**Universidad de las Ciencias Informáticas**

**Facultad 7**


# Sistema para la transmisión de imágenes médicas:

# Alas PACS DICOM Mail

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

**Autor:** Yunior González Castilla

**Tutor:** Msc. Pedro Medina Riesgo

**Co-tutor:** Msc. Héctor Raúl González Diez

Ciudad de La Habana, 23 de Junio de 2008

¨Año 50 de la Revolución¨

**DECLARACIÓN DE AUTORÍA**

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 23 días del mes de Junio del año 2008.

Yunior González Castilla

_____

Firma del Autor

Msc. Pedro Medina Riesgo

_____

Firma del Tutor

**DATOS DE CONTACTO**

TUTOR: Msc. Pedro Medina Riesgo (medina@uci.cu).

Profesor graduado de Licenciatura en Cibernética Matemática en la Universidad de La Habana. Posee categoría docente de Instructor. Ha impartido las asignaturas Programación II y Programación IV. Es profesor de la facultad 7 y se desempeña actualmente como Jefe de Proyecto dentro del Grupo de Procesamiento de Imágenes (GPI) de la Universidad de las Ciencias Informáticas (UCI).

CO-TUTOR: Msc. Héctor Raúl González Diez (hglez@uci.cu)

Profesor graduado de Licenciatura en Física Nuclear. Ha impartido las asignaturas de Física I, Física II, Matemática 3 y Matemática 4. Es profesor de la facultad 7 y se desempeña actualmente como Jefe del Polo de Procesamiento de Imágenes de la Universidad de las Ciencias Informáticas (UCI).

**AGRADECIMIENTOS**

A Lester y Pedro por que formamos un buen equipo.

A Aliuska por haberme ayudado en la realización de este trabajo.

A Karel que me respondió cuando lo necesité.

**DEDICATORIA**

En primer lugar dedico este trabajo a mis padres Elsita y Luis por darme su apoyo incondicional.
A mi hermano David para que siga esforzándose por lograr ser ingeniero.
A mi tío por haber sido mi ejemplo en la vida profesional.
A mis abuelos por sus sabios consejos.
A Aliuska por su amor, paciencia y comprensión.
A todos mis amigos que son muy importantes para mí.

**RESUMEN**

Las instituciones hospitalarias cubanas responden a un modelo de colaboración; es común que estudios radiográficos realizados en una institución sean diagnosticados en otra. Por lo que se necesita desarrollar una solución de software que permita la transmisión e intercambio global de imágenes médicas entre instituciones.

El sistema Alas PACS DICOM Mail presenta una alternativa para propiciar la transmisión de imágenes médicas de una forma personalizada. Mediante una interfaz similar a la de los correos electrónicos tradicionales, los médicos y especialistas pueden enviarse estudios, series e imágenes médicas. La transmisión de dichos ficheros, ocurre según lo establece el estándar internacional para el manejo de imágenes médicas DICOM en su versión actual 3.0. El producto fue desarrollado utilizando la herramienta de desarrollo Visual Studio 2005 y codificado en C# 2.0.

Este producto forma parte del sistema Alas PACS. Se encuentra actualmente en proceso de despliegue. Para su uso en hospitales y centros de salud con interés radiológico.

**PALABRAS CLAVES**

Alas PACS DICOM Mail, Informática Médica, Imágenes en Medicina, Transmisión de Imágenes Médicas, DICOM 3.0.

**TABLA DE CONTENIDOS**

## INTRODUCCIÓN

La evolución de la tecnología de adquisición de imágenes médicas ha supuesto entre otros avances, la incorporación de la imagen en formato digital. Han surgido numerosas modalidades en esta área como son la Tomografía Axial Computarizada (CT), Resonancia Magnética (MRI), Ultrasonido, Medicina Nuclear (NMI), Angiografía de Substracción Digital (DSA), entre otras.

Es muy común encontrarse con equipos de varios fabricantes, para las diferentes modalidades de imágenes que se generan; el tratar de integrar todos ellos en un sistema que los manipule es prácticamente imposible. Sobre esta base surgió la necesidad de estandarizar el manejo y transmisión de imágenes médicas digitales. Para ello, después de tres años de esfuerzo por lograr dicho estándar, se publicó la versión ACR[1] - NEMA[2] DICOM[3], hoy se encuentra en su versión actual DICOM 3.0.

La gran cantidad de imágenes generadas para diagnóstico en instituciones hospitalarias, ha hecho complicado su manejo. La gestión manual de dicho volumen de datos por médicos resulta muy engorrosa. Una alternativa es el manejo de imágenes digitales en forma eficiente, a través de dispositivos conectados en red, que en conjunto ofrecen una serie de servicios que dan soporte a la operatividad de un área radiológica. Sin embargo, para obtener una buena aceptación en el medio clínico, se deben considerar la facilidad, rapidez, seguridad en el acceso de imágenes y la calidad en su presentación.

Actualmente existen sistemas que realizan el manejo de imágenes; son encargados de la adquisición, almacenamiento, visualización y transmisión de imágenes médicas y son conocidos como PACS[4], donde la comunicación en ambiente de red es la parte medular para el diseño de aplicaciones.

Las autoridades cubanas de salud avanzan en la introducción de nuevas tecnologías en los diversos centros asistenciales distribuidos por la geografía nacional. La adquisición de equipos imagenológicos de alta tecnología así como el establecimiento de infraestructuras de redes computacionales en hospitales, son algunos de los esfuerzos que se están realizando hoy para la modernización del sistema nacional de salud. Sin embargo la interoperabilidad de todo el equipamiento en los hospitales y la integración inter institucional son aspectos muy poco desarrollados.

---

[1] American College of Radiology.
[2] National Electrical Manufacturers Association.
[3] Digital Imaging and Communications in Medicine.
[4] Picture Archiving and Communications Systems.

Las instituciones cubanas de salud responden a un modelo de colaboración. Es bastante común que imágenes que son generadas en una institución sean diagnosticadas en otra, por lo que es necesario un mecanismo eficaz para la transmisión e intercambio de imágenes médicas.

Los sistemas actualmente en uso dentro de las instituciones hospitalarias cubanas, intentan resolver el problema de la transmisión de imágenes mediante soluciones informáticas, aunque no cumplen con las necesidades actuales ya que presentan disímiles limitaciones en aspectos como son la facilidad de uso, la seguridad de los datos, compatibilidad con el estándar DICOM, entre otras.

Definidas las dificultades actuales en el área de la trasmisión de imágenes médicas entre instituciones hospitalarias, se plantea el problema siguiente: ¿Cómo propiciar la transmisión de imágenes médicas entre instituciones hospitalarias?

**Objetivo general.**

Desarrollar una solución de software que permita la transmisión e intercambio global de imágenes médicas entre instituciones.

**Objeto de estudio.**

La transmisión de imágenes médicas.

**Campo de acción.**

La transmisión de imágenes médicas entre instituciones hospitalarias.

**Tareas de la investigación.**

Para dar cumplimiento al objetivo propuesto se definieron las siguientes tareas:

- Analizar el estándar DICOM 3.0, en específico los protocolos de transmisión.
- Desarrollar un módulo de transmisión de imágenes médicas conforme al estándar DICOM 3.0.
- Desarrollar una estación cliente para lograr la interacción con los usuarios del sistema.
- Desarrollar un servidor de intercambio con vista a poder establecer una comunicación entre dos instancias ubicadas en subredes diferentes.
- Diseñar el modelo de datos.

**Métodos teóricos.**

**Analítico - Sintético.**

La utilización de este método permitió descubrir los distintos elementos que componen el objeto de la investigación, además de integrar los elementos en una unidad nueva, para una comprensión total de la esencia del mismo.

**Modelación.**

Este método permitió la creación de abstracciones con vistas a explicar la realidad. Se utilizó el lenguaje UML[5] para modelar los diferentes artefactos de software que ayudaron a representar el sistema de una forma simplificada.

[5] Unified Modeling Language

**Métodos empíricos.**

**Observación.**

Mediante este método se puede complementar la revisión de las características propuestas para el sistema, así como tener una herramienta de control adicional a la hora de llevar a cabo los procesos de software.

**Técnicas de recolección de información.**

**Entrevista.**

Es una técnica que aporta mucho a la hora de tener una visión detallada del asunto a tratar, debido a que la entrevista a personas bien documentadas sobre un tema, sin dudas aporta mucha información relevante.

**Estructura del documento.**

El contenido del presente documento está distribuido en cuatro capítulos. A continuación se desglosan los temas contenidos en cada uno:

- **Capítulo 1:** Fundamentación teórica.

Se aborda el estado del arte del tema a tratar a nivel internacional y nacional; las técnicas, tecnologías y software usados en la actualidad; así como los que forman parte de la solución. Se tratan los conceptos fundamentales para una correcta comprensión del tema.

- **Capítulo 2:** Características del sistema.

Este capítulo contiene la propuesta del sistema y los requerimientos, tanto funcionales como no funcionales. El modelado del dominio se presenta como alternativa para el entendimiento de los procesos de negocio. Se compone además de los actores del sistema, la especificación de los casos de uso del sistema y sus descripciones, finalizando con el diagrama de casos de uso del sistema.

- **Capítulo 3:** Arquitectura y diseño del sistema.

Se presenta una descripción de la arquitectura del sistema. Los diagramas de clases del diseño que describen la realización de los casos de uso están insertados en este capítulo. En el diseño aparecen tres tipos de clases las cuales se clasifican en: entidad, que modelan información que posee larga vida y que es a menudo persistente; interfaz, modelan la interacción entre el sistema y sus actores y por último la clase controladora, que coordina la realización de los casos de usos.

- **Capítulo 4:** Implementación y Resultados.

En este capítulo se presenta la vista de implementación, la cual está formada por los diagramas de componentes y de despliegue, este último resulta útil en la comprensión de cómo quedará distribuido el sistema en nodos físicos. Se realizan algunas pruebas para comprobar la efectividad de los mecanismos que implementa el mismo, y se muestran los resultados obtenidos.

## CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se aborda el estado del arte del tema a tratar a nivel internacional y nacional, las técnicas, tecnologías y software usados en la actualidad, así como los que formarán parte de la solución. Se tratan los conceptos fundamentales para una correcta comprensión del tema.

### 1.1. Estándar DICOM.

#### 1.1.1. DICOM (Digital Imaging and Communications in Medicine).

En las unidades de radiología de los hospitales, es muy común encontrarse con equipos de varios fabricantes, para las diferentes modalidades de imágenes que se generan; el tratar de integrar todos ellos en un sistema que los manipule es prácticamente imposible. Debido a esto surgió la necesidad de estandarizar el manejo y transmisión de imágenes médicas digitales. Este trabajo se inició en 1983, con la integración de un comité formado por el "American College of Radiology" (ACR), representando a la comunidad de radiólogos y la "National Electrical Manufacturers Association" (NEMA), representando a la industria en el área de radiología, de acuerdo a los procedimientos establecidos por NEMA. Los objetivos iniciales fueron trabajar con los diferentes problemas de compatibilidad, con el fin de homogenizar los ambientes propietarios de las diferentes modalidades de imágenes.

Específicamente:

- Promover la comunicación entre imágenes digitales independientemente del fabricante que las produjo.
- Ofrecer mayor flexibilidad a los sistemas de almacenamiento y comunicación de imágenes.
- Facilitar la creación y consulta a sistemas de diagnóstico por diferentes dispositivos y en diversos lugares locales o remotos.

Los primeros resultados en los trabajos de estandarización fueron publicados en 1985, ACR-NEMA Versión 1.0, teniendo como base ideas obtenidas de formatos ya existentes. Por ejemplo, la definición de elementos de datos de longitud variable identificados con etiquetas (formato de etiquetas), fue adoptada

de un estándar para grabar imágenes en cinta magnética, desarrollado por la Asociación Americana de Físicos en Medicina (AAPM). Sin embargo, como todas las primeras versiones, se detectaron varios errores y el comité encargado (ACR/NEMA) autorizó a los grupos de trabajo involucrados, la realización de dos revisiones en Octubre de 1986 y en Enero de 1988, que produjeron una segunda versión, ACR-NEMA Versión 2.0, en 1988.

A la par de la publicación de la segunda versión, surgió la demanda de interfaz entre dispositivos involucrados en la generación y manejo de imágenes y redes de cómputo, sin embargo, el estándar no ofrecía ningún soporte de comunicación en red. La respuesta a estas demandas implicaba grandes cambios a lo ya establecido, considerando como restricción principal el mantener la compatibilidad con las versiones anteriores, lo cual fue un gran reto para los grupos de trabajo. De esta forma, a partir de 1988 se comenzó a trabajar en una tercera versión, en donde el proceso de diseño sufrió un cambio radical adoptando modelos para simular el mundo real, modelos de capas o pila para comunicación entre sistemas heterogéneos utilizando protocolos de comunicación en red y el modelo de cómputo Cliente/Servidor para establecer asociaciones entre dispositivos compatibles, a través del envío de mensajes.

Después de tres años de esfuerzo, se publicó la versión ACR/NEMA DICOM (Digital Imaging and Communications in Medicine) llamada también DICOM 3.0, en la que participaron varias instituciones de la comunidad internacional como JIRA (Japanese Industry Radiology Apparatus) y CEN (European Committee for Standardization). Esta versión es considerada como un estándar completo, compatible con las versiones anteriores.

### 1.1.2. Comunicación DICOM.

DICOM utiliza el modelo de capas para representar conexiones virtuales entre diferentes plataformas de cómputo, utilizando protocolos de comunicación. Para establecer una conexión virtual, los dispositivos que pretenden comunicarse deben utilizar los mismos protocolos en cada capa. Además, agrega la posibilidad de conexión en red utilizando como base los protocolos TCP/IP[6] y los propuestos por ISO/OSI[7]. De esta

[6] Transmission Control Protocol/Internet Protocol

forma se aprovechan los protocolos definidos en las capas inferiores tanto de TCP/IP como de ISO/OSI y define los protocolos necesarios en las capas superiores para soportar la comunicación entre aplicaciones en forma eficiente. En el caso de ISO/OSI, aprovecha los servicios de las primeras seis capas. Para el caso de TCP/IP, especifica un protocolo de capa superior DUL[8]. Para ambos casos se define un protocolo para aplicaciones DICOM, que permite la portabilidad entre ambos ambientes sin afectar las aplicaciones ya realizadas. (Ver Figura 1.)

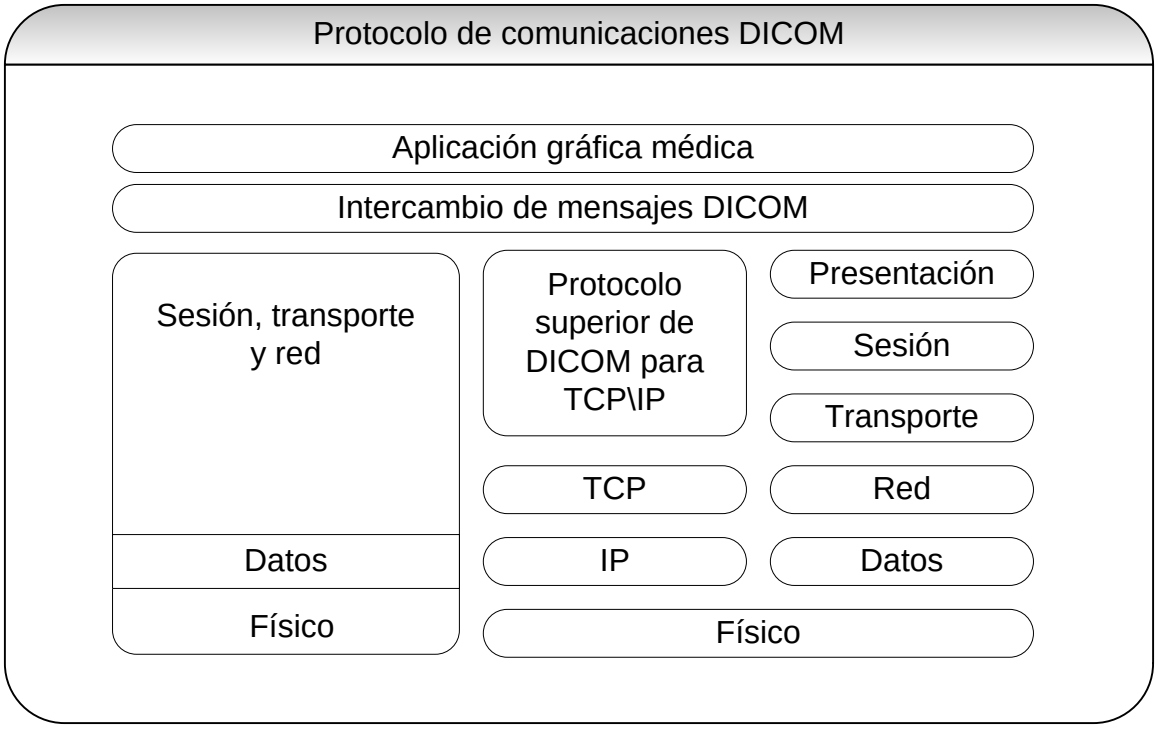



**Figura 1. Protocolo de comunicaciones DICOM.**

El estándar establece que antes de transmitir una imagen médica, debe efectuarse una asociación entre las partes involucradas, es decir las dos aplicaciones que actúan conforme a DICOM como transmisor y receptor, y que utilizan dicho protocolo para el transporte de la imagen. En esta asociación se negocian las condiciones y detalles de la transmisión, las cuales a la postre garantizan la seguridad e integridad de

[7] International Standards Organization/Open Systems Interconnection
[8] DICOM Upper Layer

los datos. Un esquema simplificado y sencillo de esta interacción se muestra en la figura a continuación. (Ver Figura 2.) (Medina, 2007)

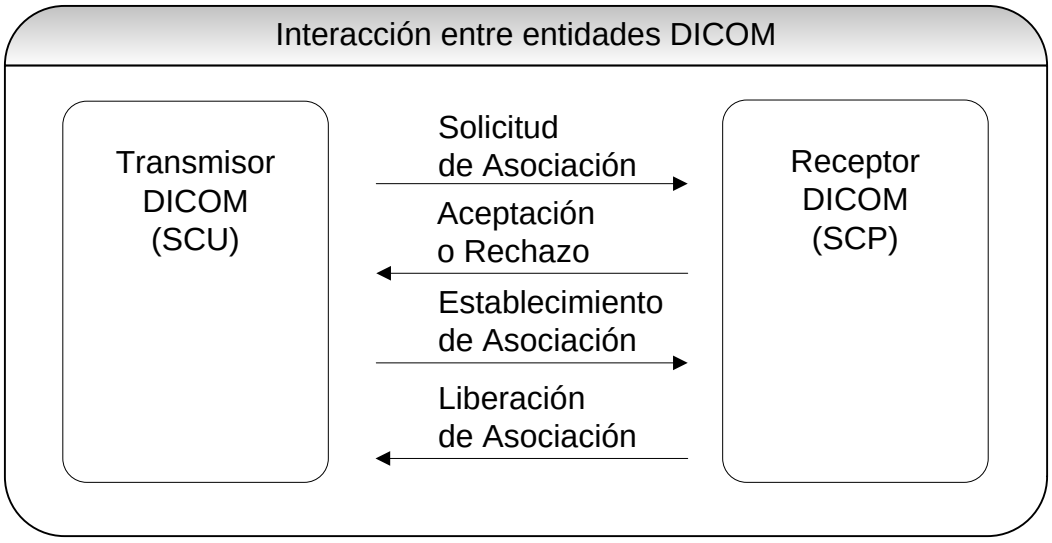



**Figura 2. Interacción entre entidades DICOM.**

### 1.2. PACS (Picture Archiving and Communications Systems).

Actualmente existen sistemas que realizan el manejo de imágenes médicas digitales, conocidos como PACS (Picture Archiving and Communications Systems), en donde la comunicación en ambiente de red es la parte medular para el diseño de aplicaciones. Para el caso de las áreas de radiología en los hospitales, también es importante tener una idea clara de la forma de operar, basada en las necesidades del hospital para poder integrar un sistema de este tipo. Algunos aspectos que se tienen en consideración a la hora de automatizar el manejo de las imágenes son:

- Los diferentes mecanismos de admisión y registro de pacientes.
- Los tipos y número de pacientes que se atienden en el servicio de radiología.
- La información relacionada al paciente, considerada como relevante para el hospital.
- Las diferentes modalidades de imágenes médicas que se manejan en el área.

- Los diferentes ambientes en la obtención de imágenes (por ejemplo, radionúclidos en imágenes de medicina nuclear).
- La existencia de otros sistemas de información en el hospital.
- El mecanismo de petición de estudios al servicio de radiología.
- La forma en que el servicio de radiología programa los estudios de pacientes.
- El mecanismo de interpretación y diagnóstico de los estudios realizados.
- Las diferentes áreas del hospital que requieren consultar imágenes.
- La ubicación física de las diferentes áreas involucradas.
- La forma de manejar la información a donde es requerida.
- La utilidad que se le da a la información, en cada servicio.
- Problemas de pérdida de información.
- Localización final de la información cuando el paciente abandona el hospital.
- La forma de organizar la información al ser almacenada.
- Consultas posteriores a la información.
- Intercambio de información con otros hospitales.

Para cubrir estas necesidades se requiere de un conjunto de dispositivos, cuyas responsabilidades son el ofrecer todos los elementos operativos demandados por el área de radiología y áreas dependientes, dentro de un hospital. Estas demandas incluyen:

- Adquisición de Imágenes.
- Almacenamiento de Información.
- Distribución de Imágenes.

- Visualización de Imágenes (consulta, interpretación o diagnóstico).
- Registro de Resultados.
- Interfaz con Otros Sistemas.
- Comunicación Remota.
- Seguridad del Sistema.

Como puede inferirse de las necesidades antes planteadas, los sistemas PACS, utilizan varios componentes (hardware y software) con funciones específicas. Estos componentes son: Digitalizadores laser para placas de Rayos X, digitalizadores de video, estaciones de trabajo con diferentes características, estaciones de consulta, medios de almacenamiento óptico y magnético, servicios de impresión, infraestructura para servicios de red, servidores de imágenes, servidores de bases de datos, dispositivos que generan imágenes médicas digitales y servicios de comunicación a sistemas remotos externos, entre otros. (Ver Figura 3.)

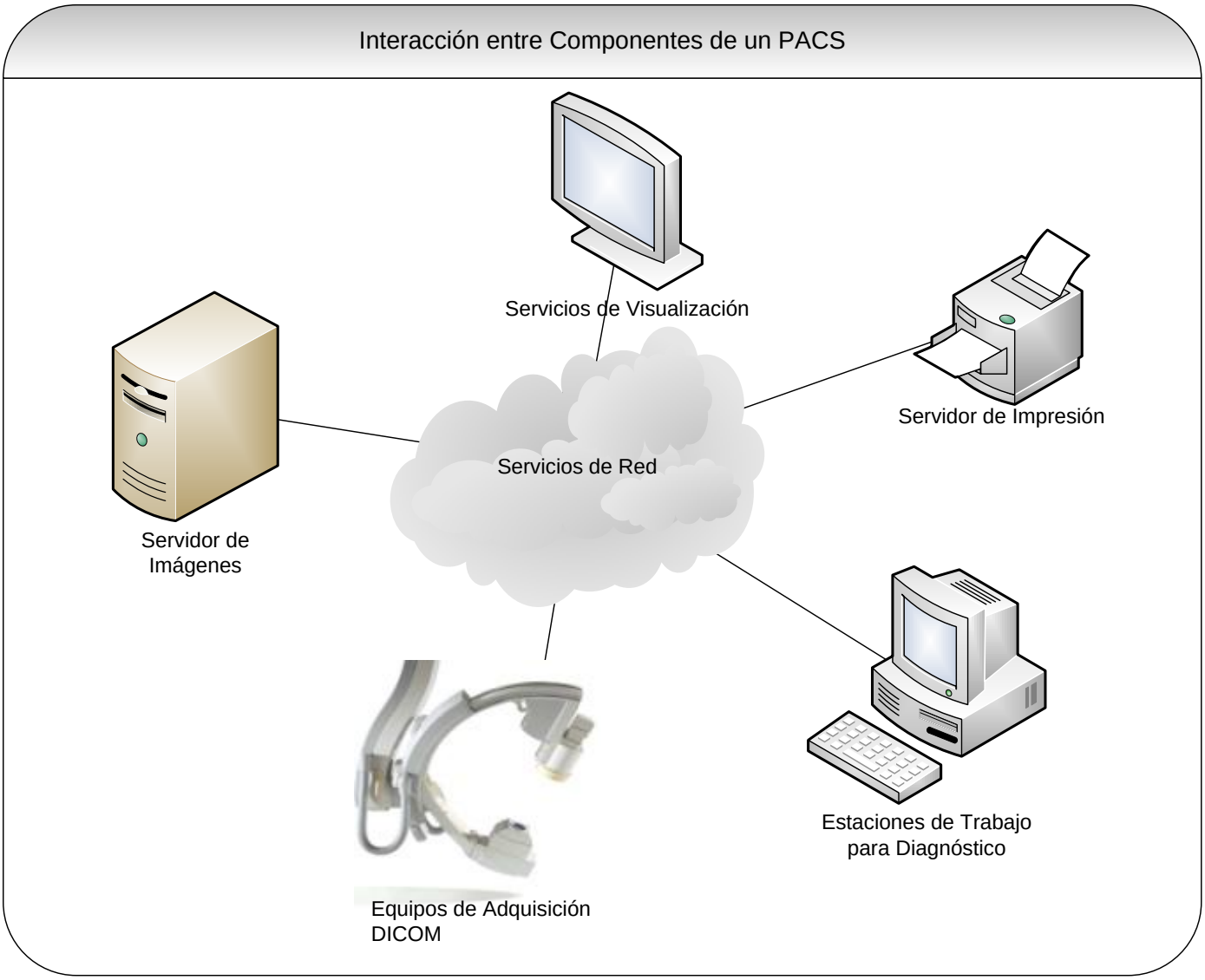



**Figura 3. Interacción entre componentes de un PACS.**

Como se observa en la Figura 3, estos componentes se integran en un esquema Cliente/Servidor en una red de cómputo, para ofrecer los diferentes servicios demandados por el área de radiología de un hospital y así cumplir con sus requerimientos operativos.

## 1.3. Estado del arte.

### 1.3.1. Mundo.

Cada día es más común en el entorno clínico el término de interconsulta. La segunda opinión se ha convertido en un método muy rico a la hora de lograr un diagnóstico certero y confiable. Para ello se necesita que las imágenes generadas en un hospital, puedan ser transmitidas a un lugar donde se realice su análisis.

El Grupo de Trabajo de Tecnología de la Información de la Sociedad Radiológica Alemana puso en marcha una iniciativa para la normalización de la telemedicina. El correo electrónico fue seleccionado como la base ideal para conexiones en teleradiología, ya que es a la vez fácil de implementar y usar. El estándar DICOM ofrece la opción de adjuntar datos de un fichero de este tipo a un correo electrónico como ocurre con cualquier adjunto tipo MIME[9], lo que permite la transmisión de los datos de un DICOM original. La primera recomendación fue presentada en la 85 Convención de Radiología Alemana, en Wiesbaden y en 2004 fue honrado el grupo de TI – SRA con el Premio de Roentgen.

Existen en el mundo varios sistemas que utilizan estos principios, a continuación se presentan algunos ejemplos de soluciones de este tipo.

- La compañía alemana Sistemas Médicos Steinhart Ltd. cuenta entre sus desarrollos con un módulo de correo electrónico DICOM. Dicho módulo se puede utilizar para enviar y recibir imágenes a través del correo electrónico de acuerdo a Suplemento 54 DICOM. Además permite el intercambio de imágenes con sistemas de otros fabricantes.

  Características del sistema:

  - Transmisión de imágenes a través de correo electrónico.
  - Encriptación PGP.
  - Compresión ZIP (factor 2).
  - Intercambio de imágenes con sistemas de otros fabricantes.

[9] Extensión de correo electrónico para multipropósito en Internet (Multipurpose Internet Mail Extensions)

- Permite dividir grandes imágenes o series según el tamaño máximo aceptado por el proveedor.
- DICOM Suplemento 54.

- La empresa norteamericana RADinfo SYSTEMS, Inc. desarrolló la solución PowerPACS DICOMmail, dicho módulo trae integrado el visor gratuito Basic RSVS de la misma empresa con el objetivo de obtener una consulta rápida.

  Características fundamentales:

  - Enviar imágenes DICOM vía correo electrónico.
  - Conversión de imágenes a JPEG, GIF, BMP en formato DICOM.
  - Anonimizar imágenes DICOM de acuerdo con parámetros de usuarios.
  - Manejo y transmisión de imágenes acorde a DICOM.
  - Permite el envío de video y audio en estudios de imágenes.

### 1.3.2. Cuba.

En el empeño por dar solución al problema de cómo transmitir imágenes médicas entre instituciones hospitalarias del país, se fueron sucediendo un grupo de soluciones que presentaban muchas dificultades en cuanto su facilidad de uso, seguridad, confiabilidad entre otros aspectos. Entre estas soluciones se encuentran las siguientes:

- **Protocolo SMTP[10].**

Una solución radical y muy engorrosa que consistía en compactar con WinRar[11] una imagen DICOM y dividirla en pequeños fragmentos, luego adjuntar cada uno a un correo tradicional ejemplo: Outlook[12],

[10] Simple Mail Transfer Protocol.
[11] Programa Compresor y Descompresor de datos.
[12] Programa Gestor de Correos Electrónicos.

luego enviar todos esos correos y en el lugar que se recibían, armar nuevamente los fragmentos para así completar la trasmisión de una sola imagen.

Este modelo de transmisión no garantiza la seguridad, integridad, confiabilidad y confidencialidad de la información del archivo DICOM.

- **Direcciones IP públicas.**

Esta transmisión ocurría según establece el estándar DICOM sólo que no existía ningún sistema que gestionara del todo el proceso. Además de que se necesitaba que ambas entidades en la comunicación tuvieran un IP real. Esto es muy costoso y trae muchos problemas en el área de seguridad informática de una institución.

- **Transmisión mediante Web Services.**

Esta solución es mejor que las anteriores solo que descuida un punto importante como es la rapidez de transmisión, esta tecnología (Web Services), en muchos casos es muy útil pero tiene la limitante de que la velocidad de transmisión, es muy lenta.

## 1.4. Tecnologías utilizadas.

### 1.4.1. .NET Framework.

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años. Para el desarrollo y ejecución de aplicaciones en este nuevo entorno tecnológico Microsoft proporciona el conjunto de herramientas conocido .NET Framework SDK, que es posible descargarlo gratuitamente de su sitio web [http://www.msdn.microsoft.com/net] e incluye compiladores de lenguajes como C#, Visual Basic.NET, Managed C++ y JScript.NET específicamente diseñados para crear aplicaciones para él.

**Elementos principales .NET Framework:**

- **CLR** El corazón de la plataforma.NET es el CLR (Common Language Runtime), que es una aplicación similar a una máquina virtual que se encarga de gestionar la ejecución de las aplicaciones para ella escritas. A estas aplicaciones les ofrece numerosos servicios que facilita su desarrollo y mantenimiento y favorecen su fiabilidad y seguridad. Entre ellos los principales son:

    - Modelo de programación consistente y sencillo, completamente orientado a objetos.
    - Eliminación del temido problema de compatibilidad entre DLLs conocido como "infierno de las DLLs".
    - Ejecución multiplataforma.
    - Ejecución multilenguaje, hasta el punto de que es posible hacer cosas como capturar en un programa escrito en C# una excepción escrita en Visual Basic.NET que a su vez hereda de un tipo de excepción escrita en Cobol.NET. Aunque más arriba se ha dicho que en el .NET Framework sólo se ofrecen compiladores de C#, MC++, VB.NET y JScript.NET, lo cierto es que aparte Microsoft y terceros han -o están- desarrollado versiones adaptadas a .NET de muchísimos otros lenguajes como APL, CAML, Cobol, Eiffel, Fortran, Haskell, Java, Mercury, ML, Mondrian, Oberon, Oz, Pascal, Perl, Python, RPG, Scheme o Smalltalk.
    - Recolección de basura.
    - Aislamiento de memoria entre procesos y comprobaciones automáticas de seguridad de tipos en las conversiones.
    - Soporte multihilo.
    - Gestión del acceso a objetos remotos que permite el desarrollo de aplicaciones distribuidas de manera transparente a la ubicación real de cada uno de los objetos utilizados en las mismas.
    - Seguridad avanzada, hasta el punto de que es posible limitar los permisos de ejecución del código en función de su procedencia (Internet, red local, CD-ROM, etc.), el usuario que lo ejecuta o la empresa que lo creó.
    - Interoperabilidad con código preexistente, de manera que es posible utilizar con facilidad cualquier librería de funciones u objetos COM y COM+ creados con anterioridad a la aparición de la plataforma .NET.

- Adecuación automática de la eficiencia de las aplicaciones a las características concretas de cada máquina donde se vaya a ejecutar.

- **El conjunto de clases del .NET Framework** es la piedra angular de cualquier desarrollador de .NET, es un rico conjunto de clases, interfaces, tipos que simplifican y optimizan el desarrollo de aplicaciones .NET además de proporcionar acceso a la funcionalidad del sistema. Como desarrolladores el dominio de este conjunto de clases es vital para un buen desarrollo en .NET.

- **Servicios Windows** Los Servicios de Windows son aplicaciones que funcionan sin interactuar directamente con el usuario y por regla general se inician junto con el sistema, sin que ningún usuario tenga que hacerlo.

El Framework .Net cuenta con las clases necesarias para crear Servicios Windows. Además permite la instalación y monitoreo de los mismos.

**Windows Forms**, parte del .NET Framework que presenta un conjunto de componentes visuales que permiten la interacción con los usuarios. (González, 2001)

## 1.5. Herramienta de desarrollo.

### 1.5.1. Visual Studio 2005.

Visual Studio .NET es un conjunto completo de aplicaciones para la creación tanto de aplicaciones Windows, aplicaciones Web como servicios Web para trabajo en equipo, permitiendo utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías para simplificar el diseño, desarrollo e implementación en equipo de soluciones para la empresa. Este entorno está creado para poder manejar proyectos que usen más de un lenguaje a la vez, teniendo en cuenta la característica multilenguaje de la plataforma.

Incluye lenguajes de programación como Visual Basic.NET, Visual C++. NET que permite tanto escribir código adaptado a la plataforma. NET como código C++ nativo, Visual C#.NET es un nuevo lenguaje de programación orientado a objetos con el que se ha desarrollado parte de la plataforma .NET y por último Visual J#.NET que no es más que la adaptación de Visual J++ para la plataforma .NET. Dichos lenguajes

aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y servicios Web XML.

Dentro de las principales características de Visual Studio se puede encontrar que facilita el diseño de formularios Windows a través de una superficie de diseño grafico que permite al programador crear una interfaz de usuario. Tiene herramientas para el diseño de formularios Windows y plantillas de aplicaciones Windows. Tiene herramientas para el diseño de formularios Web y plantillas de aplicaciones Web. Posee herramientas para el desarrollo de servicios Web basado en XML. Para el acceso a distintos tipos de datos cuenta con clases y componentes visuales. Cuenta con herramientas de depuración de código que puede utilizarse en todos los lenguajes soportados por Visual Studio.

## 1.6. Lenguaje de programación.

### 1.6.1. C# 2.0.

C# es un lenguaje de programación orientado a objetos desarrollado por la Corporación Microsoft como parte de su iniciativa titulada .NET, la cual se creó como respuesta al éxito del lenguaje de programación Java creado por Sun Microsystems. El código de fuente de C# —al igual que de los otros lenguajes .NET— es compilado en un lenguaje intermedio llamado MSIL (siglas que significan "Lenguaje Intermedio de Microsoft" en inglés). C# se deriva principalmente de los lenguajes de programación C, C++, y Java con algunas características de Visual Basic. (Chang & Wagers, 2008)

**Características de C#:**(González, 2001)

- **Sencillez:** Elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo no necesita de ficheros de cabecera o ficheros IDL. El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile, lo que facilita la portabilidad del código. No se incluyen elementos poco útiles de lenguajes tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (**.**), etc.

- **Modernidad:** Incorpora en el propio lenguaje elementos como: un tipo básico *decimal* que permita realizar operaciones de alta precisión con reales de 128 bits, la inclusión de una instrucción *foreach* que

permite recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico *string* para representar cadenas o la distinción de un tipo *bool* específico para representar valores lógicos.

- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos. Este enfoque orientado es más puro en tanto que no admite ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades, eventos o atributos.

- **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje haciendo que no sea necesario incluir instrucciones de destrucción de objetos.

- **Seguridad de tipos:** Incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evitar que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura.

- **Sistema de tipos unificado:** Todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada *System.Object*, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos")

- **Extensibilidad de tipos básicos:** Permite definir, a través de *estructuras*, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos.

- **Extensibilidad de operadores:** Permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.

- **Extensibilidad de modificadores:** Ofrece, a través del concepto de *atributos*, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET.

- **Versionable:** Incluye una *política de versionado* que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar.

- **Compatible:** Para facilitar la migración de programadores, C# mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes.

## 1.7. Herramientas de modelado.

### 1.7.1. Enterprise Architect 6.5.

Enterprise Architect ofrece las herramientas de modelado UML más potentes y flexibles para la plataforma de Windows. Es una herramienta de análisis de negocio y UML orientada a objetos para el desarrollo completo del ciclo de vida del software. Enterprise Architect provee el límite competitivo para el desarrollo de software, administración de proyecto, administración de requerimientos y análisis de negocio.

Entre sus características fundamentales se destacan: el soporte de la última especificación UML 2.1, la capacidad de Importación/Exportación XMI 2.1, motor de Reporte HTML, transformaciones MDA, perfiles y soporte de tecnologías, pruebas y rastreo de recursos y mantenimiento.

## 1.8. Librería que posibilitó el desarrollo.

### 1.8.1. MyDicom.NET SDK.

MyDicom.NET son librerías que implementan el estándar DICOM 3.0 y facilitan las tareas informáticas fundamentales como la transmisión, la utilización y almacenamiento de las imágenes médicas. Aunque en la actualidad existen varias librerías de este tipo (Java DICOM Toolkit, CTN), algunas de ellas incluso

gratis (DCMTK). Se escogieron las MyDicom debido a su facilidad de uso y que están implementadas en C#. Esto constituye un gran paso de avance por ser el mismo lenguaje de programación a utilizar en el proyecto. Entre otras funcionalidades: permiten trabajar las asociaciones múltiples lo cual confiere al servidor un poder de respuesta importante; brindan la posibilidad de la migración a la plataforma UNIX utilizando la alternativa a .NET MONO. Adicionalmente estas librerías ofrecen buenos resultados en rendimiento, estabilidad y confiabilidad. (Durañona, González, 2007)

## 1.9. Persistencia de datos.

### 1.9.1. XML (EXtensible Markup Language).

Para la persistencia de los datos se decidió usar como base XML debido a que el modelo de datos es bastante sencillo además de no presentar concurrencia en el acceso a los mismos. Las consultas son igualmente poco complejas y el volumen de información a manipular es pequeño. A continuación se explica en esencia que es XML.

XML (Lenguaje de Marca Extensible, E**X**tensible **M**arkup **L**anguage) es un lenguaje que garantizará el intercambio de cualquier tipo de información, sin que ocasione problemas de tipo "contenido" o de tipo "presentación". Este garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes.

Varias son las características que ofrece XML:

- Los documentos tienen una estructura que los hace legibles e inteligibles no sólo para los ordenadores, sino también para los humanos.
- Permite proporcionar diferentes vistas sobre los datos (HTML, PDF, voz, etc.), dependiendo de quién sea el cliente.
- Facilita la integración desde fuentes de datos heterogéneas, por ejemplo, páginas Web, distintas bases de datos, etc.
- Las aplicaciones de XML son fácilmente extensibles mediante definiciones de nuevos tipos de documento.

**Principales características.**

- Es una arquitectura más abierta y extensible. No se necesita versiones para que puedan funcionar en futuros navegadores. Los identificadores pueden crearse de manera simple y ser adaptados en el acto en internet/intranet por medio de un validador de documentos (Parser).

- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos del documento con XML.

- Se podrá hacer el intercambio de documentos entre las aplicaciones tanto en el propio PC como en una red local o extensa.

- Datos compuestos de múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje permitirá agrupar una variedad amplia de aplicaciones, desde páginas web hasta bases de datos.

## 1.10. Metodología utilizada.

### 1.10.1. RUP (El Proceso Unificado de Desarrollo).

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo. Los mismos son: Modelado del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Instalación, Administración del proyecto, Administración de cambio y configuración y Ambiente.

El ciclo de vida de RUP se caracteriza por ser Dirigido por casos de uso, Centrado en la arquitectura y además Iterativo e Incremental. (Dpto Ingeniería, 2006)

**¿Por qué usar RUP?**

El proceso de desarrollo RUP aplica varias de las mejores prácticas en el desarrollo moderno de software en una forma que se adapta a un amplio rango de proyectos y de organizaciones. Provee a cada miembro del equipo, un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software. Esta metodología permite que todos los integrantes de un equipo de trabajo, conozcan y compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software utilizando un lenguaje de modelado común: UML.

Algunas ventajas de la aplicación de esta metodología son las siguientes:

1. Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.
2. Reduce el costo del riesgo a los costos de un solo incremento.
3. Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo.
4. Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración.
5. Facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones lo cual además permite que se aprecien oportunidades de mejoras en el diseño.

## 1.11. Lenguaje de modelado.

### 1.11.1. UML (Lenguaje Unificado de Modelado).

**Lenguaje Unificado de Modelado** (**UML**, por sus siglas en inglés, **U**nified **M**odeling **L**anguage) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (BOOCH, et al., 2000). Está compuesto por diversos

elementos gráficos que se combinan para conformar diagramas. Es importante resaltar que es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

**¿Por qué usar el lenguaje de modelado UML?**

El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten (BOOCH, et al., 2000). Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

UML es además un método formal de modelado lo cual aporta las siguientes ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto. (Hernández, 2002)

En este capítulo se realizó un estudio de los principales aspectos relacionados con el tema, se analizaron los conceptos de DICOM y PACS. Se caracterizaron las principales tendencias de los sistemas de transmisión de imágenes médicas tanto a nivel nacional como internacional. El estudio de las diferentes tecnologías arrojó como resultado la selección de la herramienta de desarrollo Visual Studio 2005 así como la herramienta de modelado Enterprise Architect 6.5.

## CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Este capítulo contiene la propuesta del sistema y los requerimientos, tanto funcionales como no funcionales. El modelado del dominio se presenta como alternativa para el entendimiento de los procesos de negocio. Se compone además de los actores del sistema, la especificación de los casos de uso del sistema y sus descripciones, finalizando con el diagrama de casos de uso del sistema.

### 2.1. Propuesta del sistema.

El presente trabajo propone la automatización de los procesos de transmisión e intercambio global de imágenes médicas digitales. Para lograr esto, se propone un sistema que mediante el uso del protocolo estándar para la comunicación de imágenes digitales en medicina (DICOM 3.0), permita la transmisión de imágenes médicas.

El sistema promoverá un servicio de mensajería conforme a DICOM, que apoyado en una interfaz de correo electrónico, logrará abstraer a los usuarios de las complejidades y necesidades de la transmisión conforme al estándar. DICOM hace posible que los archivos médicos puedan viajar de forma segura entre hospitales, centros de investigación, etc. Luego esa información puede ser vista remotamente para que los médicos puedan diagnosticar desde su casa o buscar diferentes opiniones de otros expertos de una forma rápida y sencilla.

### 2.2. Modelo de dominio.

Considerando el proceso a automatizar descrito, se llega a la conclusión de que el negocio que se está estudiando tiene muy bajo nivel de estructuración por lo que se ve en la necesidad de utilizar un modelo del dominio porque permite mostrar los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema.

Se realizó un modelo del dominio debido a las características del negocio, además de no presentar procesos de negocios bien definidos. Esto ayuda a los usuarios, clientes y desarrolladores e interesados a

utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. Para capturar correctamente los requisitos y poder construir un sistema correcto se necesita tener un firme conocimiento del funcionamiento del objeto de estudio. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema.

### 2.2.1. Diagrama de dominio.

El modelo del dominio se describe mediante diagramas UML, específicamente con un diagrama de clases conceptuales significativas en el dominio del problema. (Ver Figura 4.)

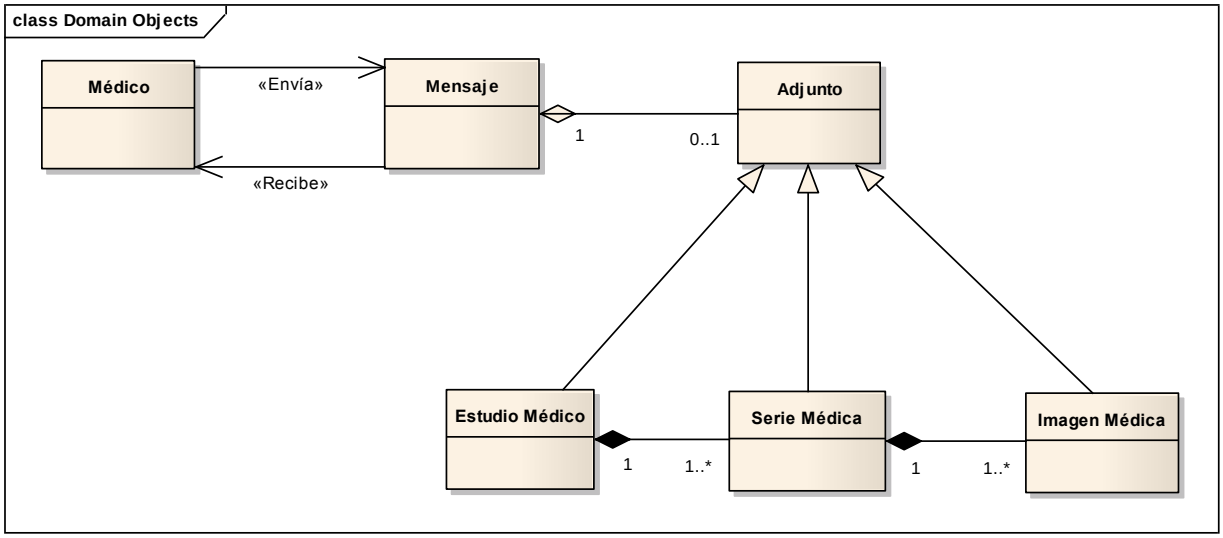



**Figura 4. Diagrama de dominio.**

## 2.3. Requerimientos.

### 2.3.1. Requerimientos Funcionales.

| Nombre del Requerimiento | Descripción |
|---|---|
| RF1 - Redactar mensaje. | El sistema debe permitir al usuario redactar un mensaje. |

| | |
|---|---|
| RF2 - Enviar mensaje. | El sistema debe permitir al usuario enviar un mensaje ya creado. |
| RF3 - Adjuntar archivo DICOM. | El sistema debe permitir al usuario adjuntar a un mensaje tanto un DICOM, una Serie, como un Estudio completo. |
| RF4 - Prioridad del envío. | El sistema debe permitir al usuario clasificar con una prioridad de envío al mensaje. |
| RF5 - Recibir mensaje. | El sistema debe permitir recibir mensajes. |
| RF6 - Responder un mensaje. | El sistema debe permitir al usuario responder un mensaje recibido. |
| RF7 - Reenviar un mensaje. | El sistema debe permitir al usuario reenviar un mensaje existente. |
| RF8 - Eliminar mensaje. | El sistema debe permitir al usuario eliminar un mensaje el cual es enviado directamente hacia la Bandeja de<br>Elementos Eliminados. |

| | |
|---|---|
| RF9 - Notificar nuevo mensaje. | El sistema debe notificar al usuario la llegada de un nuevo mensaje. |
| RF10 - Crear cuentas de usuarios. | El sistema debe permitir al usuario administrador crear una nueva cuenta de usuario. |
| RF11 - Eliminar cuentas de usuarios. | El sistema debe permitir al usuario administrador eliminar una cuenta de usuario existente. |
| RF12 - Buscar cuentas de usuarios. | El sistema debe permitir al usuario administrador buscar las cuentas de usuario existentes. |
| RF13 - Adicionar un nuevo contacto. | El sistema debe permitir al usuario adicionar un nuevo contacto al libro de direcciones. |
| RF14 - Eliminar un contacto existente. | El sistema debe permitir al usuario eliminar un contacto existente en libro de direcciones. |
| RF15 - Buscar un contacto. | El sistema debe permitir al usuario buscar un contacto existente en libro de direcciones a través de criterios determinados. |
| RF16 - Editar un contacto | El sistema debe permitir al usuario buscar un contacto existente |

| | |
|---|---|
| existente. | en libro de direcciones. |
| RF17 - Configuración del idioma. | El sistema debe permitir al usuario configurar el idioma de la Interfaz:(Español, Inglés o Chino). |
| RF18 - Configuración del servidor (IP). | El sistema debe permitir al usuario administrador configurar el IP del servidor o Proxy donde se instalará la aplicación. |
| RF19 - Comprimir archivos DICOM. | El sistema debe permitir al usuario especificar si los ficheros DICOM, serán enviados comprimidos o no. |
| RF20 - Anonimizar archivos DICOM. | El sistema debe permitir al usuario configurar la opción de seleccionar si los ficheros DICOM se anonimizarán según la configuración vigente o si debe configurarse justo antes de efectuar la transmisión. |
| RF21 - Libro de direcciones | El sistema debe permitir almacenar los contactos en un libo de direcciones. |
| RF22 - Atajo de teclado | El sistema debe brindar al usuario la opción de atajo de teclado para las operaciones que más se usan. |

### 2.3.2. Requerimientos no Funcionales.

| Nombre del Requerimiento | Descripción |
|---|---|
| RNF1 - Usabilidad | El sistema debe tener una interfaz cómoda al usuario, debe respetar al máximo la experiencia del mismo con el objetivo de facilitar su uso. |
| RNF2 - Rendimiento | En la recepción de mensajes, el sistema debe soportar alta concurrencia. Así como el envío de mensajes debe de ser de forma asincrónica. |
| RNF3 - Seguridad | El acceso al sistema deberá ser controlado por sesiones de manera tal que la información sea accedida solo por usuarios que tengan permiso. Además la información viajará encriptada por canales inseguros. |
| RNF4 - Software | Debe de estar instalado .NET Framework 2.0 o superior.<br>Microsoft Windows 98 o superior. |
| RNF5 - Alcance | El sistema debe lograr comunicación entre dos instancias ubicadas en dos puntos geográficos cualesquiera que sean, mientras que estos tengan conectividad. |

## 2.4. Definición de los casos de uso por iteraciones.

| Caso de uso | Iteración |
|---|---|
| Gestionar Mensaje. | **Primera iteración.** |
| Adjuntar ficheros DICOM. | |
| Enviar Mensaje. | |
| Recibir Mensaje. | |
| Notificar nuevo Mensaje. | |
| Anonimizar ficheros adjuntos. | **Segunda iteración.** |
| Comprimir ficheros adjuntos. | |
| Gestionar cuenta de usuario. | |
| Buscar cuenta de usuario. | |
| Autenticar usuario. | |
| Gestionar contacto. | **Tercera iteración.** |
| Buscar contacto. | |

| | |
|---|---|
| Configuración de idioma. | |
| Configuración del servidor de intercambio. | |

### 2.5. Definición de los Actores del Sistema.

| Nombre del Actor | Descripción |
|---|---|
| Médico. | Es el usuario común del sistema, Tiene acceso a las funcionalidades:<br>• Recibir, enviar y gestionar mensaje.<br>• Gestionar y buscar contacto.<br>• Comprimir ficheros.<br>• Anonimizar ficheros.<br>• Configuración de idioma. |
| CU- Recibir Mensaje. | Inicia el caso de uso Notificar Mensaje. |
| CU- Enviar Mensaje. | Inicia el caso de uso Recibir Mensaje. |
| Administrador. | Es el usuario con más privilegio en el sistema. Inicia los casos de uso:<br>• Configuración del servidor de intercambio. |

| | |
|---|---|
| | • Buscar cuenta de usuario.<br>• Gestionar cuenta de usuario. |

### 2.6. Descripción de los Casos de Uso del Sistema.

La descripción detallada de los casos de uso del sistema para la primera iteración se puede ver en el **Anexo1**. Se presenta descrito textualmente el flujo de eventos entre el actor y la respuesta que da el sistema para cada caso.

## 2.7. Diagrama de Casos de Uso del sistema.

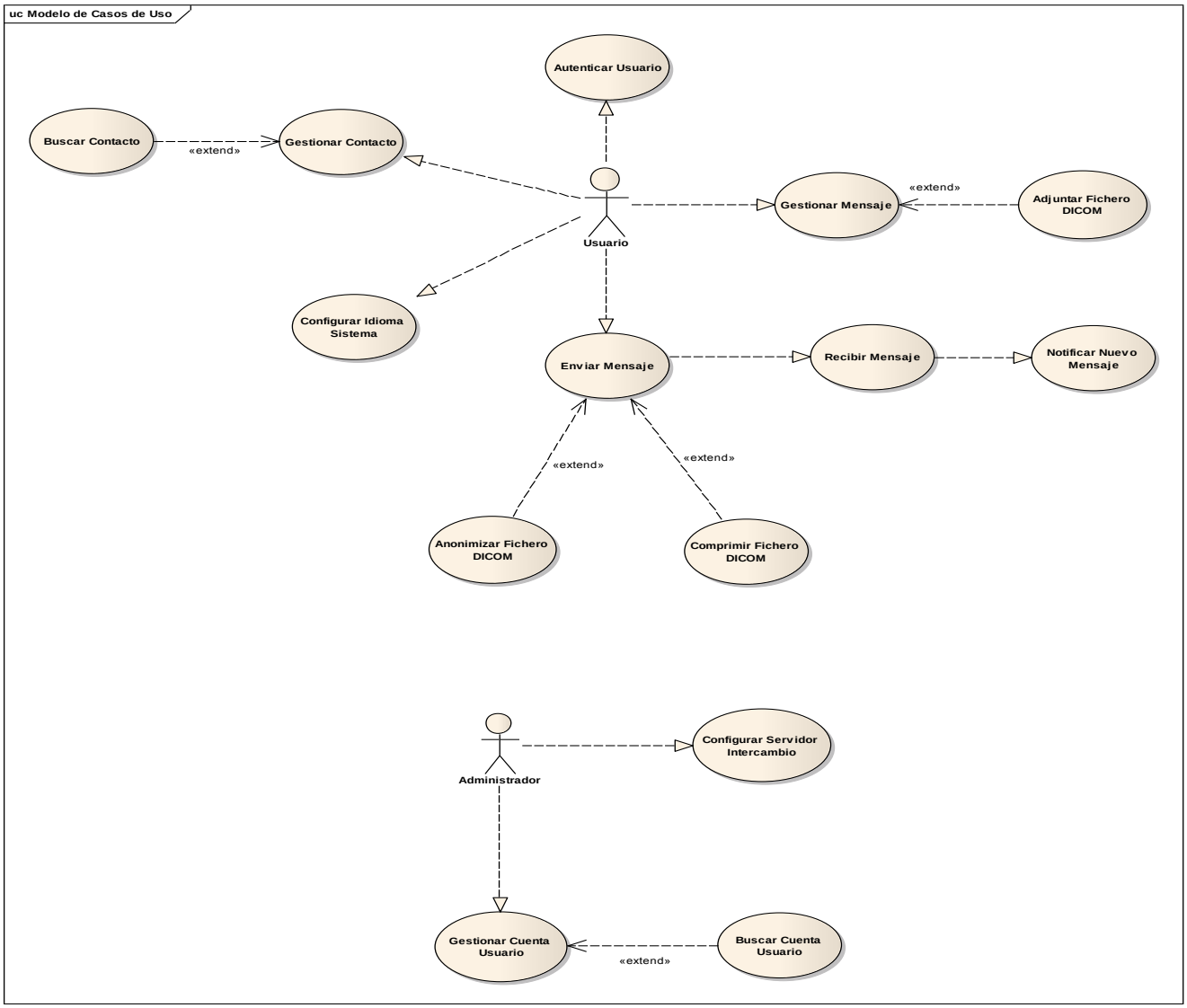



**Figura 5. Diagrama de casos de uso del sistema.**

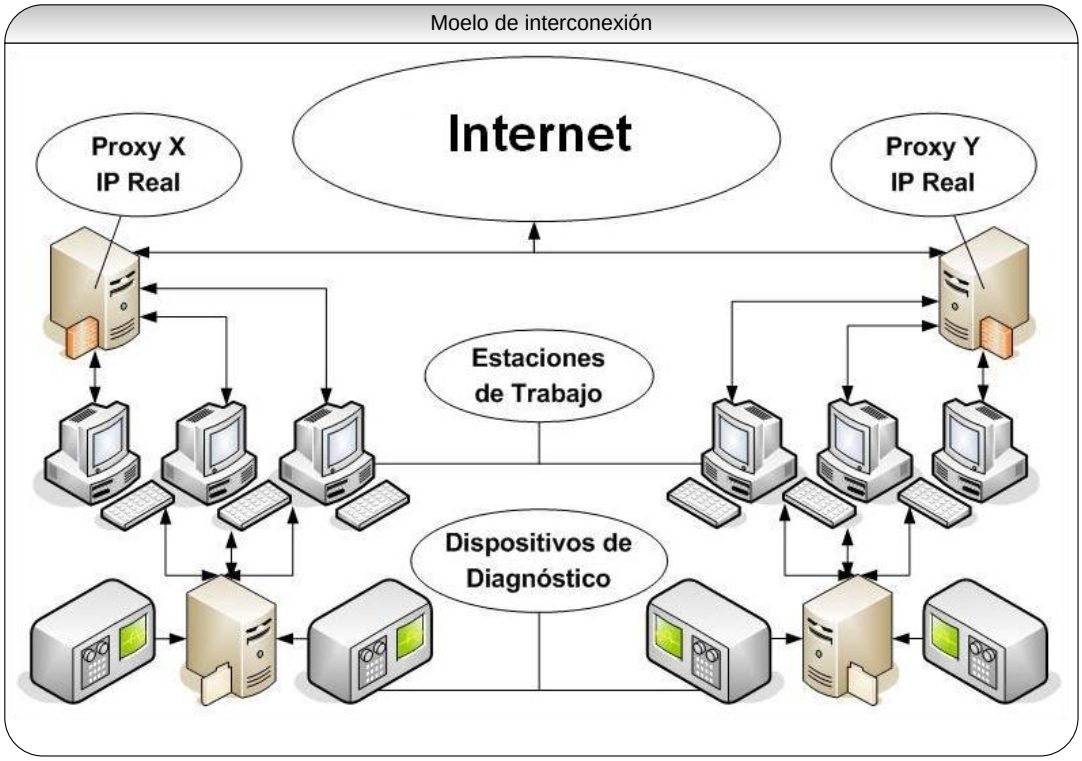
En este capítulo se abordaron aspectos cruciales del sistema. Se presentaron los requerimientos tanto funcionales como no funcionales, los cuales influyen de manera determinante en la selección de las funcionalidades del mismo. Se definieron los casos de uso a desarrollar y a su vez estos se clasificaron en tres iteraciones. A los casos de uso definidos para la primera iteración se les realizó la descripción detallada del flujo de eventos.

## CAPÍTULO 3: ARQUITECTURA Y DISEÑO DEL SISTEMA

A continuación se presenta una descripción de la propuesta de arquitectura para el sistema. Los diagramas de clases del diseño que describen la realización de los casos de uso serán insertados en este capítulo. En el diseño se encuentran tres tipos de clases las cuales se clasifican en: entidad, son las que modelan información que posee larga vida y que es a menudo persistente; interfaz, modelan la interacción entre el sistemas y sus actores; por último la clase controladora, que coordina la realización de los casos de usos.

### 3.1. Arquitectura del sistema.

En el Capítulo 1 se aborda sobre los protocolos de comunicación que establece DICOM; dentro de estos está definido que las conexiones entre aplicaciones DICOM deberán ser punto a punto. Observando el modelo de interconexión se observa un punto importante y es el problema de la visibilidad. En el Sistema Nacional de Salud Cubano, la conexión entre instituciones sanitarias, científicas y educacionales, responde en sentido general a un modelo estándar de interconexión. (Ver Figura 6.)




**(Figura 6.) Modelo general de interconexión.**

Se puede observar que en las estaciones de trabajo que están ubicadas dentro de una misma subred, sus direcciones IP privadas son internamente tratadas por el protocolo TCP/IP como direcciones IP públicas. De esta forma es posible establecer conexiones directas entre receptor y transmisor, permitiendo entonces la creación de asociaciones seguras que garantizan el cumplimiento de las políticas de seguridad que el estándar exige para la transmisión de imágenes médicas.

El problema aparece precisamente cuando se necesita globalizar la transmisión, o sea, cuando sea necesaria la transferencia de una imagen médica a otra subred. ¿Qué hacer cuando un especialista ubicado en una de las estaciones de trabajo ocultas tras el proxy X, necesite transmitir una imagen médica a otro especialista ubicado en una estación tras el proxy Y ?

La solución que propone el sistema, es la transmisión de imágenes médicas conforme al estándar DICOM. La alternativa se basa en el establecimiento de conexiones directas entre nodos consecutivos de una secuencia de equipos que comunicará al transmisor y receptor de cualquier intercambio conforme a DICOM. En este sentido la solución puede catalogarse como una variante extendida de la conexión punto a punto, la cual se ha denominado como transmisión punto a punto con intercambio.

Para comprender mejor lo anterior se presenta un ejemplo:

**Ejemplo de comunicación con intercambio.**

Si se quiere transmitir una imagen médica situada en la Estación de Trabajo # 1 (ET1X) ubicada en la subred X a la Estación de Trabajo # 2 (ET2Y) ubicada en la subred Y.

La sucesión de transmisiones de la imagen médica quedaría de la siguiente forma:

1. Transmisión DICOM de ET1X al Proxy X.
2. Transmisión DICOM del Proxy X al Proxy Y.
3. Transmisión DICOM del Proxy Y a ET2Y.

De esta forma la imagen médica alcanzará su destino, respetando los protocolos de transmisión que propone el estándar DICOM.

**Propuesta de arquitectura.**

Para poner en práctica el método antes planteado, se definió una arquitectura de tipo cliente servidor. Las estaciones clientes son las que funcionarían como destinatario y remitente de los mensajes, y las estaciones encargadas del intercambio en la transmisión, serian los servidores, cabe destacar que existirían tantos servidores como nodos de intercambio en la transmisión de un mensaje.

## 3.2. Diseño del sistema.

La realización de los casos de uso del sistema expresado en los diagramas de clases de diseño y los diagramas de secuencia que responden al flujo para cada caso de uso, están expuestos en el **Anexo2**.

En el **Anexo3** se pueden ver las descripciones de las clases de diseño que responden a los casos de uso para la primera iteración.

En este capítulo se logró dar cumplimiento al desarrollo de los artefactos de la arquitectura del software, la cual dentro del proceso de desarrollo constituye una parte importante debido a que propone los componentes que integran el sistema y la interacción entre los mismos. Se realizaron además los casos de uso en el diseño, estos brindan una vista estructural y dinámica del sistema. Los mismos se representa en clases de software que están ligadas al lenguaje de programación C#.

## CAPÍTULO 4: IMPLEMENTACIÓN Y RESULTADOS

En el presente capítulo se presenta la vista de implementación, la cual está formada por los diagramas de componentes y de despliegue, este último resulta útil en la comprensión de cómo quedará distribuido el sistema en los nodos físicos. Se presentan algunas pruebas para comprobar la efectividad de los mecanismos que implementa el mismo.

### 4.1. Diagrama de Componentes

En el diagrama de componentes se muestran las organizaciones y dependencias lógicas entre componentes de software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. En cuanto a los componentes, sólo aparecen tipos de componentes, ya que las instancias específicas de cada tipo se encuentran en el diagrama de despliegue.

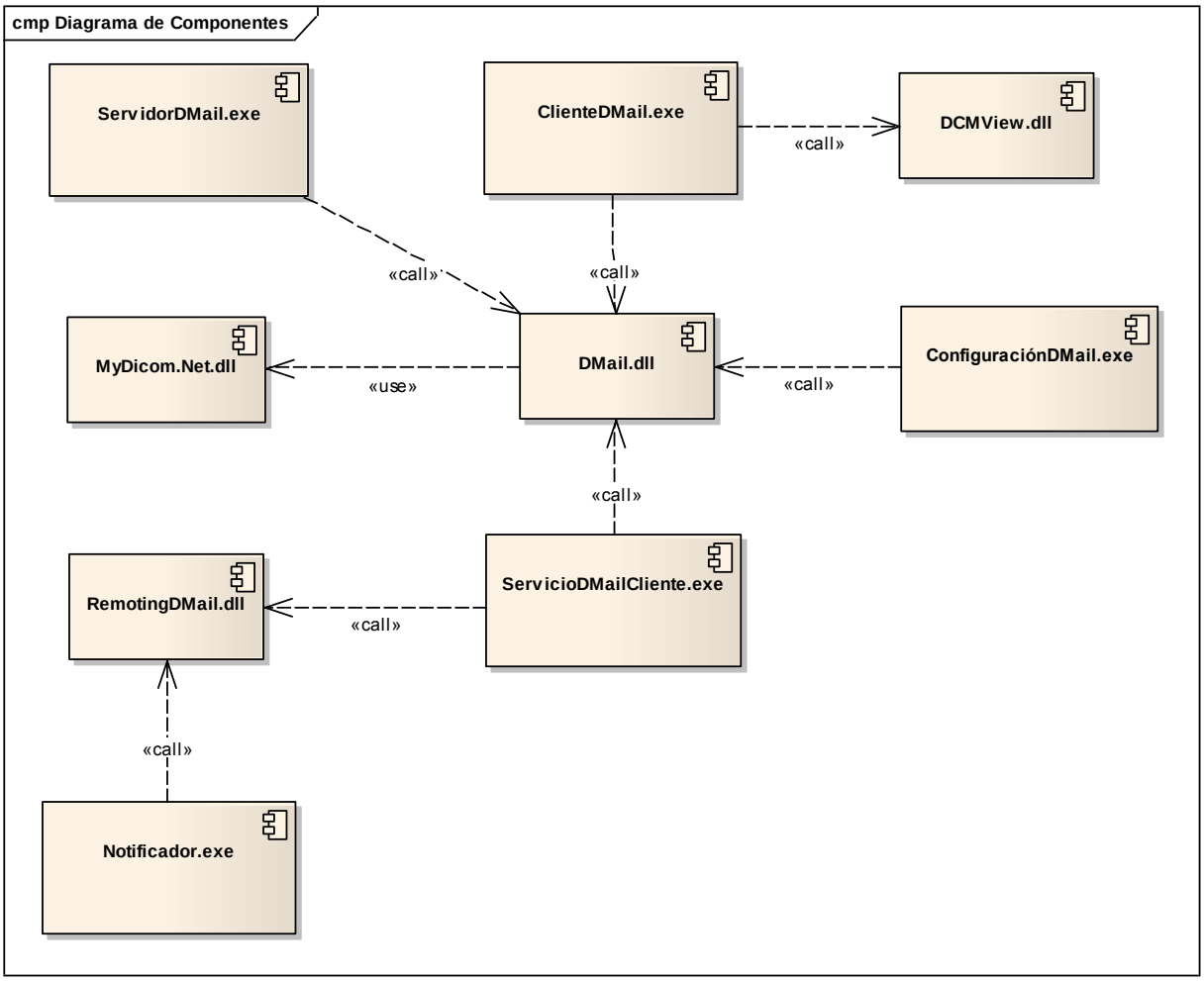



**Figura 7. Diagrama de componentes.**

## 4.2. Diagrama de despliegue.

El diagrama de despliegue muestra la disposición física de los nodos que componen el sistema. El mismo esta formado básicamente por dos módulos: Cliente y Servidor. Como se explicó en el Capítulo 3, es posible que se encuentre desplegado en varios nodos el módulo Cliente, igualmente es posible encontrar varios servidores según el alcance del sistema.

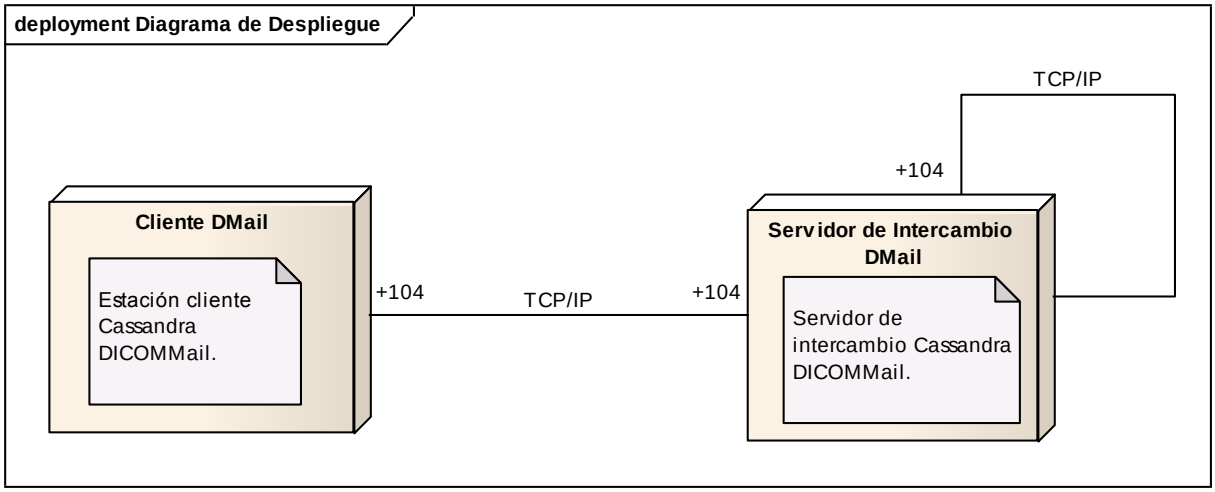



**Figura 8. Diagrama de despliegue.**

### 4.3. Resultados.

Debido a que el sistema en cuestión propone un método para hacer posible la interconsulta sobre imágenes médicas digitales entre instituciones, es necesario probar la eficiencia del mismo con el propósito de ver cuan factible es la solución propuesta. Con este objetivo, se trazó un plan de pruebas entre tres instituciones (La Universidad de las Ciencias Informáticas y el Hospital Cardiológico de la Habana William Soler) en Cuba y (el Hospital Cardiológico Infantil Latinoamericano Gilberto Rodríguez Ochoa) de la Ciudad de Caracas, Venezuela.

Las pruebas se realizaron en el mes de septiembre del 2006. Se transmitieron imágenes de varios tamaños, modalidades y equipos de adquisición. Una parte de los resultados obtenidos, se presentan a continuación de forma gráfica. (Ver Figura 9.)

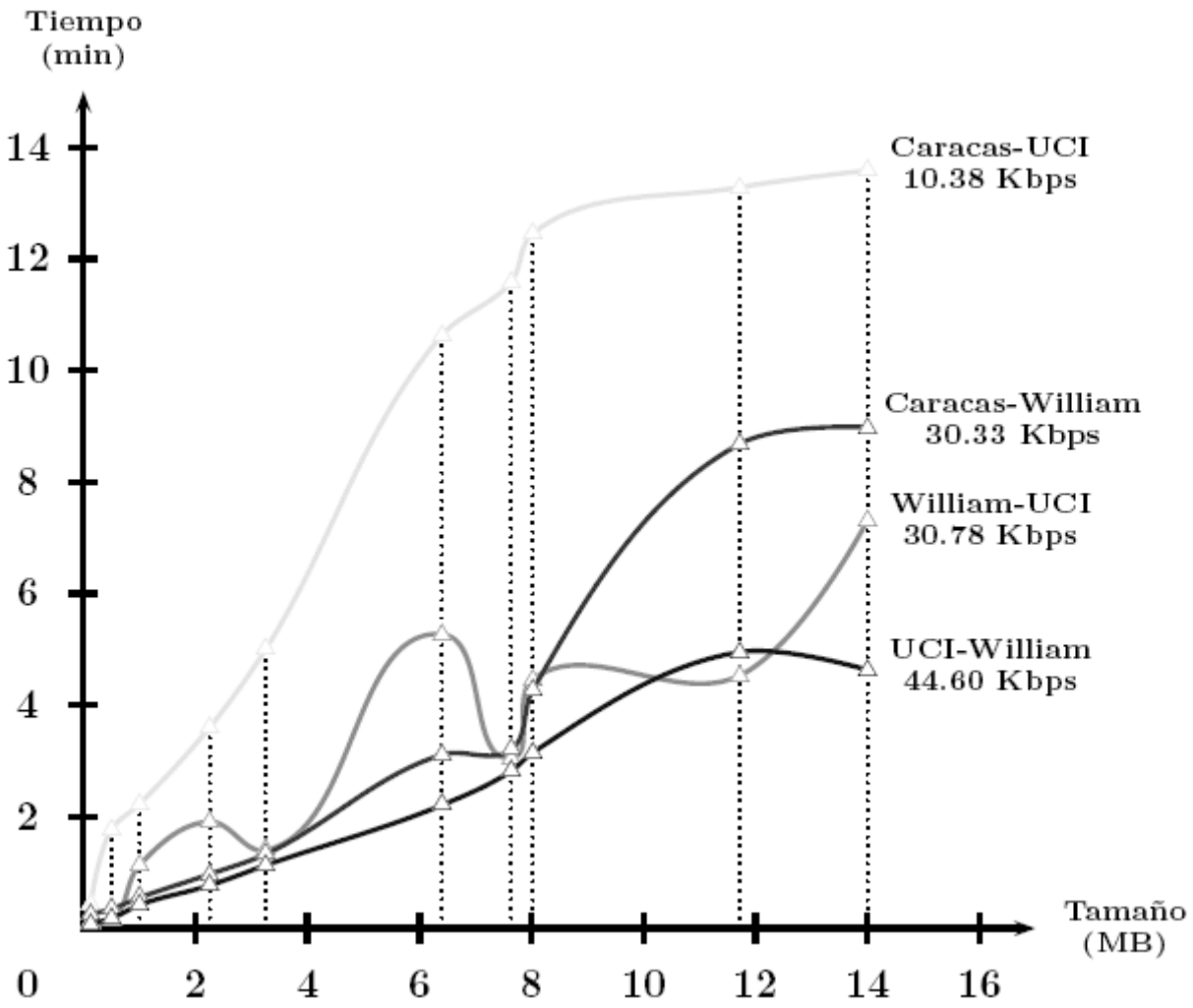



**Figura 9. Resultados de pruebas.**

En el gráfico se relacionan los parámetros tiempo y tamaño de los ficheros transmitidos. Junto a cada una de las curvas aparecen nombrados los extremos de la asociación (transmisor-receptor), así como la velocidad promedio de transferencia. Como era de esperar, el tiempo de transmisión aumenta en la medida en que crece el volumen de información a transmitir. Sin embargo, a simple vista puede apreciarse que esta tasa de crecimiento no es constante, llegando incluso a ser en extremo irregular en algunas de las transmisiones. La inestabilidad viene dada por un gran cúmulo de parámetros dentro de los cuales se pudieran mencionar el tráfico y congestionamiento de la red, ancho de banda, horario de transmisión, topología y configuración de la red, etc. Sin embargo, el parámetro más conflictivo fue la cantidad de routers (o nodos intermedios en general) que tomaban parte en las transmisiones y la ubicación física de

los mismos. A continuación se ilustra esta situación en la transmisión que más baja tasa de transferencia arrojó.

Los paquetes enviados desde la Ciudad de Caracas son dirigidos a Miami, luego a Washington, Relay, New York, London, Reino Unido y finalmente a Cuba. (Ver Figura 10.)

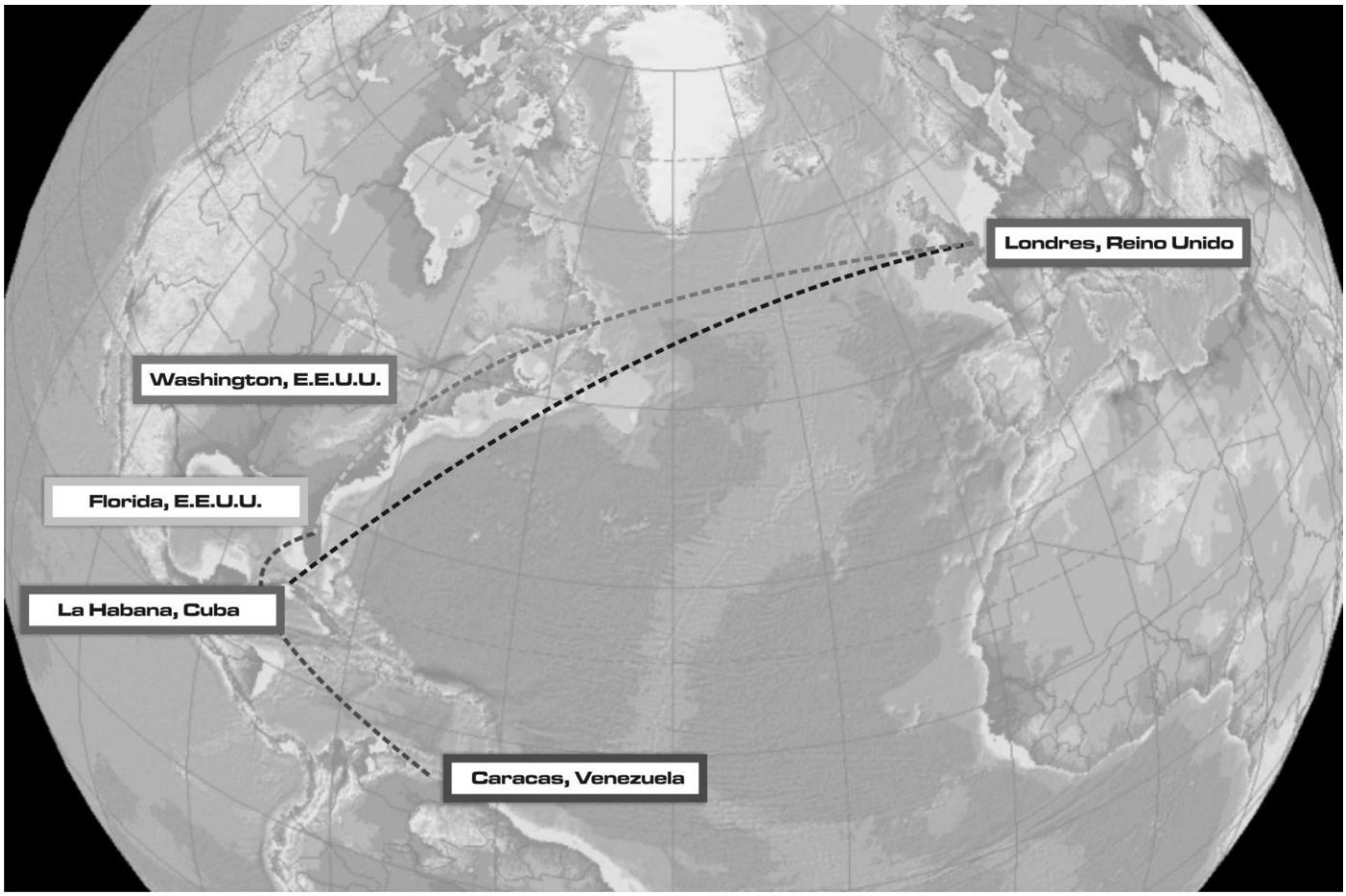



**Figura 10. Nodos intermedios para una transmisión Cuba-Venezuela.**

A pesar del largo viaje que siguieron las imágenes enviadas, y de las implicaciones que esto tiene en los tiempos de transmisión, las pruebas demostraron que el intercambio de imágenes médicas mediante la aplicación propuesta es factible. Con el uso de Alas PACS DICOM Mail, los médicos y las instituciones hospitalarias en general, pueden intercambiar imágenes, series y estudios médicos completos, logrando compatibilidad total con el estándar DICOM, asegurando así la seguridad, confiabilidad, integridad y fidelidad de la transmisión y por ende de los datos enviados.

Tener en cuenta las características de la aplicación para diseñar y configurar la red donde sesionará el sistema (minimizando por ejemplo la cantidad de nodos intermedios entre receptor y transmisor), sin duda implicaría mejoras en los tiempos de transmisión. Sin embargo, como demostró el pilotaje, no es vital para el funcionamiento del sistema. (Medina, 2007)

En este capitulo quedaron expuestos los diagramas de componente y desplieque en este último se puede apreciar como queda desplegado el sistema en nodos fisicos. Se elaboró un plan de pruebas para ver la eficacia de los mecanismos desarrollados, plasmando los resultados satisfactorios que arrojaron las pruebas. Dichas pruebas demostraron la efectividad de la solucion propuesta.

## CONCLUSIONES

En el presente trabajo de diploma se han cumplido los objetivos y tareas propuestos. Con el sistema Alas PACS DICOM Mail incluido en el producto Alas PACS se da solución a las necesidades actuales existentes en el campo de la transmisión de imágenes médicas en el país. Además proporciona una herramienta robusta y a su vez sencilla de usar por los médicos, especialistas y personal en general vinculado a la especialidad de Radiología.

Constituye el primer sistema de su tipo desarrollado en el país, basado en estándares internacionales, propiciando una solución eficiente en correspondencia con el avance tecnológico que le está imprimiendo la ciencia de la informática a la rama de la medicina.

El pilotaje al PACS comenzó en diciembre del 2006 en el CSI Dr. Salvador Allende de Chuao en la República Bolivariana de Venezuela. Actualmente se encuentra en proceso de instalación en hospitales de La Habana, para su posterior extensión a los diferentes centros hospitalarios distribuidos por la geografía nacional.

**RECOMENDACIONES**

A partir de los resultados de este trabajo y como posibles mejoras al sistema, se debe tener en cuenta algunos aspectos que debe seguir profundizándose como son:

- Continuar el despliegue el sistema propuesto en los centros donde existan áreas de radiología.
- Recopilar las opiniones de los usuarios del sistema con el fin de mejorar el software.
- Estudiar otras alternativas de trasmisión de imágenes médicas, buscando mejoras sustanciales a la propuesta en esta solución.
- Continuar el proceso de migración a Linux.
- Implementar un módulo estadístico para el sistema con el fin de conocer el tráfico de los mensajes.

**REFERENCIA BIBLIOGRÁFICA**

(ACR-NEMA, 2004) ACR-NEMA. Digital Imaging and Communications in Medicine. 2004.

(BOOCH, et al., 2000) BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El lenguaje Unificado de Modelado. 2000.

(Chang & Wagers, 2008) Chang & Wagers Asociados. C#-Online.NET. [Online] [Cited: 13 2, 2008.] http://es.csharp-online.net.

(Dpto. Ingeniería, 2006) Departamento de Ingenieria de Software, UCI. Introducción a la Ingeniería de Software. [Online] Diciembre Desde 04, 2006. http://teleformacion.uci.cu/mod/resource/view.php?id=10817&subdir=/Conferencias_IS1_05-06.

(Durañona, González, 2007) Durañona Yero, Yanoksy; González Rodríguez, Lázaro. Servidor de Imágenes Médicas (Cassandra Server).

(González, 2001) González Seco, José Antonio. El lenguaje de programación C#. 2001.

(Hernández, 2002) Hernández Orallo, Enrique. El Lenguaje Unificado de Modelado (UML). 2002.

(Martínez, 1998) Martínez M. A., Jiménez A.J.R, Medina B. V. Azpiroz L. J. Los Sistemas PACS.

(Medina, 2007) Medina Riezgo, Pedro. Cassandra DICOM Mail (Sistema para la Transmisión de Imágenes Médicas). Habana : s.n., 2007.

(Postel, 1982) Postel, J. Simple mail transfer protocol. 1982.

(Ronda, et al., 2001) Ronda, D; Ferrer, O. and Alvarez, A. IMAGIS: Sistema para la transmisión de imágenes médicas. Mayo 2001.

**BIBLIOGRAFÍA**


ACR-NEMA. Digital Imaging and Communications in Medicine. 2004.

A-G GROUP. Sitio de AGFA Corp, 2007. [Disponible en: http://www.agfa.com/en/co/index.jsp]

Bidgood, Dean. The Journal of the American Medical Informatics Association.1997. [ Disponible en: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=61235 ]

BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El lenguaje Unificado de Modelado. 2000.

Chang & Wagers Asociados. C#-Online.NET. [Online] [Cited: 13 2, 2008.] http://es.csharp-online.net.

Departamento de Ingenieria de Software, UCI. Introducción a la Ingeniería de Software. [Online] Diciembre Desde 04, 2006.
http://teleformacion.uci.cu/mod/resource/view.php?id=10817&subdir=/Conferencias_IS1_05-06.

Durañona Yero, Yanoksy; González Rodríguez, Lázaro. Servidor de Imágenes Médicas (Cassandra Server).

GE Health Care. AW Volume share. 2007. [ Disponible en: http://www.gehealthcare.com/euen/advantage-workstation/products/aw-volume-share/index.html ]

González Seco, José Antonio. El lenguaje de programación C#. 2001.

Hamilton Russell, Kill and Miles,. Learing UML 2.0. Sebastopol : O' Reilly, 2006.

Hernández Orallo, Enrique. El Lenguaje Unificado de Modelado (UML). 2002.

Huang, H.K. PACS, Basic Principles and Applications. New York : Wiley-Liss, 1999.

Martínez M. A., Jiménez A.J.R, Medina B. V. Azpiroz L. J. Los Sistemas PACS.

Medina Riezgo, Pedro. Cassandra DICOM Mail (Sistema para la Transmisión de Imágenes Médicas). Habana : s.n., 2007.

Microsoft. Visual Studio 2005 Team Suite. MSDN. 2007. [Disponible en: http://msdn2.microsoft.com/en-us/teamsystem/aa718822.aspx.]

MyDicom.SDK FAQ. 2006. [Disponible en: http://www.mydicom.net/SDKFAQ.aspx]

Pérez, J.L. and Tejeiro, J. Sistemas de Comunicación y Gestión de Imágenes Médicas .2000. Disponible en: [http://www.imedir.udc.es/pages/memoria.html/2007.Perez.et.al.Sistemas.de.comunicacion.y.Gestion..pdf ]

Postel, J. Simple mail transfer protocol. 1982.

Ronda, D; Ferrer, O. and Alvarez, A. IMAGIS: Sistema para la transmisión de imágenes médicas. Mayo 2001.

Schmuller, Joseph. Aprendiendo UML en 24 Horas. México : Pearson Education, 2000.

Systems Sparx. Enterprise Architect. 2008.
Disponible en: [http://www.sparxsystems.com.ar/products/ea.html. ]

Wikipedia. Microsoft Visual Studio. 2007.

Disponible en:[ http://en.wikipedia.org/wiki/Microsoft_Visual_Studio.]

**ANEXOS**

## 5.1. Anexo1. Descripción de los casos de uso del sistema.

### 5.1.1. Gestionar mensaje.

| | |
|---|---|
| **CASO DE USO:** | Gestionar mensaje. |
| **Propósito:** | Crear, ver o eliminar un mensaje. |
| **Complejidad:** | Alta |
| **Precondiciones:** | Para ver, o eliminar un mensaje, este debe haber sido seleccionado. |
| **REFERENCIAS** | |
| **Actores:** | Usuario. |
| **Requisitos:** | |
| **Entidades:** | Mensaje. |
| **Casos de Uso:** | Enviar Mensaje, Adjuntar Ficheros. |
| **FLUJO NORMAL DE EVENTOS** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso se inicia cuando el actor accede a la opción de Gestionar Mensaje. | |
| | 2. Permite:<br>• Crear Mensaje. Ver **Sección 1:** "Crear Mensaje." |

| | |
|---|---|
| | • Ver datos de Mensaje. Ver **Sección 2:** "Ver datos de Mensaje."<br>• Eliminar Mensaje. Ver **Sección 3**: "Eliminar Mensaje" |
| | 3. El caso de uso termina. |
| **SECCIONES** | |
| **Sección 1:** "Crear Mensaje." | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso inicia cuando el actor accede a la opción Crear Mensaje. | |
| | 2. Brinda la posibilidad de introducir los datos del mensaje.<br>Y permite:<br>• Enviar el mensaje. Ver caso de uso **Enviar Mensaje**.<br>• Cancelar operación. Ver **Alternativa 1:** "Cancelar operación." |
| 3. Introduce los datos del mensaje:<br>• Destinatario.<br>• Asunto.<br>• Cuerpo del mensaje. | |

| | |
|---|---|
| • Ficheros adjuntos, ver caso **Adjuntar Ficheros.** | |
| | 4. Valida los datos. Si hay datos incompletos, ver **Alternativa 2:** “Existen datos incompletos.”. Si hay datos incorrectos, ver **Alternativa 3:** “Existen datos incorrectos.” |
| | 5. Crea el mensaje. |
| 6. Accede a la opción Enviar mensaje. | 7. Envía mensaje, ver caso de uso Enviar Mensaje. |
| | 8. El caso de uso termina. |
| 9. El caso de uso inicia cuando el actor accede a la opción Crear Mensaje. | |
| **Sección 2:** “Ver datos del Mensaje.” | |
| **Acción del Actor** | **Acción del Actor** |
| 1. Selecciona la opción de Ver datos del Mensaje | |
| | 2. Muestra los datos del Mensaje<br>Y permite:<br>• Salir de la vista actual |
| 3. Selecciona la opción de salir de la vista actual. | |

| | |
|---|---|
| | 4. Muestra la vista anterior. |
| | 5. El caso de uso termina. |
| **Sección 3:** "Eliminar Mensaje." | |
| **Acción del Actor** | **Acción del Actor** |
| 1. Selecciona la opción de Eliminar Mensaje. | |
| | 2. Muestra el mensaje de advertencia "Se eliminará el Mensaje seleccionado". Al seleccionar Aceptar se perderán todos los datos. ¿Desea continuar?"<br>y permite:<br>• Aceptar la eliminación del Mensaje<br>• Cancelar la operación. Ver **Alternativa 1:** "Cancelar operación." |
| 3. Selecciona la opción de aceptar la eliminación del Mensaje. | |
| | 4. Elimina el Mensaje. |
| | 5. Muestra el mensaje de información "El Mensaje ha sido eliminado." |
| | 6. El caso de uso termina. |
| **FLUJOS ALTERNOS** | |

| **Alternativa 1.** "Cancelar operación." | |
|---|---|
| **Acción del Actor** | **Acción del Actor** |
| 1. Selecciona la opción de Cancelar operación. | |
| | 2. Regresa a la vista anterior. |
| | 3. El caso de uso termina. |
| **Alternativa 2.** "Existen datos incompletos." | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | 1. Muestra el mensaje de error "Existen campos vacíos que son obligatorios, por favor, complete estos datos." |
| | 2. Muestra un indicador sobre los campos vacíos. |
| **Alternativa 3.** "Existen datos incorrectos." | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | 2. Muestra el mensaje de error "Existen campos escritos incorrectamente, por favor, rectifique estos datos." |
| | 3. Muestra un indicador sobre los campos incorrectos. |
| **Poscondiciones** | Se creó, vio o eliminó un Mensaje. |

### 5.1.2. Adjuntar ficheros.

| | |
|---|---|
| **CASO DE USO:** | Adjuntar ficheros. |
| **Propósito:** | Adjuntar ficheros a un mensaje. |
| **Complejidad:** | Media. |
| **Precondiciones:** | Se debe estar creando un mensaje. |
| **REFERENCIAS** | |
| **Actores:** | Usuario. |
| **Requisitos:** | |
| **Entidades:** | Ficheros DICOM. |
| **Casos de Uso:** | |
| **FLUJO NORMAL DE EVENTOS** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso se inicia cuando el actor accede a la opción Adjuntar ficheros. | |
| | 2. Brinda la posibilidad de buscar ficheros DICOM y de seleccionar si desea adjuntar:<br>• Estudio Médico<br>• Serie Médica<br>• Imagen Médica. |

| | |
|---|---|
| | Y permite:<br>• Aceptar adjuntos.<br>• Cancelar la operación. Ver **Alternativa 1:** "Cancelar operación." |
| 3. Acepta adjuntos. | 4. Adjunta ficheros. |
| | 5. El caso de uso termina. |
| **FLUJOS ALTERNOS** | |
| **Alternativa 1.** "Cancelar operación." | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción de Cancelar operación. | |
| | 2. Regresa a la vista anterior. |
| | 3. El caso de uso termina. |
| **Poscondiciones** | Se adjuntaron ficheros al mensaje. |

### 5.1.3. Enviar mensaje.

| | |
|---|---|
| **CASO DE USO:** | Enviar mensaje. |
| **Propósito:** | Enviar un mensaje. |
| **Complejidad:** | Alta. |

| | |
|---|---|
| **Precondiciones:** | Para enviar un mensaje, este debe haber sido creado. |
| **REFERENCIAS** | |
| **Actores:** | Usuario. |
| **Requisitos:** | |
| **Entidades:** | Mensaje. |
| **Casos de Uso:** | |

| **FLUJO NORMAL DE EVENTOS** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso se inicia cuando el actor accede a la opción Enviar mensaje. | |
| | 2. Envía el mensaje. |
| | 3. Si la operación terminó con éxito, muestra el mensaje de “Operación terminada correctamente”. Si no se pudo completar la operación, ver **Alternativa 1:** “Envío incompleto.” |
| | 4. El caso de uso termina. |
| **FLUJOS ALTERNOS** | |
| **Alternativa 1.** “Envío incompleto.” | |
| **Acción del Actor** | **Respuesta del Sistema** |

| | |
|---|---|
| | 3. El mensaje es movido a una bandeja de salida para ser enviado posteriormente. |
| | 4. El caso de uso termina. |
| **Poscondiciones** | Se envió el mensaje. |

### 5.1.4. Recibir mensaje.

| | |
|---|---|
| **CASO DE USO:** | Recibir mensaje. |
| **Propósito:** | Recibir un mensaje. |
| **Complejidad:** | Alta. |
| **Precondiciones:** | |
| **REFERENCIAS** | |
| **Actores:** | Usuario. |
| **Requisitos:** | |
| **Entidades:** | Mensaje. |
| **Casos de Uso:** | |
| **FLUJO NORMAL DE EVENTOS** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso se inicia cuando un usuario envía un mensaje. | |

| | |
|---|---|
| | 2. Almacena el mensaje así como los ficheros adjuntos si tiene, creando para ello un directorio por cada mensaje. |
| | 3. Notifica la llegada de un nuevo mensaje, ver caso de uso "**Notificar nuevo mensaje**". |
| | 4. El caso de uso termina. |
| **FLUJOS ALTERNOS** | |
| **Poscondiciones** | Se recibió el mensaje y quedo notificado el usuario. |

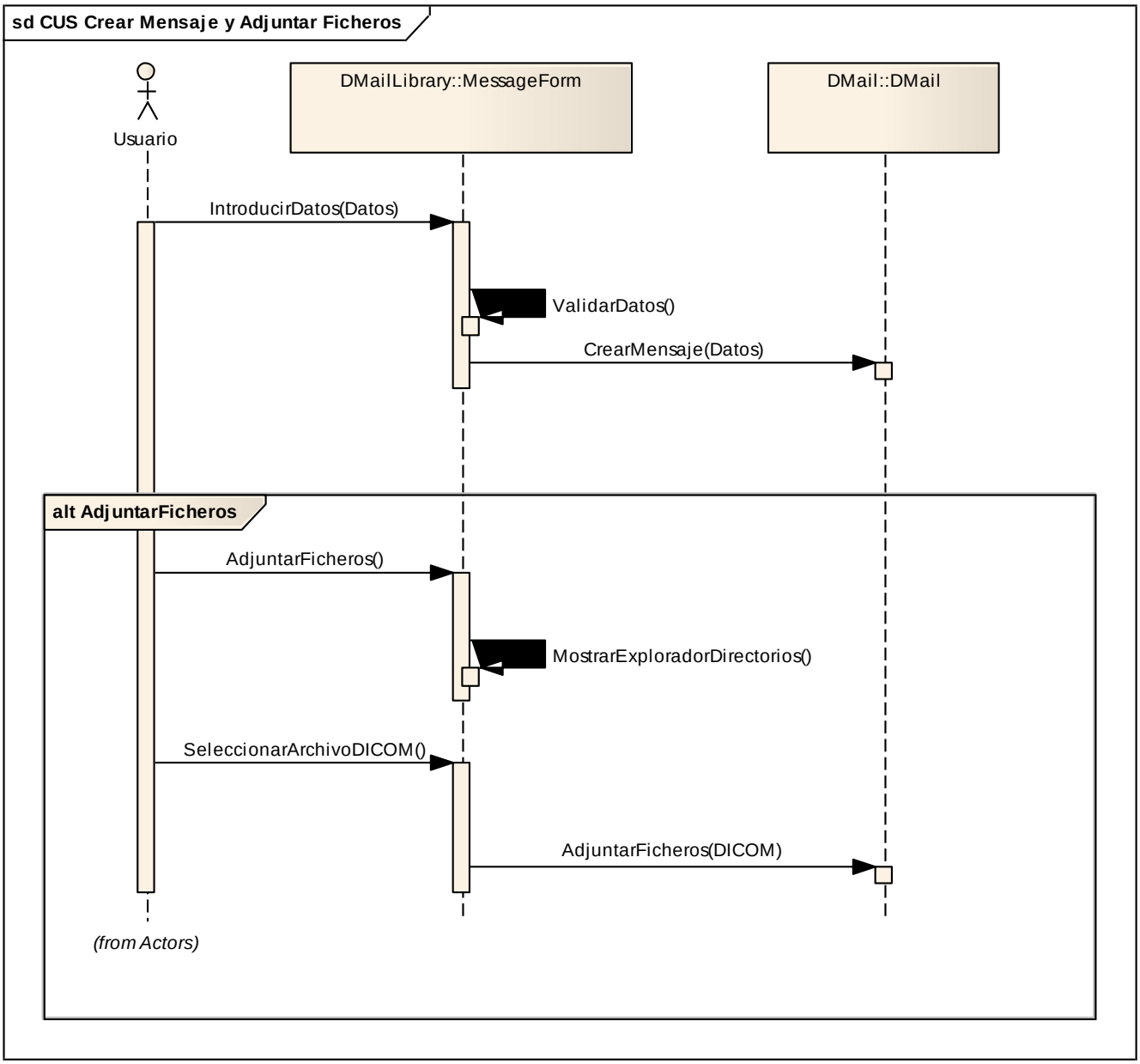
### 5.1.5. Notificar mensaje.

| | |
|---|---|
| **CASO DE USO:** | Notificar mensaje. |
| **Propósito:** | Notificar la llegada de un nuevo mensaje. |
| **Complejidad:** | Alta. |
| **Precondiciones:** | |
| **REFERENCIAS** | |
| **Actores:** | CU – Recibir mensaje. |
| **Requisitos:** | |
| **Entidades:** | Mensaje. |
| **Casos de Uso:** | |

| FLUJO NORMAL DE EVENTOS | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso se inicia cuando se recibe un nuevo mensaje. | |
| | 2. Muestra una ventana con los siguientes datos del mensaje recibido:<br>• Remitente.<br>• Destinatario.<br>• Asunto. |
| | 3. El caso de uso termina. |
| **FLUJOS ALTERNOS** | |
| **Poscondiciones** | Se recibió el mensaje y quedo notificado el usuario. |

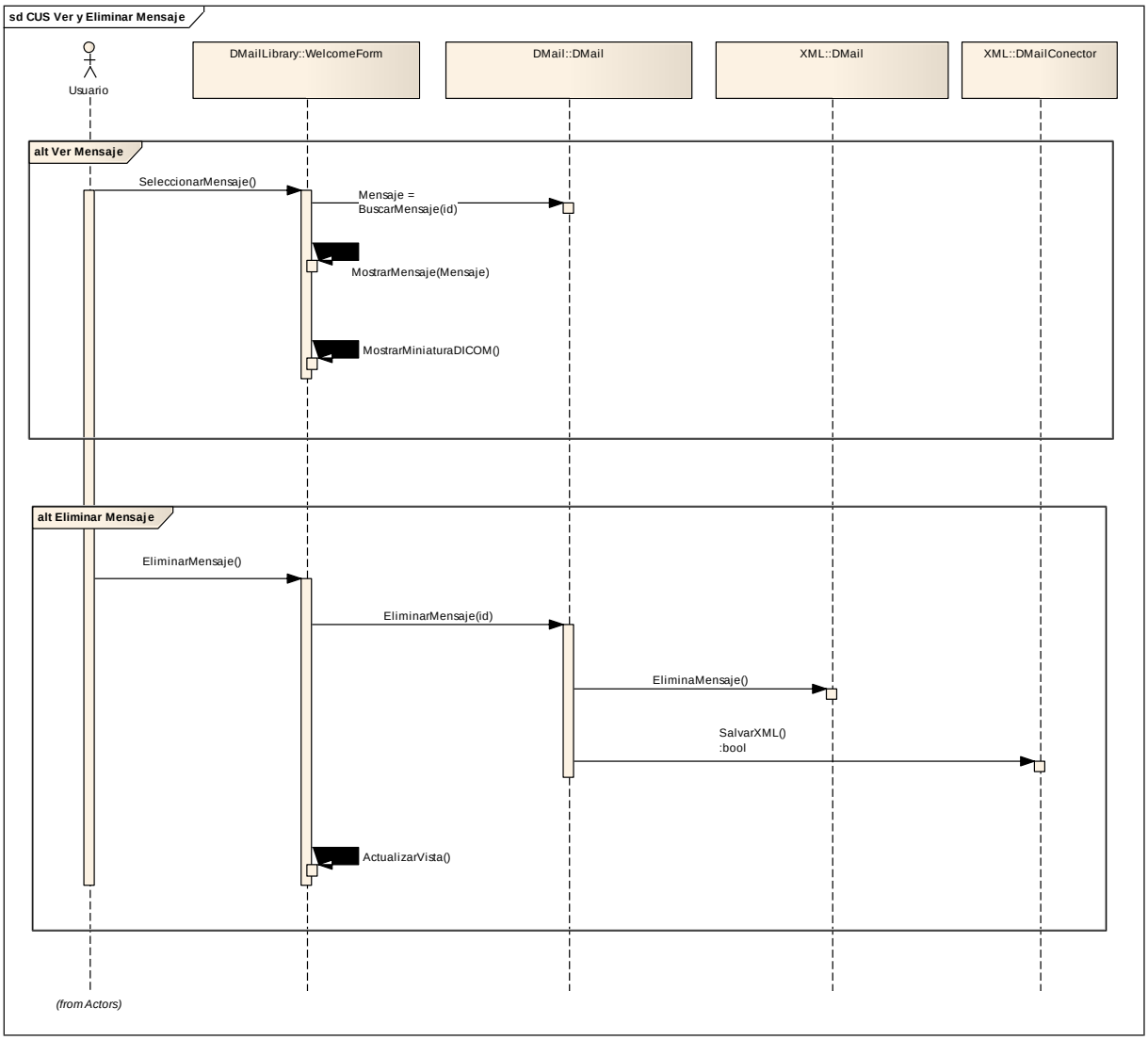
## 5.2. Anexo2. Realización de casos de uso del sistema.

### 5.2.1. Diagramas de secuencia CUS Crear Mensaje y Adjuntar Ficheros.



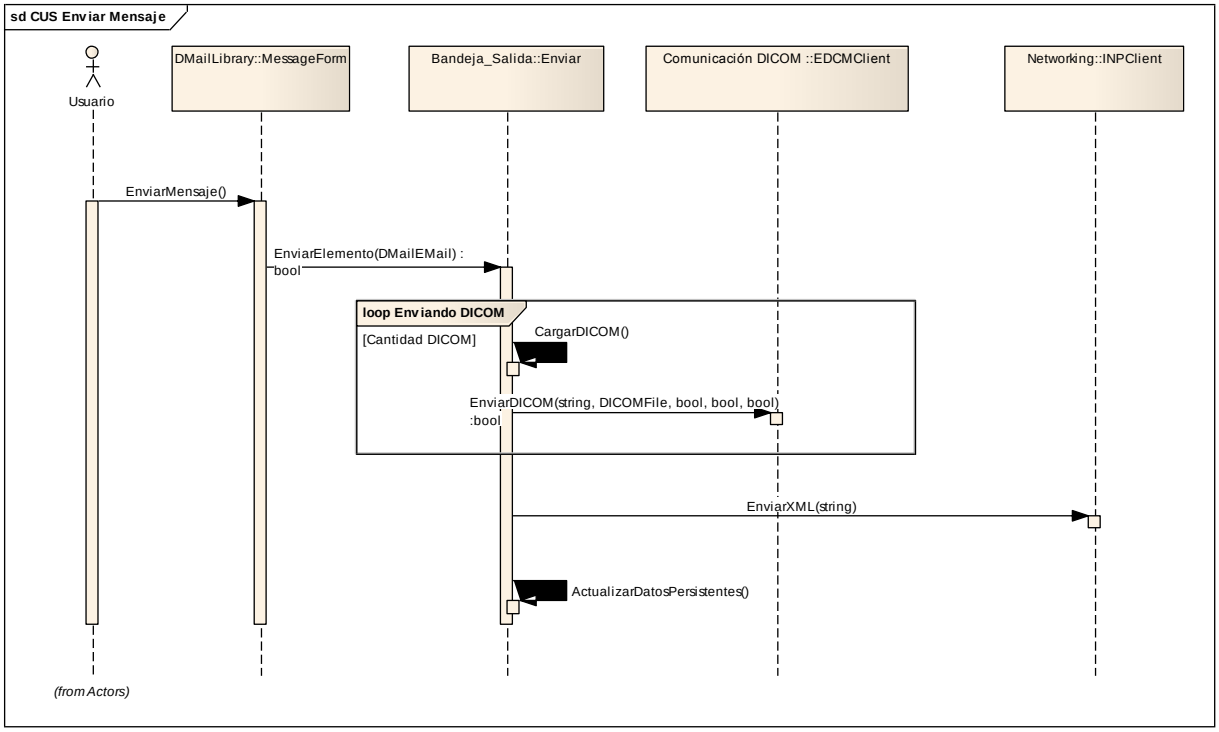
## 5.2.2. Diagrama de clases CUS Crear Mensaje y Adjuntar Ficheros.

### 5.2.3. Diagramas de secuencia CUS Ver y Eliminar Mensaje.

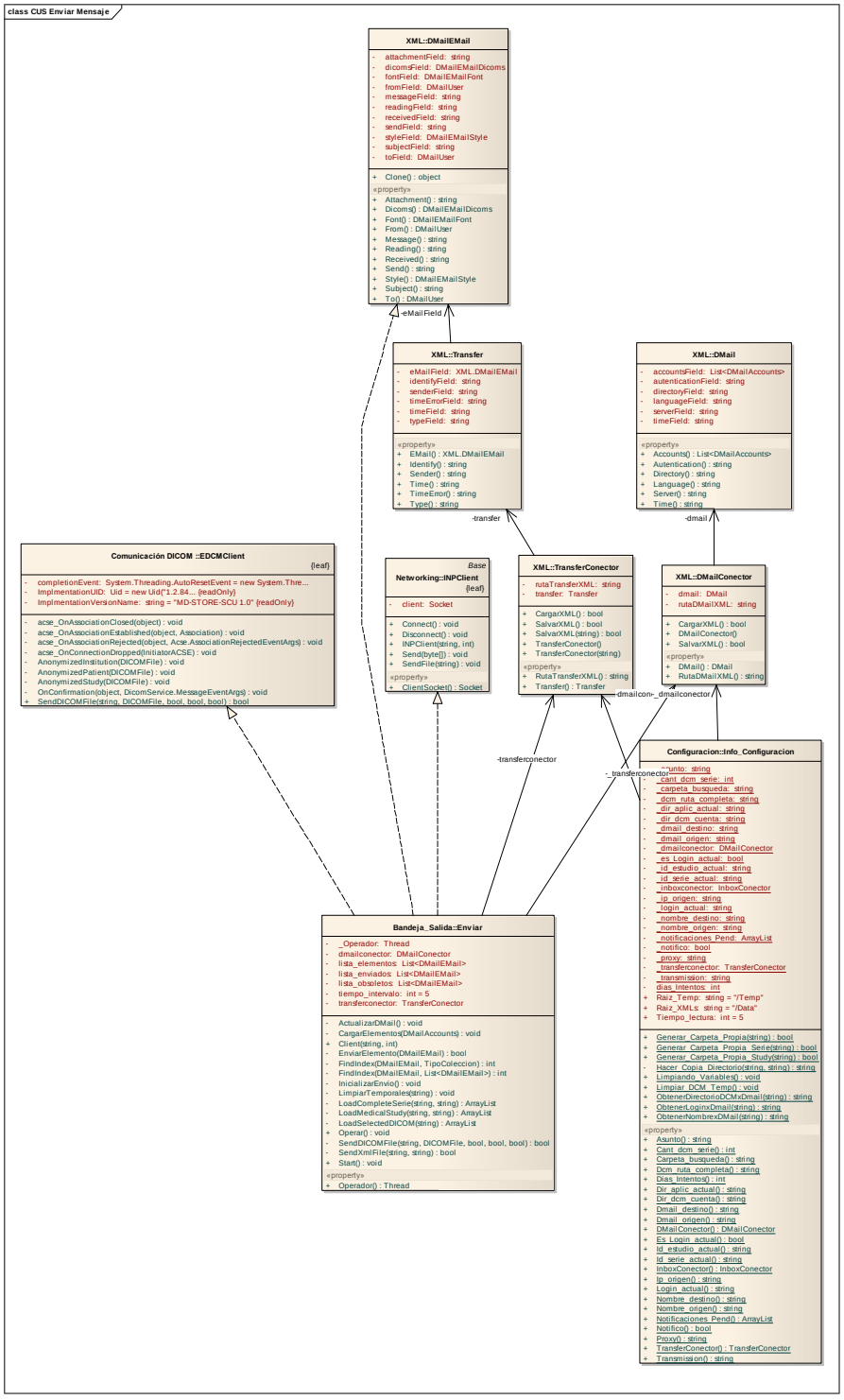


## 5.2.4. Diagrama de clases CUS Ver y Eliminar Mensaje.

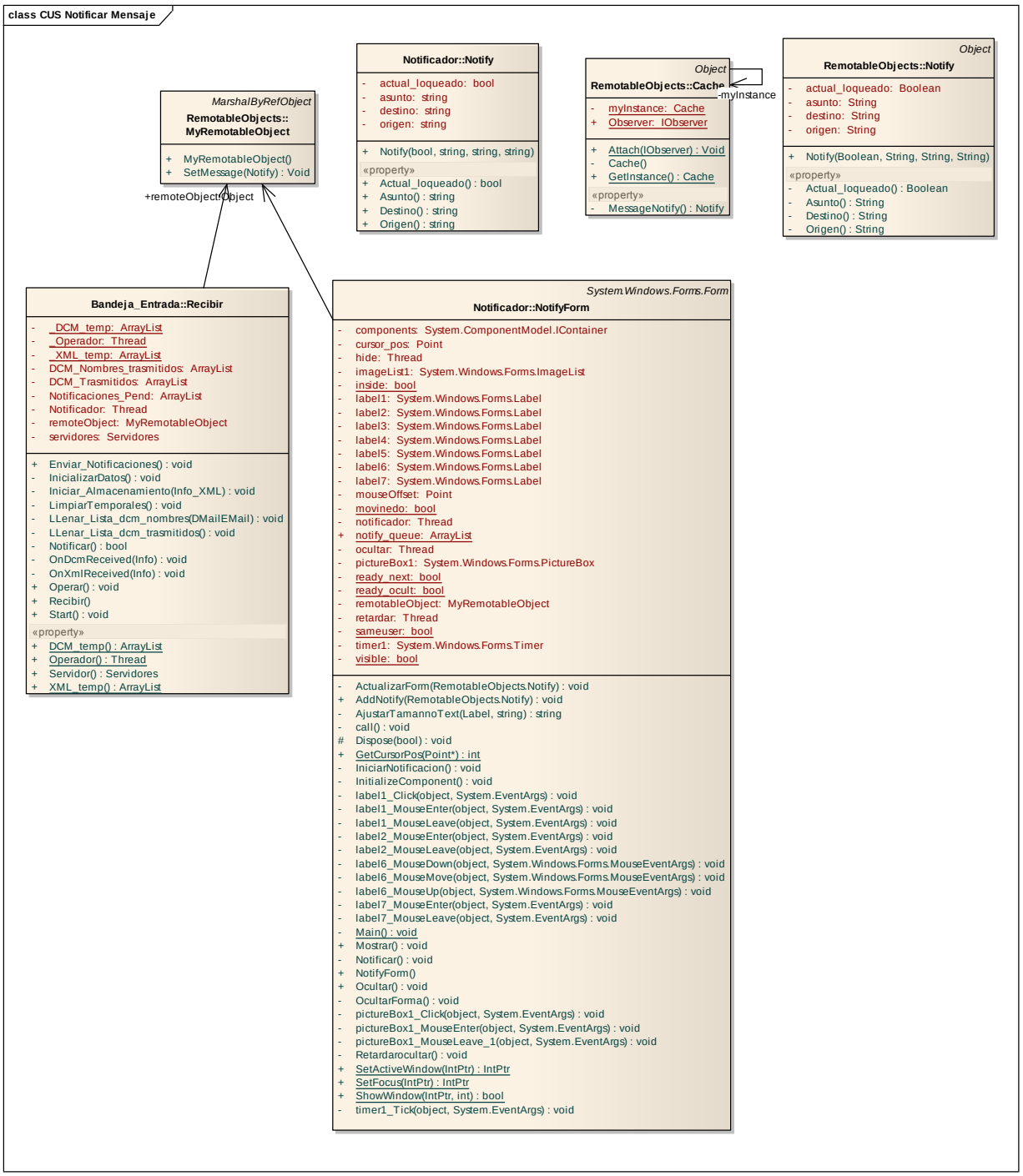
### 5.2.5. Diagramas de secuencia CUS Enviar Mensaje.



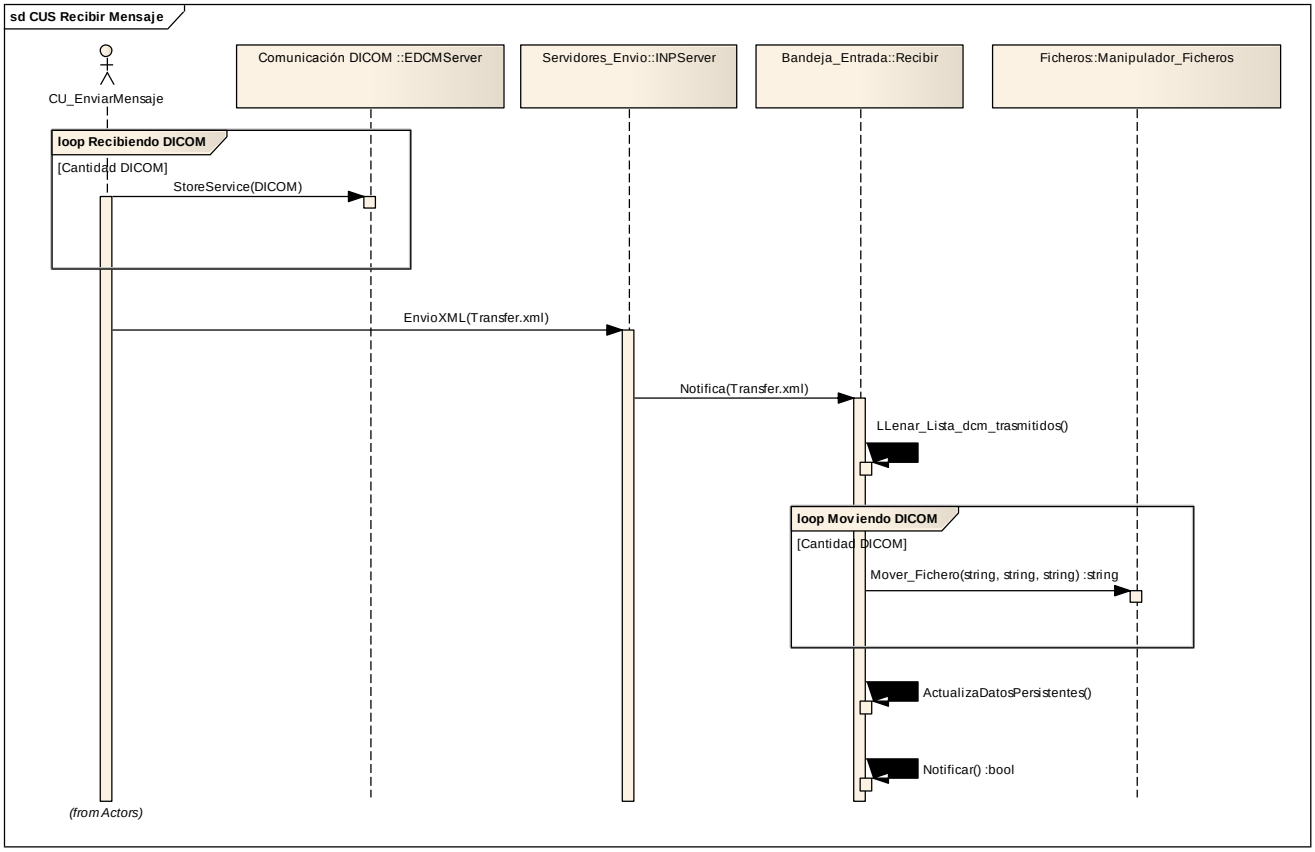
## 5.2.6. Diagrama de clases CUS Enviar Mensaje.



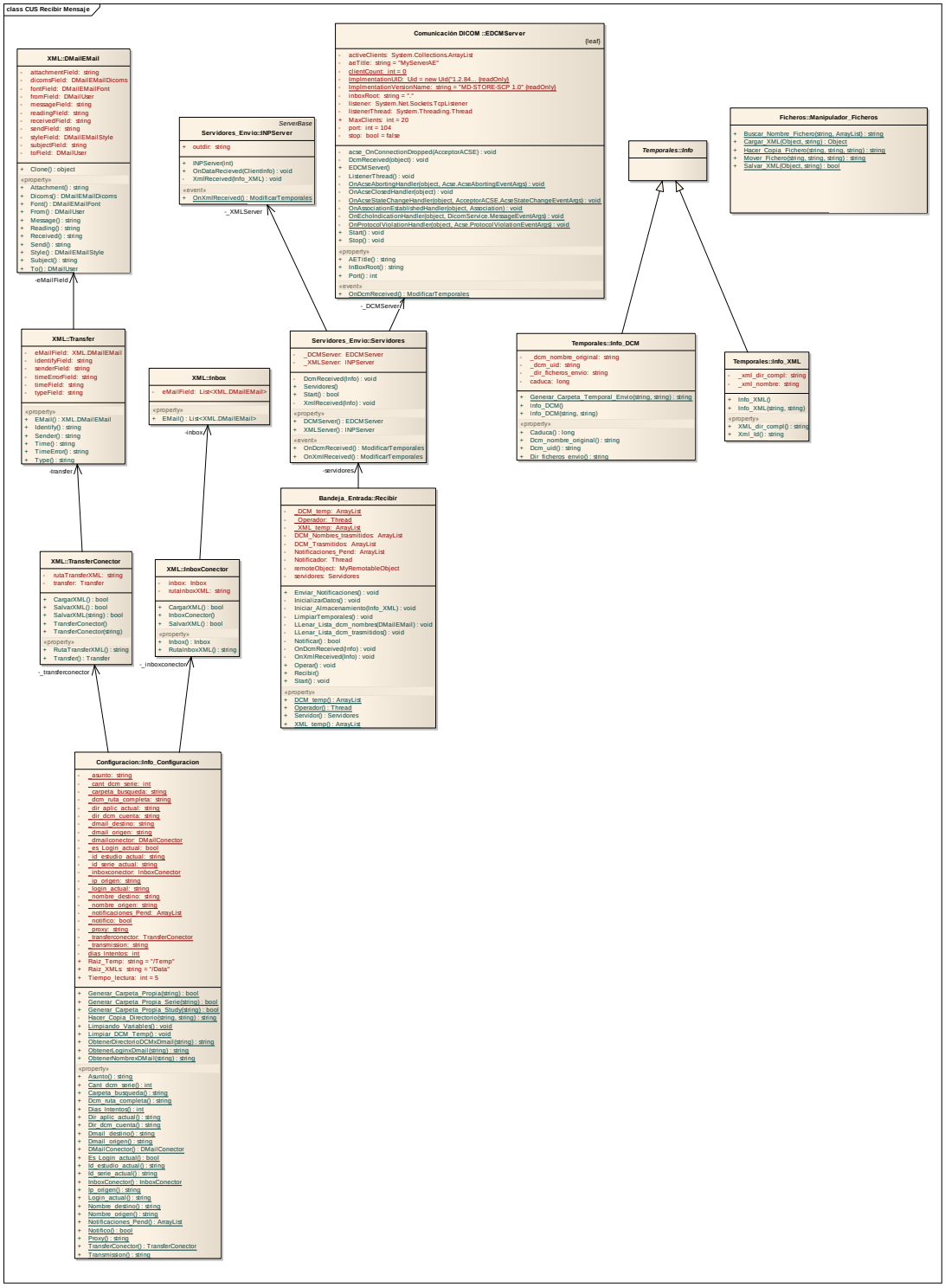
## 5.2.7. Diagrama de clases CUS Notificar Mensaje.



### 5.2.8. Diagramas de secuencia CUS Recibir Mensaje.



## 5.2.9. Diagrama de clases CUS Recibir Mensaje



## 5.3. Anexo3. Descripción detallada de las clases de diseño.

### 5.3.1. Descripción de la clase DMail.

| **public class DMail** | |
|---|---|
| **Atributo** | **Tipo** |
| accountsField | List< Accounts > |
| autenticationField | string |
| directoryField | string |
| languageField | string |
| timeField | int |
| serverField | string |
| **Miembro** | **Descripción** |
| Accounts | Lista de cuentas de usuarios. |
| Autentication | Id del usuario autenticado. |
| Directory | Directorio predefinido para los ficheros DICOM. |
| Language | Idioma definido para el sistema. |
| Server | Ip de la PC donde está instalado DMailServer. |
| Time | Tiempo en días definido para tratar de enviar un mensaje. |

### 5.3.2. Descripción de la clase DMailConector.

| **public class DMailConector** | |
|---|---|
| **Atributo** | **Tipo** |
| dmail | DMail |
| rutaDMailXML | string |
| **Miembro** | **Descripción** |
| DMail | Instancia del objeto que representa DMail.xml |
| DMailConector () | Inicializa una nueva instancia de DMailConector. |
| CargarXML() | Carga los datos de DMail.xml |
| RutaDMailXML | Ruta completa del fichero DMail.xml |
| SalvarXML() | Salva los datos para DMail.xml |

### 5.3.3. Descripción de la clase Manipulador_Ficheros.

| **internal class Manipulador_Ficheros** | |
|---|---|
| **Miembro** | **Descripción** |
| Manipulador_Ficheros() | Inicializa una nueva instancia de la clase. |
| Buscar_Nombre_Fichero(String, ArrayList) | Busca un nombre de fichero específico o parte de este en una lista de nombres. |

| | |
|---|---|
| Cargar_XML(Object, String) | Cargar objeto de un fichero xml. |
| Mover_Fichero(String, String, String) | Mueve el fichero dado para un directorio especifico si existe le hace una copia. |
| Hacer_Copia_Fichero(String, String, String) | Hace una copia de un fichero en un directorio especificado. |
| Salvar_XML(Object, String) | Salva un objeto para un fichero de tipo xml. |

### 5.3.4. Descripción de la clase DMailEMail.

| **public class DMailEMail** | |
|---|---|
| **Atributo** | **Tipo** |
| attachmentField | string |
| dicomsField | DMailEMailDicoms |
| fontField | DMailEMailFont |
| fromField | DMailUser |
| messageField | string |
| readingFieId | string |
| receivedField | string |
| sendField | string |
| styleField | DMailEMailStyle |
| subjectField | string |

| | |
|---|---|
| toField | DMailUser |
| **Miembro** | **Descripción** |
| DMailEMail () | Inicializa una nueva instancia de la clase. |
| Attachment | Dirección del fichero DICOM adjunto. |
| Clone() | Clona este objeto. |
| Dicoms | Objeto que almacena la información de los ficheros adjuntos. |
| Font | Fuente para el cuerpo del mensaje. |
| From | Usuario DMail remitente. |
| Message | Cuerpo del mensaje. |
| Reading | Fecha en la que se lee el mensaje. |
| Received | Fecha en la que se recibe el mensaje. |
| Send | Fecha en la que se envía el mensaje. |
| Style | Estilo del cuerpo del mensaje. |
| Subject | Asunto del mensaje. |
| To | Usuario DMail destinatario. |

### 5.3.5. Descripción de la clase DMailUser.

| **public class DMailUser** | |
|---|---|
| **Atributo** | **Tipo** |
| aliasField | string |
| cityField | string |
| companyField | string |
| countryField | string |
| departmentField | string |
| dMailField | string |
| nameField | string |
| phoneField | string |
| titleField | string |
| **Miembro** | **Descripción** |
| DMailUser() | Inicializa una nueva instancia de la clase. |
| Alias | Alias del usuario. |
| City | Ciudad donde radica. |
| Company | Compañía para la que trabaja. |
| Country | País de residencia. |
| Department | Departamento en el que trabaja dentro de la compañía. |
| DMail | Dirección DMail, similar a la dirección tradicional de correo electrónico. |

| | |
|---|---|
| Phone | Teléfono. |
| Title | Título. |

### 5.3.6. Descripción de la clase DMailEMailDicoms.

| **public class DMailEMailDicoms** | |
|---|---|
| **Atributo** | **Tipo** |
| compressionField | string |
| dicomField | List<DMailEMailDicomsDicom> |
| fullSizeField | string |
| institutionField | string |
| patientField | string |
| searchScopeField | string |
| selectedFolderField | string |
| studyField | string |
| transmissionField | string |
| **Miembro** | **Descripción** |
| DMailEMailDicoms() | Inicializa una nueva instancia de la clase. |
| Compression | Especifica el modo de compresión de los ficheros DICOM. |
| Dicom | Lista de los ficheros DICOM adjuntos al mensaje. |

| | |
|---|---|
| FullSize | Tamaño completo de todos los ficheros adjuntos. |
| Institution | Especifica si serán anonimizados los datos de la institución donde se generó el DICOM. |
| Patient | Especifica si serán anonimizados los datos del paciente dentro del DICOM. |
| SearchScope | Especifica el modo de búsqueda de los ficheros en el directorio DICOM. |
| SelectedFolder | Directorio seleccionado para la búsqueda. |
| Study | Especifica si serán anonimizados los datos del estudio médico del DICOM. |
| Transmission | Especifica el modo de transmisión (Estudio Médico, Serie Médica o Fichero DICOM). |

### 5.3.7. Descripción de la clase DMailEMailDicomsDicom.

| **public class DMailEMailDicomsDicom** | |
|---|---|
| **Atributo** | **Tipo** |
| inputNameField | string |
| modalityField | string |
| outputNameField | string |
| sizeField | string |
| typeField | string |
| **Miembro** | **Descripción** |
| DMailEMailDicomsDicom() | Inicializa una nueva instancia de la clase. |

| | |
|---|---|
| InputName | ID del fichero DICOM. |
| Modality | Modalidad del fichero DICOM. |
| OutputName | Nombre físico del fichero DICOM. |
| Size | Tamaño del fichero DICOM. |
| Type | Tipo del fichero DICOM. |

### 5.3.8. Descripción de la clase DMailEMailFont.

| **public class DMailEMailFont** | |
|---|---|
| **Atributo** | **Tipo** |
| boldField | string |
| italicField | string |
| nameField | string |
| sizeField | string |
| strikeoutField | string |
| underlineField | string |
| **Miembro** | **Descripción** |
| DMailEMailFont() | Inicializa una nueva instancia de la clase. |
| Bold | Negrita. |
| Italic | Cursiva. |

| | |
|---|---|
| Name | Nombre de la fuente. |
| Size | Tamaño de la fuente. |
| Strikeout | Marcado del texto. |
| Underline | Subrayado. |

### 5.3.9. Descripción de la clase DMailEMailStyle.

| **public class DMailEMailStyle** | |
|---|---|
| **Atributo** | **Tipo** |
| alignmentField | string |
| priorityField | string |
| rgbField | string |
| **Miembro** | **Descripción** |
| DMailEMailStyle() | Inicializa una nueva instancia de la clase. |
| Alignment | Alineación del texto en el mensaje. |
| Priority | Prioridad del mensaje. |
| RGB | Color del texto del cuerpo del mensaje. |

### 5.3.10. Descripción de la clase DMailAccounts.

| **public class DMailAccounts** | |
|---|---|
| **Atributo** | **Tipo** |
| addressBookField | DMailAccountsAddressBook |
| configurationSettingsField | DMailAccountsConfigurationSettings |
| deleteItemsField | DMailAccountsDeleteItems |
| dMailField | string |
| inboxField | DMailAccountsInbox |
| loginField | string |
| nameField | string |
| outboxField | DMailAccountsOutbox |
| passwordField | string |
| sentItemsField | DMailAccountsSentItems |
| **Miembro** | **Descripción** |
| DMailAccounts() | Inicializa una nueva instancia de la clase. |
| AddressBook | Libro de direcciones de contactos. |
| ConfigurationSettings | Configuración de la cuenta DMail. |
| DeleteItems | Elementos eliminados. |
| DMail | Dirección DMail de la cuenta. |
| Inbox | Bandeja de entrada. |

| | |
|---|---|
| Login | Alias del usuario DMail. |
| Name | Nombre del usuario de la cuenta. |
| Outbox | Bandeja de salida. |
| Password | Contraseña de la cuenta. |
| SentItems | Elementos enviados. |

### 5.3.11. Descripción de la clase DMailAccountsAddressBook.

| **public class DMailAccountsAddressBook** | |
|---|---|
| **Atributo** | **Tipo** |
| contactField | List<DMailUser> |
| **Miembro** | **Descripción** |
| DMailAccountsAddressBook () | Inicializa una nueva instancia de la clase. |
| Contact | Lista de contactos. |

### 5.3.12. Descripción de la clase DMailAccountsConfigurationSettings.

| **public class DMailAccountsConfigurationSettings** | |
|---|---|
| **Atributo** | **Tipo** |
| anonymizedField | string |

| | |
|---|---|
| compressionField | string |
| directoryField | string |
| institutionField | string |
| languageField | string |
| patientField | string |
| searchScopeField | string |
| studyField | string |
| transmissionField | string |

| **Miembro** | **Descripción** |
|---|---|
| DMailAccountsConfigurationSettings() | Inicializa una nueva instancia de la clase. |
| Anonymized | Especifica los campos a anonimizar en los ficheros adjuntos. |
| Compression | Especifica el modo de compresión. |
| Directory | Especifica el directorio para el cual serán almacenados los DICOM que lleguen para esta cuenta. |
| Institution | Especifica si serán anonimizados los datos de la institución donde se generó el DICOM. |
| Language | Idioma definido para el sistema por esta cuenta de usuario. |
| Patient | Especifica si serán anonimizados los datos del paciente dentro del DICOM. |
| SearchScope | Especifica el modo de búsqueda de los ficheros en el directorio DICOM. |
| Study | Especifica si serán anonimizados los datos del estudio médico |

| | |
|---|---|
| | del DICOM. |
| Transmission | Especifica el modo de transmisión (Estudio Médico, Serie Médica o Fichero DICOM). |

### 5.3.13. Descripción de la clase DMailAccountsDeleteItems.

| **public class DMailAccountsDeleteItems** | |
|---|---|
| **Atributo** | **Tipo** |
| eMailFieId | List<DMailEMail> |
| **Miembro** | **Descripción** |
| DMailAccountsDeleteItems() | Inicializa una nueva instancia de la clase. |
| EMail | Lista de mensajes eliminados. |

### 5.3.14. Descripción de la clase DMailAccountsInbox.

| **public class DMailAccountsInbox** | |
|---|---|
| **Atributo** | **Tipo** |
| eMailFieId | List<DMailEMail> |
| **Miembro** | **Descripción** |
| DMailAccountsInbox() | Inicializa una nueva instancia de la clase. |
| EMail | Lista de mensajes recibidos. |

### 5.3.15. Descripción de la clase DMailAccountsOutbox.

| **public class DMailAccountsOutbox** | |
|---|---|
| **Atributo** | **Tipo** |
| eMailField | List<DMailEMail> |
| **Miembro** | **Descripción** |
| DMailAccountsOutbox() | Inicializa una nueva instancia de la clase. |
| EMail | Lista de mensajes por enviar. |

### 5.3.16. Descripción de la clase DMailAccountsSentItems.

| public class **DMailAccountsSentItems** | |
|---|---|
| **Atributo** | **Tipo** |
| eMailField | List<DMailEMail> |
| **Miembro** | **Descripción** |
| DMailAccountsSentItems() | Inicializa una nueva instancia de la clase. |
| EMail | Lista de mensajes enviados. |

### 5.3.17. Descripción de la clase Transfer.

| **public class Transfer** | |
|---|---|
| **Atributo** | **Tipo** |
| eMailField | DMailEMail |
| identifyField | string |
| senderField | string |
| timeErrorField | string |
| timeField | string |
| typeField | string |
| **Miembro** | **Descripción** |
| Transfer() | Inicializa una nueva instancia de la clase. |
| EMail | Mensaje DMail. |
| Identify | Identificador para la transferencia. |
| Sender | Remitente. |
| Time | Fecha de composición del mensaje. |
| TimeError | Tiempo de error. |
| Type | Tipo de transferencia. |

### 5.3.18. Descripción de la clase EDCMServer.

| **public sealed class EDCMServer** | |
|---|---|
| **Atributo** | **Tipo** |
| aeTitle | string |
| clientCount | int |
| activeClients | ArrayList |
| inboxRoot | string |
| listener | System.Net.Sockets.TcpListener |
| listenerThread | System.Threading.Thread |
| port | int |
| stop | bool |
| **Miembro** | **Descripción** |
| .cctor() | |
| EDCMServer() | Inicializa una nueva instancia de la clase. |
| acse_OnConnectionDropped(AcceptorACSE) | |
| AETitle | Nombre de la entidad DICOM. |
| DcmReceived(Object) | Evento, DICOM recibido. |
| ImplmentationUID | ID de implementación. |
| ImplmentationVersionName | |

| | |
|---|---|
| InBoxRoot | Directorio de almacenamiento DICOM. |
| ListenerThread() | Hilo encargado de la escucha. |
| MaxClients | Cantidad máxima de clientes simultáneos. |
| OnAcseAbortingHandler(Object, AcseAbortingEventArgs) | |
| OnAcseClosedHandler(Object) | |
| OnAcseStateChangeHandler(Object, AcseStateChangeEventArgs) | |
| OnAssociationEstablishedHandler(Object, Association) | |
| OnDcmReceived | |
| OnEchoIndicationHandler(Object, MessageEventArgs) | |
| OnProtocolViolationHandler(Object, ProtocolViolationEventArgs) | |
| Port | Puerto para la comunicación DICOM. |
| Start() | Inicia el servidor. |
| Stop() | Detiene el servidor. |

### 5.3.19. Descripción de la clase INPClient.

| **public class INPClient** | |
|---|---|
| **Atributo** | **Tipo** |
| _mainSoc | Socket |
| _port | int |
| _serverIP | string |
| **Miembro** | **Descripción** |
| INPClient(String, Int32) | Inicializa una nueva instancia de la clase. |
| Connect() | Establece la conexión con el servidor. |
| Disconnect() | Desconecta el servidor. |
| OnDataRecieved(Byte[]()) | Evento, al recibir datos. |
| OnDataRecieved(IAsyncResult) | |
| Send(Byte[]()) | Envía un flujo de byte. |
| SendFile(String) | Envía un fichero. |
| SendFile(Object) | |
| WaitForData() | |

### 5.3.20. Descripción de la clase TransferConector.

| **public class TransferConector** | |
|---|---|
| **Atributo** | **Tipo** |
| rutaTransferXML | string |
| transfer | Transfer |
| **Miembro** | **Descripción** |
| TransferConector() | Inicializa una nueva instancia de la clase. |
| TransferConector(String) | Inicializa una nueva instancia de TransferConector a partir de un xml. |
| CargarXML() | Carga los datos de Transfer.xml. |
| RutaTransferXML | Ruta completa del fichero Transfer.xml. |
| SalvarXML() | Salva los datos para Transfer.xml. |
| SalvarXML(String) | Salva los datos para Transfer.xml en un directorio especificado. |
| Transfer | Instancia del objeto que representa a Transfer.xml. |

### 5.3.21. Descripción de la clase Info_Configuracion.

| **internal class Info_Configuracion** | |
|---|---|
| **Atributo** | **Tipo** |
| _asunto | string |

| | |
|---|---|
| _cant_dcm_serie | int |
| _carpeta_busqueda | string |
| _dcm_ruta_completa | string |
| _dir_aplic_actual | string |
| _dir_dcm_cuenta | string |
| _dmail_destino | string |
| _dmail_origen | string |
| _dmailconector | string |
| _es_Login_actual | bool |
| _id_estudio_actual | string |
| _id_serie_actual | string |
| _inboxconector | InboxConector |
| _ip_origen | string |
| _login_actual | string |
| _nombre_destino | string |
| _nombre_origen | string |
| _notificaciones_Pend | string |
| _notifico | bool |
| _proxy | string |
| _transferconector | TransferConector |

| | |
|---|---|
| _transmission | string |
| dias_Intentos | int |
| **Miembro** | **Descripción** |
| Info_ConfiguracionNew() | |
| Asunto | Asunto de este envío. |
| Cant_dcm_serie | Cantidad de DICOM de esta serie. |
| Carpeta_busqueda | Carpeta de búsqueda de los ficheros DICOM. |
| Dcm_ruta_completa | Nombre del fichero después de ser movido. |
| Dias_Intentos | Tiempo de duración de los elementos activos luego de este número de días, pasan a ser obsoletos en la bandeja de salida. |
| Dir_aplic_actual | Directorio donde está corriendo la aplicación. |
| Dir_dcm_cuenta | Directorio definido para los DICOM de esta cuenta. |
| Dmail_destino | DMail del usuario destinatario. |
| Dmail_origen | DMail del usuario remitente. |
| DMailConector | Conector de dmail.xml. |
| Es_Login_actual | Dice si la cuenta del destinatario es el actual login. |
| Generar_Carpeta_Propia(String) | Genera una carpeta específica para el destinatario dentro de su directorio raíz. |
| Generar_Carpeta_Propia_Serie(String) | Genera una carpeta con el nombre de la serie dentro del directorio raíz. |
| Generar_Carpeta_Propia_Study(String) | Genera una carpeta con el nombre del estudio dentro del directorio raíz. |

| | |
|---|---|
| Hacer_Copia_Directorio(String, String) | Si existe la carpeta con el mismo nombre, creamos una copia de la carpeta. |
| Id_estudio_actual | Identificador del estudio de este fichero DICOM. |
| Id_serie_actual | Identificador de la serie de este fichero DICOM. |
| InboxConector | Conector de inbox.xml. |
| Ip_origen | Ip de la PC que manda este mensaje. |
| Limpiando_Variables() | Después de cada almacenamiento, limpia todas las variables excepto la lista de series simultáneas. |
| Limpiar_DCM_Temp() | Limpia los temporales de la lista DCM_Temp que están caducos o sea que se trasmitieron en un envío incompleto. |
| Login_actual | Login actual. |
| Nombre_destino | Nombre del destinatario de este envío. |
| Nombre_origen | Nombre del remitente de este envío. |
| Notificaciones_Pend | Lista de notificaciones pendientes. |
| Notifico | Dice si se debe notificar el nuevo mensaje. |
| ObtenerDirectorioDCMxDmail(String) | |
| ObtenerLoginxDmail(String) | Devuelve el login del usuario con DMail de igual valor. |
| ObtenerNombrexDMail(String) | Devuelve el nombre del usuario con DMail de igual valor. |
| Proxy | Ip del Proxy de salida. |
| Raiz_Temp | Info_Configuracion.Dir_aplic_actual + Raiz_Temp diretorio raiz de los temporales comprimidos. |
| Raiz_XMLs | Info_Configuracion.Dir_aplic_actual + Raiz_Data diretorio raiz de |

| | |
|---|---|
| | los xmls. |
| Tiempo_lectura | Tiempo máximo intentando leer un fichero en minutos. |
| TransferConector | Conector de transfer.xml. |
| Transmission | Modo de transmisión para este mensaje. |

### 5.3.22. Descripción de la clase Enviar.

| **public class Enviar** | |
|---|---|
| **Atributo** | **Tipo** |
| _Operador | Thread |
| dmailconector | DMailConector |
| lista_elementos | List<DMailEMail> |
| lista_enviados | List<DMailEMail> |
| lista_obsoletos | List<DMailEMail> |
| tiempo_intervalo | int |
| transferconector | TransferConector |
| **Miembro** | **Descripción** |
| Enviar() | Inicializa una nueva instancia de la clase. |
| ActualizarDMail() | Borra de las bandejas de salida de las cuentas, los elementos enviados y los pasa para los elementos enviados. |

| | |
|---|---|
| CargarElementos(DMailAccounts) | Carga la lista de los elementos a ser enviados para esta cuenta. |
| CompressFiles(ArrayList) | Comprime los ficheros DICOM. |
| EnviarElemento(DMailEMail) | Envía el elemento especificado. |
| FindIndex(DMailEMail, TipoColeccion) | Encuentra la posición donde se encuentra el elemento. |
| FindIndex(DMailEMail, List<(Of DMailEMail>)) | Encuentra la posición donde se encuentra el elemento. |
| InicializarEnvio() | Inicializa el envío en este intervalo. |
| LimpiarTemporales(String) | Limpia todos los ficheros del directorio especificado. |
| LoadCompleteSerie(String, String) | Carga los ficheros DICOM de la serie completa. |
| LoadMedicalStudy(String, String) | Carga los ficheros DICOM del estudio completo. |
| LoadSelectedDICOM(String) | Carga el fichero DICOM seleccionado. |
| Operador | Hilo encargado de realizar las operaciones de envío. |
| Operar() | Encargado del envío de los mensajes. |
| SendDICOMFile(String, DICOMFile, Boolean, Boolean, Boolean) | Envía el fichero DICOM. |
| SendXmlFile(String, String) | Envía un fichero xml. |
| Start() | Comienza el funcionamiento del hilo encargado de los envíos. |

### 5.3.23. Descripción de la clase Notify.

| **public class Notify** | |
|---|---|
| **Atributo** | **Tipo** |
| actual_logueado | bool |
| asunto | string |
| destino | string |
| origen | string |
| **Miembro** | **Descripción** |
| Notify(Boolean, String, String, String) | Inicializa una nueva instancia de la clase. |
| Actual_loqueado | Verdadero, si el mensaje es para el usuario actual, de lo contrario, falso. |
| Asunto | Asunto de este mensaje. |
| Destino | Nombre del usuario destinatario. |
| Origen | Nombre del usuario remitente. |

### 5.3.24. Descripción de la clase Recibir.

| **public class Recibir** | |
|---|---|
| **Atributo** | **Tipo** |
| _DCM_temp | ArrayList |
| _Operador | Thread |
| _XML_temp | ArrayList |
| servidores | Servidores |
| Notificaciones_Pend | ArrayList |
| DCM_Nombres_trasmitidos | ArrayList |
| DCM_Trasmitidos | ArrayList |
| Notificador | Thread |
| **Miembro** | **Descripción** |
| DCM_temp | Lista de los DICOM recibidos aun no tratados. |
| RecibirNew() | Inicializa una nueva instancia de la clase. |
| Enviar_Notificaciones() | Encargado de enviar las notificaciones pendientes. |
| InicializarDatos() | Inicializa los datos de las instancias y las áreas de configuración. |
| Iniciar_Almacenamiento(Info_XML) | Realiza todas las operaciones cuando llega un nuevo mensaje. |
| LimpiarTemporales() | Limpia los temporales DCM y XML. |

| | |
|---|---|
| LLenar_Lista_dcm_nombres(DMailEMail) | Obtiene la lista de los nombres de todos los DICOM trasmitidos, del xml. |
| LLenar_Lista_dcm_trasmitidos() | Llena la lista de los DICOM trasmitidos para este mensaje, del la lista de Info_DCM. |
| Notificar() | Notifica con los datos especificados. |
| OnDcmReceived(Info) | Método encargado de adicionar el nuevo DICOM a la lista de los DICOM recibidos (DCM_Temp). |
| OnXmlReceived(Info) | Método encargado de adicionar el nuevo xml a la lista de los xml recibidos (XML_Temp). |
| Operador | Hilo opera los recibos. |
| Operar() | Opera sobre los recibos para tratarlos. |
| Servidor | |
| Start() | Inicia los servicios de notificación y recibo de los DICOM. |
| XML_temp | Lista de los XML recibidos aun no tratados. |

### 5.3.25. Descripción de la clase INPServer.

| **public class INPServer** | |
|---|---|
| **Atributo** | **Tipo** |
| _localIP | string |
| _mainSoc | Socket |

| | |
|---|---|
| _port | int |
| outdir | string |
| **Miembro** | **Descripción** |
| INPServer(Int32) | Inicializa una nueva instancia de la clase. |
| OnAccept(IAsyncResult) | |
| OnClientConnected(ClientInfo) | |
| OnClientDisConnected(ClientInfo) | |
| OnDataRecieved(ClientInfo) | |
| OnDataRecieved(IAsyncResult) | |
| OnXmlReceived | Evento, cuando recibe un fichero. |
| Start() | Inicia el servidor. |
| Stop() | Detiene el servidor. |
| WaitForData(ClientInfo) | |
| XmlReceived(Info_XML) | Evento, cuando recibe un fichero xml. |

### 5.3.26. Descripción de la clase Servidores.

| **public class Servidores** | |
|---|---|
| **Atributo** | **Tipo** |
| _DCMServer | EDCMServer |

| | |
|---|---|
| _XMLServer | INPServer |
| **Miembro** | **Descripción** |
| Servidores() | Inicializa una nueva instancia de la clase. |
| DcmReceived(Info) | Método que lanza el evento OnDcmReceived. |
| DCMServer | Servidor DICOM. |
| OnDcmReceived | Evento que notifica la llegada de un DICOM. |
| OnXmlReceived | Evento que notifica la llegada de un nuevo XML. |
| Start() | Inicia los servidores de DCM y XML. |
| XmlReceived(Info) | Método que lanza el evento OnXmlReceived. |
| XMLServer | Servidor destinado a los XML. |

### 5.3.27. Descripción de la clase Info_DCM.

| **public class Info_DCM** | |
|---|---|
| **Atributo** | **Tipo** |
| _dcm_nombre_original | string |
| _dcm_uid | string |
| _dir_ficheros_envio | string |
| caduca | string |
| **Miembro** | **Descripción** |

| | |
|---|---|
| Info_DCM(String, String) | Crea una nueva instancia de la clase para manipular los datos de un mensaje. |
| Caduca | Fecha en que caduca de este DICOM. |
| Dcm_nombre_original | Nombre con el que estaba copiado el fichero DICOM en la maquina origen. |
| Dcm_uid | UID del DICOM de este mensaje. |
| Dir_ficheros_envio | Directorio temporal para este DICOM. |
| Generar_Carpeta_Temporal_Envio (String, String) | Genera una carpeta temporal para cada mensaje. |

### 5.3.28. Descripción de la clase Info_XML.

| **public class Info_XML** | |
|---|---|
| **Atributo** | **Tipo** |
| _xml_dir_compl | string |
| _xml_nombre | string |
| **Miembro** | **Descripción** |
| Info_XMLNew() | Inicializa una nueva instancia de la clase. |
| Info_XMLNew(String, String) | Inicializa una nueva instancia de la clase. |
| XML_dir_compl | Dirección completa del fichero xml. |
| Xml_Id | Nombre del fichero xml recibido. |

**GLOSARIO**

| Término | Grupo | Definición |
|---|---|---|
| DICOM | Técnico | Estándar internacional para el manejo de imágenes médicas. |
| NEMA | Técnico | Asociación Nacional de Fabricantes Eléctricos (National Electrical Manufacturers Association) |
| PACS | Técnico | Sistema de Almacenamiento y Transmisión de Imágenes. (Picture Archiving and Communication System) |
| Servicio SCP | Técnico | Rol de servidor que adopta un servicio DICOM en una asociación. |
| Servicio SCU | Técnico | Role de cliente que adopta un servicio DICOM en una asociación. |
| XML | Técnico | XML o Extensible Markup Language. Es un metalenguaje extensible para la especificación de datos en documentos. |
| TCP\IP | Técnico | (Transmission Control Protocol/Internet Protocol) Protocolo de transmisión de redes. |
| SMTP | Técnico | (Simple Mail Transfer Protocol) Protocolo de correo electrónico. |