Proyecto Final de Carrera
Sistema criptográfico para la gestión segura de historiales médicos a través de la red

David Chiner Benjuya
Enginyeria d'Informàtica
Universitat Oberta de Catalunya

Jordi Castellà Roca
Consultor

7 de enero de 2008

A mi madre Nicole y a mi familia por apoyarme en mi ya larga vida de estudiante.

A todo el personal de la Universitat Oberta de Catalunya por esta estupenda modalidad *on-line* de proseguir mis estudios.

Resumen

En nuestro proyecto final de la carrera de ingeniería informática en la Universitat Oberta de Catalunya (UOC) nos centraremos en la seguridad informática. En concreto, exploraremos las posibilidades de la criptografía para garantizar la consulta y modificación seguras de historiales médicos a través de una red de comunicaciones.

Los historiales médicos contienen datos personales que deben ser protegidos de accesos no autorizados. Así, un empleado tiene derecho a ocultar a su empresa su propensión a padecer ciertas enfermedades. No obstante, la rápida consulta de un historial puede salvar vidas si permite por ejemplo que un médico sepa de la alergia a ciertos medicamentos de su paciente. Por todo ello, se hace necesario el uso de tecnologías informáticas que garanticen estos requisitos casi contradictorios: la seguridad y la rapidez de acceso.

Mediante el uso de tecnologías como Java, OpenSSL o XML implantaremos un sistema informático donde el acceso al contenido de los historiales cumplirá las condiciones de seguridad de un entorno PKI: confidencialidad, autenticidad, integridad y no-repudio. También se controlarán las altas y bajas de los protagonistas del sistema: los médicos y los pacientes.

El sistema desarrollado tendrá un carácter distribuido para simular, por ejemplo, la consulta de un mismo historial desde diferentes hospitales. En el centro de esta arquitectura situaremos al gestor del sistema, que contará con una base de datos para guardar la información tanto de los historiales como de los médicos y pacientes.

Palabras clave: historial, médico, paciente, seguridad, certificado, criptografía, Eclipse, IAIK, Java, JDOM, MySQL, PKI, RMI, SQL, UML, XML.

Área del proyecto: *Seguretat Informàtica*.

Índice general

Índice de figuras

Capítulo 1

Introducción

Contenido

Este capítulo describe las principales características de nuestro proyecto final de carrera.

1.1. Justificación y contexto del proyecto

Un historial médico es un documento con información sobre enfermedades personales y familiares (presentes y pasadas), medicamentos, vacunas, alergias y otros datos médicos importantes. Los sistemas informáticos ofrecen un soporte idóneo para gestionar de manera eficiente dicha información. Además, las redes de comunicaciones permiten consultarlos o modificarlos de forma remota. En esta línea, el Servicio Andaluz de Salud promueve el sistema informático Diraya, una historia de salud digital única que "posibilitará a los profesionales sanitarios que asisten a un mismo paciente tener acceso a la información clínica en cualquier centro sanitario de la geografía andaluza".[1]

No obstante, los historiales médicos son datos protegidos por el nivel alto de la ley orgánica de protección de datos de carácter personal (LOPD).[2] Por ello, son necesarias medidas que preserven esta información sensible frente a accesos o modificaciones no deseados.

[1] Diraya, Historia digital del ciudadano [en línea]
http://www.juntadeandalucia.es/salud/contenidos/.../Diraya_Estrategia_Digital.pdf

[2] "Los ficheros que contengan datos de ideología, religión, creencias, origen racial, salud o vida sexual así como los que contengan datos recabados para fines policiales sin consentimiento de las personas afectadas deberán reunir, además de las medidas de nivel básico y medio, las calificadas como de nivel alto." Real Decreto 994/1999.

1.2. Objetivos del PFC

En este proyecto final de carrera exploraremos las posibilidades de la criptografía informática en el contexto de la gestión segura de historiales médicos a través de una red de comunicaciones. Así, el uso de una infraestructura de clave pública (*Public Key Infrastructure* o PKI) nos permitirá garantizar las siguientes medidas de protección en la gestión de historiales:

- **Confidencialidad**. Sólo usuarios autorizados podrán consultar los historiales.
- **Autenticidad**. La autoría de los historiales no podrá ser suplantada.
- **Integridad**. Los historiales no podrán ser modificados por personas no autorizadas.
- **No-repudio**. Los usuarios que consulten o modifiquen un historial no podrán negar haberlo hecho.

Estas condiciones de seguridad se cumplirán en un entorno de consultas remotas a un sistema gestor que centralizará toda la información crítica (historiales, usuarios autorizados). Los principales casos de uso a los que se adaptará el sistema serán los siguientes:

- El sistema da de alta un usuario (médico o paciente) con los privilegios de lectura y edición correspondientes.
- El sistema autentica a un usuario y evalúa sus privilegios para determinar qué acciones puede realizar.
- El sistema permite consultar un historial a un usuario autorizado.
- El sistema añade a un historial una visita médica redactada por un usuario autorizado.

Vea en la sección 3.4. Casos de uso, pág. 18, una descripción exhaustiva de los casos de uso soportados por el sistema.

1.3. Productos obtenidos

Para implementar nuestro PFC, hemos desarrollado tanto una API en lenguaje Java como las aplicaciones detalladas en esta sección. Vea en el capítulo 8. Manual de uso, pág. 51, una descripción más detallada de estas herramientas.

1.3.1. Aplicación Gestor de Historiales

La aplicación *Gestor de Historiales* funciona en modo servidor para atender las peticiones de consulta y modificación de historiales. Desde la GUI de esta aplicación, el administrador del sistema puede configurar los diferentes parámetros del sistema (PKCS #12, base de datos de historiales, servidor RMI).

1.3.2. Aplicación Paciente

Mediante la aplicación *Paciente*, un usuario puede consultar su propio historial. Para las tareas de autenticación, debe seleccionar el archivo que contiene el PKCS #12 con sus claves, e indicar la contraseña correspondiente.

1.3.3. Aplicación Médico

Mediante la aplicación *Médico*, un médico puede realizar las siguientes operaciones:

- Consultar el historial de un paciente.
- Consultar su lista de pacientes.
- Añadir una visita al historial de un paciente.

Como el paciente, el médico se autentica indicando el fichero y la contraseña del PKCS #12 que contiene su par de claves.

1.3.4. Aplicación Gestor de Usuarios

La aplicación *Gestor de Usuarios* permite realizar las siguientes operaciones sobre los usuarios (médicos y pacientes) del sistema:

- Registrar a un usuario (médico o paciente) a partir de sus datos personales y su certificado digital.
- Dar de baja a un usuario (médico o paciente).
- Asignar un paciente a un médico.
- Eliminar la asignación de un paciente a un médico.
- Mostrar la lista de pacientes.
- Mostrar la lista de médicos.
- Mostrar la lista de pacientes de un médico.

1.4. Sumario de capítulos

Esta sección resume el contenido de cada capítulo de la memoria.

- El capítulo 1. Introducción, pág. 8, describe las principales características del proyecto.
- El capítulo 2. Desarrollo del proyecto, pág. 12, describe las etapas de elaboración del proyecto.

- El capítulo 3. Especificación, pág. 15, describe los principales requisitos del proyecto.
- El capítulo 4. Diseño, pág. 20, describe las principales decisiones de diseño del proyecto.
- El capítulo 5. Herramientas utilizadas, pág. 31, describe el entorno tecnológico utilizado para implementar el proyecto.
- El capítulo 6. Esquema criptográfico, pág. 35, describe los protocolos y las entidades protagonistas del entorno seguro de gestión de historiales.
- El capítulo 7. Esquemas XML, pág. 46, describe los esquemas de los documentos XML utilizados para intercambiar información a través de la red.
- El capítulo 8. Manual de uso, pág. 51, explica cómo utilizar las aplicaciones desarrolladas.
- El capítulo 9. Juego de pruebas, pág. 58, describe un ejemplo completo de ejecución del sistema.
- El capítulo 10. Gestión de errores, pág. 62, describe los mensajes de error mostrados por el sistema.
- El capítulo 11. Conclusiones, pág. 65, enumera las conclusiones del proyecto y apunta los aspectos pendientes que podrían desarrollarse en el futuro.

Capítulo 2

Desarrollo del proyecto

Contenido

Este capítulo describe la planificación inicial del proyecto y su cumplimiento a largo del tiempo.

2.1. Planificación del proyecto

El periodo de tiempo dedicado al proyecto es de 112 días, desde el 19 de septiembre 2007 al 7 de enero de 2008 (no distinguiremos entre días laborables y festivos). Las fechas clave del proyecto coinciden con la distribución en PAC (pruebas de evaluación continuada) regularmente entregadas al consultor.

Esta sección describe las etapas del proyecto.

2.1.1. Entorno de desarrollo (19/09/07-23/09/07)

En esta primera etapa realizamos las siguientes tareas:

- Configurar el entorno de desarrollo: J2SE, librerías criptográficas, OpenSSL, Eclipse, LaTeX, etc.
- Redactar el primer borrador de la documentación.
- Generar con OpenSSL las claves y certificados que utilizarán los actores del sistema para cifrado y autenticación.

2.1.2. Protocolos criptográficos (24/09/07-21/10/07)

En esta etapa se estudian, evalúan e implementan los protocolos descritos en la sección 6.4. Protocolos. La especificación de los protocolos es la propuesta por el consultor, a la que no hemos añadido cambios relevantes. Respecto a la implementación, los protocolos han sido codificados en lenguaje Java mediante las librerías descritas en el capítulo 5. Herramientas utilizadas, pág. 31. De esta etapa resulta el grueso de la API del proyecto, con todas las operaciones de cifrado, firma e intercambio de datos resueltas.

2.1.3. Documentos XML (22/10/07-4/11/07)

En esta etapa de especifican los documentos XML con los que intercambiar información entre los clientes (aplicaciones médico y paciente) y el servidor (aplicación gestor). La especificación incluye tanto los esquemas XML como las clases Java para crear y gestionar los documentos.

2.1.4. Invocación remota (5/11/07-18/11/07)

En esta etapa se añade soporte para invocaciones remotas al sistema gestor mediante Java RMI (Remote Method Invocation). De esta manera, el sistema, inicialmente diseñado para invocaciones locales, se adapta a un escenario distribuido.

2.1.5. Base de datos (19/11/2007-2/12/2007)

En esta etapa se integra al sistema una base de datos para guardar de forma persistente tanto los historiales como los datos de usuario. Tal como se explica en el capítulo 5. Herramientas utilizadas, pág. 31, el sistema gestor de base de datos escogido es MySQL.

2.1.6. Interfaces cliente (3/12/2007-16/12/2007)

En esta etapa se desarrollan las interfaces gráficas para que médicos y pacientes ejecuten los protocolos de consulta e inserción.

2.1.7. Interfaz gestor (17/12/2007-30/12/2007)

En esta etapa de desarrolla la interfaz gráfica desde donde administrar el funcionamiento del sistema gestor de historiales.

2.1.8. Documentación del proyecto (31/12/2007-7/01/2008)

En la etapa final del proyecto se ultima la documentación, con especial atención a los juegos de pruebas y las conclusiones.

2.2. Carga real

El principal desfase de la carga real respecto a la planificada la encontramos en la segunda etapa, 2.1.2. Protocolos criptográficos (24/09/07-21/10/07) (pág. 13). Para realizar esta etapa hemos invertido al menos una semana más del tiempo planificado. Las razones del desfase incluyen desde no leer bien los enunciados hasta cambiar de librería criptográfica a mitad de la implementación.

En cambio, para las demás etapas de desarrollo, el tiempo dedicado ha sido igual o menor al estimado, especialmente en etapas como las siguientes, que teníamos prácticamente resueltas por otras prácticas de la carrera:

- La etapa 2.1.3. Documentos XML (22/10/07-4/11/07) coincidía en muchos aspectos con la práctica de la asignatura *Criptografia*.
- La etapa 2.1.4. Invocación remota (5/11/07-18/11/07) coincidía en muchos aspectos con la práctica de la asignatura *Arquitectura de sistemes distribuïts*.

2.3. Evaluación

La carga de trabajo de nuestro proyecto ha cumplido con creces las características típicas de un PFC:

- Excede el número de créditos de la asignatura correspondiente (en nuestro caso, 9 créditos).
- Se distribuye casi a partes iguales en codificación y documentación.

Con todo, ha merecido la pena por la posibilidad de poner en práctica los conocimientos adquiridos durante la carrera, especialmente los del área de seguridad informática.

Capítulo 3

Especificación

Contenido

Durante la fase de especificación se define de forma precisa el sistema informático que se desea implementar. Este capítulo describe las conclusiones de la fase de especificación de nuestro sistema gestor de historiales.

3.1. Requisitos funcionales

Los requisitos funcionales de un sistema definen las funcionalidades que debe ofrecer a sus usuarios y resumen los Casos de uso (descritos en la sección 3.4, pág. 18). En concreto, nuestro sistema de gestión de historiales tiene los requisitos funcionales detallados en esta sección.

3.1.1. Persistencia de historiales y usuarios

El sistema debe guardar de forma persistente los historiales de los pacientes así como un registro de usuarios autorizados (médicos o pacientes).

3.1.2. Consulta de historiales

El sistema debe permitir consultar un historial médico a los siguientes usuarios:

- El titular del historial, esto es, el paciente cuyas patologías se describen.
- Un médico asignado al titular del historial.

Dicha consulta debe poder ejecutarse de manera remota (a través de una red de comunicaciones) desde una aplicación cliente con interfaz de usuario.

3.1.3. Consulta de pacientes

El sistema debe permitir que un médico consulte la lista de sus pacientes. Dicha consulta debe poder ejecutarse de manera remota, a través de una red de comunicaciones, desde una aplicación cliente con interfaz de usuario.

3.1.4. Inserción de visitas

El sistema debe permitir que un médico añada la información de una visita al historial de un paciente. Una vez insertada, dicha visita no puede ser eliminada (para que el médico no pueda negar su autoría). Las visitas añadidas al historial de un paciente deben incluir al menos la siguiente información:

- DNI del médico.
- Descripción de la visita.
- Fecha de inserción en el historial.

De esta manera, cada historial se compone de N visitas asociadas a un identificador de paciente.

La inserción de visitas debe poder ejecutarse de manera remota, a través de una red de comunicaciones, desde una aplicación cliente con una interfaz de usuario.

3.2. Requisitos no funcionales

Los requisitos no funcionales especifican propiedades del sistema como el rendimiento, la velocidad, el uso de memoria o la plataforma. Nuestro sistema de gestión de historiales tiene los requisitos no funcionales descritos en esta sección.

3.2.1. Acceso remoto desde diferentes ubicaciones

Nuestro sistema debe permitir que las aplicaciones cliente puedan ejecutarse desde diferentes ubicaciones. No obstante, la concurrencia no se considera un requisito de nuestro proyecto. Tampoco se tomarán medidas para garantizar la transaccionalidad en la inserción de visitas en un historial.

3.2.2. Conexión con diferentes bases de datos

La información del sistema (historiales, asignaciones de médicos a pacientes, certificados de usuario) debe guardarse en una base de datos externa. De esta manera, la persistencia de datos no debe estar embebida en el sistema, sino que debe permitirse el uso de diferentes DBMS para dicho cometido.

3.2.3. Aplicación multiplataforma

Nuestro sistema debe desarrollarse en una tecnología que permita su implantación en diferentes sistemas operativos. Por ejemplo, desde un cliente que se ejecuta sobre Windows debe poder consultarse un historial guardado en una base de datos que se ejecuta sobre Linux.

3.3. Actores del sistema

En esta sección describimos los tres actores de nuestro sistema: el paciente, el médico y el gestor de historiales.

3.3.1. Usuario paciente

En nuestro sistema, uno o varios pacientes pueden consultar sus historiales a través de la red. Los pacientes se identifican mediante su DNI; para las operaciones de cifrado y autenticación, cuentan con un par de claves (certificado por la Autoridad de Certificación del sistema).

3.3.2. Usuario médico

En nuestro sistema, uno o varios médicos pueden realizar las siguientes operaciones:

- Consultar historiales de sus pacientes.
- Añadir visitas a los historiales de sus pacientes.
- Consultar su lista de pacientes.

Como los pacientes, los médicos se identifican mediante su DNI y cuentan con un par de claves (certificado por la Autoridad de Certificación del sistema) para las operaciones de cifrado y autenticación.

3.3.3. Gestor de historiales

En nuestro sistema, una aplicación *Gestor* centraliza la siguiente información en una base de datos:

- Historiales de pacientes.
- Asignaciones médico-paciente.
- Certificados de los médicos y los pacientes.

Para las operaciones de cifrado y autenticación, el *Gestor* cuenta con un par de claves certificado por la Autoridad de Certificación del sistema.

El administrador del sistema *Gestor* puede realizar además las siguientes operaciones sobre los usuarios:

- Registrar médicos o pacientes en el sistema.
- Dar de baja médicos o pacientes en el sistema.
- Asignar un paciente a un médico.
- Eliminar la asignación de un paciente a un médico.

3.4. Casos de uso

Los casos de uso describen en un lenguaje no técnico la interacción del sistema con un usuario o con otro sistema para lograr un objetivo. Esta sección detalla los casos de uso de nuestro sistema gestor de historiales (figura 3.1).

Figura 3.1: Casos de uso del sistema gestor de historiales

3.4.1. Inserción de una visita

Tras atender a un paciente P, un médico M accede a su interfaz de usuario y anota el transcurso de la visita (DNI del paciente, patologías detectadas, medicamentos recetados, etc.). El médico solicita al sistema añadir la visita al historial del paciente. El sistema comprueba que P y M estén efectivamente registrados en el sistema y mantengan una relación médico-paciente. Si la comprobación tiene éxito, el sistema añade la visita al historial de P, dejando constancia del DNI del médico y la fecha de inserción.

3.4.2. Consulta de un historial por parte de un paciente

Un paciente accede a su interfaz de usuario, se identifica en el sistema y solicita consultar su historial. El sistema comprueba que el paciente esté registrado

en el sistema. Si la comprobación tiene éxito, muestra el historial en la interfaz del paciente.

3.4.3. Consulta de un historial por parte de un médico

Un médico M accede a su interfaz de usuario y solicita al sistema consultar el historial de un paciente P. El sistema comprueba que P y M mantengan una relación médico-paciente. Si la comprobación tiene éxito, el sistema muestra en la interfaz del médico el historial del paciente.

3.4.4. Consulta de los pacientes de un médico

Un médico accede a su interfaz de usuario y solicita al sistema consultar el nombre y DNI de todos sus pacientes. El sistema comprueba que el médico esté registrado como tal. Si la comprobación tiene éxito, el sistema muestra la lista de pacientes en su interfaz de usuario.

Capítulo 4

Diseño

Contenido

Este capítulo describe las decisiones de diseño adoptadas para implementar nuestro sistema.

4.1. Arquitectura

La arquitectura de nuestro sistema gestor de historiales (figura 4.1) sigue un clásico esquema cliente-servidor con las siguientes particularidades:

- Un único servidor concentra en una base de datos toda la información sobre historiales y usuarios registrados.
- Desde las aplicaciones cliente de los pacientes pueden consultarse los historiales.
- Desde las aplicaciones cliente de los médicos pueden consultarse los historiales, las listas de pacientes e insertar visitas. Al no haberse implementado la concurrencia, esta última operación, la inserción de visitas, no puede realizarse simultáneamente sobre un mismo historial desde diferentes aplicaciones médico.

Como las aplicaciones cliente deben realizar aplicaciones criptográficas de firma y cifrado, no ha sido posible implementar clientes ligeros que accedan a los servicios mediante navegador Web. Así, además de la capa de presentación, nuestros clientes incluyen una capa de datos con lógica de negocio.

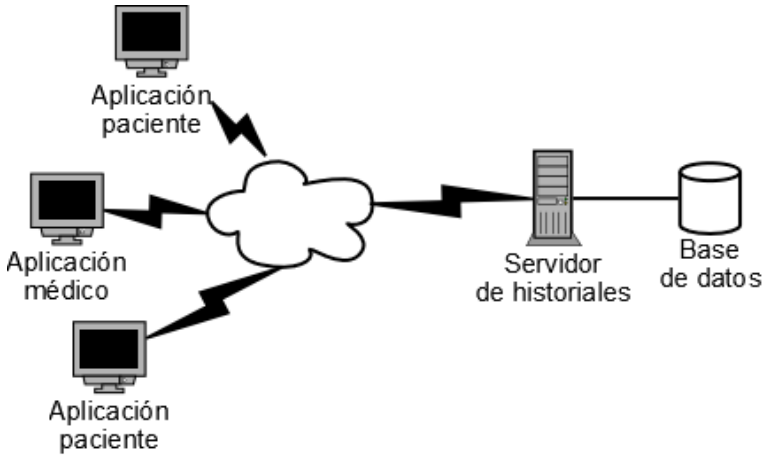

Figura 4.1: Arquitectura del sistema gestor de historiales

4.2. Diagrama de clases

Esta sección describe, en lenguaje UML, las relaciones entre las principales clases desarrolladas para nuestro proyecto.

4.2.1. Clases para gestionar documentos XML

Para gestionar los documentos XML intercambiados entre cliente y servidor hemos desarrollado las siguientes clases (figura 4.2):

- **HistXmlData**. Clase padre que define las principales operaciones y atributos.
- **HistXmlDoctorPatients**. Clase para gestionar las listas de pacientes asignados a un médico.
- **HistXmlServerResult**. Clase para gestionar los mensajes de error o confirmación del servidor.
- **HistXmlNeedhamMsg**. Clase para gestionar el intercambio de datos durante el protocolo de autenticación Needham-Schroeder.
- **HistXmlHistorial**. Clase para gestionar los historiales de paciente.
- **HistXmlVisita**. Herencia de **HistXmlHistorial** para gestionar la inserción o consulta de visitas de un historial.

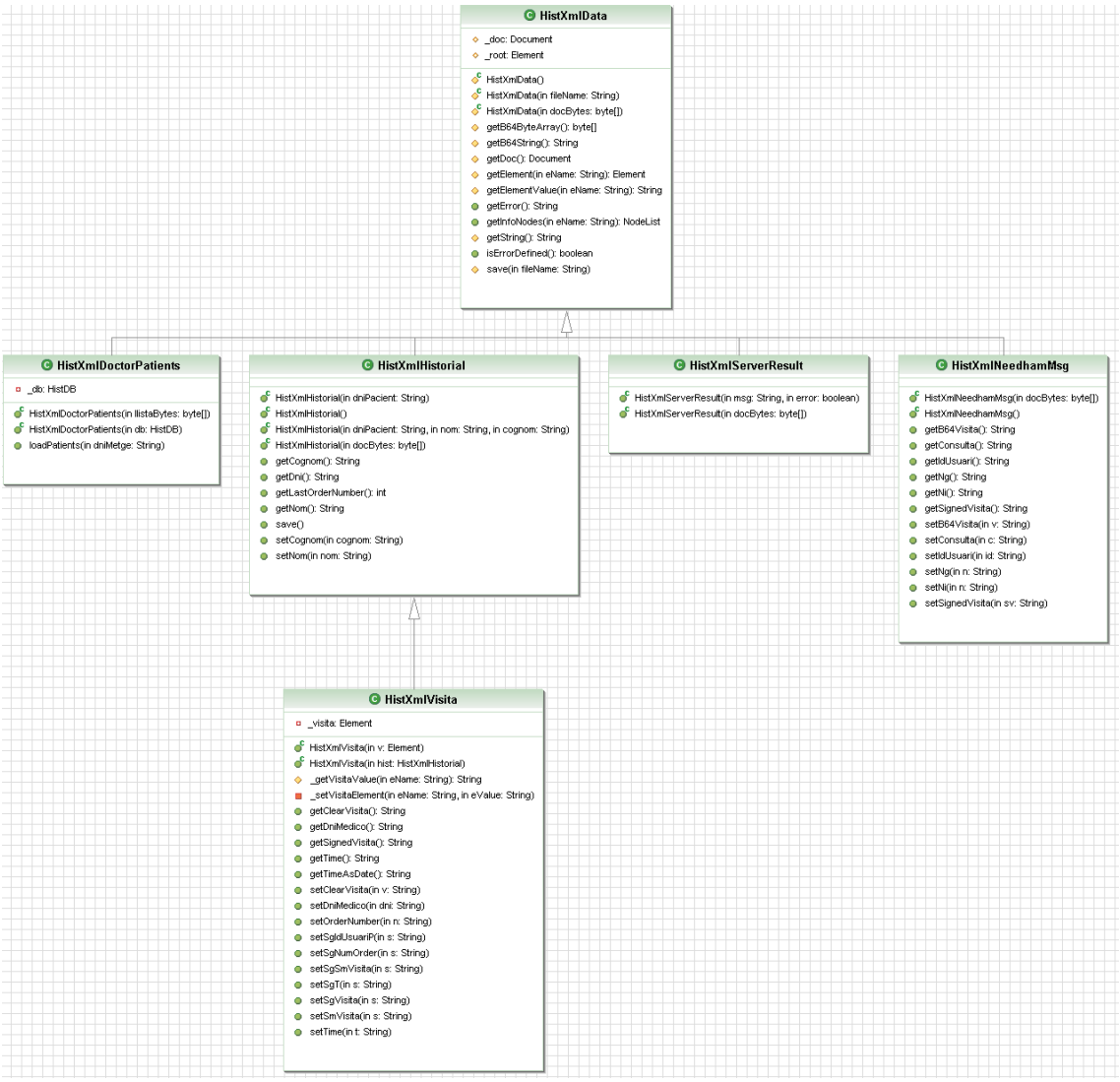

Figura 4.2: Clases para gestionar documentos XML

4.2.2. Clases para ejecutar los protocolos criptográficos

Para gestionar los protocolos criptográficos de consulta y modificación remotas de datos médicos hemos desarrollado las siguientes clases (figura 4.3):

- **HistProtocol**. Clase padre que define las principales operaciones y atributos.
- **HistProtocol3InsercionVisita**. Clase para gestionar la inserción de vi-

sitas en un historial.

- **HistProtocol2ConsultaPacientes**. Clase para consultar la lista de pacientes asignada a un médico.
- **HistProtocol1ConsultaHistorial**. Clase para consultar el historial de un paciente.

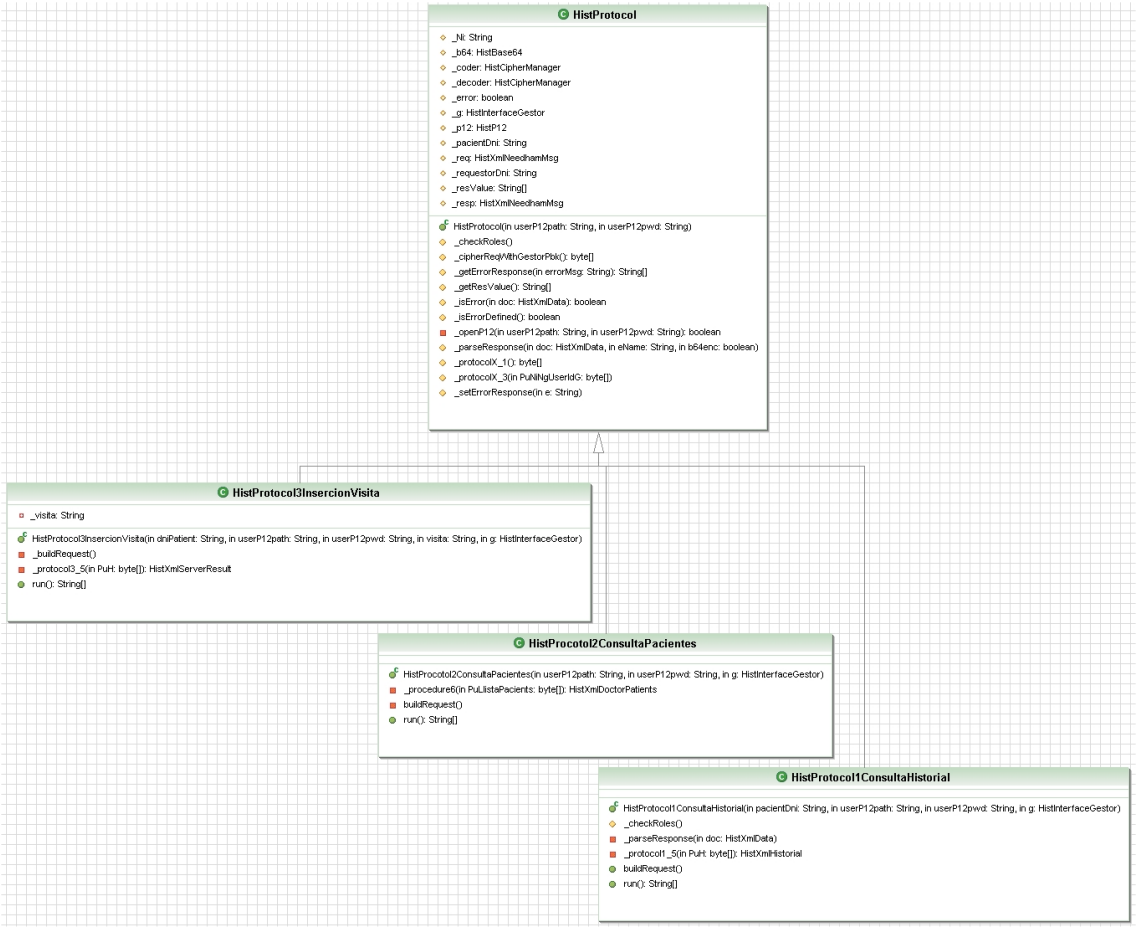

Figura 4.3: Clases para gestionar los protocolos criptográficos

4.2.3. Clases para gestionar clientes

Para gestionar los clientes (médicos y pacientes) del sistema gestor de historiales hemos desarrollado las siguientes clases (figura 4.4):

- **HistPaciente**. Clase para gestionar la consulta de historiales de un paciente.
- **HistMedico**. Herencia de **HistPaciente** que añade las operaciones de inserción de visitas y listado de pacientes asignados a un médico.

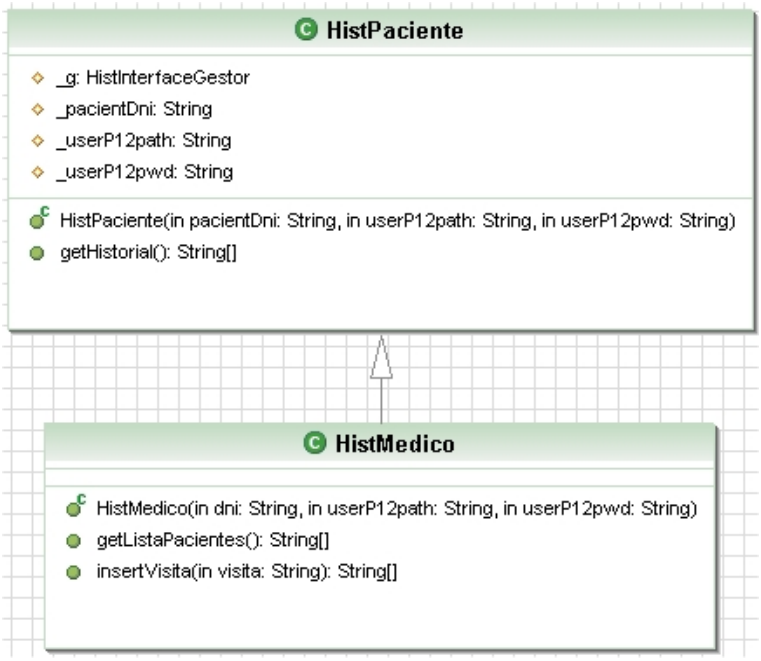

Figura 4.4: Clases para gestionar los clientes

4.2.4. Clases para gestionar la interfaz gráfica

Para gestionar la interfaz gráfica de las aplicaciones gestor, médico y paciente hemos desarrollado las siguientes clases (figura 4.5):

- **HistGuiManager**. Clase padre que define las principales operaciones y atributos.
- **HistGestorGuiManager**. Clase para gestionar la interfaz gráfica del gestor.
- **HistClientGuiManager**. Clase para gestionar la interfaz gráfica de médicos y pacientes.
- **HistFileChooser**. Clase para gestionar el cuadro de selección de archivos utilizado para importar o exportar contenidos.

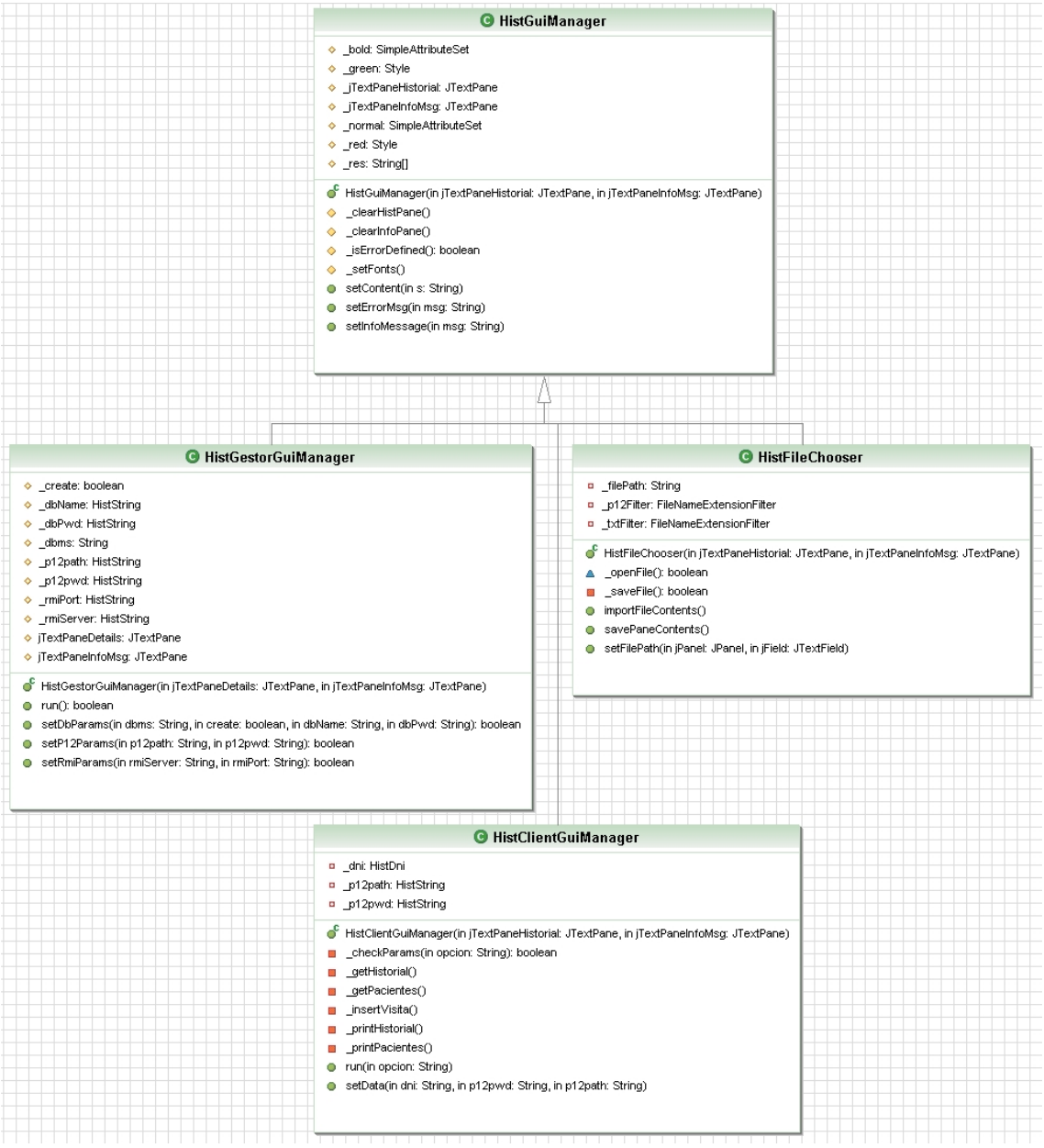

Figura 4.5: Clases para gestionar la interfaz gráfica

4.3. Modelo de base de datos

En la base de datos para guardar los historiales y la información de usuario, se han definido las siguientes tablas (figura 4.6):

- **Pacientes**. Esta tabla guarda la siguiente información sobre los pacientes registrados en el sistema: certificado, historial y DNI (clave primaria).
- **Médicos**. Esta tabla guarda la siguiente información sobre los médicos registrados en el sistema: certificado y DNI (clave primaria).
- **MedicoPacientes**. Esta tabla asociativa guarda las asignaciones de pacientes a médicos en forma de par <DNI médico, DNI paciente>.
- **Randoms**. Esta tabla guarda los valores aleatorios Ni y Ng generados durante la ejecución del Protocolo de autenticación (descrito en la sección 6.4.2, pág. 40).

Por la potencia que ofrece el formato XML, hemos preferido no repartir las diferentes visitas de un historial como entradas de una tabla. El documento XML utilizado para gestionar los historiales permite insertar visitas en forma de nuevo elemento.

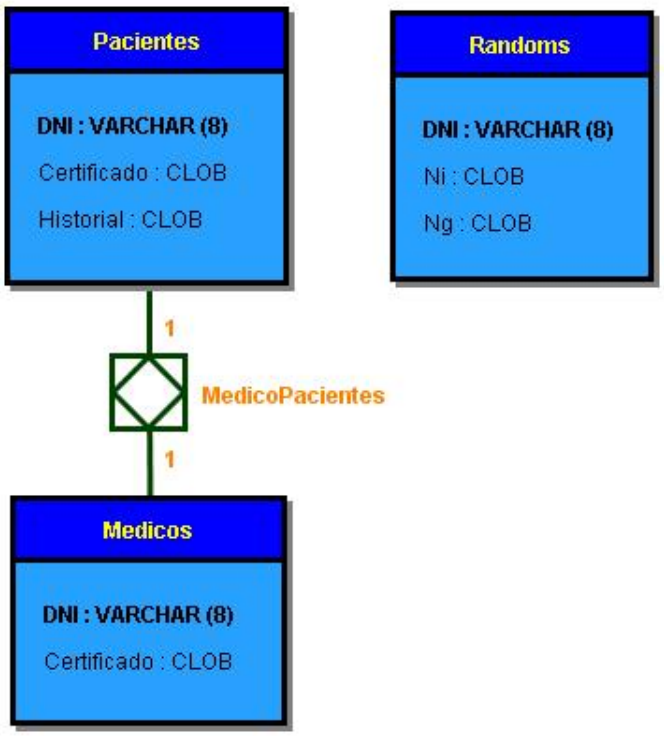

Figura 4.6: Modelo de base de datos

4.4. Diagramas de secuencia

Esta sección describe, en lenguaje UML, la secuencia de invocaciones de las principales operaciones del sistema.

4.4.1. Consulta de historiales

El siguiente diagrama de secuencia (figura 4.7) ilustra la serie de invocaciones para obtener en historial de un paciente (a partir del método *run* de la clase *HistProtocol1ConsultaHistorial*). Las invocaciones remotas se realizan sobre el *stub* de la clase *HistGestor*.

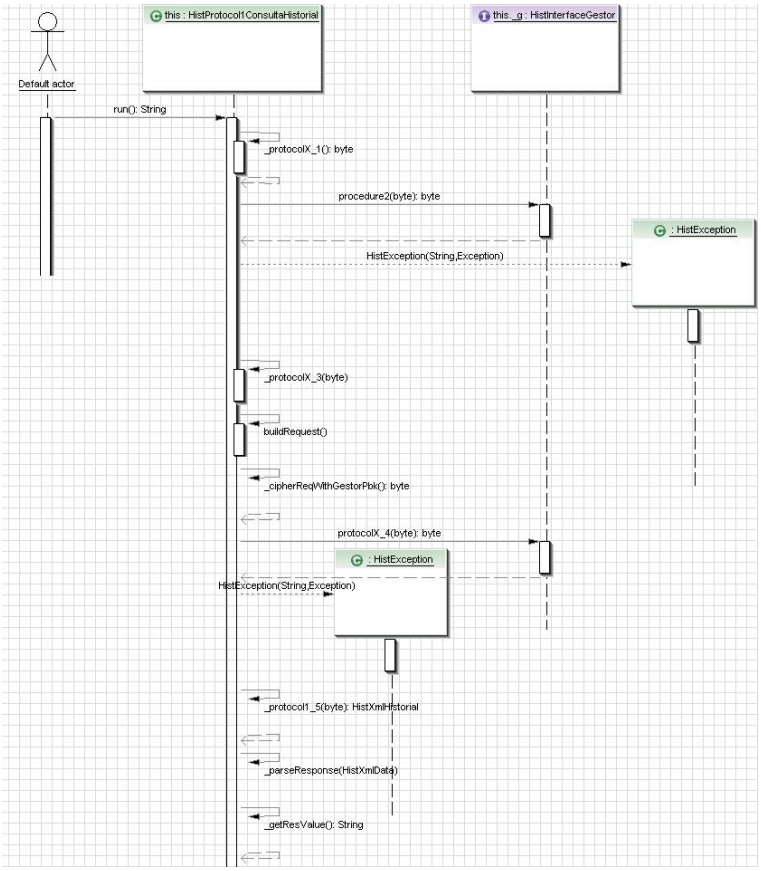

Figura 4.7: Diagrama de secuencia para consultar un historial

4.4.2. Consulta de pacientes

El siguiente diagrama de secuencia (figura 4.8) ilustra la serie de invocaciones para consultar la lista de pacientes de un médico (a partir del método *run* de la clase *HistProcotol2ConsultaPacientes*). Las invocaciones remotas se realizan sobre el *stub* de la clase *HistGestor*.

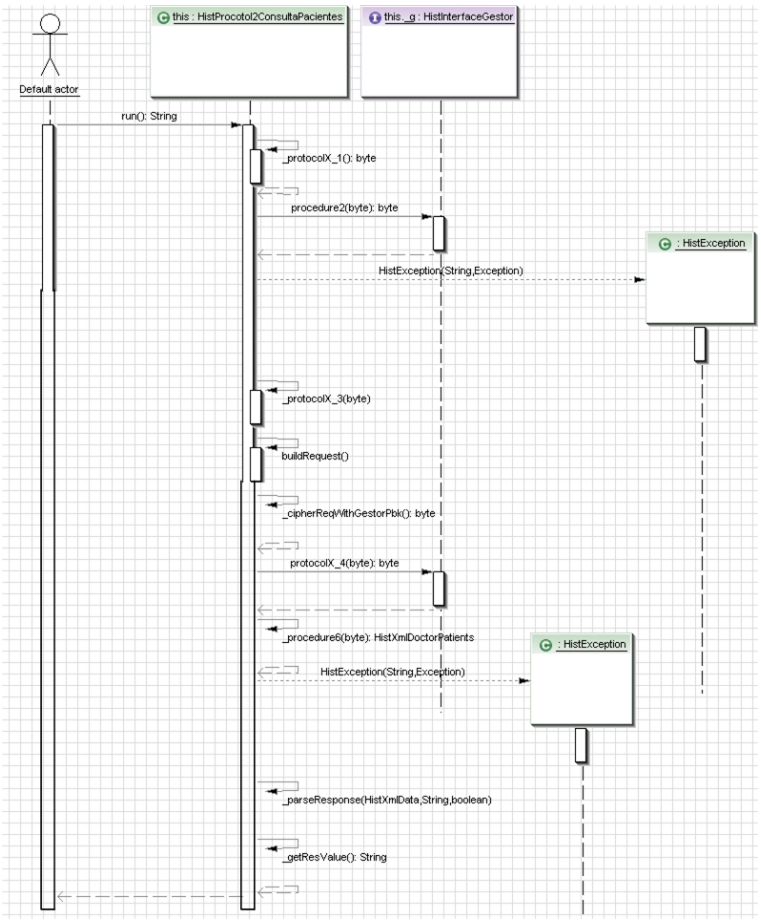

Figura 4.8: Diagrama de secuencia para consultar de pacientes

4.4.3. Inserción de visitas

El siguiente diagrama de secuencia (figura 4.9) ilustra la serie de invocaciones para insertar una visita en un historial (a partir del método *run* de la clase *HistProtocol3InsercionVisita*). Las invocaciones remotas se realizan sobre el *stub* de la clase *HistGestor*.

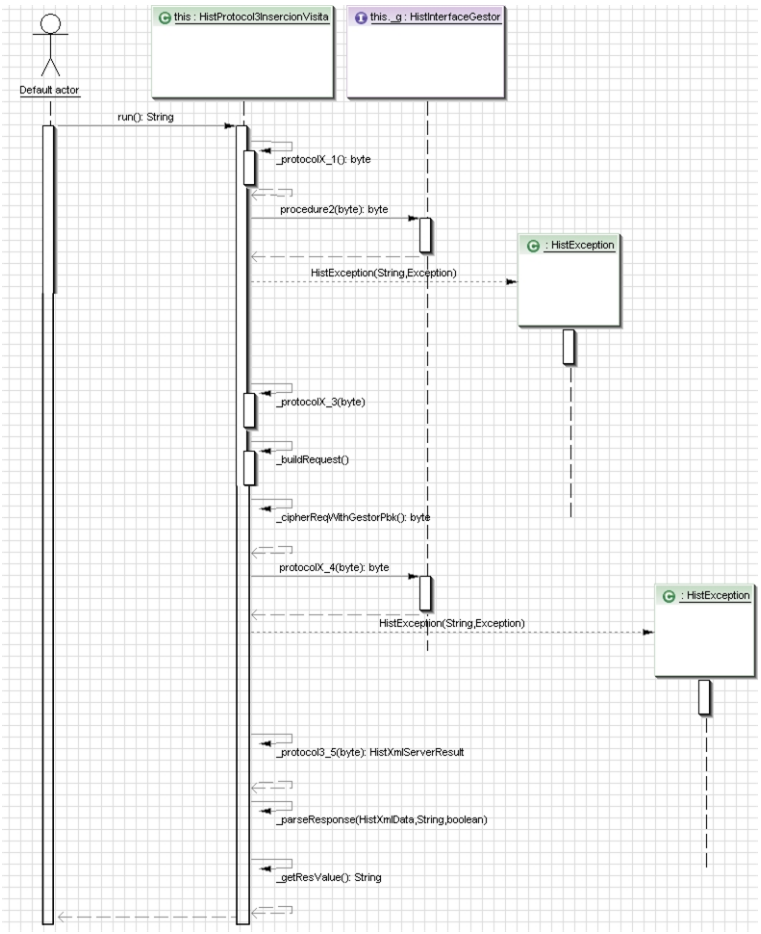

Figura 4.9: Diagrama de secuencia para insertar una visita

Capítulo 5

Herramientas utilizadas

Contenido

Este capítulo describe las herramientas utilizadas para realizar nuestro proyecto.

5.1. Java

Toda la programación del proyecto se ha realizado en lenguaje Java. Aunque algo lento en la ejecución, Java ofrece grandes ventajas como la sencillez, la orientación a objetos o el volumen de soluciones disponibles.

5.1.1. JDK 6 Update 3

El conjunto de utilidades Java de desarrollo (*Java Development Kit* o JDK) utilizado ha sido el JDK 6 Update 3 de Sun Microsystems. Este entorno de desarrollo puede obtenerse gratuitamente desde la página de descargas del fabricante (http://java.sun.com/javase/downloads) y cuenta con una completa documentación en formato HTML (http://java.sun.com/javase/6/docs/api).

Para evitar las limitaciones de longitud de clave impuestas por las políticas de exportación a ciertos países, hemos sustituido las librerías *local_policy.jar* y *US_export_policy.jar* (incluidas en la distribución nativa del JDK) por las librerías con el mismo nombre distribuidas como *JCE Unlimited Strength Jurisdiction Policy Files*. Dichas librerías pueden obtenerse gratuitamente desde la misma página de descargas del JDK.

5.1.2. IAIK

Para las operaciones criptográficas con cadenas de bytes hemos utilizado la API para Java del IAIK (*Institute for Applied Information Processing and Communication* de la Universidad politécnica de Graz). Una vez instalada la librería correspondiente, esta API se integra como Java Cryptography Extension (JCE) de la plataforma JDK y ofrece funcionalidades de cifrado, firma, resumen (*hash*) o gestión certificados.

5.1.3. Apache Derby

Apache Derby, también distribuido por Sun Microsystems como Java DB, es un DBMS enteramente desarrollado en Java para crear bases de datos relacionales en archivo. Apache Derby nos ha permitido hacer pruebas con una base de datos ligera desde las primeras fases de nuestro proyecto, antes de alcanzar la etapa de integración con una base de datos más robusta como MySQL.

5.1.4. RMI

RMI (Java Remote Invocation Method) es un mecanismo incluido en el JDK para realizar invocaciones remotas entre diferentes objetos Java. De implementación mucho más sencilla que los Web Services y que sobre todo CORBA, RMI resuelve problemas como la serialización de parámetros o la "recolección de basura".

RMI es la tecnología que hemos utilizado en nuestro proyecto para implementar las conexiones remotas con el servidor de historiales. Por su sencillez, RMI nos ha permitido añadir fácilmente invocaciones remotas a un sistema que, en una primera etapa, desarrollamos como puramente local.

5.1.5. Eclipse

Para redactar el código Java hemos utilizado Eclipse, el entorno de desarrollo integrado (IDE) para Java. Eclipse permite gestionar las librerías (propias o externas), la ejecución de código o la generación de documentación con JavaDoc. A esta funcionalidad base hemos añadido los siguientes *plug-in*:

- Eclipse Visual Editor para crear interfaces de usuario mediante un asistente gráfico.
- Omondo para obtener los diagramas UML mediante ingeniería inversa.

5.1.6. JavaDoc

La utilidad JavaDoc incluida en el JDK nos ha permitido generar en formato HTML la documentación de la API desarrollada para nuestro proyecto. Los parámetros de ejecución de JavaDoc son plenamente configurables desde el IDE Eclipse.

5.2. XML

Para los mensajes de petición y respuesta entre las aplicaciones cliente y el sistema gestor de historiales hemos utilizado el lenguaje XML (*eXtensible Markup Language* o lenguaje de marcas extensible). De este modo, cada estructura de datos intercambiada (e.g. el DNI del paciente, el texto de una visita, la descripción de un error) puede asignarse como contenido del elemento correspondiente.

Los mensajes XML viajan cifrados a través de la red. Respecto a la granularidad de cifrado, hemos optado por cifrar todo el documento. Podríamos haber cifrado sólo el material sensible (e.g. el texto de las visitas) y dejar en claro información no confidencial (e.g. el DNI del paciente). De esta manera, el documento podría ser manipulado por personal no autorizado a visualizar todo el contenido. No obstante, cifrar todo el documento simplifica la implementación, si además tenemos en cuenta que la librería criptográfica IAIK utilizada no permite seleccionar los elementos XML que se desean cifrar.

También por sencillez hemos utilizado la API JDom para *parsear* los documentos XML. Esta API se incluye de forma nativa en la JDK de Sun como la interfaz Document.

5.3. OpenSSL

Para generar las claves y certificados de nuestro proyecto hemos utilizado Win32 OpenSSL. Esta utilidad es una implementación para plataforma Windows de la librería *open source* OpenSSL. Con Win32 OpenSSL hemos generado la jerarquía de claves y certificados descrita en la sección 6.1. Claves y certificados.

5.4. MySQL

Como sistema gestor de base de datos del sistema hemos optado por MySQL, el DBMS de software libre que, para proyectos no comerciales, se distribuye bajo la licencia GNU-GPL. MySQL cuenta además con el *driver* Connector/J 3.1 para Java que puede descargarse gratuitamente desde el sitio web del proyecto (http://dev.mysql.com/downloads/connector/j/3.1.html).

De esta manera, las funcionalidades inicialmente desarrolladas para un prototipo con el DBMS Apache Derby se han podido adaptar para MySQL cambiando simplemente el *driver* (y el tipo de datos utilizado para textos largos).

5.5. LaTeX

Para generar esta memoria en formato PDF hemos utilizado el sistema de edición LaTeX. Aunque un poco abstruso al principio, LaTeX permite redactar en texto plano una especificación exacta del resultado deseado. De esta manera,

se evitan sorpresas de última hora comunes entre los editores WYSIWG tipo Microsoft Word. Además, LaTeX resuelve automáticamente aspectos de diseño como la paginación.

5.5.1. MiKTeX

Para compilar el código LaTeX hemos utilizado MiKTeX, la distribución LaTeX para Microsoft Windows. MiKTeX ofrece entre otras ventajas la facilidad de actualización o la posibilidad de descargar e instalar durante la compilación los módulos aún no disponibles.

5.5.2. TeXnicCenter

TeXnicCenter es una IDE *open source* para LaTeX sobre plataforma Windows con la que hemos editado el texto de la memoria. TeXnicCenter soporta los correctores ortográficos de Open Office y permite configurar diferentes tipos de compilación para obtener documentos DVI, PostScript o PDF.

Capítulo 6

Esquema criptográfico

Contenido

Este capítulo describe el esquema criptográfico del proyecto, desde las entidades hasta los procedimientos utilizados.

6.1. Claves y certificados

Esta sección describe las claves y certificados digitales utilizados por los actores del sistema para tareas de autenticación o cifrado. Para generar estas entidades, hemos utilizado la librería criptográfica Win32 OpenSSL descrita en la sección 5.3. OpenSSL, pág. 33.

6.1.1. Claves y certificado de la Autoridad de Certificación

Para firmar los certificados de los actores del sistema (paciente, médico, gestor de historiales) hemos generado un par de claves y un certificado autofirmado de Autoridad de Certificación raíz. Las propiedades del certificado son las siguientes:

- **Algoritmo de firma**: sha1RSA
- **Emisor**: E = capfc@uoc.edu, CN = CA_PFC_Seguretat, OU = PFC_Seguretat, O = UOC, L = Barcelona, S = Barcelona, C = ES
- **Válido desde**: jueves, 11 de octubre de 2007 11:30:15
- **Válido hasta**: domingo, 05 de octubre de 2008 11:30:15

- **Titular**: E = capfc@uoc.edu, CN = CA_PFC_Seguretat, OU = PFC_Seguretat, O = UOC, L = Barcelona, S = Barcelona, C = ES
- **Clave pública**: RSA de 2048 bits
- **Restricciones básicas**: Tipo de asunto=Entidad emisora de certificados (CA)

6.1.2. Claves y certificado del paciente

Para cada actor con el rol de paciente en el sistema de gestión de historiales se ha generado un par de claves RSA de 1024 bits. Para certificar dicho par de claves, se ha generado un certificado (firmado por la CA del sistema) con propiedades como las siguientes:

- **Algoritmo de firma**: sha1RSA
- **Emisor**: E = capfc@uoc.edu, CN = CA_PFC_Seguretat, OU = PFC_Seguretat, O = UOC, L = Barcelona, S = Barcelona, C = ES
- **Válido desde**: jueves, 11 de octubre de 2007 11:47:34
- **Válido hasta**: viernes, 10 de octubre de 2008 11:47:34
- **Titular**: E = jesquifit@uoc.edu, CN = Joan Esquifit [DNI:40105232], OU = Pacients, O = UOC, S = Barcelona, C = ES
- **Clave pública**: RSA de 1024 bits
- **Restricciones básicas**: Tipo de titular=Entidad final
- **Uso de clave**: Firma digital, No-repudio, Cifrado de clave

Cada par de claves se ha incluido además con su cadena de certificación en un contenedor PKCS #12 protegido mediante contraseña.

6.1.3. Claves y certificado del médico

Para cada actor con el rol de médico en el sistema de gestión de historiales se ha generado un par de claves RSA de 1024 bits. Para certificar dicho par de claves, se ha generado un certificado (firmado por la CA del sistema) con propiedades como las siguientes:

- **Algoritmo de firma**: sha1RSA
- **Emisor**: E = E = capfc@uoc.edu, CN = CA_PFC_Seguretat, OU = PFC_Seguretat, O = UOC, L = Barcelona, S = Barcelona, C = ES
- **Válido desde**: jueves, 11 de octubre de 2007 12:02:00
- **Válido hasta**: viernes, 10 de octubre de 2008 12:02:00

- **Titular**: E = mcurat@uoc.edu, CN = Marc Curat [DNI:38105232], OU = Metges, O = UOC, S = Barcelona, C = ES
- **Clave pública**: RSA de 1024 bits
- **Restricciones básicas**: Tipo de titular=Entidad final
- **Uso de clave**: Firma digital, No-repudio, Cifrado de clave

Cada par de claves se ha incluido además con su cadena de certificación en un contenedor PKCS #12 protegido mediante contraseña.

6.1.4. Claves y certificado del gestor

Para el actor con el rol de gestor del sistema se ha generado un par de claves RSA de 1024 bits. Para certificar dicho par de claves, se ha generado un certificado (firmado por la CA del sistema) con las siguientes propiedades:

- **Algoritmo de firma**: sha1RSA
- **Emisor**: E = capfc@uoc.edu, CN = CA_PFC_Seguretat, OU = PFC_Seguretat, O = UOC, L = Barcelona, S = Barcelona, C = ES
- **Válido desde**: jueves, 11 de octubre de 2007 12:18:38
- **Válido hasta**: viernes, 10 de octubre de 2008 12:18:38
- **Titular**: E = gestorhistorials@uoc.edu, CN = Gestor, OU = Gestors_Historials, O = UOC, S = Barcelona, C = ES
- **Clave pública**: RSA de 1024 bits
- **Restricciones básicas**: Tipo de titular=Entidad final
- **Uso de clave**: Firma digital, No-repudio, Cifrado de clave

La clave privada y el certificado se han incluido además en un contenedor PKCS #12 protegido mediante contraseña.

6.2. Notación

Esta sección describe la notación utilizada en los protocolos y procedimientos para referirse a los certificados, claves, firmas digitales, etc.

- $user\_id_U$: Identificador del usuario U.
- K: Clave de criptosistema simétrico.
- $E_K(M)$: Cifrado del mensaje M con la clave simétrica K.
- $D_K(M)$: Descifrado del mensaje M con la clave simétrica K.

- $(P_{Entitat}, S_{Entitat})$: Par de claves asimétricas propiedad de *Entidad*, donde $P_{Entitat}$ es la clave pública y $S_{Entitat}$ la privada.
- $S_{Entitat}[M]$: Firma del mensaje M con la clave $S_{Entitat}$.
- $P_{Entitat}[M]$: Cifrado del mensaje M con la clave $P_{Entitat}$.
- $H(M)$: Resultado de aplicar la función de resumen (*hash*) al mensaje M.

6.3. Procedimientos

Esta sección explica los procedimientos criptográficos aplicados en la ejecución de los protocolos. Vea en la sección 6.2. Notación una descripción de la notación utilizada.

6.3.1. Procedimiento 1

El procedimiento 1 contiene una parte de la autenticación del protocolo Needham-Schroeder y es utilizado por médicos y pacientes. Este procedimiento recibe como entrada el valor P_G y ejecuta los siguientes pasos:

1. Generar de forma aleatoria un valor N_i.
2. Cifrar N_i y $user\_id_U$ con la clave pública de G, $P_G[N_i, user\_id_U]$.
3. Enviar $P_G[N_i, user\_id_U]$ a G.

6.3.2. Procedimiento 2

El procedimiento 2 implementa otra parte de la autenticación del protocolo Needham-Schroeder y es ejecutado por el sistema gestor. Este procedimiento recibe como entrada el valor $P_G[N_i, user\_id_U]$ y ejecuta los siguientes pasos:

1. Descifrar $P_G[N_i, user\_id_U]$ con S_G para obtener N_i y $user\_id_U$.
2. A partir de $user\_id_U$ obtener de la base de datos el certificado de U (y su correspondiente clave privada P_U).
3. Generar de forma aleatoria un valor N_G.
4. Guardar en la base de datos los valores N_i y N_G asociados a U.
5. Cifrar los valores N_i, N_G y $user\_id_G$ con la clave P_U, $P_U[N_i, N_G, user\_id_G]$.
6. Devolver el valor $P_U[N_i, N_G, user\_id_G]$.

6.3.3. Procedimiento 3

El procedimiento 3 es utilizado por el sistema gestor para encontrar un historial y cifrarlo con la clave del usuario que desea consultarlo. Este procedimiento recibe como entrada los valores $user\_id_U$ y P_U y ejecuta los siguientes pasos:

1. Obtener el historial H del usuario $user\_id_U$.
2. Descifrar la parte de H que ha sido cifrada mediante S_G, la clave privada de G.
3. Cifrar H con la clave pública P_U para obtener $P_U[H]$.
4. Devolver el valor $P_U[H]$.

6.3.4. Procedimiento 4

El procedimiento 4 es utilizado por el usuario (médico o paciente) para descifrar un historial enviado por el sistema gestor y verificar su validez. Este procedimiento recibe como entrada $P_U[H]$ y ejecuta los siguientes pasos:

1. Descifrar $P_U[H]$ con la clave privada S_U para obtener $S_U[P_U[H]]$.
2. Para cada entrada firmada del historial H, realizar lo siguiente:
   a) Verificar la firma digital de U.
   b) Verificar la firma digital de G.
   c) Verificar la secuencia.
3. Retornar H.

6.3.5. Procedimiento 5

El procedimiento 5 es utilizado por el gestor G del sistema para obtener los historiales de los pacientes del médico $user\_id_U$. Este procedimiento recibe como entrada $user\_id_U$ y P_U y ejecuta los siguientes pasos:

1. Buscar en la base de datos los pacientes asignados al médico $user\_id_U$ para obtener $\{user\_id_1, ..., user\_id_n\}$.
2. Firmar $\{user\_id_1, ..., user\_id_n\}$ con la clave privada S_G de G, $S_G[\{user\_id_1, ..., user\_id_n\}]$.
3. Cifrar $\{user\_id_1, ..., user\_id_n\}$ y $S_G[\{user\_id_1, ..., user\_id_n\}]$ con la clave pública P_U de $user\_id_U$, $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$.
4. Devolver $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$.

6.3.6. Procedimiento 6

El procedimiento 6 es utilizado por el médico para obtener la lista de sus pacientes y verificar que ha sido generada por el gestor G. Este procedimiento recibe como entrada $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$ y ejecuta los siguientes pasos:

1. Descifrar $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$ con la clave privada S_U de U, $S_U[P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]]$ para obtener $S_G[\{user\_id_1, ..., user\_id_n\}]$.
2. Verificar la firma digital $S_G[\{user\_id_1, ..., user\_id_n\}]$ con la clave pública P_G de G.
3. Si la verificación anterior es correcta, devolver $\{user\_id_1, ..., user\_id_n\}$.

6.4. Protocolos

En esta sección se describen los protocolos criptográficos utilizados en cada caso de uso del sistema. Dichos protocolos siguen las propuestas del consultor del proyecto, apostando siempre por la solución más sencilla. Vea en la sección 6.2. Notación una descripción de la notación utilizada.

6.4.1. Protocolo de identificación

La identificación es el proceso que establece la identidad de un usuario en el sistema. En nuestro sistema de gestión de historiales, el usuario (médico o paciente) se identifica simplemente mediante su DNI.

El DNI se incluye además en el campo "Titular" del certificado digital de cada usuario y se utiliza como clave primaria en la tabla de certificados de la base de datos del gestor. De esta forma, a partir del DNI de un usuario, el sistema puede obtener el certificado correspondiente.

6.4.2. Protocolo de autenticación

La autenticación es el proceso que confirma que un usuario es quien dice ser. En nuestro sistema usuario P_i y gestor G se autentican bilateralmente mediante el protocolo de Needham-Schroeder.[1] Dicho protocolo se utilizará para implementar la autenticación en los protocolos de consumo de servicios del sistema:

- Protocolo de consulta de un historial descrito en la sección 6.4.3, pág. 41.
- Protocolo de consulta de los pacientes asignados a un médico descrito en la sección 6.4.4, pág. 42.

[1] Para simplificar la implementación, la autenticación se repetirá antes de atender cada petición. Aunque sea computacionalmente más eficiente, no se conservará la autenticación realizada en un contexto de sesión.

- Protocolo de inserción de datos en un historial descrito en la sección 6.4.5, 44.

El protocolo Needham-Schroeder consta de los siguientes pasos:

1. El usuario U realiza las siguientes operaciones:
   a) Obtener un valor aleatorio N_i.
   b) Cifrar N_i y $user\_id_U$ con la clave pública G, $E_G[N_i, user\_id_U]$.
   c) Enviar $E_G[N_i, user\_id_U]$ a G.
2. G realiza las siguientes operaciones:
   a) Descifrar $E_G[N_i, user\_id_U]$ con S_G para obtener N_i y $user\_id_U$.
   b) A partir de $user\_id_U$ obtener el certificado de U. A partir del certificado obtener la clave pública P_U de U.
   c) Obtener un valor aleatorio N_G.
   d) Cifrar N_i, N_G, $user\_idU$ con la clave pública P_U de U, $P_U[N_i, N_G, user\_id_U]$.
   e) Enviar $P_U[N_i, N_G, user\_id_U]$ a U.
3. El usuario U realiza los siguientes pasos:
   a) Descifrar $P_U[N_i, N_G, user\_id_U]$ con su clave privada S_U para obtener N_i ,N_G ,$user\_id_U$.
   b) Cifrar N_G con la clave pública P_G de G, $P_G[N_G]$.
   c) Enviar $P_G[N_G]$ a G.
4. G realiza las operaciones siguientes:
   a) Descifrar $P_G[N_G]$ con la clave privada S_G para obtener N'_G.
   b) Si $N'_G == N_G$, entonces G y U se han autenticado bilateralmente.

6.4.3. Protocolo de consulta de un historial

El protocolo de consulta de historiales garantiza la autenticación y la confidencialidad en los siguientes casos:

- Un usuario de tipo *paciente* consulta su historial.
- Un usuario de tipo *médico* consulta el historial de un usuario que es paciente suyo.

Dicho protocolo consta de los siguientes pasos:

1. El usuario U realiza las siguientes operaciones:
   a) Ejecutar el Procedimiento 1 (descrito en la sección 6.3.1, pág. 38) con la clave pública del gestor P_G para obtener $P_G[N_i, user\_id_U]$.

b) Enviar $P_G[N_i, user\_id_U]$ a G.

2. El gestor G realiza las siguientes operaciones:

   a) Ejecutar el Procedimiento 2 (descrito en la sección 6.3.2, pág. 38) con $P_G[N_i, user\_id_U]$ para obtener $P_U[N_i, N_G, user\_id_G]$

   b) Enviar $P_U[N_i, N_G, user\_id_G]$ a U.

3. U realiza las siguientes operaciones:

   a) Descifrar $P_U[N_i, N_G, user\_id_G]$ con su clave privada S_U para obtener N'_i, N_G y $user\_id_G$.

   b) Si $N'_i == N_i$ hacer:

      I. Cifrar N_G, *consulta* y *user_id* con la clave pública P_G de G, $P_G[N_G, Consulta, user\_id]$. El valor *consulta* indica que se desea consultar el historial del usuario *user_id*.

      II. Enviar $P_G[N_G, Consulta, user\_id]$ a G.

   c) Si no, retornar error.

4. G realiza las siguientes operaciones:

   a) Descifrar $P_G[N_G, consulta, user\_id]$ con la clave privada S_G para obtener N'_G, $Consulta$ y *user_id*.

   b) Recuperar N_G de la base de datos (donde fue guardado junto con N_i en el paso 4 del Procedimiento 4 descrito en la sección 6.3.4).

   c) Si $N'_G == N_G$ hacer:

      I. Si ($user\_id_U ==$ *user_id*) o ($user\_id_U$ médico y *user_id* paciente suyo), ejecutar el Procedimiento 3 (descrito en la sección 6.3.3, 39) con *user_id* y P_U para obtener $P_U[H]$; enviar $P_U[H]$ a U.

      II. Si no, enviar error.

   d) Si no, enviar error.

   e) Borrar N_i y N_G de la base de datos.

5. U realiza las siguientes operaciones:

   a) Ejecutar el Procedimiento 4 (descrito en la sección 6.3.4, pág. 39) con $P_U[H]$ para obtener H.

   b) Mostrar H.

6.4.4. Protocolo de consulta de los pacientes asignados a un médico

Para consultar de forma segura los pacientes asignados a un médico, utilizaremos un protocolo con los siguientes pasos:

1. El médico U realiza las siguientes operaciones:

   a) Ejecutar el Procedimiento 1 (descrito en la sección 6.3.1, pág. 38) con la clave pública del gestor P_G para obtener $P_G[N_i, user\_id_U]$.
   b) Enviar $P_G[N_i, user\_id_U]$ a G.

2. El gestor G realiza las siguientes operaciones:
   a) Ejecutar el Procedimiento 2 (descrito en la sección 6.3.2, pág. 38) con $P_G[N_i, user\_id_U]$ para obtener $P_U[N_i, N_G, user\_id_G]$
   b) Enviar $P_U[N_i, N_G, user\_id_G]$ a U.

3. U realiza las siguientes operaciones:
   a) Descifrar $P_U[N_i, N_G, user\_id_G]$ con la clave privada S_U para obtener N_i,N'_G, $user\_id_G$.
   b) Si $N'_i == N_i$ entonces cifrar N_G y $lista\_pacientes$[2] con la clave pública P_G, $P_G[N_G, lista\_pacientes]$; enviar $P_G[N_G, lista\_pacientes]$ a G.
   c) Si no, devolver error.

4. G realiza las siguientes operaciones:
   a) Descifrar $P_G[N_G, lista\_pacientes]$ con la clave privada S_G para obtener N_G y $lista\_pacientes$].
   b) Recuperar N_G de la base de datos (donde fue guardado junto con N_i en el paso 4 del Procedimiento 4 descrito en la sección 6.3.4).
   c) Si $N'_G == N_G$ y el usuario $user\_id_U$ es médico[3] hacer:
      i. Ejecutar el Procedimiento 5 (descrito en la sección 6.3.5, pág. 39) con $user\_id_U$ y P_U para obtener $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$.
      ii. Enviar $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$ a U.
   d) Si no, devolver error.
   e) Borrar N_G y N_i de la base de datos.

5. U realiza las siguientes operaciones:
   a) Ejecutar el Procedimiento 6 (descrito en la sección 6.3.6. pág. 40) sobre $P_U[\{user\_id_1, ..., user\_id_n\}, S_G[\{user\_id_1, ..., user\_id_n\}]]$ para obtener $\{user\_id_1, ..., user\_id_n\}$.
   b) Mostrar $\{user\_id_1, ..., user\_id_n\}$.

[2] El valor *lista_pacientes* sirve para solicitar una lista de los pacientes del médico $user\_id_U$.
[3] El campo *titular* del certificado de usuario indica si se trata de un médico o un paciente

6.4.5. Protocolo de inserción de datos en un historial

El protocolo de inserción de datos en un historial médico sirve para que un usuario M de tipo *médico* añada una visita V al historial de un usuario P de tipo *paciente*. Para proteger los historiales de ataques *man-in-the-middle*,[4] cuando el gestor recibe una visita V adopta las siguientes medidas:

1. Verifica que V realmente ha sido firmada por el médico M asignado al paciente P.
2. Añade V una marca temporal T y un número de serie X para dejar constancia del momento de la visita y su posición en el historial.
3. Firma la visita V, el tiempo T y el número de serie X. De esta manera, un atacante que elimine una visita no podrá generar nuevos números de serie (porque no dispondrá de la clave de firma del gestor S_G).
4. Cifra la visita V con su clave pública (para preservar su confidencialidad) y la añade a la base de datos.
5. Añade al historial una entrada firmada con el número de serie X de la última visita V. Si un atacante borra la visita V de la base de datos se detectará un salto entre la última visita disponible y el último número de serie X.

En concreto, el protocolo de inserción de datos en un historial consta de los siguientes pasos:

1. El médico M realiza las siguientes operaciones:
   a) Ejecutar el Procedimiento 1 (descrito en la sección 6.3.1. pág. 38) con su clave pública P_M para obtener $P_G[N_i, user\_id_M]$.
   b) Enviar $P_G[N_i, user\_id_M]$ a G.
2. El gestor G realiza las siguientes operaciones:
   a) Ejecutar el Procedimiento 2 (descrito en la sección 6.3.2. pág. 38) con $P_G[N_i, user\_id_M]$ para obtener $P_U[N_i, N_G, user\_id_G]$.
   b) Enviar $P_M[N_i, N_G, user\_id_G]$ a M.
3. El médico M realiza las siguientes operaciones:
   a) Descifrar $P_M[N_i, N_G, user\_id_G]$ con la clave privada S_M para obtener N_G, N'_i y $user\_id_G$.
   b) Si $N'_i == N_i$ hacer:

[4] "En criptografía, un ataque man-in-the-middle (MitM o intermediario, en castellano) es un ataque en el que el enemigo adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido violado. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas." (http://es.wikipedia.org/wiki/Ataque_Man-in-the-middle)

i. Obtener los datos de la visita V. La visita debería incluir como mínimo $user\_id_P$.
ii. Firmar V con la clave privada S_M de M, $S_M[V]$.
iii. Cifrar $N_G, V, S_M[V]$ con la clave pública P_G de G, $P_G[N_G, insertar\_visita, V, S_M[V]]$. El valor $insertar\_visita$ indica que se desea insertar V al historial del paciente P.
iv. Enviar $P_G[N_G, insertar\_visita, V, S_M[V]]$ a G.

c) Si no, devolver error.

4. G realiza las siguientes operaciones:

a) Descifrar $P_G[N_G, insertar\_visita, V, S_M[V]]$ con la clave privada S_G para obtener N'_G, $insertar\_visita$, V y $S_M[V]$.

b) Recuperar N_G de la base de datos (donde fue guardado junto con N_i en el paso 4 del Procedimiento 4 descrito en la sección 6.3.4).

c) Si $N'_G == N_G$ hacer:

i. Obtener $user\_id_P$ a partir de V.
ii. Verificar que $user\_id_M$ es médico.[5]
iii. Verificar en la base de datos que $user\_id_P$ es un paciente asignado a $user\_id_M$.
iv. Si las verificaciones anteriores son correctas:
A. Verificar la firma digital $S_M[V]$ con la clave pública P_M.
B. Obtener el instante de tiempo actual T.
C. Obtener el número de serie X de la última visita del historial H del paciente $user\_id_P$.
D. Incrementar en una unidad X, $X + 1$.
E. Firmar V, $S_M[V]$, T y $X + 1$ con la clave privada S_G de G, $S_G[V, S_M[V], T, X + 1]$.
F. Cifrar V y $S_M[V]$ con la clave pública S_G de G, $P_G[V, S_M[V]]$.
G. Firmar $user\_id$ y $X+1$ con la clave privada S_G de G, $S_G[X+1, user\_id_P]$.
H. Guardar en la base de datos $P_G[V, S_M[V]]$, $X+1$, T, $S_G[V, S_M[V], T, X + 1]$ y $S_G[X + 1, user\_id_P]$.
v. Si no, devolver error.

d) Si no, devolver error.

[5] El campo **Organizational Unit Name** (e.g. section) del nombre distintivo del certificado de usuario indica si el usuario es médico o paciente

Capítulo 7

Esquemas XML

Contenido

Los mensajes de petición y respuesta de nuestro sistema de gestión de historiales se intercambian en formato XML. Este capítulo describe el contenido de dichos mensajes mediante el uso de esquemas XML. En concreto, para cada mensaje incluimos la siguiente información:

- Representación gráfica del esquema XML, generada mediante el editor Altova XMLSpy.
- Referencia de elementos.

La codificación XML completa de los esquemas se incluye en el apéndice A. Código XML de los esquemas, pág. 71.

7.1. Mensajes del protocolo Needham-Schroeder

Esta sección describe la estructura XML de los mensajes de petición y respuesta intercambiados por el usuario y el gestor para autenticarse mediante el Protocolo de autenticación Needham-Schroeder (descrito en la sección 6.4.2, 40). Dichos mensajes de cifran con la clave pública del usuario (pasos 1 y 3) del gestor (pasos 2 y 4).

7.1.1. Esquema de los mensajes Needham-Schroeder

La siguiente figura 7.1 describe el esquema XML de los mensajes de autenticación del protocolo Needham-Schroeder.

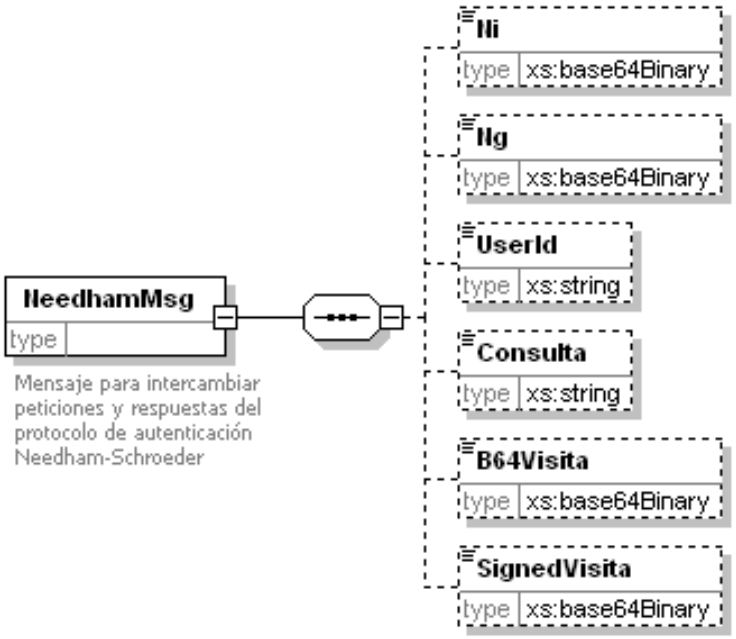

Figura 7.1: Esquema de los mensajes Needham-Schroeder

7.1.2. Elementos XML de los mensajes Needham-Schroeder

Esta sección describe los elementos XML de los mensajes del protocolo Needham-Schroeder.

- **Ni**. (Opcional) Valor aleatorio generado por el usuario durante el protocolo de autenticación, codificado en base 64.
- **Ng**. (Opcional) Valor aleatorio generado por el gestor durante el protocolo de autenticación, codificado en base 64.
- **Consulta**. (Opcional) Identificador del servicio solicitado, cifrado con la clave pública del usuario.
- **B64Visita**. (Opcional) Nueva visita que se desea insertar en un historial, codificada en base 64.
- **SignedVisita**. (Opcional) Firma de la nueva visita, codificada en base 64.

7.2. Historiales médicos

Esta sección describe la estructura XML de los historiales, tal como se guardan en la base de datos (cifrados con la clave pública del gestor) o se envían

(firmados con la clave pública del receptor) cuando se solicitan mediante el Protocolo de consulta de un historial (descrito en la sección 6.4.3, pág. 41).

7.2.1. Esquemas de los historiales

La siguiente figura 7.2 describe el esquema XML de los historiales.

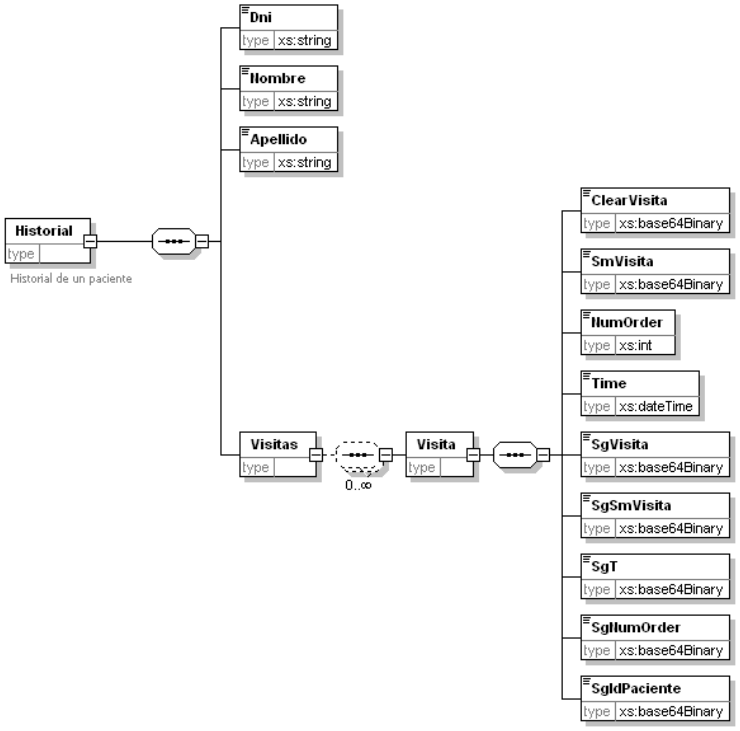

Figura 7.2: Esquema de los historiales

7.2.2. Elementos XML de los historiales

Esta sección describe los elementos XML de los historiales.

- **Dni**. (Obligatorio) DNI del paciente.
- **Nombre**. (Obligatorio) Nombre del paciente.
- **Apellido**. (Obligatorio) Apellido del paciente.
- **Visitas**. (Obligatorio) Lista de elementos <Visita> con las visitas del paciente.

- **Visita** (Opcional) Visita realizada a un paciente. Este elemento contiene los siguientes subelementos:
  - **ClearVisita**. (Obligatorio) Texto de la visita, codificado en base 64.
  - **SmVisita**. (Obligatorio) Texto de la visita, firmado con la clave privada del médico y codificado en base 64.
  - **NumOrder**. (Obligatorio) Número de orden de la visita en el historial.
  - **Time** (Obligatorio) Fecha de inserción de la visita en el historial.
  - **SgVisita**. (Obligatorio) Texto de la visita, firmado con la clave privada del gestor y codificado en base 64.
  - **SgSmVisita**. (Obligatorio) Valor del elemento <SmVisita>, firmado con la clave privada del gestor y codificado en base 64.
  - **SgT**. (Obligatorio) Valor del elemento <Time>, firmado con la clave privada del gestor y codificado en base 64.
  - **SgNumOrder**. (Obligatorio) Valor del elemento <NumOrder>, firmado con la clave privada del gestor y codificado en base 64.
  - **SgDniPaciente**. (Obligatorio) DNI del paciente, firmado con la clave privada del gestor y codificado en base 64.

7.3. Lista de pacientes de un médico

Esta sección describe la estructura XML de las listas de pacientes asignados a un médico tal como se envían (firmadas con la clave pública del receptor) cuando se solicitan mediante el Protocolo de consulta de los pacientes asignados a un médico (descrito en la sección 6.4.4, pág. 42).

7.3.1. Esquema de las listas de pacientes de un médico

La siguiente figura 7.3 describe el esquema XML de las listas de pacientes asignados a un médico.

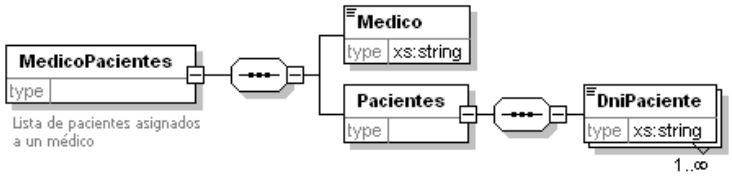

Figura 7.3: Esquema de las listas de pacientes asignados a un médico

7.3.2. Elementos XML de las listas de pacientes de un médico

Esta sección describe los elementos XML de las listas de pacientes asignados a un médico.

- **Médico**. (Obligatorio) DNI del médico.
- **Pacientes**. (Obligatorio) Lista de al menos un elemento <Paciente>.
- **Paciente**. DNI de un paciente del médico.

7.4. Mensaje de respuesta del servidor

Esta sección describe la estructura XML de los mensajes de respuesta con los que el servidor indica un éxito o fracaso al procesar una petición (e.g. insertar una visita). Dichos mensajes se envían cifrados con la clave pública del receptor.

7.4.1. Esquema de los mensajes de respuesta del servidor

La siguiente figura 7.4 describe el esquema XML de los mensajes de error devueltos por el servidor.

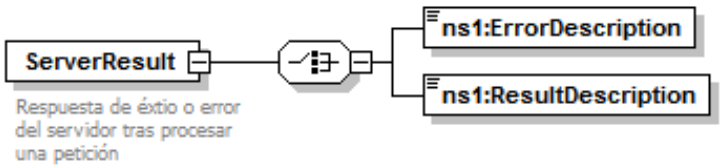

Figura 7.4: Esquema de los mensajes de respuesta del servidor

7.4.2. Elementos XML de los mensajes de respuesta del servidor

Esta sección describe los elementos XML de los mensajes de respuesta del servidor. En concreto, dichos mensajes contienen un *choice* entre los dos elementos siguientes:

- **ErrorDescription**. Si la petición no se ha podido procesar, este elemento describe el error surgido.
- **ResultDescription**. Si la petición de ha podido procesar con éxito, este elemento incluye un confirmación.

Capítulo 8

Manual de uso

Contenido

Este capítulo explica cómo ejecutar las operaciones soportadas por el sistema. Vea en el capítulo 9. Juego de pruebas, pág. 58, un ejemplo completo de ejecución.

8.1. Puesta en marcha del servidor

Esta sección explica cómo poner en marcha el servidor del sistema gestor de historiales.

Para poner en marcha el servidor:

1. Genere la clase *stub* mediante la siguiente línea de comandos:

   rmic uoc.historials.HistGestor

2. Inicie el servidor RMI mediante la siguiente línea de comandos:

   start rmiregistry

3. Ejecute la aplicación gestora mediante la siguiente línea de comandos:

   start java uoc.historials.HistGestorApp

4. Rellene los campos de la GUI (parámetros de conexión a la base de datos, inicio del servidor RMI y acceso al PKCS #12). Si aún no ha creado las tablas en la base de datos, active la casilla **Crear las tablas** (figura 8.1).

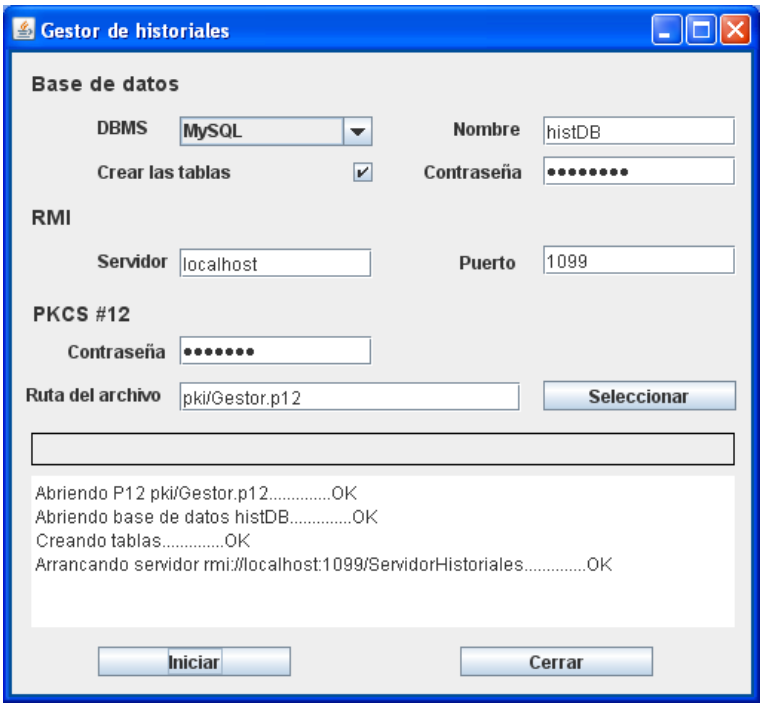

Figura 8.1: Interfaz gráfica del gestor del sistema

5. Pulse el botón **Iniciar** (figura 8.1). Si los parámetros son correctos, el sistema creará las tablas en la base de datos y pondrá en marcha el servidor RMI.

8.2. Inicialización de la base de datos con HistDbManager

Para cargar en la base de datos del sistema un juego predefinido de médicos y pacientes puede utilizarse la utilidad *HistDbManager*. En concreto, esta herramienta debe invocarse mediante la siguiente línea de comandos:

java uoc.historials.HistTestDB -dbms <mySQL|derby> -dbname <dbName> [-dbpwd <dbPassword>] [-createTables]

Donde:

- <dbms>: Identificador del sistema gestor de base de datos. Los valores soportados por este parámetro son *Derby* y *MySQL*. Si trabaja con el DBMS de prueba Apache Derby, cierre antes cualquier aplicación que acceda a la base de datos (porque no soporta accesos concurrentes).
- <dbName>: Nombre de la base de datos.

- <dbPassword>: Contraseña para acceder a la base de datos (si el sistema gestor escogido es MySQL).
- -createTables: Añada este comando si aún no ha creado las tablas en la base de datos.

Ejemplo:
Cuando se ejecuta mediante la siguiente línea de comandos, la utilidad *HistDbManager* accede a la base de datos *histDB* sobre sistema gestor MySQL (protegido con la contraseña *demodemo*), crea las tablas y carga los valores por defecto.

java uoc.historials.HistDbManager -dbms mysql -dbname histDB -dbpwd demodemo -createtables

8.3. Gestión de la base de datos en modo interactivo

El contenido de la base de datos del sistema se puede administrar en modo interactivo desde la consola de gestión de usuarios. Mediante dicha consola no pueden crearse las tablas del sistema, que deben ser generadas previamente de una de las siguientes maneras:

- Desde la GUI de gestión del sistema (figura 8.1) tal como se explica en la sección 8.1. Puesta en marcha del servidor, pág. 51.
- Desde la utilidad *HistDbManager* para inicializar la base de datos, tal como se explica en la sección 8.2. Inicialización de la base de datos con HistDbManager, pág. 52.

Los siguientes pasos explican cómo utilizar la consola de gestión de usuarios.
Para administrar la base de datos en modo interactivo:

1. Ejecute la siguiente línea de comandos para poner en marcha la consola de gestión de usuarios (figura 8.2):

   java uoc.historials.HistUserManager

2. Indique en la consola (figura 8.2) los parámetros de conexión a la base de datos: nombre del DBMS, nombre de la base de datos, contraseña de la base de datos.

3. Ejecute los comandos de la consola para registrar los médicos y pacientes. En concreto, los comandos de gestión de usuarios son los siguientes:

   - **register**. Registra a un usuario (médico o paciente).
   - **assig**. Asigna un paciente a un médico.
   - **unassig**. Elimina una asignación de paciente a médico.

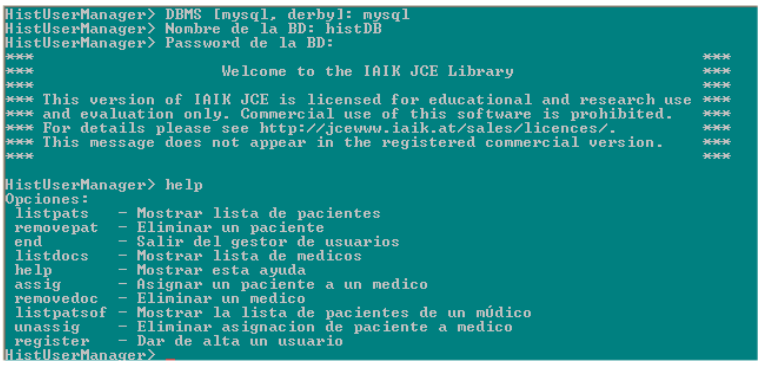

Figura 8.2: Consola de gestión de usuarios

- **listpats**. Muestra la lista de pacientes registrados.
- **removepat**. Elimina un paciente del sistema.
- **listpatsof**. Muestra la lista de pacientes asignados a un médico.
- **listdocs**. Muestra la lista de médicos registrados.
- **removedoc**. Elimina un medico del sistema.
- **help**. Muestra la lista de comandos.
- **end**. Finaliza la aplicación de gestión de usuarios.

8.4. Consulta de pacientes

Esta sección explica cómo consultar la lista de pacientes de un médico.
Para consultar la lista de pacientes de un médico:

1. Ejecute la aplicación médico (figura 8.3) mediante la siguiente línea de comandos:

   java uoc.historials.HistMedicoApp

2. Seleccione la opción *Consultar pacientes* en la lista desplegable **Operación** (figura 8.3).

3. Rellene los campos **Contraseña del PKCS #12 del médico** y **Ruta del PKCS #12 del médico** (figura 8.3).

4. Pulse el botón **Ejecutar**. Si el sistema valida satisfactoriamente las credenciales, mostrará la lista de pacientes en el panel de contenido (figura 8.3). Pulse el botón **Exportar** para guardar la lista en archivo.

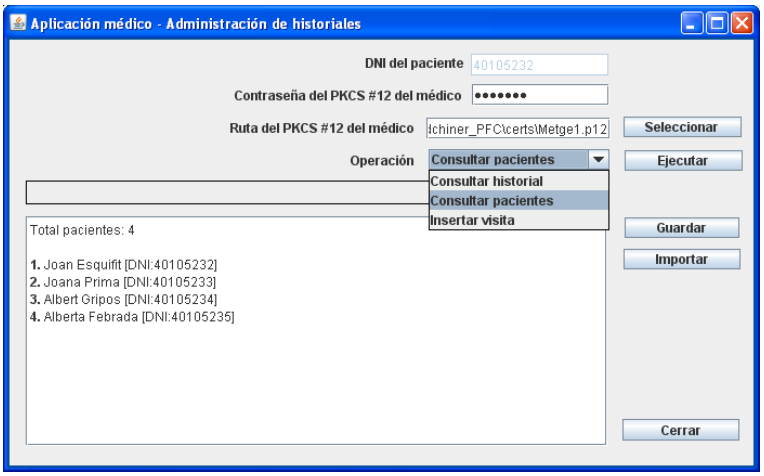

Figura 8.3: Consulta de pacientes de un médico

8.5. Inserción de visitas

Esta sección explica cómo insertar una visita en el historial de un paciente.
Para insertar una visita en un historial:

1. Ejecute la aplicación médico (figura 8.4) mediante la siguiente línea de comandos:

   java uoc.historials.HistMedicoApp

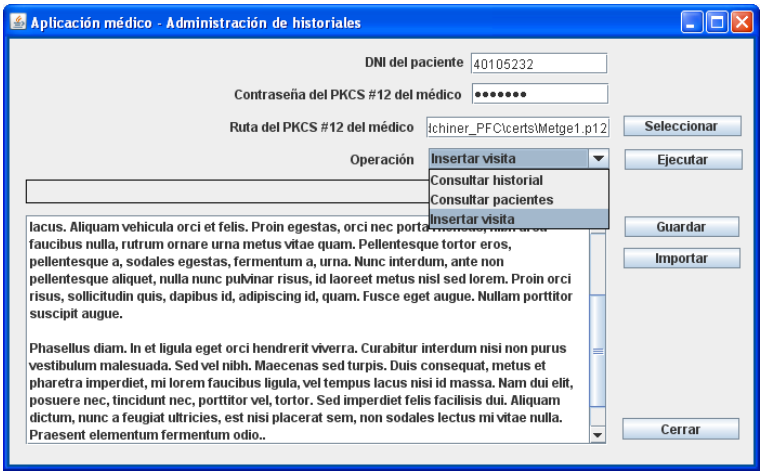

Figura 8.4: Inserción de una visita

2. Seleccione la opción *Insertar visita* de la lista desplegable **Operación**

(figura 8.4).

3. Rellene los campos **DNI del paciente**, **Contraseña del PKCS #12 del médico** y **Ruta del PKCS #12 del médico** (figura 8.4).

4. Escriba el texto de la visita en el panel de contenido, o pulse el botón **Importar** (figura 8.4) para cargar el texto desde archivo.

5. Pulse el botón **Ejecutar**. Si el sistema valida satisfactoriamente las credenciales, mostrará un mensaje de confirmación en la GUI de aplicación (figura 8.4).

8.6. Consulta de historiales

Esta sección explica cómo consultar el historial de un paciente desde las aplicaciones cliente del sistema.

Para consultar un historial:

1. Ejecute la aplicación cliente para pacientes (*uoc.historials.HistPacienteApp*) o médicos (*uoc.historials.HistPacienteApp*).

2. Indique los siguientes datos en la interfaz de usuario (figura 8.5): DNI del paciente cuyo historial desee consultar; archivo que contiene el PKCS #12 del solicitante del historial; contraseña del PKCS #12.

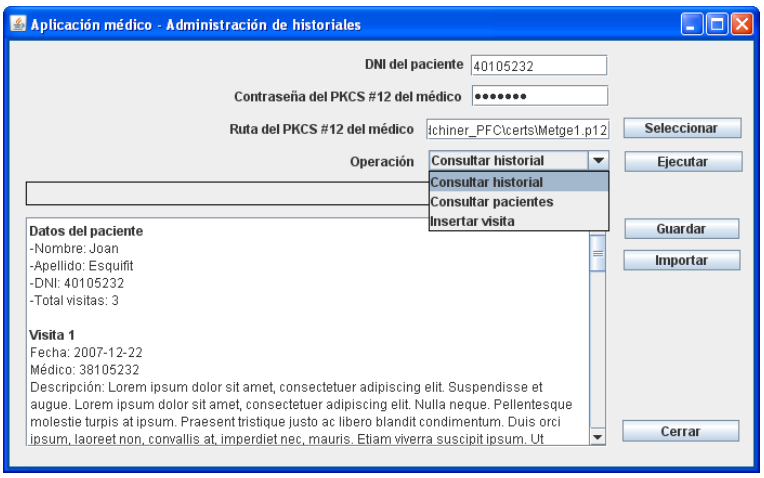

Figura 8.5: Consulta de historial

3. Seleccione la opción *Consultar historial* en la lista desplegable **Operación**.

4. Pulse el botón **Ejecutar**. Si el paciente existe en la base de datos y el proceso de autenticación es correcto, el sistema gestor mostrará los datos

del historial en el panel de contenido (figura 8.5). Pulse el botón **Exportar** para guardar el historial en archivo.

Capítulo 9

Juego de pruebas

Este capítulo describe un ejemplo de ejecución del sistema gestor de historiales. En concreto, los siguientes pasos permiten evaluar las principales operaciones soportadas por el sistema.

1. Creamos en el servidor MySQL una base de datos de nombre *histDB*.
2. Establecemos las variables de entorno y compilamos el proyecto.
3. Levantamos el servidor RMI y ejecutamos el gestor del sistema (figura 8.1) mediante las siguientes líneas de comandos:

   rmic uoc.historials.HistGestor

   start rmiregistry

   start java uoc.historials.HistGestorApp
4. Rellenamos los campos de la GUI del gestor (figura 8.1), activamos la casilla **Crear las tablas** y pulsamos el botón **Iniciar**.
5. Ejecutamos la consola de gestión de usuarios:

   java uoc.historials.HistUserManager
6. Damos de alta tres pacientes a partir de los certificados guardados en el subdirectorio *pki*.

```
HistUserManager> DBMS [mysql, derby]: mysql
HistUserManager> Nombre de la BD: histDB
HistUserManager> Password de la BD:
HistUserManager> register
HistUserManager> Tipo de usuario [medico, paciente]: paciente
HistUserManager> Nombre del archivo con el certificado: pki/Pacient1.crt
HistUserManager> Nombre del paciente: Joan
HistUserManager> Apellido del paciente: Esquifit
HistUserManager> Registrar paciente Joan Esquifit [DNI:40105232]? [si, no]: si
```

```
HistUserManager> Usuario Joan Esquifit [DNI:40105232] registrado como paciente
HistUserManager> register
HistUserManager> Tipo de usuario [medico, paciente]: paciente
HistUserManager> Nombre del archivo con el certificado: pki/Pacient2.crt
HistUserManager> Nombre del paciente: Joana
HistUserManager> Apellido del paciente: Prima
HistUserManager> Registrar paciente Joana Prima [DNI:40105233]? [si, no]: si
HistUserManager> Usuario Joana Prima [DNI:40105233] registrado como paciente
HistUserManager> register
HistUserManager> Tipo de usuario [medico, paciente]: paciente
HistUserManager> Nombre del archivo con el certificado: pki/Pacient3.crt
HistUserManager> Nombre del paciente: Albert
HistUserManager> Apellido del paciente: Gripos
HistUserManager> Registrar paciente Albert Gripos [DNI:40105234]? [si, no]: si
HistUserManager> Usuario Albert Gripos [DNI:40105234] registrado como paciente
```

7. Damos de alta dos médicos, también a partir de los certificados guardados en el subdirectorio *pki*.

```
HistUserManager> register
HistUserManager> Tipo de usuario [medico, paciente]: medico
HistUserManager> Nombre del archivo con el certificado: pki/Metge1.crt
HistUserManager> Registrar medico Marc Curat [DNI:38105232]? [si, no]: si
HistUserManager> Usuario Marc Curat [DNI:38105232] registrado como medico
HistUserManager> register
HistUserManager> Tipo de usuario [medico, paciente]: medico
HistUserManager> Nombre del archivo con el certificado: pki/Metge2.crt
HistUserManager> Registrar medico Lluis Moltbo [DNI:38105344]? [si, no]: si
HistUserManager> Usuario Lluis Moltbo [DNI:38105344] registrado como medico
```

8. Asignamos los tres pacientes al médico Marc Curat.

```
HistUserManager> assig
HistUserManager> DNI del medico: 38105232
HistUserManager> DNI del paciente: 40105232
HistUserManager> Asignar el paciente 40105232 al medico 38105232? [si, no]: si
HistUserManager> Paciente 40105232 asignado al medico 38105232
HistUserManager> assig
HistUserManager> DNI del medico: 38105232
HistUserManager> DNI del paciente: 40105233
HistUserManager> Asignar el paciente 40105233 al medico 38105232? [si, no]: si
HistUserManager> Paciente 40105233 asignado al medico 38105232
HistUserManager> assig
HistUserManager> DNI del medico: 38105232
HistUserManager> DNI del paciente: 40105234
HistUserManager> Asignar el paciente 40105234 al medico 38105232? [si, no]: si
HistUserManager> Paciente 40105234 asignado al medico 38105232
```

9. Asignamos los dos primeros pacientes al médico Lluis Moltbo.

```
HistUserManager> assig
HistUserManager> DNI del medico: 38105344
HistUserManager> DNI del paciente: 40105232
HistUserManager> Asignar el paciente 40105232 al medico 38105344? [si, no]: si
HistUserManager> Paciente 40105232 asignado al medico 38105344
HistUserManager> assig
HistUserManager> DNI del medico: 38105344
HistUserManager> DNI del paciente: 40105233
HistUserManager> Asignar el paciente 40105233 al medico 38105344? [si, no]: si
HistUserManager> Paciente 40105233 asignado al medico 38105344
```

10. Ejecutamos la aplicación médico:

    java uoc.historials.HistMedicoApp

11. Consultamos la lista de pacientes del médico Marc Curat (figura 8.3).

    - Contraseña del PKCS #12 del médico: uoc0708
    - Ruta del PKCS #12 del médico: pki/Metge1.p12
    - Operación: Consultar pacientes

12. Añadimos una visita al historial del paciente Albert Gripos (figura 8.4).

    - DNI del paciente: 40105234
    - Contraseña del PKCS #12 del médico: uoc0708
    - Ruta del PKCS #12 del médico: pki/Metge1.p12
    - Operación: Insertar visita

13. Repetimos la inserción varias veces para generar un historial con varias entradas.

14. Intentamos consultar el historial de Albert Gripos como médico Lluis Moltbo.

    - DNI del paciente: 40105234
    - Contraseña del PKCS #12 del médico: uoc0708
    - Ruta del PKCS #12 del médico: pki/Metge2.p12
    - Operación: Consultar historial

15. Obtenemos un mensaje de error (figura 9.1).

16. Desde la consola de gestión de usuarios, asignamos el paciente Albert Gripos al médico Lluis Moltbo.

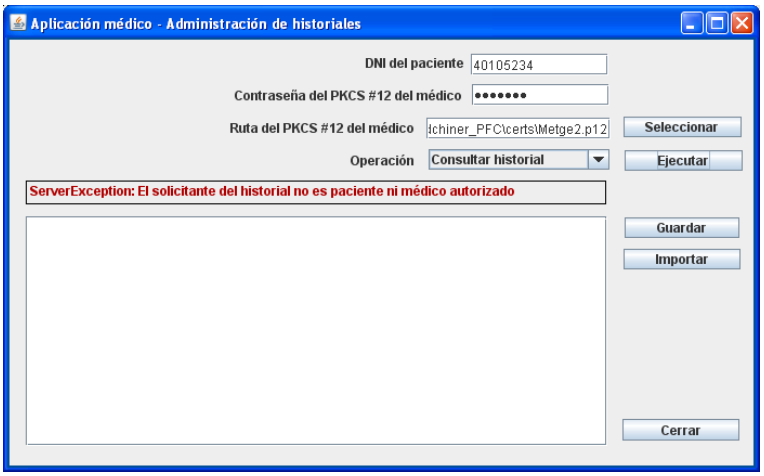

Figura 9.1: Mensaje de error al acceder a un historial

```
HistUserManager> assig
HistUserManager> DNI del medico: 38105344
HistUserManager> DNI del paciente: 40105234
HistUserManager> Asignar el paciente 40105234 al medico 38105344? [si, no]: si
HistUserManager> Paciente 40105234 asignado al medico 38105344
```

17. Repetimos la consulta del historial de Albert Gripos como médico Lluis Moltbo y obtenemos el historial (figura 8.5).

18. Desde la consola de gestión de usuarios, eliminamos a Albert Gripos como paciente de Marc Curat.

```
HistUserManager> unassig
HistUserManager> DNI del medico: 38105232
HistUserManager> DNI del paciente: 40105234
HistUserManager> Eliminar asignación del paciente 40105234 al medico 38105232? [
si, no]: si
HistUserManager> asignación del paciente 40105234 con el medico 38105232 elimina
da
```

19. Intentamos añadir una visita al historial de Albert Gripos como médico Marc Curat.

    - DNI del paciente: 40105234
    - Contraseña del PKCS #12 del médico: uoc0708
    - Ruta del PKCS #12 del médico: pki/Metge1.p12
    - Operación: Insertar visita

20. Obtenemos un mensaje de error (figura 9.1).

Capítulo 10

Gestión de errores

Contenido

Este capítulo describe los mensajes de error que pueden visualizarse en las GUI de las aplicaciones de nuestro sistema gestor de historiales.

Este capítulo no describe las excepciones internas por mal funcionamiento de la aplicación. En concreto, la operación permite dos modos de funcionamiento según el valor asignado a la constante *FAULT_TOLERANT* de la clase *HistConsts*:

- **false**. Las excepciones interrumpen la ejecución de la aplicación.
- **true**. La aplicación puede seguir funcionando tras una excepción.

En ambos casos, los detalles de la excepción se guardan en un archivo de *log*.

10.1. GUI de todas las aplicaciones

Esta sección describe los mensajes de error que pueden visualizarse desde la interfaz del médico (figura 8.5), el paciente o el gestor de historiales (figura 8.1).

- **Debe rellenar todos los campos**. Cuando se han dejado campos vacíos en algún formulario.
- **Contraseña de PKCS #12 incorrecta**. Cuando la contraseña introducida para abrir un PKCS #12 de cliente o gestor es incorrecta.
- **Ruta de archivo incorrecta**. Cuando la ruta indicada no corresponde a un archivo (de certificado o PKCS #12).

10.2. GUI del gestor de historiales

Esta sección describe los mensajes de error que sólo pueden visualizarse desde la interfaz del gestor de historiales (figura 8.1).

- **Error abriendo la base de datos**. Cuando el gestor no consigue acceder a la base de datos con los parámetros indicados.
- **Problema en rebind de HistServidor**. Cuando el gestor no consigue levantar el servidor RMI con los parámetros indicados.

10.3. GUI de las aplicaciones médico y paciente

Esta sección describe los mensajes de error que sólo pueden visualizarse desde la interfaz del médico (figura 8.5) o el paciente.

- **El DNI debe ser una cadena de 8 dígitos**. Cuando el valor introducido en el campo **DNI del paciente** no tiene formato de DNI.
- **El titular del certificado seleccionado no es médico**. Cuando el usuario intenta identificarse como médico con un certificado inválido.
- **El solicitante del historial no es paciente ni médico autorizado**. Cuando el usuario no está autorizado a realizar una operación de consulta o modificación como paciente ni como médico.
- **No existe historial del paciente indicado**. Cuando se intenta consultar el historial de un paciente que no existe.

10.4. Consola de administración de usuarios

Esta sección describe los mensajes de error que sólo pueden visualizarse desde la consola de administración de usuarios (figura 8.2).

- **Nombre de archivo incorrecto**. Cuando la ruta indicada no corresponde a un archivo.
- **El usuario no está registrado como médico**. Cuando el usuario seleccionado (para asignarle un paciente, dar de baja, etc.) no es médico.
- **El usuario no está registrado como paciente**. Cuando el usuario indicado (para asignar a un médico, dar de baja, etc.) no es paciente.
- **El certificado indicado no es de médico**. Cuando el certificado seleccionado no corresponde a un médico.
- **El usuario ya está registrado como médico**. Cuando el usuario que se pretende registrar ya está registrado como médico.

- **El certificado indicado no es de paciente**. Cuando el certificado seleccionado no corresponde a un paciente.
- **El usuario ya está registrado como paciente**. Cuando el usuario que se pretende registrar ya corresponde a un paciente.
- **El paciente <dni_paciente> ya estaba asignado al médico <dni_medico>**. Cuando se intenta asignar a un médico un usuario que ya es paciente suyo.
- **El paciente <dni_paciente> no está asignado al médico <dni_medico>**. Cuando se intenta dar de baja como paciente de un médico a un usuario que no era paciente de dicho médico.
- **El DNI debe ser una cadena de 8 dígitos**. Cuando el valor indicado no tiene formato de DNI.

Capítulo 11

Conclusiones

Contenido

Este capítulo enumera los principales objetivos cumplidos y plantea tareas pendientes para el futuro.

11.1. Objetivos cumplidos

Nuestro proyecto ha cumplido con creces el propósito de todo PFC: sintetizar y consolidar todo lo aprendido durante la carrera. En nuestro segundo ciclo de ingeniería informática en la UOC hemos seguido los itinerarios formativos *Disseny, gestió i seguretat de xarxes* y *Construcció d'aplicacions i sistemes distribuïts*. El proyecto ha supuesto por tanto una oportunidad de aplicar los conceptos aprendidos sobre criptografía y redes.

Además, la propuesta de proyecto, la gestión de historiales, ha sido una motivación idónea para trabajar todos los niveles de aplicación, desde la amigabilidad de la interfaz de usuario hasta la inserción de información cifrada en una base de datos remota.

En definitiva, se ha cumplido el objetivo inicial de desarrollar aplicaciones cliente y servidor para la gestión segura de historiales a través de la red. A este sistema se le ha añadido además una aplicación para administrar los usuarios del sistema en modo interactivo.

11.2. Tareas pendientes

A partir del prototipo desarrollado para nuestro PFC, quedan muchos temas pendientes que podrían desarrollarse en un proyecto de mayor envergadura:

- Mejorar el formato de la firma y cifrado de elementos de un documento XML. En concreto, el cifrado de información debería limitarse a la información sensible, para facilitar por ejemplo la manipulación de los documentos por parte de personal administrativo.
- Mejorar las GUI de aplicación para soportar la complejidad de un verdadero historial médico.
- Profundizar en el carácter distribuido del sistema, mediante tecnologías como Web Services.
- Controlar la concurrencia en los accesos de escritura.

Todas estas mejoras corresponderían por ejemplo a una aplicación real de la aplicación, que puliese y extendiese nuestro prototipo.

Glosario

Autenticación Proceso que confirma que un usuario es quien dice ser, 40

Autenticidad Garantía de que el autor del documento es realmente quien dice ser, 9

Certificado Documento electrónico que incorpora una firma digital para vincular una clave pública y una identidad, 35

Confidencialidad Protección de datos frente a accesos no autorizados, 9

Historial médico Documento con información sobre enfermedades personales y familiares, 8

IAIK API criptográfica del Institute for Applied Information Processing and Communication, 31

Identificación Proceso que establece la identidad de un usuario en el sistema, 40

JDK Conjunto de utilidades para desarrollar aplicaciones en lenguaje Java, 31

LaTeX Lenguaje de marcado y sistema de preparación de documentos, 33

MySQL Sistema gestor de base de datos que para proyectos no comerciales se distribuye bajo licencia GNU-GPL, 33

Needham-Schroeder Protocolo para la autenticación mutua de dos partes a través de una red, 40

No-repudio Imposibilidad de eludir la responsabilidad en autoría, 9

OpenSSL Librería *open-source* para generar claves y certificados, 33

RMI Java Remote Invocation Method o Método Java de Invocaciones Remotas, 32

UML Unified Modeling Language o Lenguaje de Modelado Unificado para especificar en modo gráfico un sistema software, 20

XML eXtensible Markup Language o lenguaje de marcas extensible, 32

Bibliografía

[1] Materiales de la asignatura *Criptografia*, Universitat Oberta de Catalunya, Barcelona 2006.

[2] Materiales de la asignatura *Metodologia i Gestió de Projectes Informàtics*, Universitat Oberta de Catalunya, Barcelona 2003.

[3] Materiales de la asignatura *Competència comunicativa per a professionals de les TIC*, Universitat Oberta de Catalunya, Barcelona 2007.

[4] Materiales de la asignatura *Comerç electrònic*, Universitat Oberta de Catalunya, 2002.

[5] Materiales de la asignatura *Auditoria, peritatges i aspectes legals per a informàtics*, Universitat Oberta de Catalunya, 2005.

[6] Java Platform, Standard Edition 6 API Specification [en línea] http://java.sun.com/javase/6/docs/api

[7] The Not So Short Introduction to LaTeX 2_ε [en línea] http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

Apéndices

Apéndice A

Código XML de los esquemas

Este apéndice incluye la codificación de los Esquemas XML descritos en el capítulo 7, pág. 46.

A.1. Esquema de los mensajes de autenticación Needham-Schroeder

El siguiente código XML es el esquema de los mensajes intercambiados durante el protocolo de autenticación Needham-Schroeder.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.uoc.edu/informatica/pfc/seguretat"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.uoc.edu/informatica/pfc/seguretat"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="NeedhamMsg">
    <xs:annotation>
      <xs:documentation>Mensaje para intercambiar peticiones y respuestas
      del protocolo de autenticación Needham-Schroeder</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Ni" type="xs:base64Binary" minOccurs="0"/>
        <xs:element name="Ng" type="xs:base64Binary" minOccurs="0"/>
        <xs:element name="UserId" type="xs:string" minOccurs="0"/>
        <xs:element name="Consulta" type="xs:string" fixed="" minOccurs="0"/>
        <xs:element name="B64Visita" type="xs:base64Binary" minOccurs="0"/>
        <xs:element name="SignedVisita" type="xs:base64Binary" minOccurs="0"/>
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.2. Esquema de los historiales

El siguiente código XML es el esquema de los historiales de pacientes.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.uoc.edu/informatica/pfc/seguretat"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.uoc.edu/informatica/pfc/seguretat"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="Historial">
    <xs:annotation>
      <xs:documentation>Historial de un paciente</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Dni" type="xs:string"/>
        <xs:element name="Nombre" type="xs:string"/>
        <xs:element name="Apellido" type="xs:string"/>
        <xs:element name="Visitas">
          <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:element name="Visita">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ClearVisita" type="xs:base64Binary"/>
                    <xs:element name="SmVisita" type="xs:base64Binary"/>
                    <xs:element name="NumOrder" type="xs:int"/>
                    <xs:element name="Time" type="xs:dateTime"/>
                    <xs:element name="SgVisita" type="xs:base64Binary"/>
                    <xs:element name="SgSmVisita" type="xs:base64Binary"/>
                    <xs:element name="SgT" type="xs:base64Binary"/>
                    <xs:element name="SgNumOrder" type="xs:base64Binary"/>
                    <xs:element name="SgIdPaciente" type="xs:base64Binary"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.3. Esquema de las listas de pacientes de un médico

El siguiente código XML es el esquema de las listas de pacientes asignados a un médico.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.uoc.edu/informatica/pfc/seguretat"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.uoc.edu/informatica/pfc/seguretat"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="MedicoPacientes">
    <xs:annotation>
      <xs:documentation>Lista de pacientes asignados a un médico</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Medico" type="xs:string"/>
        <xs:element name="Pacientes">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DniPaciente" type="xs:string" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.4. Esquema de los mensajes de respuesta del servidor

El siguiente código XML es el esquema de los mensajes de respuesta del servidor para indicar el éxito o fracaso de procesar una petición.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.w3.org/2001/04/xmlenc#"
```

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://www.uoc.edu/informatica/pfc/seguretat"
targetNamespace="http://www.uoc.edu/informatica/pfc/seguretat"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="ServerResult">
    <xs:annotation>
      <xs:documentation>Respuesta de éxtio o error del servidor
      tras procesar una petición</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:choice>
        <xs:element name="ErrorDescription" type="xs:string"/>
        <xs:element name="ResultDescription" type="xs:string"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```