionadas con la firma digital en los procesos de facturación, ha supuesto dependiendo de su modelo de negocio, que el ahorro por concepto de administración de facturas (recepción, almacenaje, búsqueda, firma, devolución, pago, envío, etc.) sea entre el 40% y el 80%. Entre los motivos se encuentran:

- Menor tiempo de recepción y el envío.
- Ahorro en el gasto de papelería.
- Mejora en los procesos de auditoría.
- Seguridad incrementada en el resguardo de los documentos.
- No hay posibilidad apenas posibilidad de falsificación.
- Agilidad en la localización de información.
- Reducción del espacio físico.
- Procesos con la administración más rápidos y eficientes.

Actualmente en España nos encontramos inmersos en un momento de profundo cambio en las transacciones electrónicas. Desde la Administración Estatal se promulga, a través del Plan Avanza del Ministerio de Industria, la obligatoriedad para finales de 2010 de la facturación electrónica, que servirá de impulso al uso de las Tecnologías de la Información y Comunicación (TIC) en el ámbito empresarial.

Si hasta ahora eran los certificados de la FNMT los que permitían a empresas, autónomos y ciudadanos su identidad digital, la llegada del DNIe le ha traído muchas facilidades de uso a estos últimos porque siempre pueden llevar un certificado digital consigo en forma de tarjeta inteligente sin necesidad de portar el certificado de la FNMT. Incluso con el DNIe la descarga del certificado de la FNMT es inmediato sin necesidad de personarse físicamente en ninguna institución, sólo visitando http://www.cert.fnmt.es/.

Así pues, vemos que muchas instituciones públicas han apostado por estas tecnologías desarrollando y poniendo a disposición de ciudadanos y empresas aplicaciones completas y API, para que dado el caso, puedan elaborar las suyas propias con los formatos que se ajustan a los formatos elegidos.

Nosotros, como ya hemos explicado, hemos escogido las librerías que proporciona Java, suficientes para la programación de firmas digitales, tanto XMLDSig como XAdES.