Capítulo 9
Aplicación desarrollada para la central del HUVR

9.1. Introducción

En este proyecto lo que se ha desarrollado ha sido básicamente la aplicación cliente de los centros ambulatorios y la aplicación de contestación de la unidad de CPL y QUE[1]. Por este motivo, en este capítulo vamos a presentar en primer lugar una breve descripción del funcionamiento del servidor DICOM del Sistema de gestión de imágenes médicas en el que se trató de integrar nuestro proyecto [10], sin adentrarnos en cuestiones más complejas referentes a su implementación. A continuación, en el apartado 9.3 se explicará la base de datos que se propone para almacenar las distintas fichas de los pacientes que se envían al servidor desde los distintos centros ambulatorios y concluiremos el capítulo con la aplicación que se ha implementado (también en JAVA en entorno JBuilder) para enviar el formulario de contestación desde la Unidad de CPL y QUE a la Gestoría de Usuarios.

[1] Cirugía Plástica y Quemados

9.2. Servidor DICOM [10]

El Servidor DICOM actúa como un Servidor de Almacenamiento, ya que su misión consiste en ir almacenando las distintas imágenes recibidas desde las aplicaciones cliente de los distintos centros ambulatorios. Para que esto sea posible, estas aplicaciones tienen que generar y enviar ficheros de datos que sigan el Estándar DICOM, que permite la interoperabilidad entre los distintos equipos de imágenes médicas. La estructura del Servidor DICOM la podemos encontrar en la figura 9.1, en la que se ve también que la herramienta que se utilizó para la implementación del mismo fueron las librerías DCMTK[2] de *Offis* para Visual C++.

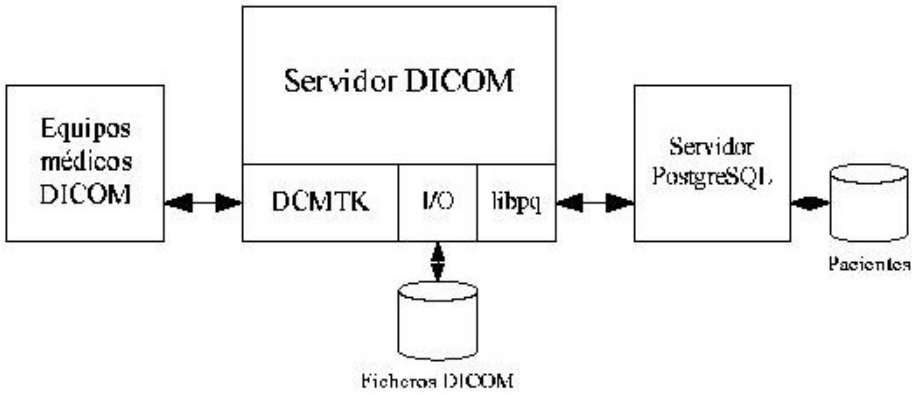

Figura 9.1. Estructura del servidor DICOM

9.2.1. Fichero de configuración del servidor

El Servidor DICOM utiliza un fichero de configuración para leer las variables necesarias para su funcionamiento. Dicho fichero se llama *dicomsvr.cfg*, y por defecto la aplicación lo busca en el fichero actual de la aplicación. Si no lo encuentra en dicho directorio, lo busca en el path indicado por la variable de entorno *DICOMSVR_CONFIG*. Si dicha variable no está definida o no se encuentra el fichero de configuración en el directorio indicado, el Servidor DICOM muestra un mensaje de error y finaliza su ejecución, ya que es imprescindible para su funcionamiento.

Las variables que se encuentran dentro del fichero de configuración son:

[2] Dicom Toolkit

- *port*: puerto de trabajo DICOM. Es el puerto en el que el Servidor DICOM escuchará las distintas peticiones de equipos DICOM. El número de puerto reservado para DICOM es el 104.
- *timeout*: tiempo máximo de espera del servidor en segundos. Si tiene el valor "0", indica que el tiempo de espera es indefinido, o lo que es lo mismo, que estamos trabajando en modo de bloqueo, es decir, el servidor queda bloqueado a la espera de alguna asociación.
- *silent*: indica si el servidor debe mostrar información por pantalla o si debe permanecer en silencio. El valor "1" indica el modo silencioso, mientras que el "0" indica el modo normal de trabajo.
- *logfile*: Si se especifica alguno, el servidor irá registrando en este fichero los distintos sucesos.
- *dcmpath*: path de la base de datos de ficheros DICOM.
- *dbaddr*: dirección IP del servidor PostgreSQL.
- *dbport*: puerto de trabajo de PostgreSQL, normalmente el 5432.
- *dbuser*: nombre de usuario a utilizar al conectar a la base de datos.
- *dbpasswd*: clave del usuario.
- *dbname*: nombre de la base de datos.

9.2.2. Diagramas de flujo de la aplicación

En las figuras 9.2, 9.3, 9.4, 9.5, 9.6 y 9.7 podemos ver el diagrama de flujo completo de la aplicación. Como ya dijimos antes, se trata de observar el funcionamiento de modo introductorio, por lo que no entraremos en muchos detalles.

Diagrama de flujo principal

Como ya se hemos comentado, lo primero que hará la aplicación será leer del fichero de configuración los valores de las variables necesarias para su funcionamiento. Si no encuentra el fichero de configuración o no puede leerlo, dará error y saldrá de la aplicación.

Tenemos que tener en cuenta que si estamos trabajando con plataformas *Microsoft Windows*, antes de poder hacer cualquier operación relacionada con redes

necesitamos inicializar las librerías *Winsock*, que contienen las APIs necesarias para trabajar con los distintos protocolos de red que ofrece esta familia de sistemas operativos. Asimismo, cuando ya no vayamos a utilizar más las funciones relaciona-das con la red, necesitamos liberar los recursos reservados en las librerías *Winsock*.

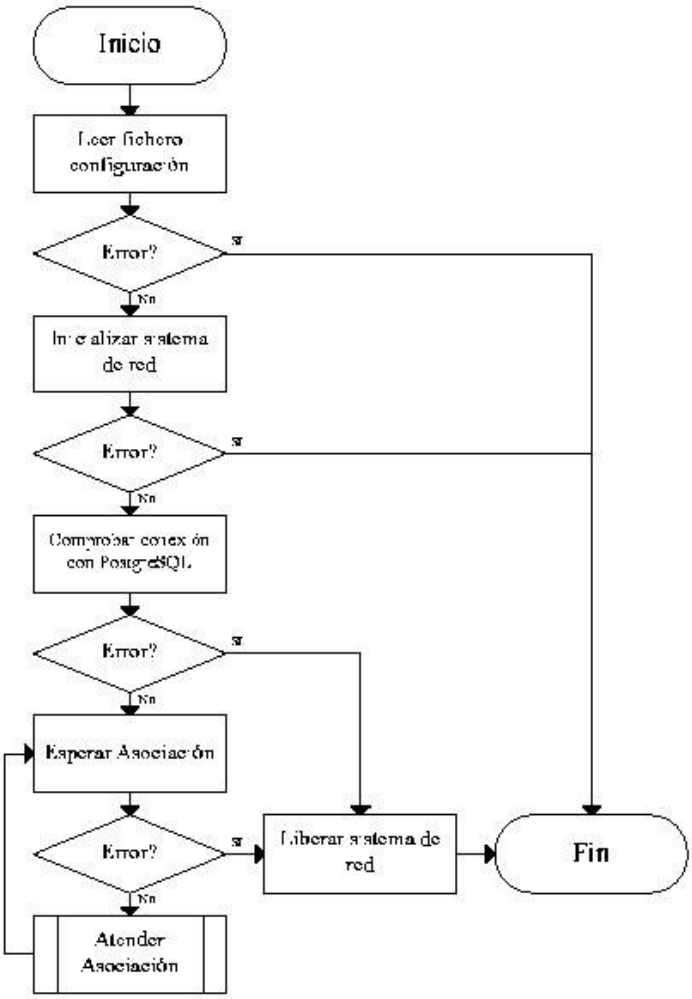

Figura 9.2. Diagrama de Flujo Principal

También hay que inicializar el subsistema de red de las librerías DCMTK, de forma que la aplicación registre en ellas qué tipo de papel va a jugar en las asociaciones (siempre receptora de asociaciones, nunca iniciadora), puerto en el que se va a trabajar, tiempo máximo de espera de asociaciones, etc. Cuando ya no vayamos a utilizar más las funciones de red de las librerías DCMTK, debemos liberar los recursos reservados.

Conexión con PostgreSQL

La conexión con el Servidor PostgreSQL se realiza mediante las librerías cliente *libpq*. En ellas encontramos todas las funciones necesarias para realizar consultas SQL sobre la base de datos.

En el caso de que no se pueda establecer la conexión con PostgreSQL, la aplicación finaliza su ejecución, ya que no va a poder insertar ningún fichero en la misma, y por tanto no a poder hacer nada. Si se establece correctamente la conexión, esta se libera a continuación ya que no hará falta hasta que tengamos que insertar un fichero en la base de datos.

Bucle principal

Una vez iniciados todos los subsistemas necesarios para la aplicación, nos encontramos con el bucle principal de la misma. La aplicación permanecerá en el bucle hasta que se den una de las siguientes posibilidades:

- Se produce un error en alguna de las funciones de asociación.
- No llega ninguna petición de asociación en el tiempo máximo especificado (si se trabaja en el modo sin bloqueo).
- Se mata al proceso correspondiente a la aplicación.

Respecto a los tiempos de espera, hay dos modos de trabajo:

- *Modo sin bloqueo*: la aplicación espera la llegada de una asociación durante un tiempo máximo de espera. Pasado ese tiempo, si no se ha recibido nada, dará error.
- *Modo con bloqueo*: la aplicación queda en estado de bloqueo hasta la llegada de alguna asociación, durante un tiempo indefinido. Sólo dará error si éste se produce en los niveles inferiores.

Tanto en un modo como en otro, si llega una asociación, la aplicación recibirá un código de éxito, con lo que pasará a atender la asociación mediante la negociación de

sus parámetros. Mediante la negociación se establecen los Contextos de Presentación que puede aceptar el Servidor DICOM, y se comparan con los contextos propuestos por la entidad llamante. Si no se puede aceptar ninguno, se rechaza la asociación. Si la negociación de los parámetros de la asociación tiene éxito, se aceptará la asociación y se enviarán las oportunas primitivas de respuesta a la Entidad llamante. Así, una vez enviada la primitiva de respuesta, el servidor pasa a estar a la espera de mensajes.

Despacho de mensajes

Una vez establecida la asociación, el servidor espera a que le lleguen mensajes. Según los mensajes que reciba, tendrá que hacer una cosa u otra. Por un lado puede recibir mensajes DIMSE, relativos a los distintos servicios definidos en dicho protocolo, como *DIMSE_C_ECHO_RQ* para el Servicio de Verificación, o *DIMSE_C_STORE_RQ* para el Servicio de Almacenamiento. Además de estos mensajes también pueden llegar mensajes relativos a la Capa DUL (*Dicom Upper Layer*), como mensajes de liberación remota *DUL_PEERREQUESTEDRELEASE*, o *DUL_PEERABORTEDASSOCIATION* , indicando que el extremo ha abortado la asociación.

La función que se ocupa de recibir los mensajes por la asociación se utiliza dentro de un bucle, de forma que el servidor estará siempre esperando mensajes por la asociación hasta que el extremo la libera o la aborta, o hasta que se produce un error en capas inferiores y la libera el proveedor de servicios. Según el tipo de mensaje DIMSE recibido se iniciará un servicio u otro. El Servidor DICOM sólo aborta la asociación si recibe un comando DIMSE que no es capaz de manejar (esto sería un error DIMSE, ya que la aplicación espera mensajes DIMSE de acuerdo a los Contextos de Presentación aceptados). En el caso de que el extremo remoto haya iniciado la liberación, el Servidor la acepta y la libera. Si el extremo remoto la abortó, ya no hay nada más que hacer. En cualquier otro caso es que se ha producido un error inesperado, y se procede a abortar la asociación desde el lado del servidor.

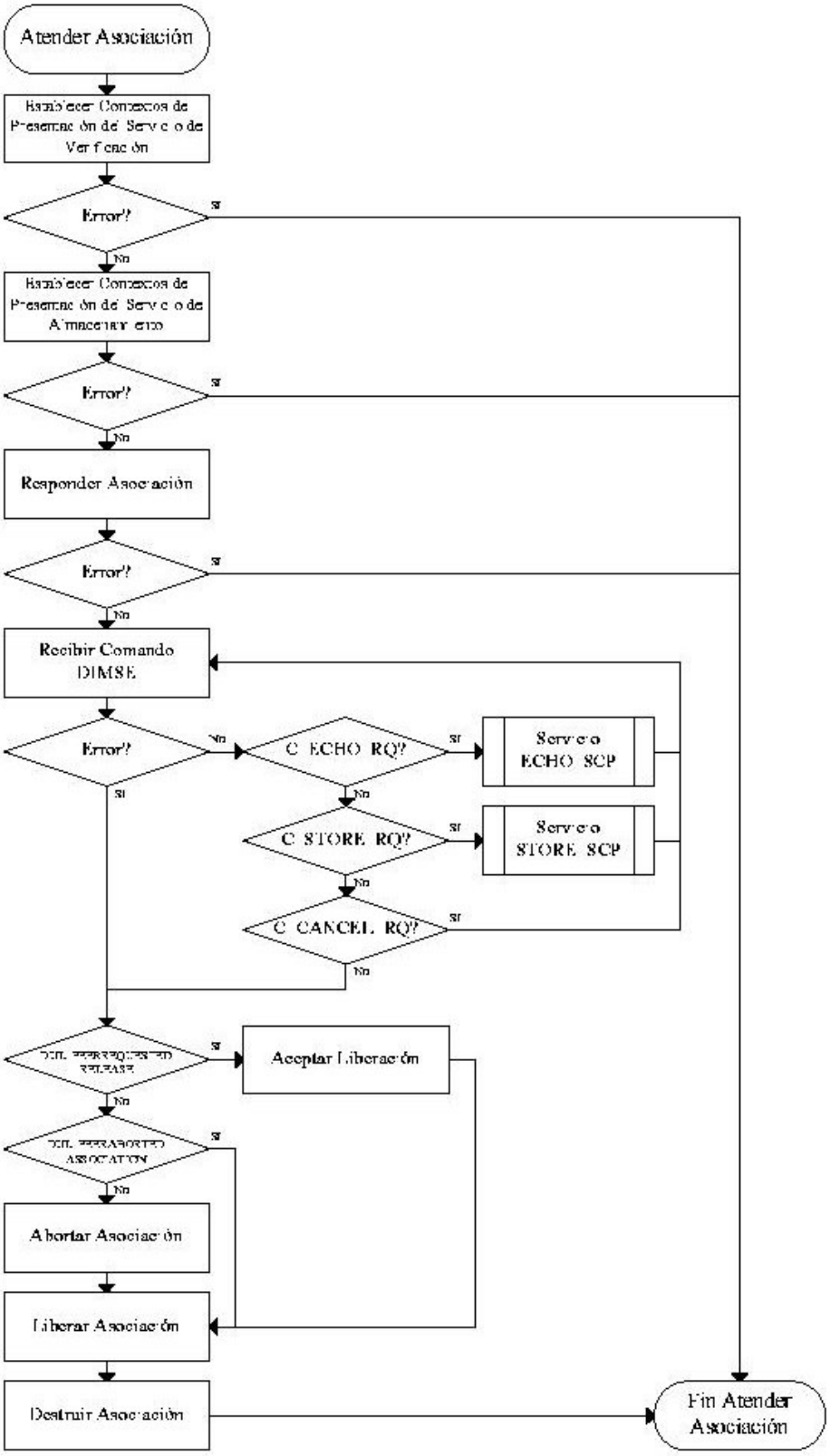

Figura 9.3. Diagrama de Flujo Atender Asociación

Servicio de Verificación ECHO SCP

El Servicio de Verificación (en el papel de SCP) consiste únicamente en el envío de la respuesta DIMSE C-ECHO RESPONSE a la aplicación cliente que actúa como SCU. Como se puede ver en la figura 9.4, se trata de un servicio muy simple.

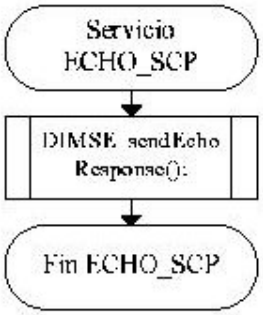

Figura 9.4. Diagrama de Flujo del Servicio C-ECHO-SCP

Servicio de Almacenamiento STORE SCP

El Servicio de Almacenamiento se lleva a cabo en una función que es relativamente simple, ya que casi todo el trabajo lo realiza la función de retrollamada StoreProgressCallback. Dicha función es pasada como argumento a la función DIMSE_storeProvider() de las librerías DCMTK, que implementa el Servicio C STORE SCP.

En primer lugar, se prepara el nombre del fichero en el que se va a almacenar la Instancia SOP recibida. Para ello se comprueba que se trata efectivamente de una Instancia SOP de Almacenamiento. Si es así, se coge el *path* de la base de datos de ficheros DICOM, y se le añade el UID de la Instancia. Como este UID es único, se asegura así que nunca se vayan a sobrescribir ficheros que contengan distinta información.

Si la Instancia SOP recibida no pertenece a una Clase SOP de Almacena miento, no puede ser tratada, así que se generará un nombre de fichero nulo. Aunque se

dé este caso, el proceso debe continuar, ya que no puede interrumpirse hasta que esté dentro de la función de retrollamada.

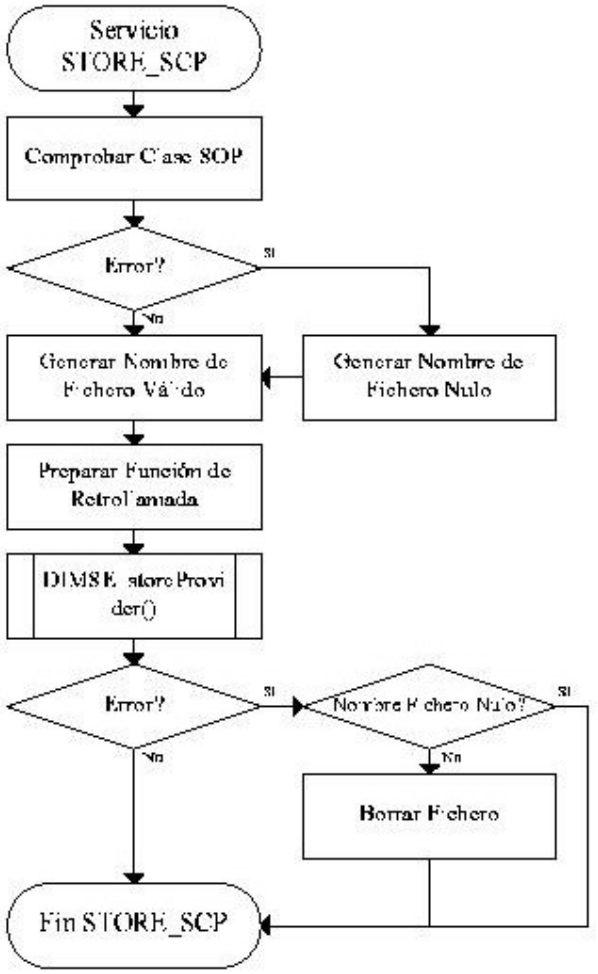

Figura 9.5. Diagrama de Flujo del Servicio C-STORE-SCP

A continuación, hay que preparar la información relativa a la función de retrollamada, de forma que una vez dentro de ella, se pueda tener el contexto del servicio que se está llevando a cabo.

Por último, se llama a la función de la librería que implementa el Servicio de Almacenamiento. En caso de que esta función devuelva un código de error, habrá que borrar el fichero creado en la base de datos.

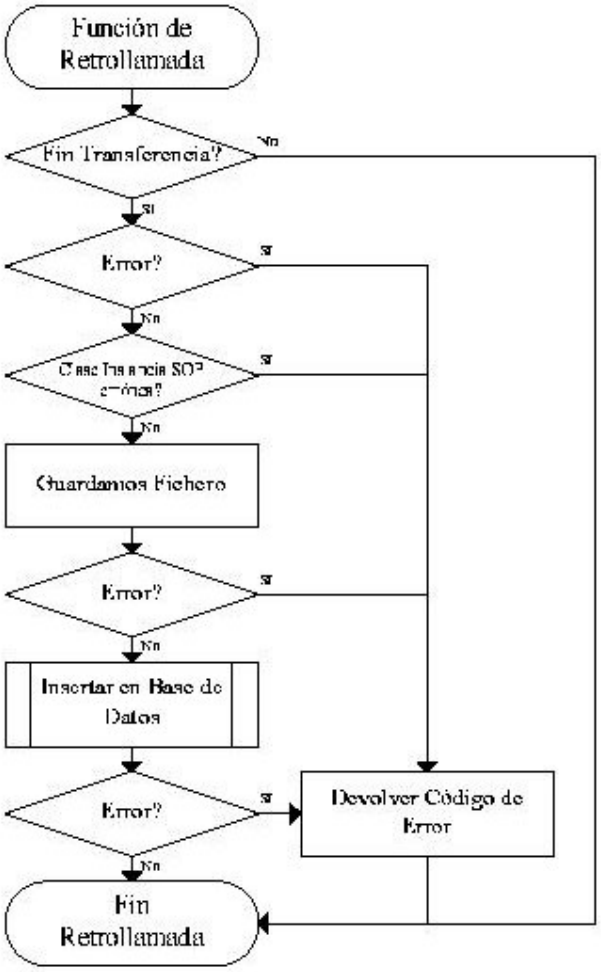

Figura 9.6. Diagrama de Flujo de la Función de Retrollamada del Servicio C-STORE-SCP

Función de retrollamada

Esta función es la que realmente realiza el trabajo asociado al Servicio de Almacenamiento, como almacenar el fichero con la Instancia SOP recibida, actualizar la base de datos de pacientes, etc. Para poder realizar su trabajo, la función necesita conocer el contexto del servicio.

Una vez que la transferencia del archivo ha terminado, es cuando realmente empieza el trabajo de esta función. En primer lugar, se recupera el contexto para utilizarlo más adelante. A continuación, se realizan una serie de comprobaciones sobre los datos recibidos, para ver si se ajustan a la información contenida en la primitiva de petición del servicio.

Si todo es correcto, se guardará la Instancia SOP recibida en un fichero DICOM. Una vez guardado el fichero en la base de datos de ficheros DICOM, hay que actualizar la información contenida en la base de datos. Si la información de la base de datos se actualiza sin problemas, la aplicación sale del servicio.

Actualización de la base de datos

La actualización de la base de datos es un proceso crítico, ya que si se produce algún error, la base de datos puede quedar corrompida, de forma que la información que contenga sea errónea. Para evitar estos problemas se utilizó una característica muy útil implementada por PostgreSQL, como son las transacciones. Así, antes de hacer ningún cambio, se indica el comienzo de una transacción. Todas las operaciones que se realicen sobre la base de datos quedan registradas, de forma que si hay un error, se puede volver al estado inicial y no se quedan a medias las transacciones. Si todo va bien, se le indicará a la base de datos, que procederá a volcar los cambios sobre los datos reales.

Lo primero que se hace es crear una conexión con PostgreSQL. Al comienzo de la ejecución del Servidor se hizo una conexión de prueba, así que en este punto ya se sabe que al menos los datos de configuración son correctos. No obstante, podría producirse error en la conexión si el Servidor PostgreSQL deja de estar accesible por cualquier circunstancia.

A continuación se recopila toda la información necesaria acerca del paciente, prueba y serie a la que pertenece la Instancia SOP recibida y contenida en el fichero destino. Obtenido el identificador del paciente, se comprueba que existe en la base de datos de pacientes. Si no existe, no se puede agregar la información, así que se sale, cancelando antes la transacción. Después se comprueba que la instancia contiene una imagen gráfica o una ECG (en el caso de nuestra aplicación cliente siempre va a ser una imagen gráfica), ya que si no, el servidor no sería capaz de tratarla. Ya sólo resta ir insertando la información en la base de datos. Si se produce algún error, simplemente se cancelará la transacción. Si todo es correcto, se indicará a PostgreSQL que la transacción ha finalizado con éxito y que puede aplicar los cambios sobre los datos reales.

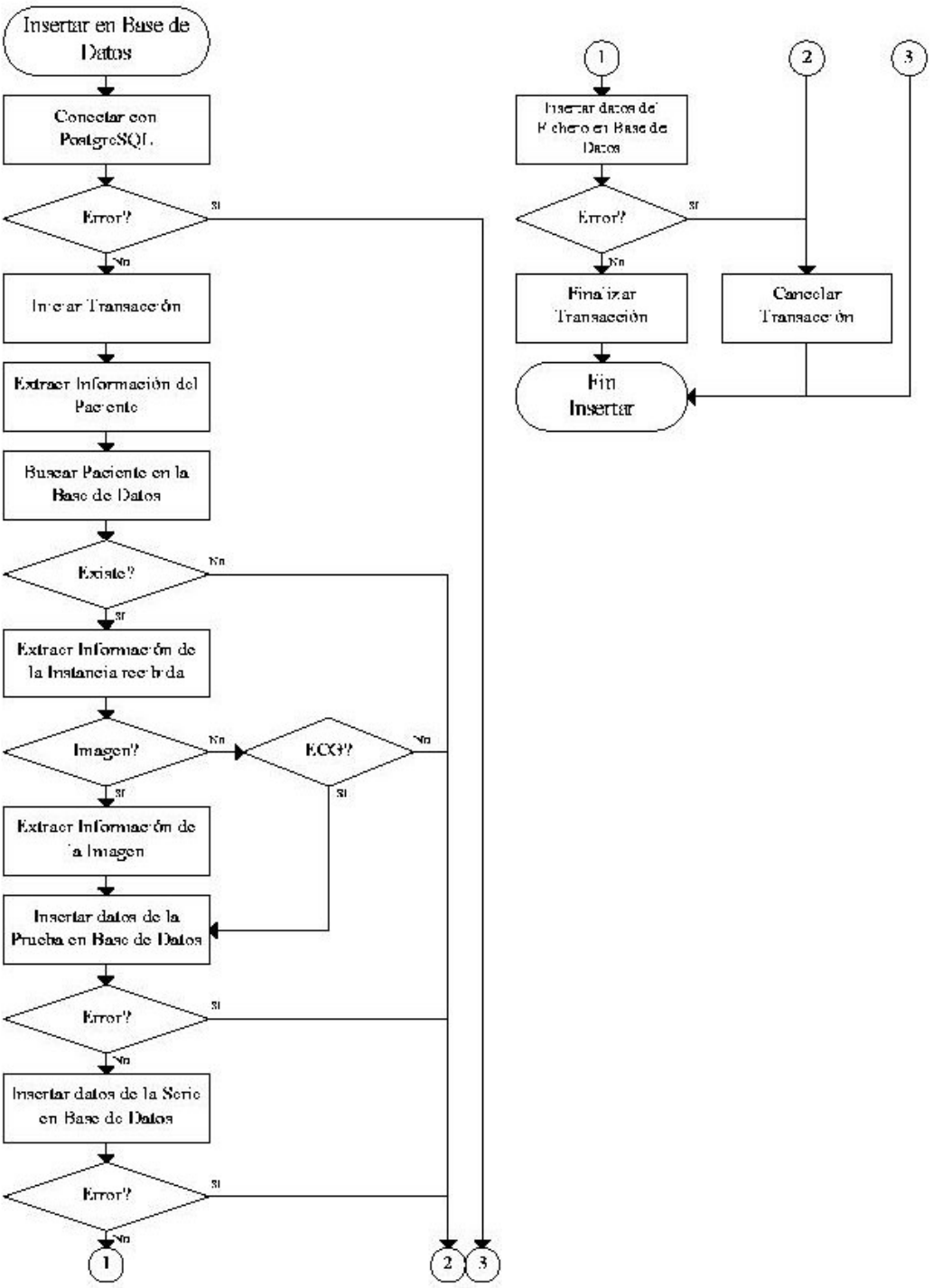

Figura 9.7. Diagrama de Flujo de la Función de Inserción en la Base de Datos

9.3. La Base de Datos

En este apartado vamos a analizar la estructura de la base de datos que se propone para el diseño de nuestro sistema. En principio, es bastante similar a la base de datos que se utilizó en el Sistema de Gestión de Imágenes Médicas en el que se trató de integrar nuestro proyecto [10] con ligeras modificaciones, ya que habrá que añadir algunos campos referentes a los datos del paciente. Para almacenar la información que va a gestionar el Sistema se utilizará una base de datos con un modelo Entidad - Relación, ya que es el que mejor se adapta al Modelo de Información definido en DICOM (ver modelo Entidad - Relación de los Objetos de Información Compuestos DICOM, en la sección 5.4).

El esqueleto de esta base de datos será el que vemos en la figura 9.8, donde podemos observar las distintas entidades que componen el modelo. Podemos encontrar el código completo de la estructura propuesta para la base de datos en el anexo C.

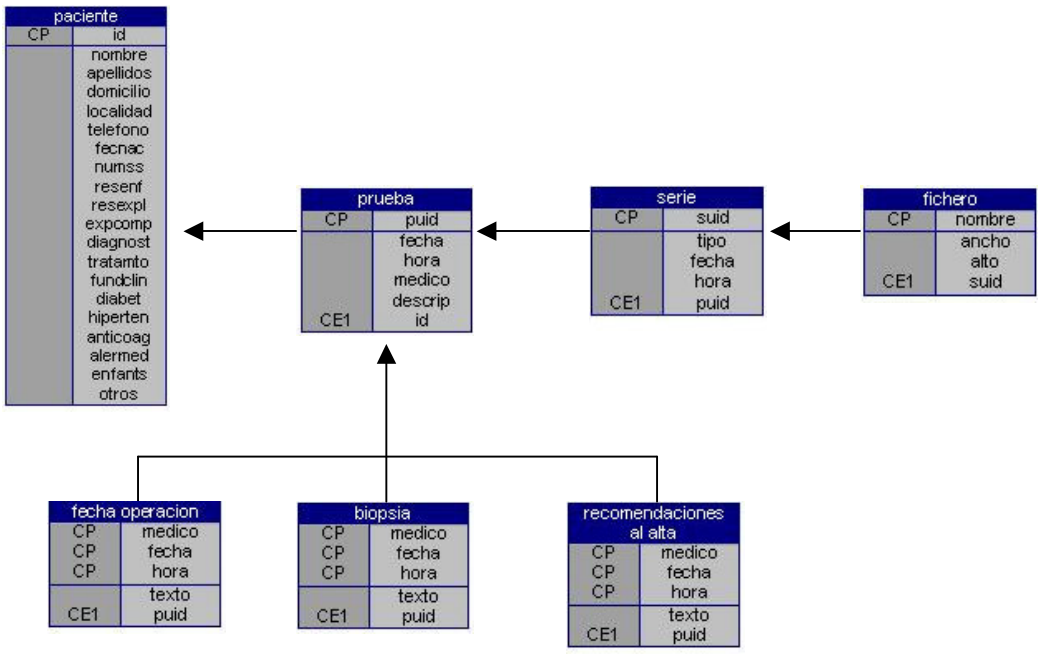

Figura 9.8. Esqueleto propuesto para la Base de Datos de Pacientes

A continuación vamos a estudiar cada una de las Entidades detalladamente.

Entidad paciente

Es la entidad principal, a partir de la cual se extiende todo el esqueleto. Se corresponde básicamente con la entidad Paciente definida en el modelo E-R de los Objetos de Información Compuestos DICOM. Contiene la información personal y clínica del paciente, como su nombre, apellidos, diagnóstico del facultativo del centro ambulatorio, etc.

Los atributos de que se compone esta entidad son los siguientes:

- *id*: identificador del paciente en el Hospital. Ha de ser único, al menos en el ámbito del centro y se genera por medio de los procedimientos internos del mismo, debiendo tener como máximo 10 caracteres.
- *nombre*: nombre del paciente. Admite como máximo 64 caracteres.
- *apellidos*: apellidos completos del paciente. Máximo 256 caracteres.
- *domicilio*: domicilio de residencia habitual del paciente. Admite hasta 256 caracteres.
- *localidad*: localidad en la que se encuentra el domicilio de residencia habitual del paciente. Admitirá hasta 40 caracteres.
- *telefono*: teléfono de contacto del paciente. Máximo 16 cifras.
- *fecnac*: fecha de nacimiento del paciente. El formato a seguir es el mismo que el definido en el Estándar DICOM, esto es, AAAA-MM-DD, donde:
  - **AAAA**: año, siempre con cuatro dígitos.
  - **MM**: mes
  - **DD**: día.
- *numss*: Número de la Seguridad Social del paciente. El formato a seguir debe ser 12 dígitos sin barras ni espacios. Si el paciente no tiene dicho número, tomará el número de tarjeta sanitaria.
- *resenf*: Resumen de la enfermedad que ha llevado al paciente al centro ambulatorio. Este y los cinco campos que siguen a continuación son para almacenar los datos contenidos en el panel de datos clínicos que vimos en la aplicación cliente de los centros ambulatorios. El formato de estos seis campos

será el correspondiente al de un valor de tipo texto, por lo que sólo está limitado por la capacidad del gestor de bases de datos para este tipo de datos.

- *resexpl*: Resumen de la exploración realizada al paciente en los centros ambulatorios. Será tal y como hemos dicho un valor de tipo texto.
- *expcomp*: Aquí se recogerán exploraciones complementarias realizadas al paciente si se enviaron. Valor de tipo texto.
- *diagnost*: Campo que almacena un diagnóstico provisional o definitivo que se emitió el facultativo del centro ambulatorio. Valor de tipo texto.
- *tratamto*: Resume tratamientos anteriores o actuales que estuviese siguiendo el paciente. Valor de tipo texto.
- *fundclin*: Campo que almacena un valor de tipo texto con el fundamento clínico de la petición.
- *diabet*: Los atributos que se presentan desde este hasta el último son los que recogen los datos que se inserten en el panel correspondiente a las patologías intercurrentes que vimos en la aplicación cliente. Este en concreto será un campo para un solo carácter que sólo puede valer "S" si el paciente es diabético o "N" si no lo es.
- *hiperten*: Al igual que el anterior, este es un campo para un carácter que valdrá "S" si el paciente padece hipertensión o "N" si no la padece.
- *anticoag*: Campo para un carácter que valdrá "S" si el paciente está siguiendo un tratamiento con anticoagulantes o "N" en caso contrario.
- *alermed*: Campo para valor de tipo texto que recoge posibles alergias a medicamentos del paciente.
- *enfants*: Campo para valor de tipo texto que recoge la información que contenga el área de texto para enfermedades anteriores sufridas por el paciente de nuestra interfaz de la aplicación cliente.
- *otros*: Aquí se recogería en un valor de tipo texto otros posibles comentarios que hubiese insertado el médico del centro ambulatorio en el campo correspondiente.

Como clave primaria de esta entidad se utiliza el atributo id, ya que debe ser único, al menos en el ámbito en el que trabajaremos.

Entidad Prueba

La entidad Prueba se corresponde básicamente con un Estudio tal como se define en DICOM. Recoge la información relativa a una prueba realizada a un paciente, como fecha y hora, médico responsable y la descripción de la misma. Una prueba en si no contiene ninguna imagen, si no que se compone de Series, como veremos más adelante.

Los atributos de esta entidad son:

- *puid*: identificador único de la prueba. Es un identificador único global, de forma que no puede haber dos pruebas con el mismo puid. Se corresponde con el uid asignado por DICOM a la prueba, siguiendo los procedimientos definidos en el Estándar, y consta de hasta 64 caracteres.
- *fecha*: fecha de inicio de la prueba, en el formato AAAA-MM-DD, descrito anteriormente. Indica la fecha en la que se dio de alta la prueba. Posteriormente a esa fecha se pueden agregar nuevas series a la misma prueba.
- *hora*: hora de inicio de la prueba, en el formato HH:MM:SS.
- *medico*: nombre del médico responsable de la prueba. Admite hasta 256 caracteres.
- *descripcion*: descripción de la prueba realizada. Es un campo de texto, limitado únicamente por el sistema de bases de datos utilizados.
- *id*: identificador del paciente al que hace referencia esta prueba (ver entidad paciente).

Como clave primaria de esta entidad se utiliza el atributo puid, ya que por su propia definición en el Estándar DICOM se garantiza su unicidad. La relación con la entidad paciente se establece mediante la clave externa id, que referencia al paciente es cuestión. Si dicho paciente es borrado, el sistema de base de datos se encargará de eliminar también todas las pruebas relacionadas con él.

Entidad Serie

La entidad Serie se corresponde con la entidad del mismo nombre definida en DICOM. Engloba la información común de una serie de imágenes generadas por un determinado equipo en el transcurso de un examen médico.

Se compone de los siguientes atributos:

- *suid*: identificador único de la serie. Es un identificador único global, de forma que no puede haber dos series con el mismo suid. Se corresponde con el uid asignado por DICOM a la serie, y consta de 64 caracteres.
- *tipo*: identifica el tipo de máquina utilizada para generar la serie de imágenes, que en nuestro caso será siempre una webcam (aparato de toma de imágenes digitales). Es un campo de 16 caracteres, cuyos valores permitidos se encuentran recogidos en el Estándar DICOM. Algunos ejemplos son:
  - XC: Equipo de Fotografía con Cámara Externa
  - CR: Equipo de Radiografía Computarizada
  - MR: Equipo de Resonancia Magnética
  - US: Equipo de Ultrasonidos
  - ...
- *fecha*: fecha de realización de la serie, en el formato AAAA-MM-DD, descrito anteriormente.
- *hora*: hora de realización de la serie, en el formato HH:MM:SS.
- *medico*: nombre del médico responsable de la serie. Admite hasta 256 caracteres.
- *puid*: identificador único de la prueba a la que pertenece esta serie (ver entidad prueba).

Como clave primaria de esta entidad se utiliza el atributo suid, que por propia definición es único. La relación con la entidad prueba se establece mediante la clave externa puid.

Entidad Fichero

La entidad Fichero representa un fichero DICOM. Como sabemos un fichero DICOM contiene un Registro de Datos, correspondiente a una Instancia SOP codificada. Instancias SOP puede haber de muchas clases, aunque nosotros nos vamos a ceñir únicamente a las relacionas con imágenes. El resto, como las orientadas a gestión de impresión, documentos estructurados, etc. escapan al ámbito de este Proyecto.

Se compone de los siguientes atributos:

- *nombre*: nombre del fichero DICOM, con el *path* relativo al directorio donde se encuentra la base de datos de ficheros. Con esto separamos la ubicación real de los ficheros de la información contenida en la base de datos, permitiendo así el traslado de la base de datos de ficheros en caso de que fuese necesario. Admite como máximo 2048 caracteres, más que suficiente para cualquier necesidad.
- *ancho*: ancho de la imagen en píxeles. Es un valor numérico entero.
- *alto*: alto de la imagen en píxeles. Al igual que el anterior, es un valor numérico entero.
- *suid*: identificador único de la serie a la que pertenece esta imagen.

Como clave primaria de esta entidad se utiliza el atributo *path*, que debe ser único, ya que no puede haber dos ficheros distintos con el mismo nombre. La relación del fichero con la serie se establece mediante la clave externa suid.

Entidad Fecha Operación

Esta entidad y las que vienen a continuación constituyen un mecanismo para la gestión de la historia clínica del paciente tras la intervención, por cuanto que sirven para que el médico responsable del área de CPL y QUE inserte datos referentes a la operación a la que se sometió al paciente. Esta entidad en concreto sirve para insertar la fecha en que fue dicha operación. Los atributos de los que se compone son los siguientes:

- *medico*: nombre del médico que realizó la intervención. Admite hasta 256 caracteres.
- *fecha*: fecha en la que se están insertando los datos. Tiene el formato AAAA-MM-DD que vimos antes.
- *hora*: hora. Tiene el formato HH:MM:SS.
- *texto*: la fecha de la intervención en sí. Se le ha asignado el nombre texto para no confundirla con la fecha en que se están insertando los datos. Tiene el formato AAAA-MM-DD.
- *puid*: identificador de la prueba a la que hace referencia esta entidad.

Como clave primaria de la entidad se utilizan la combinación de atributos *medico*, *fecha* y *hora*. Se garantiza su unicidad ya que un mismo médico no puede escribir al mismo tiempo dos resúmenes distintos. La relación con la prueba a la que se refiere se establece mediante la clave externa puid.

Entidad Biopsia

La entidad biopsia se utiliza cuando se ha habido necesidad de realizar este tipo de intervención al paciente, para recoger los comentarios que el médico haya querido hacer sobre ello. Los atributos de los que se compone esta entidad son los siguientes:

- *medico*: nombre del médico que realiza la exploración. Admite hasta 256 caracteres.
- *fecha*: fecha en la que se realiza la exploración. Tiene el formato AAAA-MM-DD.
- *hora*: hora. Tiene el formato HH:MM:SS.
- *texto*: los comentarios referentes a la biopsia en sí. Es un valor de tipo texto, por lo que sólo está limitado por la capacidad del gestor de bases de datos para este tipo de datos.
- *puid*: identificador de la prueba a la que hace referencia esta exploración.

Como clave primaria de la entidad se utilizan la combinación de atributos medico, fecha y hora. Al igual que en el punto anterior, podemos garantizar su unicidad. La relación con la prueba a la que se refiere se establece mediante la clave externa puid.

Entidad Recomendaciones al Alta

En esta entidad se almacenarán las recomendaciones que el médico cree pertinentes hacer al paciente para el momento en que éste reciba el alta médica. Los atributos que contiene son:

- *medico*: nombre del médico que realiza las exploraciones complementarias. Admite hasta 256 caracteres.
- *fecha*: fecha en la que se realizan dichas exploraciones. Tiene el formato AAAA-MM-DD.
- *hora*: hora. Tiene el formato HH:MM:SS.
- *texto*: Aquí es donde se recogen verdaderamente las recomendaciones del médico al alta. Es un valor de tipo texto.
- *puid*: identificador de la prueba a la que hace referencia dicho resumen.

Como clave primaria de la entidad volvemos a utilizar la combinación de atributos medico, fecha y hora, y con ello volvemos a garantizar su unicidad. La relación con la prueba se establece también en este caso mediante la clave externa puid.

9.4. Interfaz Web de usuario

En este apartado nos vamos a limitar a hacer una breve explicación del funcionamiento de la interfaz Web del Sistema de Gestión de imágenes médicas [10] para poder entender el modo en que se consultaría un fichero de imagen en la central del HUVR porque esta es una cuestión que queda fuera del ámbito de nuestro proyecto.

Como ya dijimos en la sección 4.3, la consulta de las fichas de los pacientes se hará mediante una interfaz de usuario basada en páginas Web. Para desarrollar dicha interfaz de usuario, el objetivo principal fue conseguir la facilidad de uso y la funcionalidad de la misma. Esto se hizo teniendo en cuenta que los usuarios del sistema serán principalmente personal médico y especialistas, por lo que sus conocimientos informáticos no tienen por qué ser superiores a los de un usuario normal. Solo se requerirá un mínimo de experiencia en el manejo de navegadores Web. Esto fue posible

gracias a que se trató de ocultar todos los detalles relativos a DICOM que no aportan ninguna información al usuario, consiguiendo así simplificar al máximo la interfaz.

Para el desarrollo de las páginas se optó por utilizar el lenguaje de *script* PHP, ya que permite realizar funciones muy avanzadas sin necesidad de utilizar ningún navegador especial, pues todo se ejecuta en el Servidor Web. Para ello será necesario instalar en el Servidor Web un módulo especial que se encargue de interpretar las páginas PHP, que normalmente tienen la extensión *.php*.

El Servidor Web que se escogió para el Proyecto fue Apache, ya que está muy extendido y está disponible en gran variedad de plataformas. No obstante, en principio podría usarse cualquier otro, como *Microsoft Internet Information Services*. El único requisito es que disponga de soporte para PHP.

9.4.1. Estructura general de las páginas

Esto será lo único que veremos de la interfaz. A la hora de diseñar las páginas Web utilizando el lenguaje PHP, se optó por utilizar las facilidades que éste proporciona relativas al manejo de objetos. Dicho lenguaje proporciona orientación a objetos, aunque a un nivel un tanto básico pero suficiente para las necesidades que existían.

De esta forma, se desarrollaron varias clases, que aportan la funcionalidad necesaria para generar las páginas. Las clases que se utilizaron fueron las siguientes:

- *CPaginaWEB*: esta clase representa una página cualquiera del Proyecto, y proporciona los métodos necesarios para mantener una estructura y estilo coherentes en todo el sistema.
- *CPaciente*: esta clase representa a un paciente y proporciona los métodos para la gestión de los datos del mismo así como todas las funciones relacionadas. Tiene la particularidad de que sus atributos se corresponden básicamente con los de la entidad Paciente de la base de datos.
- *Clase CPrueba*: representa una prueba de un paciente. Los atributos, al igual que en el caso anterior, se corresponden con los de la entidad Prueba de la base de datos.

- *Clase CSerie*: representa a una serie de una prueba, y se corresponde de nuevo con la entidad Serie de la base de datos.
- *Clase CFichero*: representa a un fichero DICOM y se corresponde con la entidad Fichero de la base de datos.
- *Clase CListado*: se utiliza para generar los listados de pacientes que van a permitir seleccionar el paciente de trabajo en las páginas Web. Permite ordenar el listado en orden ascendente o descendente según id, nombre, apellidos o fecha de nacimiento.
- *Clase CPostgreSQL*: se encarga de gestionar todo lo relacionado con la utilización del sistema de bases de datos PostgreSQL, permitiendo a las clases un acceso simplificado a la misma.

9.5. Aplicación de contestación a la Gestoría de Usuarios

Esta aplicación, al igual que la aplicación cliente de los centros ambulatorios, fue implementada en JAVA con entorno de desarrollo JBuilder, pero en este caso no necesitaremos bibliotecas adicionales puesto que todas las funciones que utilizaremos vienen incluidas en el SDK[3] en su versión 1.4.2.

En esta aplicación volvemos a tener una interfaz gráfica de usuario en la que el facultativo del HUVR señalará si el paciente es apto o no para la intervención de CmA y enviará el resultado a la Gestoría de Usuarios para que desde allí se encarguen de avisar al paciente con la información que corresponda. Pero veamos la aplicación con más detalle:

9.5.1. Interfaz Gráfica de Usuario

El diseño definitivo de la interfaz puede verse en la figura 9.9. Para ello se volvieron a utilizar componentes de la biblioteca *javax.swing*, por lo que no vamos a entrar en detalles innecesarios acerca del código (además, éste puede encontrarse en el anexo C). Como puede comprobarse, la GUI se encuentra dividida en tres partes, pero ahora es mucho más sencillo que en el caso de la aplicación cliente (que también tenía

[3] Software Development Kit

su GUI dividida en partes según necesidades) porque no hemos necesitado usar ningún *JTabbedPane*. De este modo, se puede observar que en la parte superior de la interfaz tenemos un *JPanel* para ubicar los datos personales del paciente, en la zona central tenemos otro para los datos relativos al resultado de la consulta y la parte inferior contiene los botones que gestionan la aplicación.

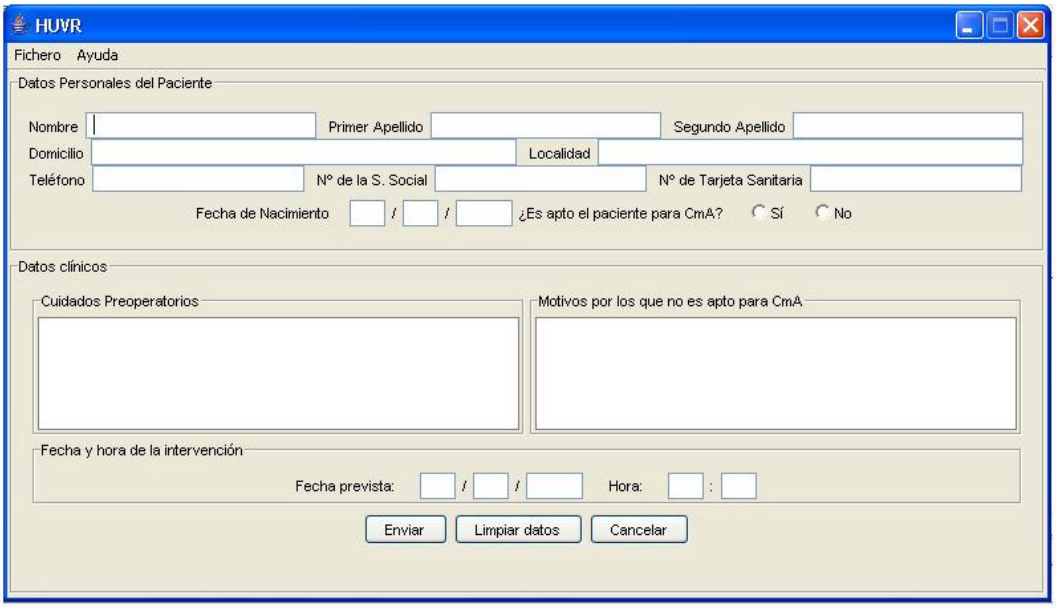

Figura 9.9. Diseño definitivo de la GUI de contestación a la Gestoría de Usuarios

Merece la pena mencionar que también para esta GUI se ha implementado la herramienta de ayuda para rellenar los campos de datos con el método *setToolTipText()* que ya vimos cuando explicamos la aplicación cliente. Además, es preciso rellenar todos los campos de datos personales del paciente y señalar si éste resulta ser apto o no para CmA en su modalidad de acto único para que se pueda enviar la contestación a la Gestoría de Usuarios. Si no se rellena alguno de los datos mencionados, la aplicación lo avisará con un mensaje semejante al de la aplicación cliente (sección 8.3) y esperará a que todos los campos necesarios estén rellenos para proceder a crear y enviar el archivo de respuesta a la Gestoría de Usuarios.

9.5.2. Panel para Datos Personales del Paciente

Como puede verse en la figura 9.10, es muy similar al que se implementó para la aplicación cliente, salvo que en este caso se cambia el *CheckboxGroup* (grupo de casillas de verificación) de la parte inferior del *JPanel* en el que se señalaba si era la primera vez que el paciente acudía a la consulta por otro *CheckboxGroup* en el que el facultativo responsable en la central del HUVR indicará el resultado de la consulta, es decir, si el paciente es apto para CmA en su modalidad de acto único o no lo es.

Figura 9.10. Panel para Datos Personales del Paciente de la GUI de contestación a la Gestoría de Usuarios

Como ya hemos dicho, es muy similar al Panel para Datos Personales del Paciente que desarrollamos para la aplicación cliente (sección 8.2.1), por lo que no vamos a profundizar más en este apartado.

9.5.3. Panel para Datos Clínicos

Se puede observar en la figura 9.11 y la forma de rellenarlo es sencilla e intuitiva, como mostramos a continuación:

- Si el paciente resultó ser apto para CmA en su modalidad de acto único, el responsable del área de CPL y QUE indicará en los campos correspondientes la fecha prevista para la intervención, así como la hora de la misma. Además, insertará los cuidados preoperatorios que el paciente debería seguir en el área de texto del mismo nombre.
- Por el contrario, si el paciente resultó no ser apto para CmA en su modalidad de acto único, el facultativo indicará los motivos en el campo correspondiente y desde la Gestoría de Usuarios se informará al paciente del modo en que debe seguir el cauce convencional que vimos en el capítulo 2.

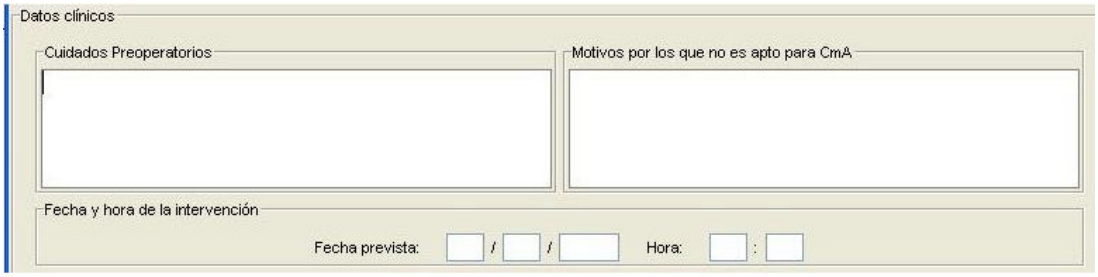

Figura 9.11. Panel para Datos Clínicos de la GUI de contestación a la Gestoría de Usuarios

En este caso también se utilizaron componentes de *javax.swing* para la implementación, por lo que tampoco profundizaremos más aquí.

9.5.4. Gestión de los eventos de la GUI

Los botones que gestionan los eventos correspondientes a nuestra interfaz los podemos encontrar en la figura 9.12, y también han sido implementados con componentes de *javax.swing*.

Figura 9.12. Botones para la gestión de los eventos de la GUI de contestación a la Gestoría de Usuarios

Botón Enviar

Lo primero que hace el oyente de eventos asociado a este botón es comprobar si los campos de datos personales del paciente están todos rellenos y si se ha seleccionado la opción de aptitud o no aptitud del paciente. En caso de que falte alguno por rellenar, aparecerá un mensaje de error semejante al que vimos en el caso de la aplicación cliente (ver figura 8.11).

Una vez que está todo correcto, se generará un fichero de texto que tendrá un formato ú otro según la clasificación del paciente para CmA en su modalidad de acto único. Para poder hacer esto, se implementó el método *boolean creaArchivo()* dentro de la clase *FuncionesGUI()* que a su vez se encuentra incluida en el *package huvr.nucleo*.

Este método pregunta el nombre que quiere darse al fichero que se va a crear (ver figura 9.13) y posteriormente, lo ubicará en la carpeta *C:/ficheros respuesta/* de manera que en el PC desde el que se envía el formulario de contestación a la gestoría de usuarios quede una copia de los ficheros enviados.

Podemos ver el formato que adquieren estos ficheros en las figuras 9.14 y 9.15. A continuación, vamos a proceder a explicar un poco la forma en que se ha implementado lo que acabamos de ver, aunque como ya se ha dicho, el código completo de esta aplicación lo podemos encontrar en el anexo C.

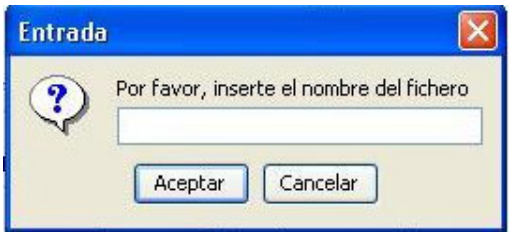

Figura 9.13. Cuadro de Diálogo para la inserción del Nombre del Fichero de Texto

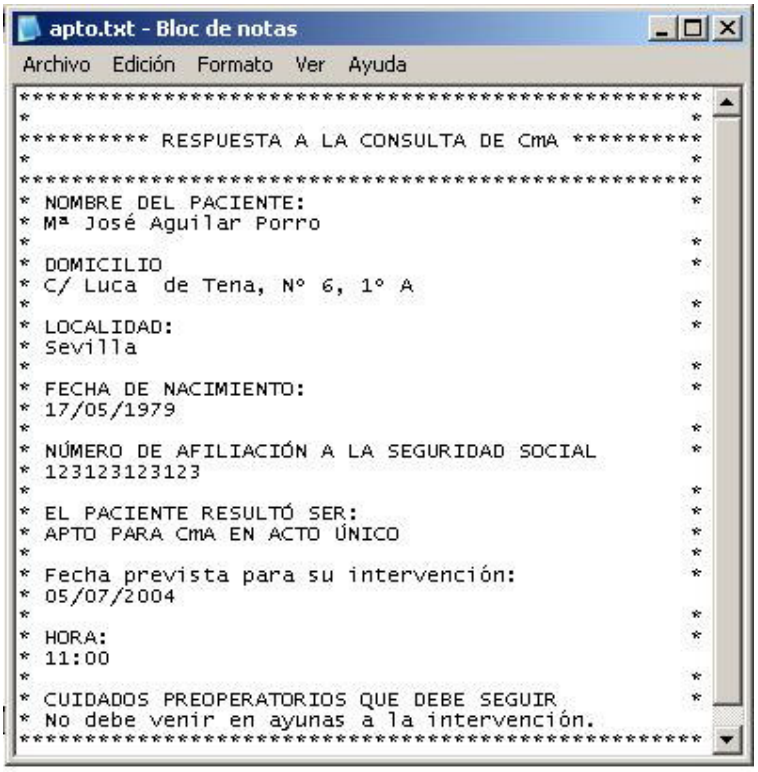

Figura 9.14. Ejemplo del Archivo de Texto generado cuando del paciente es Apto para CmA en Acto Único

Para la generación de los archivos, hemos utilizado clases y métodos contenidos en el paquete java.io como son *FileWriter*, *BufferedWriter* y *PrintWriter*. Para insertar el contenido de los campos de la interfaz simplemente tuvimos que hacer uso del método *getText()* que se encuentra dentro de la clase *JTextField*. Pero veamos esto más detenidamente:

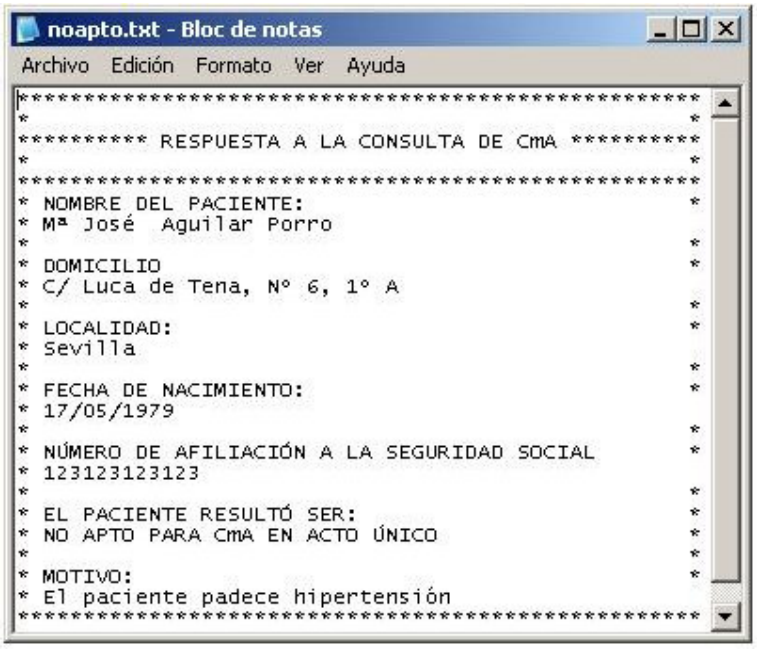

Figura 9.15. Ejemplo del Archivo de Texto generado cuando del paciente no es Apto para CmA en Acto Único

```
public boolean creaArchivo(){
   boolean f;
   try {

/**
*  Creamos un fichero de texto para ir copiando los datos de la
*  GUI en él.
*/
   Object o = javax.swing.JOptionPane.showInputDialog(null,"Por
        favor, inserte el nombre del fichero");
   String nombreFich = o.toString();
   FileWriter fw = new FileWriter ("c:/ficheros/" + nombreFich +
        ".txt");
   BufferedWriter bw = new BufferedWriter(fw);
   PrintWriter salida = new PrintWriter(bw);

. . .

/**
*  Vemos cómo se inserta un dato
*/
   salida.println("* DOMICILIO                                    *");
```

```
        salida.println("* " +
             PanelDatosPersonales.domicilioCampo.getText());

    . . .
```

La discriminación del formato del archivo según la clasificación del paciente se hace en un bucle muy sencillo que podemos ver a continuación:

```
    . . .

    salida.println("* EL PACIENTE RESULTÓ SER:                    *");
    if (PanelDatosPersonales.si.getState()){
       salida.println("* APTO PARA CmA EN ACTO ÚNICO              *");

    . . .

    }
    else if (PanelDatosPersonales.no.getState()){
       salida.println("* NO APTO PARA CmA EN ACTO ÚNICO           *");

    . . .
```

Ya sólo nos queda por ver el modo en que se envían los ficheros a la Gestoría de Usuarios. Cuando se trata de enviar ficheros a través una red que a su vez se rige por TCP / IP, lo más lógico es emplear *FTP*[4] como protocolo de transmisión. Este protocolo se utiliza para enviar ficheros de un sistema a otro bajo el control del usuario.

FTP permite transmitir ficheros tanto de texto como en binario, y además permite controlar el acceso de los usuarios. Cuando un usuario solicita la transferencia de un fichero, el FTP establece una conexión TCP con el sistema de destino para intercambiar mensajes de control. Esta conexión permite al usuario transmitir su identificador y contraseña, además de la identificación del fichero junto con las acciones a realizar sobre el mismo. Una vez que el fichero se haya especificado y su transferencia haya sido aceptada, se establecerá una segunda conexión TCP a través de la cual se materializará la transferencia. El fichero se transmite a través de la segunda conexión, sin necesidad de enviar información extra, o cabeceras generadas por la capa de aplicación. Cuando la transferencia finaliza, se utiliza la conexión de control para indicar el fin, además esta misma conexión estará disponible para aceptar nuevas órdenes de transferencia. [21]

[4] File Transfer Protocol

En Internet, el acceso a los recursos de un ordenador remoto se especifica mediante lo que se denomina *URL* (*Uniform Resource Locator*) que es una cadena de caracteres mediante la cuál queda totalmente especificada la situación en la red de cualquier tipo de información, de páginas Web, bases de datos… Con la URL, se identifica el protocolo, el servidor con el que se quiere establecer la conexión, el puerto por el que se va a hacer y el recurso al que se quiere acceder. [11]

Para implementar los URLs con JAVA, disponemos de la clase *URL* y de sus métodos, que pertenecen al paquete *java.net*. Los constructores para esta clase son diferentes dependiendo de la forma en que nos refiramos a la información buscada. Por eso, utilizaremos la siguiente versión del constructor de la clase:

```
public URL (String recurso)
```

Donde la cadena que se le pasa como parámetro contiene la especificación completa del recurso del servidor FTP de la Gestoría de Usuarios al que queremos enviar el archivo de texto de contestación. Así, en nuestro programa, establecemos la conexión con el servidor FTP de la siguiente forma:

```
. . .

try{
// Primero, crea un objeto de tipo URL para conectarse con el
// servidor ftp
URL url = new URL("ftp://ftp.servidor_gestoría.sas");
// A continuación, establece la conexión
URLConnection urlc = url.openConnection();
// Obtiene los canales de entrada y salida de datos
InputStream is = urlc.getInputStream();
OutputStream os = urlc.getOutputStream();
// Envía el fichero mediante la clase enviaFich
enviaFich fich = new enviaFich(os);

. . .
```

Mediante algunas de las clases contenidas en el paquete *java.io* podemos enviar datos a través de la red byte a byte e incluso carácter a carácter, pero lo que nosotros necesitamos mandar son archivos de texto completos. Por este motivo, no nos queda más remedio que recurrir a la serialización, que es un proceso por el cuál un objeto cualquiera se convierte en una secuencia de bytes con la que más tarde se reconstruirá dicho objeto manteniendo el valor de sus variables. Esto permite guardar un objeto en un archivo o mandar un archivo por la red. Para que una clase pueda utilizar la

serialización, debe implementar la interfaz *Serializable*. La forma de implementarlo fue la siguiente:

```
class enviaFich  implements Serializable{
   public enviaFich(OutputStream os){
      try{
      // Asocia el ObjectOutputStream al OutputStream por el que
      // se enviará el fichero
      ObjectOutputStream objout = new ObjectOutputStream(os);
      objout.writeObject("c:/ficheros/" + FuncionesGUI.nombreFich
            + ".txt");
      // Cerramos el canal de salida
      objout.close();
      }
   catch(Throwable t){

. . .

  }
}
```

Como hemos visto, para escribir un objeto en el canal de salida se utiliza la clase ObjectOutputStream mediante el método writeObject(Object o). Se le pasa como parámetro el canal de salida OutputStream que se obtuvo de la conexión URL que establecimos. Llegados a este punto sólo nos quedaría cerrar los canales de entrada y salida que todavía quedan abiertos:

```
// Ya solo nos queda cerrar la conexión
is.close(); // Cierre del InputStream
os.close(); // Cierre del OutputStream
```

Botón Limpiar Datos

Al igual que en el caso de la aplicación cliente, este botón simplemente limpia de datos todos los campos de la interfaz. Esta función se implementó para facilitar y agilizar la tarea del médico encargado de rellenar los formularios de contestación a la Gestoría de Usuarios con los resultados de las consultas referentes a CmA en su modalidad de acto único procedentes de los centros ambulatorios.

Botón Cancelar

Del mismo modo que vimos en la aplicación cliente, el oyente de eventos asociado a este botón simplemente hace que salgamos de la aplicación (en este caso de la de contestación a la Gestoría de Usuarios) sin guardar cambios.