Capítulo 10
Aplicación Visor DICOM

10.1. Introducción

Hemos querido analizar esta aplicación, que fue la última que se desarrolló para nuestro proyecto, en un capítulo aparte porque en realidad se ha concebido como una herramienta de ayuda para visualizar archivos que siguen el formato definido por DICOM.

Como ya se vio en su momento, la aplicación cliente de los centros ambulatorios genera un archivo (con extensión *.dcm* y que responde a las especificaciones marcadas por el estándar DICOM) a partir de los datos personales y clínicos del paciente junto con una imagen digital de la lesión que lo llevó a la consulta. Este archivo se guarda en el ordenador desde el que se está haciendo la consulta en la carpeta *C:/dicoms/*, y será lo que se envíe a través de la red hacia el servidor de almacenamiento que se encuentra ubicado en la central del HUVR y cuyo estudio ya se presentó en el capítulo 9.

Esta nueva aplicación se ha creado para poder validar la aplicación cliente que se desarrolló para los centros ambulatorios, ya que los archivos DICOM no pueden ser visualizados con ningún programa de los que existen normalmente en cualquier terminal de usuario. De este modo y gracias a la aplicación VISOR podremos visualizar archivos DICOM de una imagen en color o en escala de grises así como los datos clínicos y personales del paciente que acompañan a dicha imagen y que fueron insertados mediante la aplicación cliente de los centros ambulatorios.

También podemos decir que esta nueva herramienta puede resultar muy útil cuando se desarrolle el servidor de nuestro sistema porque podría hacer innecesario el uso de un navegador Web para visualizar los archivos de imágenes de lesiones de los pacientes, pero esto lo veremos más detenidamente en el capítulo 11.

10.2. Aplicación VISOR

Para la implementación de este visor de archivos DICOM volvimos a utilizar JAVA como lenguaje de programación, JBuilder como entorno de desarrollo y las librerías JDT como herramienta de apoyo para el manejo de clases y tipos de datos para trabajar con DICOM.

En este caso volvemos a tener una interfaz gráfica de usuario dividida en dos partes: en la mitad superior se encuentra el panel en el que se visualizará la imagen de la lesión y los botones manejadores de eventos y en la mitad inferior se dispuso un *JTabbedPane* para ver los datos personales, clínicos y aquellos referentes a patologías intercurrentes que pudiesen hacer que el paciente no fuese apto par CmA en su modalidad de acto único. El diseño definitivo de la GUI lo podemos ver en la figura 10.1, y el código completo de la aplicación se encuentra en el anexo D.

Tenemos dos manejadores de eventos:

- *Botón Selección de Archivo*: Cuando se pulsa este botón, se abre un cuadro de diálogo de selección de archivos, se elige el archivo que se quiere abrir y es cuando aparecen en nuestro visor de imágenes y archivos DICOM la imagen de la lesión y los datos del paciente que fueron insertados en la aplicación cliente de los centros ambulatorios. Para ello, se implementó la clase *OyenteEventoSeleccion()*, que vamos a ver más detenidamente a continuación.
- *Botón Cancelar*: El oyente del evento asociado a este botón simplemente hace que salgamos de la aplicación como ya vimos en capítulos anteriores.

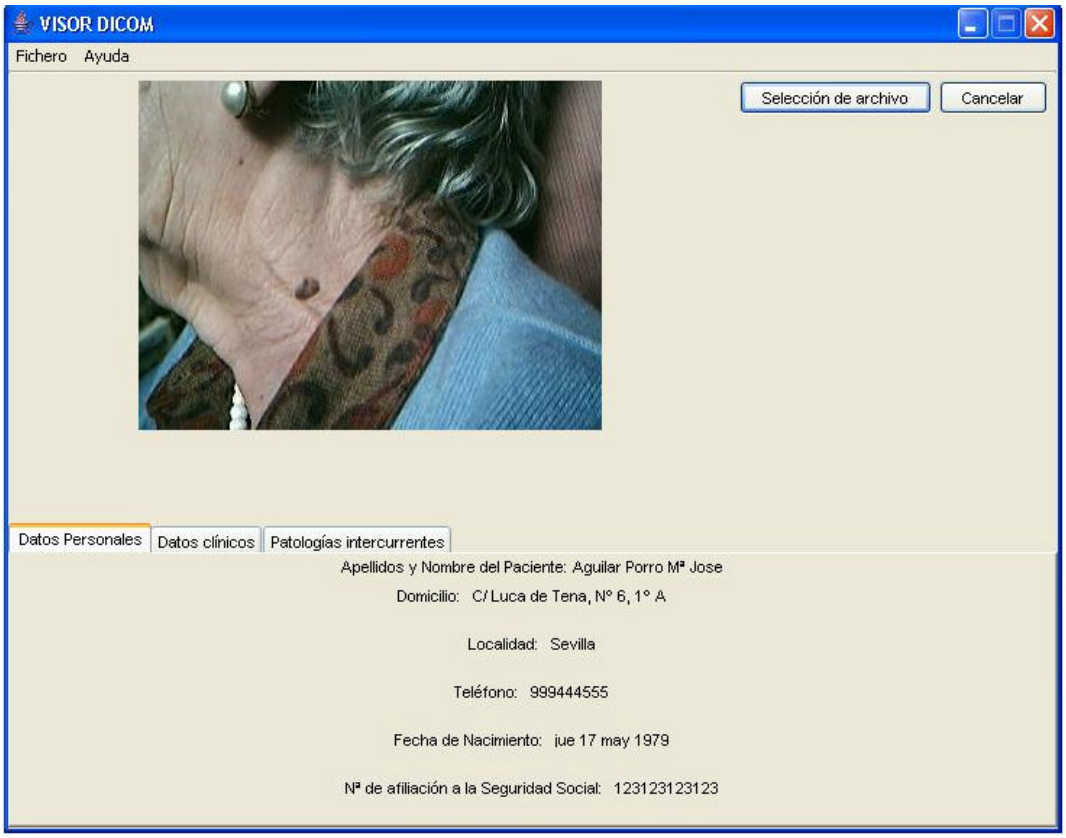

Figura 10.1. Diseño definitivo de la GUI del visor de archivos DICOM

Cuando seleccionamos el archivo DICOM que queremos abrir, lo que hará la clase que hemos mencionado será guardarlo primero en un objeto de la clase *DicomObject* y después irá extrayendo los datos pertinentes, tanto los de la imagen como los de texto. Pero vayamos por partes: en las siguientes líneas de código podemos ver el modo en que se abre el archivo DICOM mediante el selector de archivos *choose*r

(de la clase *JFileChooser*) y se guarda en un *DicomObject* la información que necesitamos mediante el método *read (InputStream ins, boolean f)* de dicha clase.

```
class OyenteEventoSeleccion
   implements java.awt.event.ActionListener {
   public void actionPerformed(java.awt.event.ActionEvent e) {

   javax.swing.JFileChooser chooser = new
       javax.swing.JFileChooser();
   int returnVal = chooser.showOpenDialog(parent);
   if (returnVal == javax.swing.JFileChooser.CANCEL_OPTION) {
     parent.getParent();
   }
   if (returnVal == javax.swing.JFileChooser.APPROVE_OPTION) {
     try {
       openFileName = chooser.getSelectedFile().getAbsolutePath();
       f = new File(openFileName);
       fin = new FileInputStream(f);
       bis = new BufferedInputStream(fin);
       dcm.read(bis, true);

   . . .
```

En las secciones que vienen a continuación podemos encontrar la forma de extraer los distintos tipos de datos del objeto DICOM que hemos creado.

10.2.1. Panel de Visualización de la imagen

El panel para la visualización de la imagen que hay dentro del archivo DICOM es el que ocupa la mitad superior de la GUI. Como se puede ver en la figura 10.1, alberga además de la imagen del archivo los manejadores de eventos.

Para poder extraer la imagen del archivo DICOM hemos utilizado la clase *ImageIO* del paquete *com.dicom.archimed.image*, que a su vez alberga el método *getImageProducer( )*. Mediante dicho método obtenemos una instancia de la clase *ImageProducer* (de *java.awt.image*) que nos permite visualizar la imagen en nuestro panel visor de la siguiente forma:

```
class OyenteEventoSeleccion
implements java.awt.event.ActionListener {
public void actionPerformed(java.awt.event.ActionEvent e) {

. . .

ImageIO imagenio = new ImageIO(dcm);
// Obtenemos el ImageProducer a partir de imagenio
ImageProducer imagep = imagenio.getImageProducer();
```

```
      foto = new ImageIcon(devImagen(imagep));
      // Ya sólo nos queda colocar la imagen en el marco
      marcofoto.setIcon(foto);
   }


/**
*  El siguiente método devuelve una instancia de la clase Image
*  a partir del ImageProducer que se le pasa como parámetro.
*/

   public Image devImagen(ImageProducer imp){
      Image im = createImage(imp);
      return im;
   }
```

10.2.2. Panel para Datos Personales

Este panel contiene los datos personales del paciente que es objeto de la consulta y es el primero de los que alberga el *JTabbedPane* de la mitad inferior de nuestra GUI. Lo podemos ver en la figura 10.2.

Figura 10.2. Panel para Datos Personales del paciente

Todos los datos que alberga este panel se encuentran recogidos en el diccionario de datos del estándar DICOM (Parte 3) [12], y la forma de extraerlos del *DicomObject* es muy sencilla: utilizaremos el método *get (int dname)* que está recogido en esta clase y que nos devuelve un objeto con el valor del atributo que se indica con *dname*. En nuestro caso estamos leyendo valores insertados en forma de cadenas de caracteres, por lo que tenemos que convertir el objeto que nos devuelven en un *String* (mediante el método *toString( )*)y ya sólo nos queda insertarlo en el panel ayudándonos con una *JLabel* de *javax.swing*. A continuación mostramos cómo se hizo:

```
class OyenteEventoSeleccion implements ActionListener{
   public void actionPerformed(ActionEvent e) {

   . . .


   String paciente = dcm.get(DDict.dPatientName).toString();
   nombrePacBis.setText(paciente);
   String domicil = dcm.get(DDict.dPatientAddress).toString();
   domicilioBis.setText(domicil);
   String localid = dcm.get(DDict.dRegionOfResidence).toString();
   localidadBis.setText(localid);

   . . .}
```

10.2.3. Panel para Datos Clínicos

Este es el segundo de los paneles contenidos en el JTabbedPane de la mitad inferior de la GUI y alberga los campos correspondientes a los datos resultantes de la exploración clínica a la que se sometió el paciente. Estos datos no están incluidos en el diccionario de datos del estándar DICOM y los tuvimos que crear nosotros mediante la asociación de nuevos pares grupo – elemento con el valor del atributo al que representaban. Podemos ver este panel en la figura 10.3.

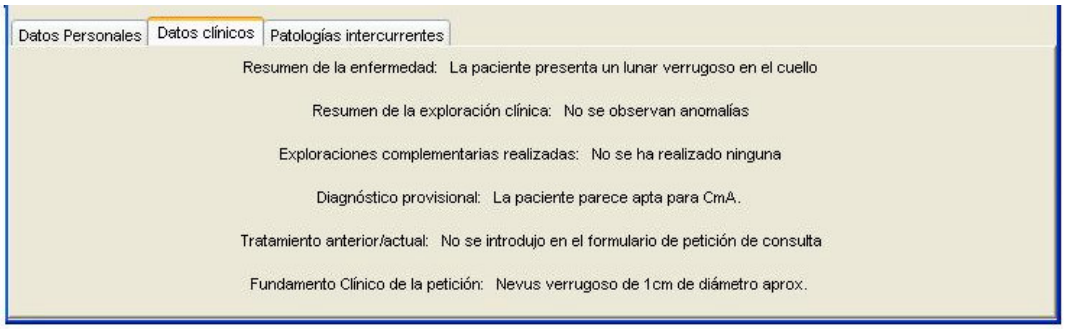

Figura 10.3. Panel para Datos Clínicos del visor

Como ya se ha dicho, estos campos de datos no se encuentran incluidos en el diccionario de datos de DICOM, de manera que no podemos utilizar el método get (int dname) para extraerlos. Sin embargo, también dentro de la clase DicomObject encontramos un método que cumplía con nuestros requerimientos: *getS_ge(int group, int element)* nos devuelve un *String* con el valor contenido en el atributo que tiene *group* por número de grupo y *element* por número de elemento. De este modo, ya sólo nos

queda añadirlo al visor mediante una *JLabel*. La implementación sería por tanto como sigue:

```
/**
 *  Si recordamos que para el campo resumen de la enfermedad
 *  y el par grupo,elemento) era (0010,4001)
 */
String resEnf = dcm.getS_ge(0010,4001);
resenfBis.setText(resEnf);
// Y hacemos esto con el resto de los campos del Panel de Datos
// Clínicos de la aplicación cliente
```

10.2.4. Panel para Patologías Intercurrentes

Si volvemos a la aplicación cliente de los centros ambulatorios, recordaremos que se dispuso un panel para introducir posibles patologías que pudiesen hacer que un paciente susceptible de ser apto para CmA en acto único se clasificase como no apto para ella. Pues bien, los datos de dicho panel serán los que se incluyan en este último panel del JTabbedPane de la mitad inferior de nuestra GUI que podemos observar en la figura 10.4.

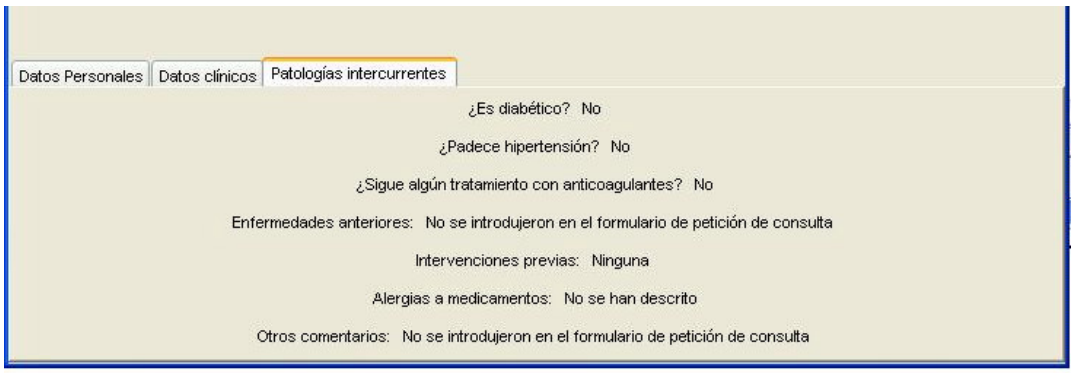

Figura 10.4. Panel para Patologías Intercurrentes del Visor

Tampoco en este caso los campos de datos estaban recogidos en el diccionario DICOM, de modo que volvimos a utilizar el método que se mencionó en el apartado anterior: *getS_ge(int group, int element)* de *DicomObject* para insertar los datos correspondientes en nuestro panel de la siguiente forma:

```
// El valor del par (grupo,elemento) para el campo "Diabetes"
// era (0010,4011)
String diaBet = dcm.getS_ge(0010,4011);
diabetBis.setText(diaBet);     // Y hacemos igual para los
      // campos del panel Patologías de la aplicación cliente.
```