Sistema PACS mínimo basado en el estándar DICOM.

Tesis presentada por:

Jiménez Herrera Armando.

Para obtener el grado de:

Maestro en Ciencias de la Computación.

Asesor:

Dr. Carlos Avilés Cruz.

México, D. F. Julio del 2006.

Agradecimientos:

El conocimiento es el tesoro; pero el juicio es el tesorero del hombre sabio. El que tiene más conocimiento que juicio ha sido hecho para servir a otros más que a si mismo.

-Willeam Penn.-

A mis padres Minerva y Armando.

Agradezco su infinito apoyo durante mi vida y formación.

A Irma y familia, Daniel y familia, y Bety.

Por haberme alentado a llegar a una meta más.

A mi Asesor de Tesis Dr. Carlos Avilés Cruz.

Por su apoyo en la dirección y asesoria desde el principio hasta la conclusión de este proyecto.

A Familiares y amigos.

Por compartir conmigo su entusiasmo a la superación.

A la Universidad Autónoma Metropolitana.

Por "La beca para alumnos de maestría y doctorado que se encuentren en el proceso de elaboración de la idónea comunicación de resultados o tesis, durante el periodo escolar 2005-2006".

Índice

Índice de tablas.

Índice de figuras.

Introducción.

Desde 1993, la integración de los sistemas de manejo de imágenes e información se ha acentuado. Esto incluye al Sistema de almacenamiento y comunicación de imágenes (PACS Picture Archiving and Communications System), que manejan los informes e imágenes de imagenologia y se comunica con los sistemas de archivo de registros médicos, el Sistema de Información Hospitalario (HIS, Hospital Information Systems), el Sistema de información de Radiología (RIS, Radiologic Information Systems), los sistemas de información de laboratorio, y los archivos en línea de CD-ROM, todo lo cual esta integrado y poderlo utilizar en red. Por lo que el rumbo que se toma es hacia el "Hospital Electrónico Integrado".

Actualmente encontramos en forma comercial Sistemas PACS creados por las empresas que producen equipos médicos como AGFA, Siemens , entre otros, los cuales tiene como estándar a DICOM (Digital Imaging and Communications in Medicine / Estándar de Comunicación e Imágenes Médicas). Los sistemas existenes comercialmente son vendidos a altos costo aunado a los honerosos costos de mantenimiento por lo que solo algunos hospitales tienen la capacidad para instalarlos a nivel nacional. El presente trabajo pretende sentar las bases para un sistema PACS con futuras aplicaciones en hospitales.

La estandarización de archivos médicos hace posible que, mediante una transmisión segura, los datos de los pacientes puedan viajar de departamento en departamento, de hospital en hospital, lo que hace que esa información pueda ser vista remotamente desde la zona de adquisición de las imágenes. Esto permite que los médicos puedan diagnosticar desde su casa, buscar diferentes opiniones de otros médicos expertos de una forma sencilla y rápida, un orden y estructura de los datos más efectivo y seguro, y muchos otros tipos de ventajas.

DICOM es el estándar industrial para transferencia de imágenes digitales e información médica entre computadoras. DICOM permite la comunicación digital entre equipos de diagnóstico, terapéuticos y entre sistemas de diferentes fabricantes. Se puede observar, la gran importancia de este estándar, ya que dá la posibilidad de interconectar sistemas informáticos de diferentes fabricantes y hace posible que se comuniquen entre sí, en un hospital, donde los aparatos médicos son de muchas marcas, debido a la especialización, es tremendamente interesante y necesario.

En el campo de la medicina, existe gran cantidad de imágenes para diagnóstico tales como: Rayos X, Radiografía Computarizada (CR), Tomografía Computarizada (CT), Resonancia Magnética (MRI), Ultrasonido, Medicina Nuclear (NMI), Angiografía de Sustracción Digital (DSA), entre otras. El manejo de dichas imágenes se ha vuelto complicado, principalmente cuando deben imprimirse y archivarse. En la figura 0.1 se muestra que las imágenes de estudios se digitalizan para un mejor manejo de estos.

El proliferar de los Sistemas Digitales en los años 80, ha conducido al desarrollo de la idea de constituir un departamento de radiología o imagenología prácticamente digital. Este departamento emplearía una red de estaciones de visualización junto con los sistemas de almacenamiento y adquisición de imágenes. Un sistema completo de este tipo se conoce bajo el nombre de un sistema PACS ("Picture Archiving and Communications System"). El empleo generalizado de este tipo de sistemas en nuestro país traería un cambio fundamental en el esquema de funcionamiento de los departamentos de radiología o imagenología, mejorando significativamente la eficiencia de los mismos, conjuntamente con una mejoría importante de la calidad de la atención médica que se les brinda a los pacientes. En este caso no solo se trataría de reproducir el paradigma de visualización en medios electrónicos, sino que al incorporar otro tipo de información antes no disponible, como el despliegue de imágenes multimodalidad, el realce de imágenes y el diagnóstico asistido por computadora.

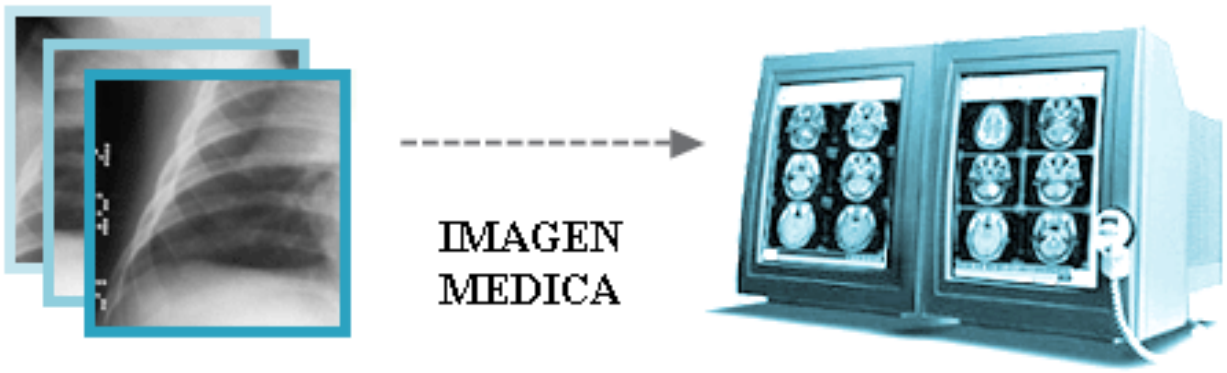


Figura 0.1 Digitalización de las Imágenes para un mejor manejo.

Esta investigación tuvo como objetivo el realizar el Análisis, Diseño e Implementación de un Sistema PACS basado en el estándar DICOM.

Esté proyecto se realizo en cuatro etapas: Análisis.- se identificó las necesidades del sistema en base a Reconocimiento del problema, Modelado y Especificación. Diseño.- se definio al sistema para su interpretación y realización física, desarrollando el diseño de las Base de datos, diseño de la interfaz a usuario y diseño de las Clases. Implementación.- Se implementaron la interfaz Gráfica y WEB. Pruebas.- Las pruebas que se aplicaron a la interfaz gráfica para la adquisición de imagen, interfaz WEB para la visualización de la información fueron de funcionalidad.

Se obtuvo como resultado un sistema con las siguientes características: Maneja la Base de Datos de la información del paciente; cuenta con las librerías para los distintos tipos de imágenes; Un servidor de Base de Dato e Imágenes que es capas de atender a múltiples clientes y que soporte el protocolo DICOM para adquirir imágenes médicas cargando Objetos de Información compuesto (IOD-Compuesto); cuenta con una interfaz gráfica para la adquisición de imágenes, el sistema muestra la información del paciente, estudio, equipo, serie, e imagen según lo solicite el usuario (Visualizador) utilizando una interfaz WEB;Se implementaron mecanismos de protección de la privacidad del sistema controlando el acceso y privacidad de los datos para la interfaz WEB utilizando SSL, la interfaz gráfica de adquisición de imagen controla el acceso.

1 Antecedentes y revisión de estado del arte.

Este punto esta formado por tres parte. En la primera describimos como esta formado un sistema PACS. En la segunda parte damos una introducción del estándar DICOM. Y en la última parte mencionamos algunos trabajos de investigación ya realizados.

1.1 Fundamentos de un PACS.

La figura 1.1 muestra una realización pequeña de un sistema PACS. Se trata de la implantación de una red de estaciones de consulta, diagnóstico y visualización basada en computadoras personales. Se desarrolló entre la Université de Technologie de Compiègne y el Centro Hospitalario de Amiens, en el norte de Francia [4].

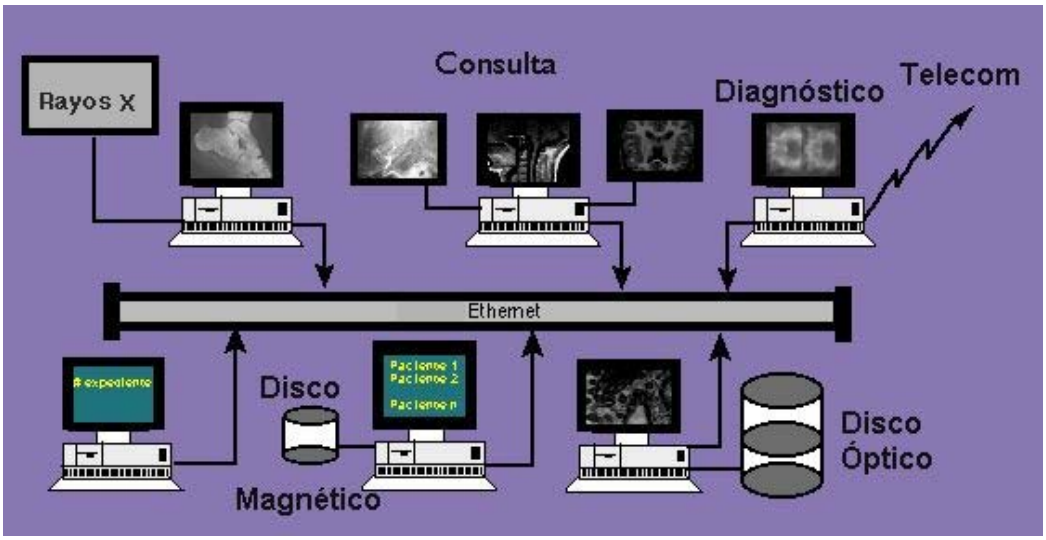


Figura 1.1 Red MicroPACS, Université de Technologie de Compiègne y el Centro Hospitalario de Amiens, en el norte de Francia.

Los sistemas PACS, utilizan varios componentes (hardware y software) con funciones específicas.

Son cinco componentes básicos de los sistemas PACS[4]:

- Adquisición de Imágenes.
- Red de Comunicación.
- Bases de Datos.
- Estaciones de Diagnóstico y Visualización.
- Sistemas de Almacenamiento.

A continuación describimos a manera de bloque, dichos componentes. Ver figura 1.2.

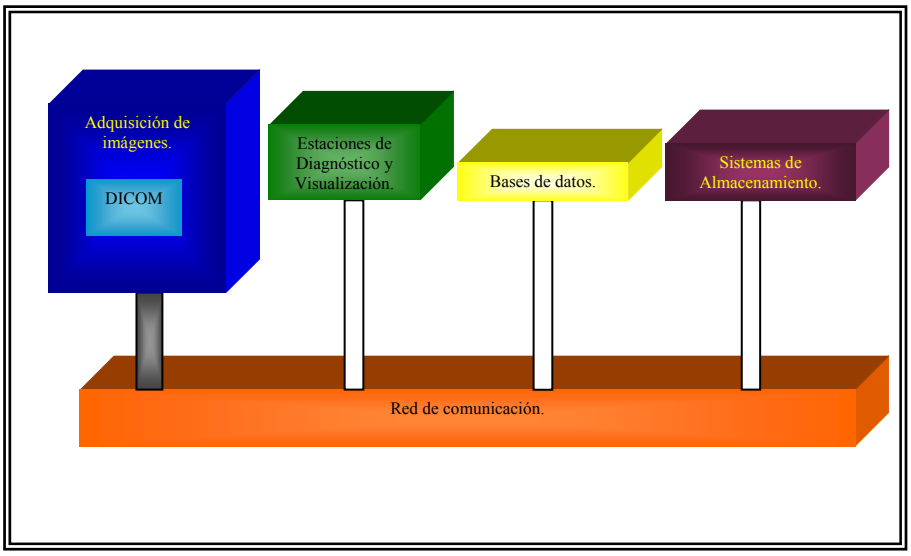


Figura 1.2 Componentes de un Sistema PACS.

Adquisición.

La adquisición de las imágenes tiene dos modalidades principales. En el caso de una cantidad importante de tipos de imagen, debido a su naturaleza, se tiene que éstas ya se encuentran en un formato digital. Las imágenes de Tomografía Axial Computarizada, Resonancia Magnética Nuclear, Medicina Nuclear, entre otras son digitales y se imprimen en placa por comodidad únicamente. En estos casos, el reto es encontrar la manera de obtener la información digital directamente de la máquina y transmitirla a través de la red al archivo de imágenes. Es común encontrar que las imágenes se proporcionan bajo un formato no estándar, que depende del fabricante. En otros casos, se presentan los datos digitales siguiendo el estándar DICOM, en cuyo caso es posible leerlos y almacenarlos utilizando esta norma.

En el caso que se tengan las imágenes disponibles únicamente en placa, se tendrán que digitalizar manualmente, por medio de un digitalizador de placas (scanner).

Los equipos de diagnóstico por imagen.

Cada uno de los equipos de diagnóstico que obtienen imágenes del paciente se denomina modalidad. Como por ejemplo:

- Tomografía Computarizada (CT).
- Resonancia Magnética (MRI).
- Radiografía Computarizada (CR).
- Radiografía Directa (DR).
- Película Digitalizada (FD).
- Ultrasonido (US).
- Medicina Nuclear (NM).
- Fluoroscopia Digital (DF).
- Radiología Angiografía (Angi-R) Cardiología Angiografía (Angi-C).

La adquisición de la imagen médica.

Considerando el formato de la información original, existen dos fuentes de imágenes:

- Imágenes producidas sobre película (Radiografías ó Ecografía) que deberían ser digitalizadas para poder ser explotadas digitalmente.
- Imágenes generadas en formato digital de forma directa: Tomografía Computarizada, Medicina Nuclear, Resonancia Magnética, Radiografía Computarizada.

La mayor calidad de información se obtiene con la conexión digital directa de las modalidades, que permite tener toda la información de la exploración.

Dentro de los equipos que generan imágenes médicas no digitales encontramos dos tipos diferentes de equipos: Los que generan la imagen sobre placas radiográficas (rayos-X) y los que presentan las imágenes en video, dependiendo del origen de la imagen se han de seguir metodologías diferentes para su digitalización.

Placas convencionales de Rayos X.

Las imágenes obtenidas sobre película convencional deben ser convertidas a formato digital para incorporarlas a la carpeta del paciente. El proceso consiste en una lectura punto a punto de cada película con un digitalizador, que puede ser de tres tipos: cámara de video CCD, barrido por CCD, o barrido por láser. La mejor calidad se obtiene con los digitalizadores láser, obtienen resolución superior a 2000x2000 píxeles y una gama de densidades de 12 bits (4096 tonos) por píxel. Con la cámara de video CCD, limitada 8 bits (256 grises) y a resoluciones inferiores a 1024x1024 píxeles, la calidad es muy limitada, (aunque hay prototipos a 2048x2048 píxeles).

Este proceso de conversión es siempre costoso ya que dúplica el registro analógico, además precisa personal para la manipulación de las películas, y con los equipos digitalizadores menos sofisticados disminuye la calidad de la imagen.

Digitalización de video.

La interfaz digital de los equipos con señal de video, se pueden realizar con digitalizadores de video "Frame Grabber" que toma la imagen de un monitor del equipo de exploración y la convierte en un archivo gráfico. La resolución espacial oscila alrededor de 800x800 píxeles y 8 bits (256 grises), que no corresponden con los datos originales de adquisición sino con la ventana o ajuste del monitor. Los digitalizadores de video son válidos en Ecografías, en Fluoroscopia Digital e incluso en Resonancia Magnética, pero su ventana máxima de 256 niveles de grises, claramente insuficiente en la Tomografía Computarizada. Se requiere almacenar 4000 unidades Hounsfield (12 bits).

Imagen Digitales.

La introducción de modalidades radiológicas con adquisición digital: Tomografía Computarizada, Radiología Angiografía, Cardiología Angiografía, Medicina Nuclear, Resonancia Magnética y sobre todo la Radiografía Computarizada, y el progreso de las tecnologías de la comunicación e informática, han facilitado el desarrollo de la administración directa de las imágenes en formato digital.

La imagen médica digital constituye un paradigma de requerimientos para cualquier sistema computarizado: las imágenes presentan un volumen muy elevado de información, tanto por sus características de resolución espacial, como por el volumen de datos o números de imágenes por exploración.

Modalidad	Tamaño de la imagen			Por Estudio			
				Numero de Imágenes		Almacenamiento en MB	
	X (píxeles)	Y (píxeles)	Almacenamiento en MB.	Típico	Rango	Típico	Rango
Tomografía computarizada (CT)	512	512	0,524	60	40-300	31,4	21-157
Resonancia Magnética (MRI)	256	256	0,131	160	80-1000	21,0	10,5-131
Radiografía Computarizada (CR)	2000	2500	10,00	3	2-5	30,0	20-50
Radiografía Directa (DR)	3000	3000	18,00	3	2-5	54,0	36-90
Película digitalizada (FD)	2000	2500	10,00	3	2-5	30,0	20-50
Ultrasonido (US)	640	480	0,614	30	20-60	18,40	12,3-36,8
Medicina Nuclear (NM)	256	256	0,131	10	4-30	1,31	0,52-3,93
Fluoroscopia Digital (DF)	1024	1024	1,05	20	10-50	21,0	10,5-52,5
Radiología angiografía (Angi-R)	1024	1024	1,05	15	10-30	15,8	10,5-31,5
Cardiología angiografía (Angi-C)	1024	1024	1,05	70	40-120	73,5	42-126

Tabla 1.1 Imagen típica y tamaño de estudio por Modalidad.

En la Tabla 1.1 se muestra el tamaño y el número de las imágenes en cada estudio así como el tamaño requerido en MB.

Redes de comunicación.

La red de comunicación es un elemento fundamental de los sistemas PACS. Esta puede ser una red simple tipo Ethernet en un sistema mínimo, pero comúnmente se cuenta con una serie de elementos con distintas velocidades de acceso, que dependen de las necesidades de velocidad de transferencia de información. Comunmente se cuenta con una red de alta velocidad dentro del departamento de imagenología, que puede ser FDDI (transmisión de datos en LAN que opera sobre fibra óptica a 100 Mbps) o Gigabit Ethernet (1000 megabytes por segundo), un red de menor capacidad dentro del hospital, como Ethernet convencional y un sistema de acceso exterior puede ser tan lento como el acceso telefónico, el empleo de la red digital de servicios integrados, o canales de mayor velocidad. Estos esquemas se basan en el hecho de que la mayor parte del tráfico de información se encontrará dentro de la misma unidad de imagenología, donde se hará la mayor parte del diagnóstico radiológico y donde se generán los informes por parte de los especialistas. Esta demanda de ancho de banda justifica la instalación de una red de alta velocidad.

En el caso de la conexión al resto del hospital, la velocidad de transferencia no tiene que ser tan alta, ya que la demanda es menor. Es común que se tengan enlaces entre los sistemas generales de información hospitalaria, donde se encuentran los expedientes de los pacientes, y sistema de información radiológica. En algunos casos, todo el hospital está cableado con la misma tecnología (frecuentemente se trata de fibra óptica), por lo que la intercomunicación en sistemas de información se facilita. Para las comunicaciones con el exterior se debe hacer un estudio cuidadoso del ancho de banda que se requiere, ya que los costos de renta para RDSI y otras opciones pueden ser altos.

Bases de datos.

El diseño de un sistema de bases de datos y su implantación son fundamentales para el buen funcionamiento de un sistema PACS. Se deben almacenar tanto imágenes como voz (el informe oral del radiólogo) y texto. El diseño de la base de datos debe ser orientado a objetos para que su manejo sea más intuitivo. Se debe tener una estrategia para el almacenamiento de información: En las horas siguientes a la adquisición de una imagen, ésta se consulta con más frecuencia. A lo largo del tiempo la probabilidad de que esta imagen sea consultada disminuye significativamente. Debido a esto, el almacenamiento a corto plazo (plazos de horas) debe hacerse en los sistemas locales (memoria y disco). A mediano plazo (días), el almacenamiento debe hacerse en servidores locales, mientras que el almacenamiento permanente y a largo plazo puede hacerse ya sea en unidades de disco óptico o en cinta magnética. Unido a esto, debe existir un módulo que se encargue de efectuar una recuperación inteligente de las imágenes que probablemente se solicitarán (prefetch), junto con un sistema de compresión y descompresión en línea. Un ejemplo de esta aplicación es el precargado de las imágenes de un determinado paciente, el día de su consulta. Así, los médicos podrán hacer un seguimiento a largo plazo de sus padecimientos y podrán solicitar cualquiera de sus imágenes, si así lo desean. El programa estaría encargado de revisar la agenda de visitas programadas y de precargar las imágenes que ordinariamente se encuentran en almacenamiento a largo plazo.

Estaciones de diagnóstico y visualización.

Las estaciones de diagnóstico y visualización también son elementos importantes en un sistema PACS. Estos son los elementos que presentan la información visual a los médicos y deben cumplir con las normas de calidad adecuadas.

Para el caso de las estaciones de diagnóstico, que se encuentran dentro del departamento de imagenología, éstas deben tener una muy alta resolución y se deben poder presentar imágenes en monitores múltiples de 2048 x 2048 píxeles y un tamaño de no menos de 19". Para las estaciones de visualización que se encontrarán en todas partes dentro de un hospital, y que recibirán las imágenes ya analizadas por los especialistas, éstas deberán tener una resolución de alrededor de 1024 x 1024 píxeles y 17" de diámetro. En ambas situaciones es deseable incorporar funciones básicas de procesamiento de imágenes para poder hacer operaciones de cambio de contraste y de intensidad por lo menos. Es deseable además que se incorporen otras funciones tales como audio (informes orales, traducción automática de audio a reporte escrito) y despliegue de otros tipos de información en tiempo real (Ayuda en línea, marcado de áreas de interés), todo bajo una interfaz para el usuario amigable.

Las estaciones de diagnóstico y visualización deben contar con algunas funciones de procesamiento de imágenes. Estas son las funciones de base, que consisten en:

- Modificación de Contraste.
- Acercamientos (Zoom).
- Mediciones Cuantitativas.
- Anotación sobre la imagen.
- Ecualización de histogramas.
- Análisis de texturas.
- Despliegue en 3D.
- Filtrado.
- Registro.

Las funciones básicas deben estar disponibles en ambos tipos de estaciones, mientras que las funciones avanzadas de procesamiento deben incluirse en las estaciones de diagnóstico. La diferencia en la disponibilidad de estas funciones obedece al hecho de que el primer tipo no altera las características fundamentales de las imágenes y sirven para mejorar el despliegue de las mismas, mientras que el segundo tipo, en las estaciones de diagnóstico serán manejadas por expertos que podrán generar nuevas imágenes con realce que estarán disponibles en los archivos radiológicos y que servirán para complementar la información existente. La implantación de estas funciones implica el proporcionar un cierto grado de capacidades de cálculo a ambos tipos de estaciones.

Sistemas de almacenamiento.

Los sistemas de almacenamiento de imágenes deben seguir una estructura jerárquica que dependerá de la probabilidad de demanda de la imagen. En general las imágenes recientemente adquiridas se consultan con mucha frecuencia en los minutos siguientes a su adquisición y su frecuencia de consulta disminuye rápidamente con el tiempo. En la figura 1.3 se muestra una gráfica de la demanda de visualización vs. Tiempo.

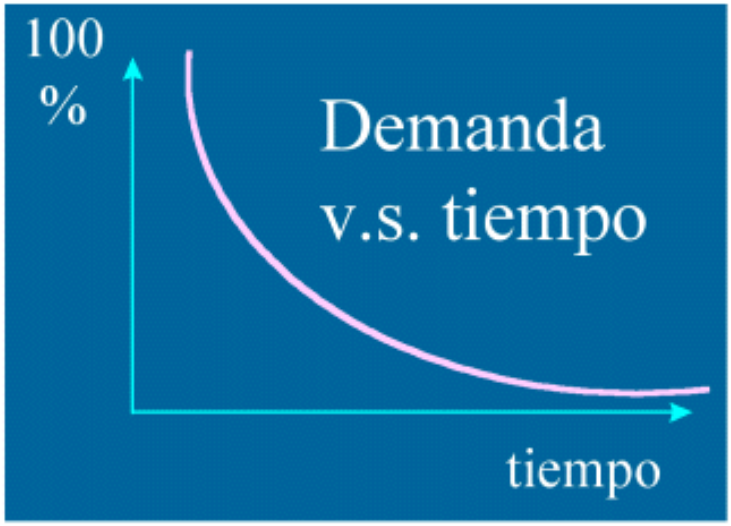


Figura 1.3 Gráfica de la demanda de visualización de una imagen a lo largo del tiempo.

La estructura jerárquica toma en cuenta estas características para reducir los costos, mientras que se aumenta el rendimiento, consiste de:

- Discos magnéticos locales.
- Discos magnéticos remotos.
- Discos ópticos.
- Cinta.

El almacenamiento a corto plazo (local) tiene las siguientes características:

- Decenas de GB.
- Transferencia de alrededor de 50 imágenes por minuto.
- 1-15 días de almacenamiento.

El almacenamiento a largo plazo debe cumplir con los siguientes requisitos:

- Capacidad de varios Terabytes.
- Empleo de robots o "jukeboxes" de discos ópticos.
- Capacidad de almacenamiento de dos años de información.
- Empleo de cinta e imágenes comprimidas para almacenamiento a plazos mayores.

La compresión de imágenes se puede emplear para multiplicar el espacio en disco, y para reducir el tiempo de transferencia. Se pueden emplear varios criterios:

- Compresión reversible con tasas de 3:1 para imágenes de referencia o para almacenamiento a corto plazo.
- Compresión irreversible con tasas de 10-20:1 para almacenamiento a largo plazo.
- En la actualidad el problema del tipo de compresión adecuado para un determinado tipo de imágenes no está resuelto y sigue siendo un tema de investigación.

1.2 Generalidad del estándar DICOM.

DICOM (Digital Imaging and Communications in Medicine) es el estándar que define los métodos para la transferencia de imágenes médicas para diagnóstico y la información asociada a ellas, entre equipos de imagenología y sistemas de fabricantes distintos.

El objetivo principal del Comité de Estándares DICOM es crear y mantener estándares internacionales para la comunicación de información biomédica diagnóstica y terapéutica para las disciplinas que utilizan imágenes digitales y datos relacionados.

Como resultado, DICOM es y será utilizado por prácticamente cualquier disciplina médica que utilice imágenes dentro de la industria del cuidado de la salud. Esto incluye neurología y neurocirugía, cardiología, endoscopía, mamografía, oftalmología, ortopedia, patología, pediatría, radioterapia, cirugía, odontología y veterinaria.

El estándar DICOM 3.0 (1992) evolucionó de los estándares ACR-NEMA versiones 1.0 (1985) [21] y 2.0 (1988) desarrollados por ACR (American College of Radiology) y NEMA (National Electrical Manufacturers Association) que en 1983 formaron un comité para desarrollar un estándar que cumpliera los siguientes objetivos:

- Promover la comunicación de imágenes digitales, independientemente del fabricante del equipo
- Facilitar el desarrollo y expansión de los sistemas de almacenamiento y comunicación de imágenes (PACS) capaces de comunicarse también con otros sistemas de información hospitalaria.
- Permitir la creación de bases de datos de información diagnóstica que pudiesen ser consultadas por una amplia variedad de dispositivos remotos.

Los estándares ACR-NEMA 1.0 y 2.0 especificaban un hardware para interfaz, y un conjunto mínimo de software, comandos y un grupo consistente de formatos de datos.

El estándar actualmente conocido como DICOM 3.0 (Digital Imaging and Communications in Medicine) [21] incorpora las siguientes mejoras con respecto al estándar ACR-NEMA:

- Es aplicable en un ambiente de red mediante el protocolo de red estándar TCP/IP.
- Es aplicable en un ambiente con medios fuera de línea como CD-R y MOD, y sistemas de archivo lógicos como ISO 9660 y sistemas de archivo para PC (FAT 16)
- Especifica la semántica de los comandos y los datos asociados mediante el concepto de Clases de Servicios (Service Classes).
- Especifica niveles de conformidad. Describe explícitamente como un desarrollador debe estructurar las sentencias para seleccionar opciones específicas.
- Incluye además de imágenes y gráficos, Objetos de Información para graficación de presiones por curvas, reportes, estudios, pacientes, impresiones y otros grupos de datos.
- Establece una técnica para identificar claramente cualquier Objeto de Información conforme son transmitidos a través de la red.

1.3 Trabajos Principales.

Los trabajos estudiados son:

- Instalación y operación de sistemas PACs (almacenamiento y comunicación de imágenes): características fundamentales[12].
- Sistemas PACS-CNR: Una propuesta tecnológica[5].
- Introduction to Image Acquisition, Archive Managers, and Storege[32].
- Real-time teleconsulting solution for teleradiology based on PACS [33].
- The Development of a Client/Server Architecture for Standardized Medical Application Network Services[34].
- Manejo de imágenes DICOM (Digital Imaging and Communications in Medicine) Mediante un sistema en Internet (Tesis)[1].
- Desarrollo de aplicaciones DICOM para la gestión de imágenes biomédicas(Tesis)[45].

Instalación y operación de sistemas PACs (almacenamiento y comunicación de imágenes): características fundamentales.

En el trabajo presentado por AZPIROZ LEEHAN et Al.

Publicado en Revista Mexicana de Ingeniería Biomédica • volumen XIX • número 3 • noviembre 1998

Resumen:

Este trabajo describe en forma resumida el problema del almacenamiento y manejo de las imágenes médicas en nuestro país. Los cuales son: La existencia de un almacenamiento de la información deficiente; El desperdicio de los recursos disponibles; Una disminución en la calidad de la atención al paciente.

Hace un planteamiento de las soluciones a los problemas anteriormente expuestos que se basan en la aplicación de la tecnología digital para implantar sistemas de manejo y comunicación de las imágenes. Las soluciones pueden incluir desde el diseño y construcción de un sistema simple pero funcional, hasta la adquisición de un sistema PACS comercial completo.

Describe los componentes básicos de un sistema PACS que son: Adquisición de imágenes; Red de comunicación; Bases de datos; Estaciones de Diagnóstico y Visualización; Sistemas de Almacenamiento.

Y finalmente resume las ventajas de estos sistemas.

Sistemas PACS-CNR: Una propuesta tecnológica.

En el trabajo presentado por la Josefina Gutiérrez et al .

Publicado en la Revista Mexicana de Ingeniería Biomédica; Número 1; Marzo 2003; Volumen 24. pág. 77-85.

Resumen: En el Centro Nacional de Rehabilitación (CNR), al igual que en otros centros de salud, el incremento en la demanda de almacenamiento y manipulación de imágenes médicas, la necesidad de transferirlas entre los equipos que las generan hasta las áreas médicas que las requieren así como la evolución de la tecnología, ha conducido a proponer que resulta más conveniente y eficiente emplear un sistema formado por una red de computadoras, con características de eficiencia, que integre estaciones de adquisición, visualización, consulta, diagnóstico, almacenamiento e impresión de imágenes médicas, conocido como PACS, como una alternativa para reducir los inconvenientes que ocasionan los procedimientos

convencionales del manejo de información radiográfica. Con base en lo anterior, se presenta el estudio realizado acerca de la conveniencia institucional, la factibilidad técnica y el beneficio económico en el diseño, desarrollo e instalación de un sistema PACS ad-hoc, distribuido a lo largo de todo el Centro Nacional de Rehabilitación, según los requerimientos del área médica, con la finalidad de disminuir la dependencia tecnológica médica, reducir los costos de autoría y mantenimiento, así como disponer de una plataforma de desarrollo que esté en constante actualización de acuerdo a los últimos avances científicos y tecnológicos

La propuesta del desarrollo del proyecto está basada por un lado, en el estándar DICOM, estandarización aceptada mundialmente en el manejo de imágenes médicas, y por el otro en los procedimientos establecidos por Ingeniería de Software. Incluirá un conjunto de librerías para las entidades de aplicación, la implantación de sistemas de compresión de imágenes de última generación, el diseño de bases de datos adaptados al manejo de imágenes médicas y las herramientas de operación específicas a las necesidades del CNR.

La validación del producto generado estará a cargo de médicos radiólogos de la División de Imagenología del CNR, de acuerdo a los criterios establecidos por la American College of Radiology, con relación al manejo de imágenes digitales para su adquisición, calidad presentada en la pantalla, velocidad de transmisión y medios de almacenamiento.

El empleo de este tipo de sistemas traerá como consecuencia un cambio fundamental en el funcionamiento de los departamentos de radiología e imagenología del CNR, mejorando significativamente la eficiencia de los mismos, conjuntamente con una mejoría importante en la calidad de la atención médica que se brinda a los pacientes.

Introduction to Image Acquisition, Archive Managers, and Storege.

En el trabajo presentado por Edgar M. Smith.

Publicado en el libro "Archiving Issues in the Medical Enterprise" Reiner, Bruce I., Siegel, Eliot L., Smith, Edward M. Great Falls, Va. : Society for Computer Applications in Radiology, c 2001 Capitulo 1.

Resumen: Muestra la interrelación de los archivos y otros archivos de imágenes y los componentes de sistema de comunicación para los PACS, así como la configuración de tolerancia a fallos para los archivos mostrando la tecnología "Heart-beat" y la carga balanceada (Sharing).

Por otra parte muestra las características de imágenes típicas y tamaño de estudios por modalidad; describe los componentes de lo archivos así como su manejador y su almacenamiento.

Real-time teleconsulting solution for teleradiology based on PACS.

En el trabajo presentado por Uri Shani, et al

Publicado en Israel en el Laboratorio de investigación de IBM Haifa

http://www.haifa.il.ibm.com/projects/software/idmr/papers/teleradiology.htm

Resumen: El alto nivel de especialización radiológica existe típicamente adentro de centro médico importante, este tipo de especialistas se necesita con frecuencia en los hospitales o las clínicas rurales. La aparición de los PACS (Picture Archiving and Communication Systems) en los último años, ha introducido la posibilidad de teleconsulta en tiempo real en imágenes médicas entre los sitios a distancias grandes.

Al extender PACS sirve apoyar los desafíos de numerosas teleconsultas. Las imágenes médicas pueden ser muy grandes (4 Mb o más), mientras que las manipulaciones en tiempo real de la imagen (tales como zoom o rotación) deben ocurrir casi simultáneamente para que los doctores estudien la información desde cualquier punto al momento de se requiera esta información. Además, la puesta en práctica de PACS funciona en PC's y estaciones de trabajo Unix, con cada

plataforma teniendo un diverso sistema gráfico de ventanas (Windows y X-11, respectivamente), y diverso hardware. Ellos han considerado estas dos posibilidades en el diseño, una misma-plataforma y una solución heterogénea. La solución de la plataforma heterogénea servirá como la base para desarrollar un protocolo estándar para ver la imagen de los vendedores de PAC. La solución de una plataforma igual se ha puesto en ejecución y se ha probado entre dos hospitales en Israel, y se ha demostrado ser eficaz utilizando un angosto ancho de banda. Los doctores en el hospital rural pueden llamar a un experto en el departamento central de radiología del hospital, y discutir el caso en tiempo real. La aparición de la Web Mundial (WWW) alternativas que son discutidos también (ya que este esta pensado como maestro/esclavo).

The Development of a Client/Server Architecture for Standardized Medical Application Network Services.

En el trabajo presentado por Sherif M. Yacoub, et al

Publicado en la Universidad De Virginia Occidental y IEEE

http://csdl.computer.org/comp/proceedings/asset/1999/0122/00/01220002abs.htm

Resumen: En este proyecto, se especificó y diseño el comportamiento del cliente y del servidor del servicio superior de la capa del estándar de DICOM usando un lenguaje orientada objeto. El principio de la Arquitectura del statechart es utilizado para facilitar la comprensibilidad del estándar proporcionando un formalismo visual para la especificación. La especificación distingue el comportamiento de los servicios superiores de la capa como un cliente o servidor. Una alta estructura de cliente/servidor. Finalmente, una lengua orientada objeto del patrón de statechart se utiliza para traducir la especificación a diseño, y un diagrama de la clase para el servidor se presenta en la notación de UML.

Manejo de imágenes DICOM (Digital Imaging and Communications in Medicine) mediante un sistema en Internet.

Es la tesis de Alicia Morales Reyes.

Universidad Nacional Autónoma de México.

México, D.F. 2002

Resumen: La propuesta de este trabajo es el diseño e implementación de un sistema que cumpla con los requerimientos del estándar DICOM para comunicarse con un sistema PACS remoto y realizar tanto búsquedas como despliegues de las imágenes almacenadas en él. Se utilizará la tecnología de Internet, con el fin de tener una sistema cuyos requerimientos mínimos sean un equipo de cómputo con un navegador instalado, donde el usuario indique la dirección del servidor donde está implementado el sistema y tenga acceso a una serie de formularios en los cuales podrá indicar los criterios de búsqueda y recuperar las imágenes que requiera para despliegue en el navegador, así como también para su almacenamiento local. El sistema será compatible con diferentes plataformas como son: UNIX, Linux, Windows NT, etcétera.

Desarrollo de aplicaciones DICOM para la gestión de imágenes biomédicas.

Esta es la tesis de Fernando Ballesteros Herranz.

Universidad Politécnica de Madrid.

Octubre 2003.

Resumen: En esté trabajo, describe un sistema multiplataforma desarrollado para el manejo, procesamiento y auditoría de Imágenes de Diagnóstico Médico a través de redes (Intra o Internet). El sistema está basado en un diseño Cliente/Servidor orientado a objetos, para el cual se utilizó como lenguaje de programación JAVA, y se realizó de acuerdo a la norma DICOM 3.0. Para la realización de la aplicación se ha trabajado con librerías JDT (Java Dicom Toolkit), con las librerías JDK 1.4.1., en el entorno JBuilder Enterprise 7. Fue desarrollado enteramente en JAVA y puede ser utilizable en cualquier sistema operativo.

2 Justificación y Objetivos.

En este punto desarrollaremos por que se justifica este proyecto y cuales son los objetivos que se buscan del mismo.

2.1 Justificación.

En la práctica Clínica los Sistemas PACS aumentan la competitividad del personal de un centro médico. Los médicos ya no estarán más dispuestos a esperar para tener en sus manos la información del paciente para poder hacer una interpretación de una imagen.

El planteamiento de ahorros de tiempo, película, almacenamiento, entre otros se utiliza para la justificación de este sistema. Pero esto llega a ser insignificativo comparado con los cambios en la productividad que puede esperarse. El diagnóstico remoto bien puede ser la fuente de ganancia más importante ya que otros especialistas desde su estado local pueden aportar su experiencia para cada estudio.

Actualmente encontramos en forma comercial Sistemas PACS creados por empresas que producen equipos médicos como AGFA, Siemens, entre otros, los cuales tiene como estándar a DICOM pero son vendidos en alto precio. Por lo que estos sistemas tienen un uso limitado en los Hospitales Nacionales. Con este trabajo se plantearan las bases que permitirá al médico acceder a imágenes médicas para analizarlas y procesarlas, para disponer así de más datos para la toma de decisiones.

El propósito de obtener este Sistema es disminuir la dependencia tecnológica sin necesidad de pagar el costo intelectual del producto generado. Por lo que podrá facilitar las bases para la implementación del Sistema PACS en nuestros Hospitales. Reduciendo sus costos de implementación y mantenimiento.

2.2 Objetivos.

En la Maestría en Ciencias de la Computación se adquirieron los conocimientos, se desarrollaron las habilidades y actitudes para contribuir al entendimiento y la solución de problemas generales y particulares por medio del desarrollo e implementación de sistemas computacionales. Aplicaremos este conocimento en forma integral y creativa, los fundamentos y técnicas de la computación para el desarrollo de esté proyecto.

El Sistema PACS mínimo basado en el estándar DICOM tiene dos tipos de objetivos: a) Objetivos generales; b) Objetivos Particulares. Los cuales describimos a continuación.

Objetivos Generales.

Esta investigación tiene como Objetivo General, realizar el Análisis, Diseño e Implementación de un Sistema PACS basado en el estándar DICOM, con el propósito de obtener un software de calidad que permita disminuir la dependencia tecnológica sin necesidad de pagar el costo intelectual del producto generado.

Objetivos Particulares.

Realizar el Análisis, Diseño e Implementación de:

Un servidor de Base de Datos e Imágenes que sea capas de atender a múltiples clientes y que soporte el protocolo DICOM para atender los servicios solicitados por el cliente.

La Aplicación para Cliente que sea capas de comunicarse con el servidor de Base de Datos e Imágenes bajo el estándar DICOM.

Las Librerías para los distintos tipos de imágenes (codificación para cada tipo de imagen).

El Manejo de la Base de Datos de la información del paciente.

Que el sistema tenga una comunicación segura utilizando técnicas de encriptamiento utilizando el Algoritmo TDES.

Pruebas de Validación.

3 Fundamentos Teóricos.

En el desarrollo de este proyecto es necesario estudiar:

- Unified Modeling Language (UML).
- El estándar Digital Imaging and Communications in Medicine (DICOM).
- El Admisntirado de Base de Datos MySQL.
- Acceso a la Base de Datos utilizando Java Database Connectivity (JDBC).
- Diseño de Base de Datos.
- Criptografía.
- Algoritmo de encriptamiento Data Encryption Standard (DES).
- Algoritmo de encriptamiento Triple DES (TDES).
- Seguridad.
- Secure Socket Layer (SSL).

Para un mejor entendimiento para el desarrollo de este proyecto.

3.1 Unified Modeling Language (UML).

UML define una notación que se expresa como diagramas sirven para representar modelos/subsistemas o partes de ellos

El 80 por ciento de la mayoría de los problemas pueden modelarse usando alrededor del 20 por ciento de UML-- Grady Booch [10].

Aquí presentamos el significado, historial, perspectivas, definiciones de Modelos y Diagramas, Organización de Modelos para UML.

Significado de UML.

UML su significado es "Unified Modeling Language". Es lenguaje de propósito general para el modelado orientado a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales se pueden modelar sistemas.

1) Diagramas de Casos de Uso para modelar los procesos 'business'.
2) Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
3) Diagramas de Colaboración para modelar interacciones entre objetos.
4) Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
5) Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
6) Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
7) Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
8) Diagramas de Componentes para modelar componentes.
9) Diagramas de Implementación para modelar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

Aun así, UML no prescribe un proceso o método estándar para desarrollar un sistema. Hay varias metodologías existentes; entre las más populares se incluyen las siguientes:

- Catalysis: Un método orientado a objetos que fusiona mucho del trabajo reciente en métodos orientados a objetos, y además ofrece técnicas específicas para modelar componentes distribuidos.
- Objetory: Un método de Caso de Uso guiado para el desarrollo, creado por Ivar Jacobson.
- Shlaer/Mellor: El método para diseñar sistemas de tiempo real, puesto en marcha por Sally Shlaer y Steven Mellor en dos libros de 1991, Ciclos de vida de Objetos, modelando el Mundo en Estados y Ciclos de vida de Objetos, Modelando el mundo en Datos (Prentice Hall). Shlaer/Mellor continúan actualizando su método continuamente (la actualización más reciente es el OOA96 report), y recientemente publicaron una guía sobre cómo usar la notación UML con Shlaer/Mellor.
- Fusión: Desarrollado en Hewlett Packard a mediados de los noventa como primer intento de un método de diseño orientado a objetos estándar. Combina OMT y Booch con tarjetas CRC y métodos formales. (www.hpl.hp.com/fusion/file/teameps.pdf)
- OMT: La Técnica de Modelado de Objetos fue desarrollada por James Rumbaugh y otros, y publicada en el libro de gran influencia "Diseño y Modelado Orientado a Objetos" (Prentice Hall, 1991). Un método que propone análisis y diseño 'iterative', más centrado en el lado del análisis.
- Booch: Parecido al OMT, y también muy popular, la primera y segunda edición de "Diseño Orientado a Objetos, con Aplicaciones" (Benjamin Cummings, 1991 y 1994), (Object-Oriented Design, With Applications), detallan un método ofreciendo también diseño y análisis 'iterative', centrándose en el lado del diseño.

Además, muchas organizaciones han desarrollado sus propias metodologías internas, usando diferentes diagramas y técnicas con orígenes varios. Ejemplos son el método Catalyst por Computer Sciences Corporation (CSC) o el Worlwide Solution Design and Delivery Method (WSDDM) por IBM. Estas metodologías difieren, pero generalmente combinan análisis de flujo de trabajo, captura de los requisitos, y modelado de negocio con modelado de datos, con modelado de objetos usando varias notaciones (OMT, Booch, etc), y algunas veces incluyendo técnicas adicionales de modelado de objetos como Casos de Uso y tarjetas CRC. La mayoría de estas organizaciones están adoptando e incorporando el UML como la notación orientada a objetos de sus metodologías.

Algunos modeladores usarán un subconjunto de UML para modelar 'what they're after', por ejemplo simplemente el diagrama de clases, o solo los diagramas de clases y de secuencia con Casos de Uso. Otros usarán una suite más completa, incluyendo los diagramas de estado y actividad para modelar sistemas de tiempo real, y el diagrama de implementación para modelar sistemas distribuidos. Aun así, otros no estarán satisfechos con los diagramas ofrecidos por UML, y necesitarán extender UML con otros diagramas como modelos relacionales de datos y 'CRC cards'.

Historia del UML

Comenzó como el "Método Unificado", con la participación de Grady Booch y Jim Rumbaugh. En 1995 se unió Ivar Jacobson. Los "Tres Amigos" son socios en la compañía Rational Software. Herramienta CASE Rational Rose. La figura 3.1 muestra los enfoque de programación Orientada a Objetos.

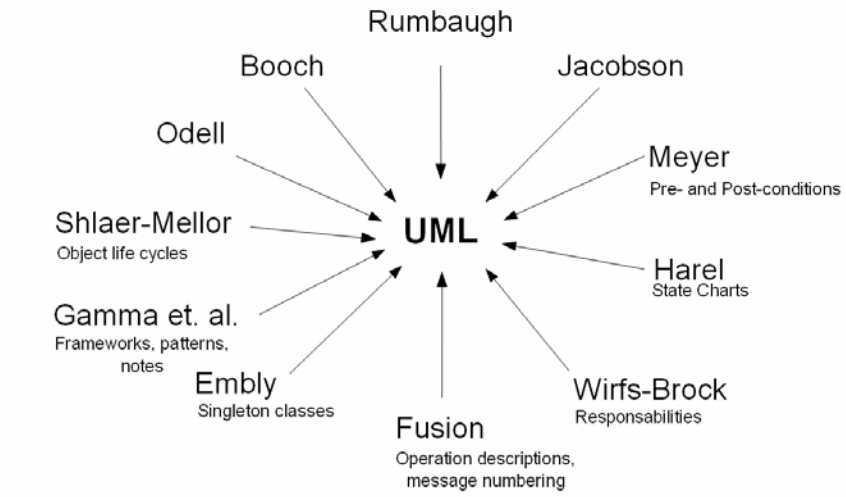


Figura 3.1 UML "Aglutina" enfoques de programación Orientada a Objetos.

En enero de 1997 es lanzado UML 1.0 participando: Rational Software (Grady Booch, Jim Rumbaugh y Ivar Jacobson), Digital Equipment , Hewlett-Packard, i-Logix (David Harel), IBM, ICON Computing (Desmond D'Souza), Intellicorp and James Martin & co. (James Odell), MCI Systemhouse, Microsoft, ObjecTime, Oracle Corp., Platinium Technology, Sterling Software, Taskon, Texas Instruments, Unisys (ver figura 3.1).

La Figura 3.2 se muestra como fue evolucionando UML desde 1995 hasta el 2003.

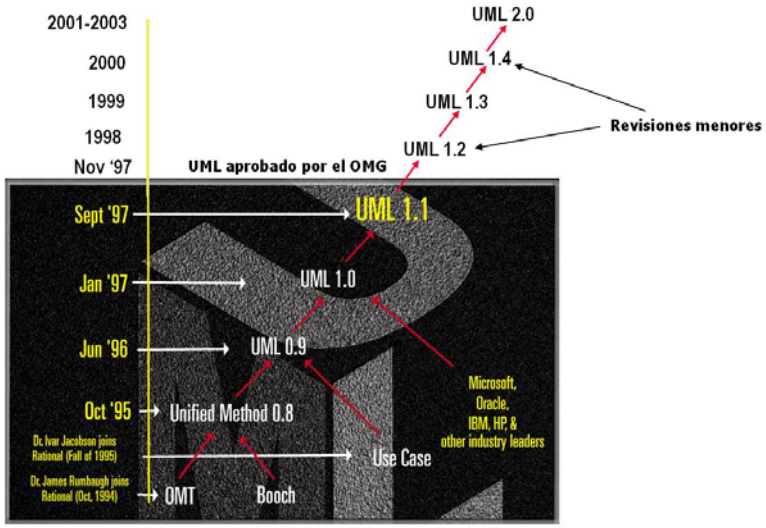


Figura 3.2 Historia de UML.

Perspectivas de UML.

UML será el lenguaje de modelado orientado a objetos estándar predominante los próximos años.

Razones:

- Participación de metodologías influyentes.
- Participación de importantes empresas.
- Aceptación del OMG como notación estándar.

Evidencias:

- Herramientas que proveen la notación UML.
- "Edición" de libros.
- Congresos, cursos, etc.
- UML aglutina Enfoque de programación Orientado a Objetos.

Definición de modelos y diagramas.

Un modelo captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.

El Diagrama es una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos.

Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés.

El código fuente del sistema es el modelo más detallado del sistema (y además es ejecutable). Sin embargo, se requieren otros modelos.

Cada modelo es completo desde su punto de vista del sistema, sin embargo, existen relaciones de trazabilidad entre los diferentes modelos.

Diagramas de UML.

Los Diagramas que contempla son: Diagrama de Casos de Uso, Diagrama de Clases, Diagrama de Objetos, Diagramas de Comportamiento, Diagrama de Estados, Diagrama de Actividad, Diagramas de Interacción, Diagrama de Secuencia , Diagrama de Colaboración, Diagramas de implementación, Diagrama de Componentes, Diagrama de Despliegue, Los diagramas expresan gráficamente partes de un modelo.

En la figura 3.3 muestra que los diagramas expresan gráficamente partes de un modelo.

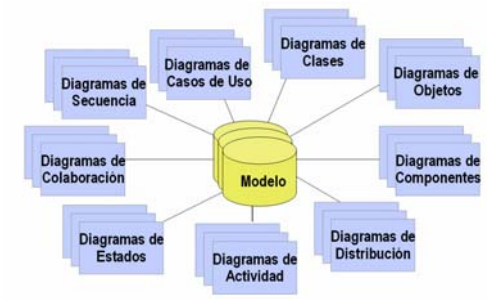


Figura 3.3 Un modelo se puede expresar por medio de diagramas.

Propuesta de Rational Unified Process (RUP).

Esta propuesta contiene los siguientes modelos:

- Casos de Uso del Negocio (Business Use-Case Model).
- Objetos del Negocio (Business Object Model).
- Casos de Uso (Use-Case Model).
- Análisis (Analysis Model).
- Diseño (Design Model).
- Despliegue (Deployment Model).
- Datos (Data Model).
- Implementación (Implementation Model).
- Pruebas (Test Model).

Dado que es necesario modelar el sistema se busco un software. Por lo que ArgoUML es la herramienta que utilizamos para el diseño bajo UML. A continuación mostramos la información de ArgoUML.

ArgoUML .

ArgoUML es una herramienta utilizada en el modelado de sistemas, mediante la cual se realizan diseños en UML ("Unified Markup Language") llevados acabo en el análisis y prediseño de Sistemas de Software. En la Figura 3.4 se muestra como es la pantalla de este software.

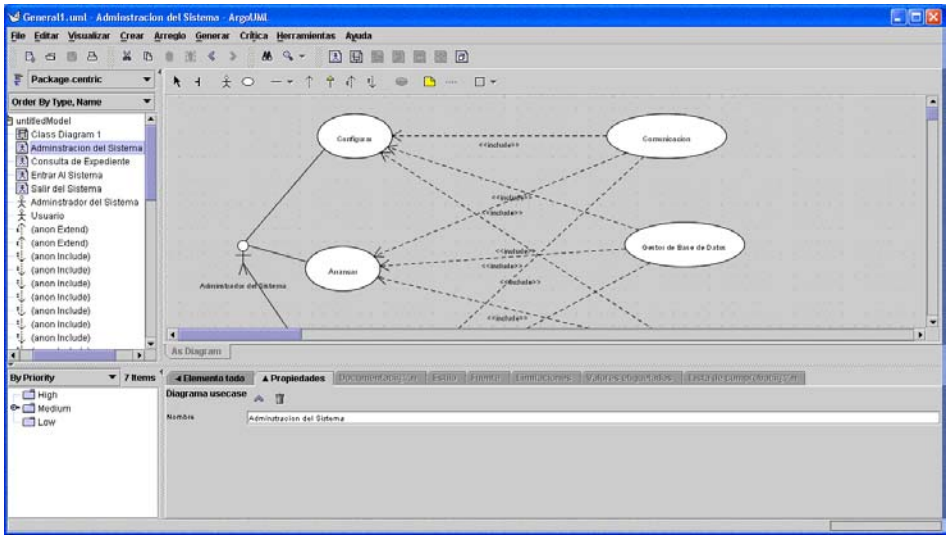


Figura 3.4 Presentación de Argouml.

Lista de características.

Las características de Argouml son:

- Multiplataforma (Java 1.2).
- Metamodelo de UML estándar.
- Clase.
- Estado.
- Casos de uso.
- Actividad.
- Colaboración.
- Despliegue.
- Secuencia.
- Soporte para bases de datos.
- Exporta los diagramas a distintos formatos.
- Generación de código (parcial).

Soporte cognitivo:

- Preactivo (criticas de diseñó, listas de cosas por hacer, correcciones automáticas).
- Comprensión y solución del problema.

Rational Rose, Poseidon vs. Argouml

En el ramo para modelado de Software existen diversas herramientas que también son capaces de realizar diseños en UML, entre las principales se encuentran Rational Rose y Together, e inclusive existe un derivado comercial de ArgoUML llamado Poseidon.

Esta última herramienta llamada Poseidon se encuentra disponible en diversas versiones, la cual en su modalidad más básica (Community) es gratuita, y dado que se encuentra basada en ArgoUML, presenta sus mismas funcionalidades con opciones de compra para usuarios que requieran funciones avanzadas no disponibles en ArgoUML.

En lo que concierne las otras opciones comerciales -- antes mencionadas -- Éstas por lo general son empleadas en proyectos de misión critica, no sólo por las funcionalidades avanzadas que ofrecen en las etapas de diseñó y modelación de software, sino por su costo de licenciamiento que puede oscilar desde los $1,000 Dlls (U.S) hasta $4,500 Dlls (U.S) por licencia/usuario.

3.2 El estándar Digital Imaging and Communications in Medicine (DICOM).

Se presenta una breve historia del Estándar DICOM, sus alcances, el modelo de protocolo de comunicación, la estructura del estándar, Aplicaciones distribuidas DICOM para la implementación en modalidades.

Historia.

En 1983 la ACR y la NEMA formaron un comité de trabajo con el fin de desarrollar un estándar que permitiera a los usuarios de equipos de imágenes médicas (como son Tomografías Computarizadas, Imágenes de resonancia magnética, Medicina nuclear, etc.) Conectar diferentes dispositivos para compartir recursos, como visualizadores, impresoras, etc. El comité encargado de buscar una interfaz entre equipos de imágenes médicas se llamo ACR-NEMA Digital Imaging and comunications Estandars Comité. Las especificaciones deberían de incluir un diccionario de datos el cual se especificaran los elementos necesarios para visualizar interpretar y almacenar las imágenes médicas, así como las características del hardware de estos sistemas.

Este comité estudio los estándares de interfaz existentes hasta esa fecha sin encontrar alguien que cumpliera los requerimientos. Un Año antes la AAPM (American Association of Phisicists in Medicine) había desarrollado un estándar para el almacenamiento de Imágenes en cintas magnéticas. Este estándar definía los archivos formados por cabecera (head) y matriz de datos. En la Cabecera se usaba el concepto de <etiqueta> <descripción>. Estos conceptos fueron aprovechados para definir las bases del estándar DICOM[21].

En el congreso anual RSNA 1985 fue publicada la primera versión del estándar por NEMA. Se llamó ACR-NEMA Versión 1.0. dado que esta versión tenía una serie de errores se tuvo que estudiar las propuestas de usuarios y fabricantes para crear la versión 2.0. En 1988 el ACR-NEMA Versión 2.0 (también llamado ACR-NEMA 300-1988) fue publicado conservando las especificaciones de hardware de V1.0 pero añadiendo nuevos datos y mejoras.

La evolución de la tecnología plantea la necesidad de comunicarse diferentes dispositivos la necesidad de los usuarios de utilizar las redes, por lo que se tuvo que rediseñar todo el estándar y el método adoptado fue el diseño orientado a objetos.

Un exhausto examen de los servicios que debería de ofrecer el estándar para la comunicación sobre diferentes redes mostraba que la definición de estos servicios básicas deberían definirse para permitir la comunicación entre proceso (al nivel de aplicación). El grupo de trabajo seleccionó como protocolo TCP/IP de mas bajo nivel sobre el que se asentaría el nuevo estándar y como modelo de diseño el ISO-OSI.

El estándar esta en la versión 3 que fue publicada en 1998. Hoy en día, DICOM esta centrando su atención a la evolución de los estándares vinculados a Internet. Su estrategia es integrar sus recomendaciones tan pronto como sea estables y ampliamente adoptadas por productos comerciales.

Por lo tanto, DICOM esta en continuo cambio y expansión, añadiendo nuevos equipos y/o servicios o debatiendo en alguno de sus grupos de trabajo sobre la definición de nuevos objetos y servicios.

Alcances de DICOM.

El contenido del estándar DICOM va más allá de la definición del formato de intercambio de imágenes, sus alcances principales son:

- Estructuras de datos (formatos) para imágenes médicas y datos relacionados.
- Servicios orientados a red, como: Transmisión de imágenes, búsqueda de imágenes, impresión y modalidades de integración entre un sistema PACS y un sistema general de información de un hospital (HIS o RIS).

- Formatos para intercambio entre medios de almacenamiento.
- Requerimientos de conformidad de los equipos y aplicaciones.

El modelo de protocolo de comunicación de DICOM.

El protocolo se modela como una serie de capas de forma independiente pero compatibles entre si, llamadas "stacks". En la Versión existente 2.0 el "stacks" definido en una conexión punto a punto era uno. Y posteriormente se agregaron dos mas: El Protocolo de Control de Transmisión / Internet de Protocolo (TCP / IP) y la Organización Internacional de Estándares de Interconexión de Sistemas (ISO-OSI).

La figura 3.5. muestra el modelo de comunicación que fue desarrollado. Este modelo tiene las siguientes capas: aplicación de imágenes médicas; capa de intercambio de mensajes de aplicación DICOM; posteriormente se divide en dos ambientes: comunicación en red (soporte de protocolo TCP/IP y OSI) y comunicación punto a punto (capas física, de enlace y SNT (sesión, transporte y red) de DICOM). La filosofía básica de diseño era que una aplicación de imagen médica determinada (fuera del alcance del estándar) podría comunicar sobre cualquier de los stacks de otro dispositivo que use la misma stack. Con la adherencia al estándar, sería posible cambiar las comunicaciones de stacks sin tener que revisar los programas de computadora de la aplicación.

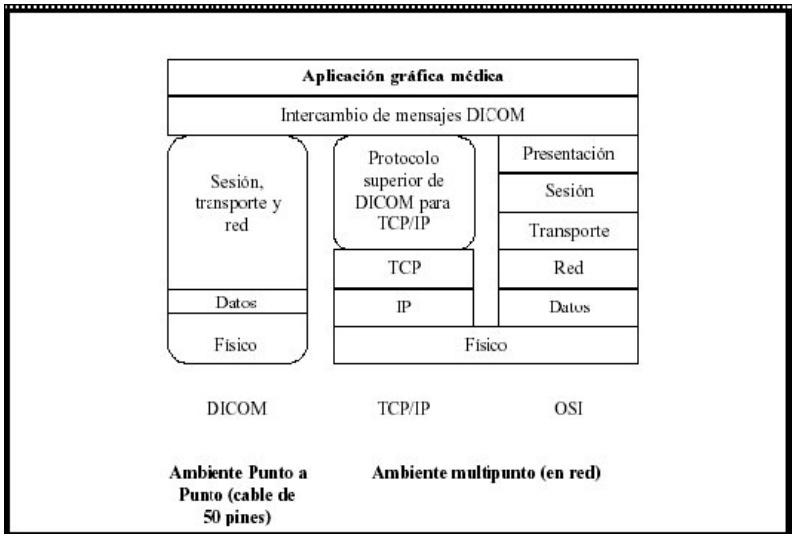


Figura 3.5 Modelo de protocolo de comunicación DICOM.

Estructura.

El estándar esta estructurado como un documento en multipartes usando como guía el Estándares Internacionales Dibujo y Presentación (ISO/IEC Directives, 1989 Part 3: Drafting and Presentation of International Standards).

En su revisión de 2004 el Estándar DICOM versión 3.1 tiene las siguientes partes:

- PS 3.1. Introducción y vista general: La primera parte contiene una panorámica del estándar en si mismo, con descripción de los principios básicos, como pueden ser los conceptos de SOP Class o Service Class Provider.
- PS 3.2. Conformación: Describe la definición de conformación para DICOM, es decir se le solicita a los desarrolladores y vendedores de equipos y sistemas describir claramente como es su adhesión al estándar DICOM.

- PS 3.3. Definición de los Objetos de Información (IOD): Especifica la estructura y atributos de los objetos que se operan por Clases de Servicio. Estos objetos pueden ser Paciente, Estudio, Serie, Imagen, etc. Cada definición de IOD esta agrupada en módulos. Algunos IOD pueden tener grupos de atributos idénticos, que se definen en módulos comunes. Estos objetos compuestos son por ejemplo: la imagen de TC o RMN que contienen atributos inherentes a la misma entidad del mundo real y otros que no lo son.

- PS 3.4.Especificación de las clases de servicios: Se define las funciones que operan sobre los Objetos de información para proporcionar un servicio específico. La especificación de las Clases de Servicios (SOP Class) se base en las operaciones que deben actuar sobre los IOD. Tales SOP Class son.- certificación, memorización, petición/consulta de imágenes e información, contenida en el estudio, administración del paciente, administración del examen, administración del parte médico, administración de la documentación. Cuando una aplicación DICOM genera una serie de datos, esta tiene que ser decodificada para poder ser insertada en los mensajes de comunicación.

- PS 3.5.Estructura de datos y codificación: Especifica la codificación de los datos en los mensajes que se intercambian para lograr el funcionamiento de las Clases de Servicio. La principal función es definir el lenguaje que dos aparatos tiene que utilizar en la comunicación.

- PS 3.6.Diccionario de datos: Define los atributos de información individuales que representan los datos de todos los IOD definidos en la parte 3.3. También se especifica los valores de algunos de estos atributos.

- PS 3.7.Intercambio de Mensajes: Especifica el funcionamiento de los mensajes a intercambiar. Estos mensajes se necesitan para poder utilizar los servicios definidos por las Clases de Servicio (parte 3.4).

- PS 3.8.Soporte de las comunicaciones por red para el intercambio de mensajes: Define los servicios y protocolo de intercambio de mensajes (Parte 3.7) directamente en OSI y redes TCP/IP. En el entorno DICOM, el protocolo de comunicación utilizado es el TCP/IP.

- PS 3.9. Retirado

- PS 3.10.Medios de almacenamiento y formato de archivos para intercambio de datos: Define los formatos lógicos para guardar la información de DICOM sobre varios medios de comunicación, entre ello los archivos tratados.

- PS 3.11.Perfiles de aplicación para medios de almacenamiento: Define los medios para usuario y vendedores, especificando la sección de medios de comunicación entre los sistemas definidos en la Parte 3.12 y los objetos de información definidos en la parte 3.3

- PS 3.12.Formatos y medios físicos para el intercambio de datos: Las especificaciones de la industria referentes a los medios Físicos de Almacenamiento o medios de Comunicación que estructuran los sistemas de archivos.

- PS 3.13. Retirado

- PS 3.14.Estándar para la función de representación de escala de grises: En está parte se especifica la estandarización de las características de los monitores para la representación en la escala de grises de las imágenes.

- PS 3.15. Perfiles de Seguridad: Perfiles de usuario en aplicaciones.

- PS 3.16. Recurso para el Mapeo de Contenido: Templetes para estructuración de documentación de objetos de información DICOM; conjunto de términos codificados para uso en objetos de información; un léxico de definición de términos para DICOM; traducción de especificaciones del país por medio de código

- PS 3.17. Interpretación de la información: anexos de información y normativa.

- PS 3.18. Acceso WEB de objeto persistente DICOM (WADO): Especifica la solicitud para el acceso de un objeto persistente DICOM que puede ser expresada en un HTTP URL.

Aplicaciones distribuidas DICOM para la implementación en modalidades[43].

El modelo de procesamiento distribuido es utilizado para explicar el mecanismo y la terminología del estandar DICOM.

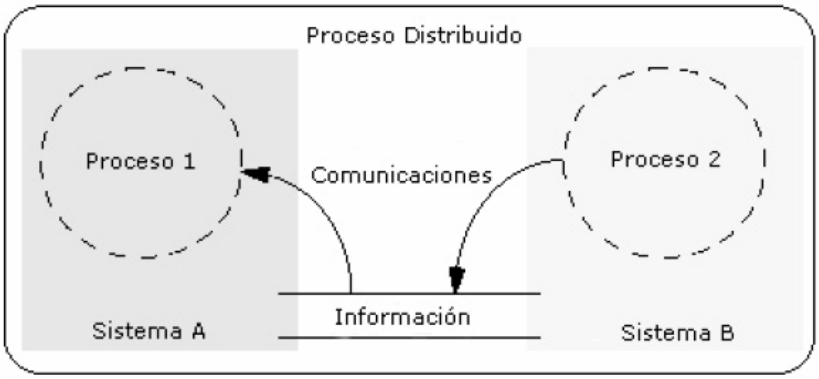


Figura 3.6 Modelo simple de procesamiento distribuido.

Un proceso distribuido comparten información dos (o más) procesos y cada uno realiza su propio trabajo. Pero al mismo tiempo depende de la funcionalidad del otro. Ver figura 3.6.

Varios procesos distribuidos actuando en conjunto proporciona servicios (adquisición, almacenamiento, despliegue de imagen). Por lo general en el escenario de un proceso distribuido los procesos de aplicación son independientes a los procesos de comunicación encargados de la transmisión de datos entre sistemas y compensa a los valores que son internamente representados en diferentes sistemas. Ver figura 3.7.

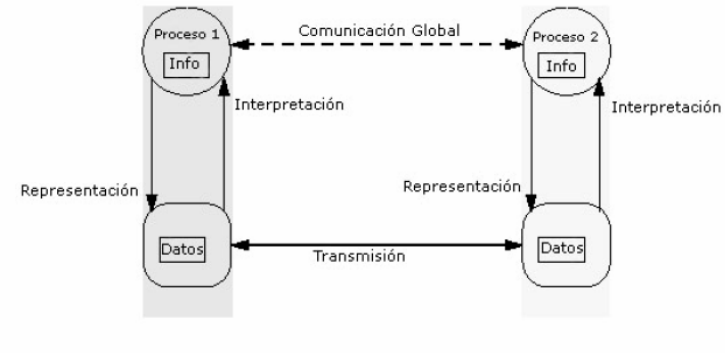


Figura 3.7 Diagrama de operación de un proceso distribuido.

Para que dos procesos puedan actuar en conjunto es necesario primero definir el rol de cada uno de ellos, teniendo una visión de la información y operaciones que cada proceso ejecutara. Por lo que el rol de cada entidad debe definirse como cliente o servidor. Las expectativas que estas entidades tienen entre sí están definidas por la relación compartida. La relación define que entidad y bajo que condiciones tomará la iniciativa en el proceso. En la mayoría de los instancias el Cliente arranca el proceso, pero algunas veces el Servidor es la parte que inicia. Ver figura 3.8.

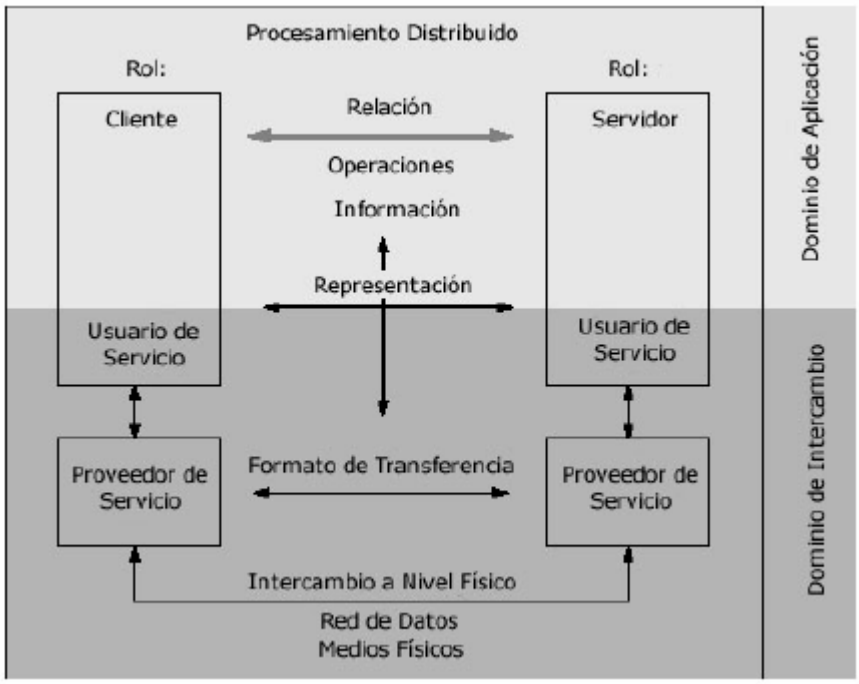


Figura 3.8 Diagrama a bloques del procesamiento distribuido.

Además de los roles, ambas entidades deben acordar el tipo de información a intercambiar. Considerando la semántica de la información y no su forma de representación (sintaxis). La información está definida por el contexto del servicio en que se implementa el proceso distribuido. Cada proceso individual debe tener una visión selectiva y a la vez consistente del contexto completo de la información.

Las operaciones se definen como la información intercambiada se debe de procesar en el otro lado, tal como almacenamiento de información, regreso de resultados, etc.

La combinación del contexto, relación, operaciones e información es lo más importante en el procesamiento distribuido y debe determinarse antes para lograr una implementación exitosa. Estos aspectos son parte del dominio de aplicación de los procesos distribuidos, no intervienen en la forma en que la información se intercambia, pero dependen de los servicios de bajo nivel (por ejemplo TCP/IP) provistos en el dominio de intercambio.

Tanto el cliente como el servidor deben ser capaces de hacer solicitudes a servicios de bajo nivel. Estos manejan el intercambio de datos y son ocultos para la parte del dominio de aplicación. La entidad que solicita el servicio de bajo nivel es el Usuario del Servicio, quien da respuesta es el Proveedor del Servicio. Ambos pueden tener diferentes implementaciones, pero comparten el mismo conocimiento acerca del intercambio de datos (protocolos) y tienen la misma interfaz lógica (formato de solicitud).

Ambas partes deben determinar cual será la representación de la información (formatos). El Proveedor del Servicio debe determinar el formato de la información a transmitir y convertirla a la representación esperada por el dominio de aplicación. La representación se conoce por el Usuario y Proveedor del Servicio en cada lado. Después de realizarse el intercambio, la información de estado debe ser la misma para las dos entidades.

El intercambio a nivel físico entre los Proveedores del Servicio puede ser a través de una red de datos o medios físicos. Cada mecanismo tiene su propia forma de manejar la representación de la información.

A continuación desarrollaresmos los siguientes puntos:

- El procesamiento distribuido y DICOM.
- Modelo de Información de imágenes DICOM.
- Clasificación de datos de la imagen.
- Tipos de imágenes.

El procesamiento distribuido y DICOM.

DICOM utiliza su propia terminología para describir los conceptos del procesamiento distribuido. En esta sección se explicará una forma de adecuar el modelo de procesamiento distribuido, aplicando los términos equivalentes en DICOM. Para ello describimos los siguientes puntos:

- Clase de Servicio y Clase de Servicio Par-Objeto.
- Definiciones de Objeto de Información (IOD).
- Atributos.
- Elementos de Servicio.
- Clase SOP.
- Identificación.
- Relaciones.
- Representación del Valor.
- Sintaxis de Transferencia.
- Flujo de codificación y decodificación.
- Conceptos de red.
- Entidad de Aplicación.
- Dirección de Presentación.
- Negociación de Asociación.
- Contexto de Presentación.
- Protocolos de Red.
- Protocolo TCP/IP.

Clase de Servicio y Clase de Servicio Par-Objeto.

La *clase de servicio* describe los roles que cada entidad desempeña y define el contexto. En DICOM cada entidad se nombra como: Usuario de Clase de Servicio (SCU) y Proveedor de Clase de Servicio (SCP), ver figura 3.9.

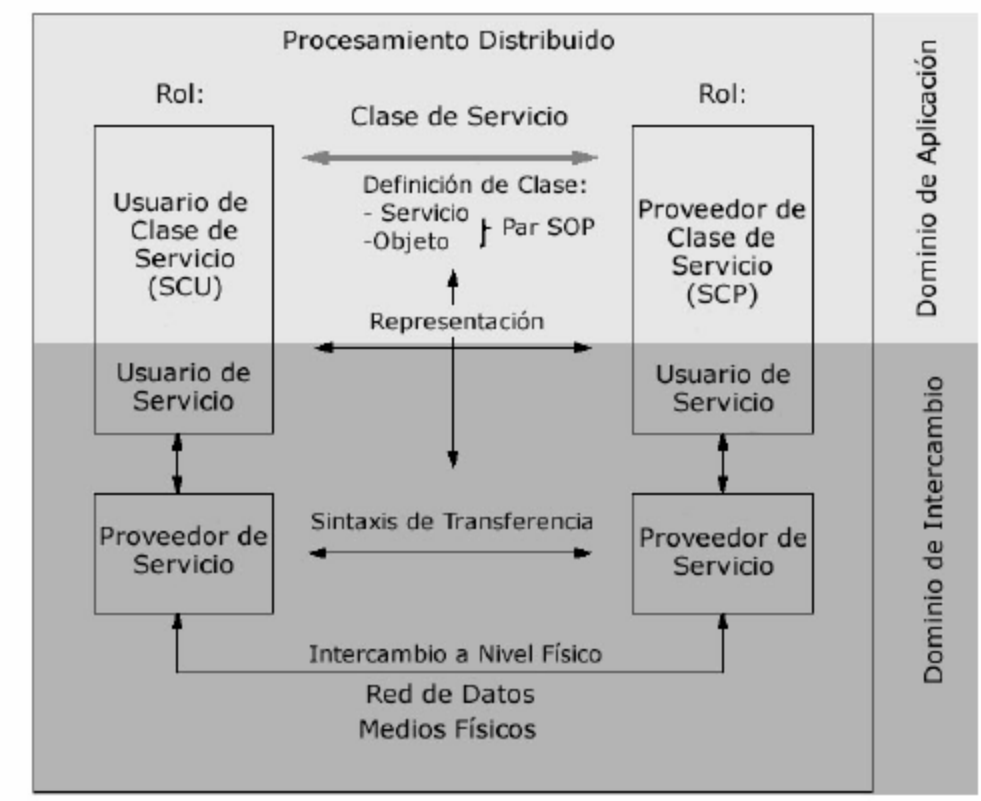


Figura 3.9 Diagrama de conceptos DICOM adecuado al procesamiento distribuido.

Las Clases de Servicio establecen la descripción de la información y las operaciones. En el estándar DICOM las Clases de Servicio se combinan con la definición de Clase (orientada a objetos), llamada Clase de Servicio Par-Objeto (Clase SOP). En cada definición de Clase SOP se combina una sola Definición de Objeto de Información (IOD) con uno o más servicios. Para cada uno de estos servicios, se deben definir con detalle el rol que cada entidad desempeña. En una Clase de Servicio pueden existir más de una Clase SOP.

La Clase SOP identifica las capacidades de una Clase de Servicio en un proceso distribuido especifico. Cuando ambas partes están de acuerdo en utilizar una Clase SOP, deberán asegurar que su desempeño será dentro del contexto limitado por la Clase de Servicio. Antes del intercambio de información debe realizarse la identificación de la Clase SOP. El mecanismo utilizado depende del tipo de intercambio: redes de datos o medios físicos.

Mediante las Clases de servicio, las partes que operan en un ambiente de procesamiento distribuido, lo hacen en conjunto por medio de los servicios provistos por el dominio de intercambio.

Definiciones de Objeto de Información (IOD).

La información de una Clase SOP se especifica en una Definición de Objeto de Información (IOD), que es una colección de piezas de información relacionadas, que se agrupan en Entidades de Información (IE). Cada Entidad contiene datos referentes a una sola característica, por ejemplo, un paciente, un estudio, datos de imagen, etcétera. Dependiendo del contexto definido por la Clase de Servicio un IOD puede ser una sola Entidad de Información (IOD Normalizada) o una combinación de Entidades de Información (IOD Compuesta) Las Clases de Servicio que implementan funciones de administración utilizan IODs normalizados, aquellas que manejan el flujo de datos (imágenes) utilizan IODs Compuestos. La Figura 3.10. muestra las relación de IODs y atributos.

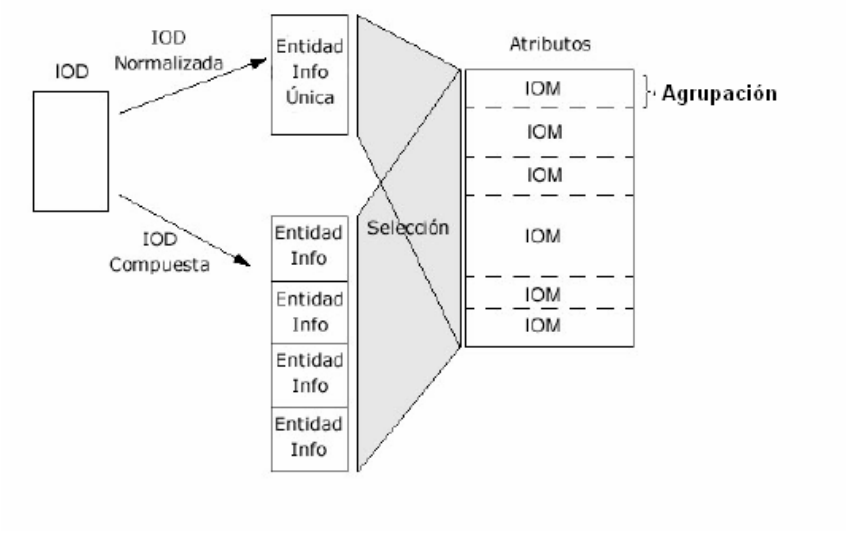


Figura 3.10 Relación IODs y atributos.

La relación entre diferentes Entidades de Información (estructura) de un IOD compuesto se describe mediante un Modelo de Información, mientras que para un IOD normalizado no hay necesidad de estructurar.

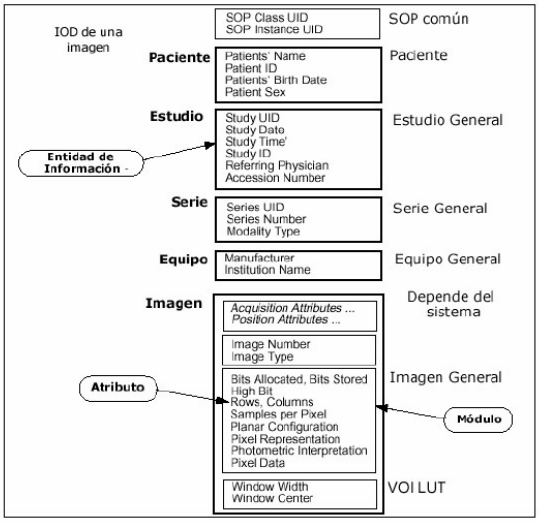


Figura 3.11 Definición de Objeto de Información Compuesto (Imagen).

Las Entidades de Información poseen Atributos descritos en una sola pieza de información, por ejemplo, el nombre de un paciente. Los Atributos que tengan relación entre sí, son agrupados en Módulos de Objeto de Información (IOM). Los IOM están definidos de tal manera que

pueden ser utilizados en más de un IOD. La figura 3.11. muestra un ejemplo de un IOD compuesto de una imagen.

Atributos.

Atributos son las entidades de información básicas y deben ser descritas a detalle. Las características definidas por DICOM para un Atributo son las siguientes:

- Nombre del Atributo único.
- Etiqueta del Atributo única.
- Descripción del Atributo (semántica).
- Representación del Valor (sintaxis).
- Tipo de Clasificación: 1, 1C, 2, 2C o 3 (su uso depende del contexto de la Clase SOP, el rol dentro de la Clase de Servicio, etcétera).

El tipo de clasificación especifica el uso del Atributo relacionado con la Clase SOP y el rol como SCU o SCP. Dependiendo de la situación, cada atributo debe tener un valor forzado (tipo 1), con o sin valor (tipo 2) u opcional (tipo 3).

Dentro de un IOD, los Atributos agrupados o individuales pueden ser condicionados a los usos del mismo. Por ejemplo, para un estudio de imagenología donde se utiliza contraste, puede tenerse un atributo especial para esta característica y estar disponible o no para el módulo dependiendo de su uso.

Elementos de Servicio.

Los elementos de servicio son las operaciones permitidas sobre los IODs para una cierta Clase SOP.

El grupo de elementos de servicio que pertenecen a una Clase SOP se llama grupo de servicio. El grupo de servicio de una Clase SOP se selecciona de una lista fija de elementos de servicio DICOM. Algunos de estos elementos de servicio se destinan para uso con IODs compuestas o normalizadas. Una tercera categoría, son los elementos de servicio relacionados con medios de almacenaje, que manejan instancias de Clases SOP compuestas y normalizadas en archivos.

El contexto descrito por la Clase de Servicio está limitado cuando se utilizan IODs compuestas (transferencia de imagen). Donde los elementos de servicio tienen un significado complejo (STORE, FIND, MOVE). En contraste, las Clases de Servicio que utilizan IODs normalizadas tienen un contexto más amplio (funciones de administración). Emplean Elementos de Servicio primitivos para operaciones con piezas simples de información:GET, SET, ACTION, etc.

Cada Clase SOP utiliza uno o más Elementos de Servicio de cualquiera de los dos tipos, del grupo compuesto (C-XXXX) o del grupo normalizado (N-XXXX). Se dispone de los siguientes elementos de servicio: C-STORE, C-FIND, C-MOVE, C-GET, C-CANCEL, C-ECHO, N-GET, N-SET, N-ACTION, N-CREATE, N-DELETE, N-EVENT-REPORT. La semántica de los elementos de servicio depende de la clase de servicio y la Clase SOP que los utiliza.

Los Elementos de Servicio relacionados con medios de almacenaje son: M-WRITE, M-READ, M-DELETE, M-INQUIRE-FILE-SET, M-INQUIRE-FILE que definen funciones primitivas para la manipulación de conjuntos de archivos.

Clase SOP.

Después de acordar las Clases SOP a utilizar (e implícitamente la Clase de Servicio) y como se dividirón los roles (SCU o SCP), se generan los Instancias SOP mediante los Atributos con los valores correctos, posteriormente se puede realizar el intercambio de Instancias SOP entre ambas entidades. La codificación de la información se hace en los formatos definidos por DICOM, utilizando una Etiqueta (*Tag*) y la Representación del Valor (VR) para crear un

Conjunto de Datos DICOM (*Data Set*), en los cuales cada Atributo se codifica en un Elemento de Dato (*Data Element*). El Proveedor del Servicio administra al Conjunto de Datos, asegurando que la otra entidad reciba un Conjunto de Datos igual. Las diferencias en los sistemas referentes a la representación específica, se toman en cuenta durante el intercambio, asegurando que los valores semánticos queden intactos. El receptor decodificará el Conjunto de Datos para extraer la información que necesita y actuar de acuerdo a la semántica de la Clase SOP.

Identificación.

Como parte del proceso de creación de un Clase SOP, se genera una identificación considerada como un Atributo del mismo. La identificación se usa en los sistemas de información y tiene dos características: Identificación de Clase (*Class Identification*) e Identificación de Instancia (*Instance Identification*).

Para asegurar que es única la identificación se utiliza un mecanismo para generar cadenas de caracteres, llamado Identificador Único (UID), con el siguiente formato: <raíz>.<sufijo> .La raíz es la parte que asigna una autoridad (las organizaciones de estándares a empresas y hospitales) que garantiza que nadie más utilizará esta raíz, asegurando que es único. El sufijo lo genera dinámicamente el sistema durante la creación de una Instancia SOP. Si se hacen copias de un Instancia SOP sin ninguna modificación, deberán tener el mismo UID, de otra forma podrían existir dos piezas con la misma información y distintas identificaciones, lo que puede traer confusiones.

Relaciones.

Además de la identificación de la Clase SOP y el instancia SOP, los UIDs también se utilizan para identificar la relación entre instancias. Por ejemplo, para un instancia SOP Compuesto de una imagen que pertenece a una serie de imágenes, la Entidad de Información que posee los datos de la serie, es común a todas las imágenes de esa serie. La relación se define por el uso del mismo UID que identifica a la serie. Para un instancia SOP Normalizado solamente se pueden hacer referencias externas y se requiere de la combinación de identificadores de clase e instancia. Con el uso de UIDs es posible comparar instancias iguales. El valor del UID no tiene significado y no puede usarse para ordenar, etc. Al utilizar otros atributos con significado como la fecha y hora, es posible establecer relaciones entre la información.

Representación del Valor.

Para cada atributo se define una Representación del Valor (VR), define la codificación del Atributo en un Elemento de Dato y se comparte por ambas entidades durante el intercambio de información. El proceso de codificación y decodificación de la información debe ser cuidadoso al seleccionar la Representación del Valor correcta para un Atributo.

Existen dos formas de compartir esta información: mediante un Diccionario de Datos que contiene todos los posibles atributos a intercambiar o incluyendo la Representación del Valor como parte del Elemento de Dato. La última opción incrementa por mucho el intercambio de información, pero es mucho más flexible en comparación al uso de un Diccionario de Datos compartido, sobre todo cuando se trata de sincronizarlo dentro de un ambiente multi-fabricante.

Cuando se incluye la Representación del Valor, la información se codifica como "explícita" (*Explicit* VR), de lo contrario la codificación será "implícita" (*Implicit* VR).

Sintaxis de Transferencia.

Después de que un instancia SOP se codifica en un Conjunto de Datos para su intercambio, se debe definir la forma en que estos datos serán codificados en una cadena de *bytes*. La forma de codificar se especifica en la Sintaxis de Transferencia.

La Sintaxis de Transferencia debe definir tres aspectos:

- Especificación de la Representación del Valor (VR).
- Ordenamiento en *bytes* (*Little Endian* o *Big Endian*).
- En instancia de compresión, definir el formato.

El manejo de la Sintaxis de Transferencia lo hace el Proveedor del Servicio. Sin embargo, los procesos deben establecer el contexto para una correcta Sintaxis de Transferencia aceptable para ambos. Análogamente a la identificación de la Clase SOP, la Sintaxis de Transferencia debe identificarse por un UID.

Flujo de codificación y decodificación.

El proceso de codificación y decodificación tiene dos estados: Primero, codifica la representación interna al formato definido por DICOM (Conjunto de Datos), donde cada atributo se almacena de acuerdo a la Representación del Valor. Segundo, codifica el conjunto de datos a una cadena de *bytes* la cual puede ser manejada por las capas más bajas. Para esta segunda etapa, la cadena de *bytes* debe ser utilizada de acuerdo a la Sintaxis de Transferencia. Para la decodificación, los estados que se siguen son los mismos en orden invertido. La aplicación que este utilizando la información debe conocer el significado de los datos dentro de los objetos de información. Ver figura 3.12.

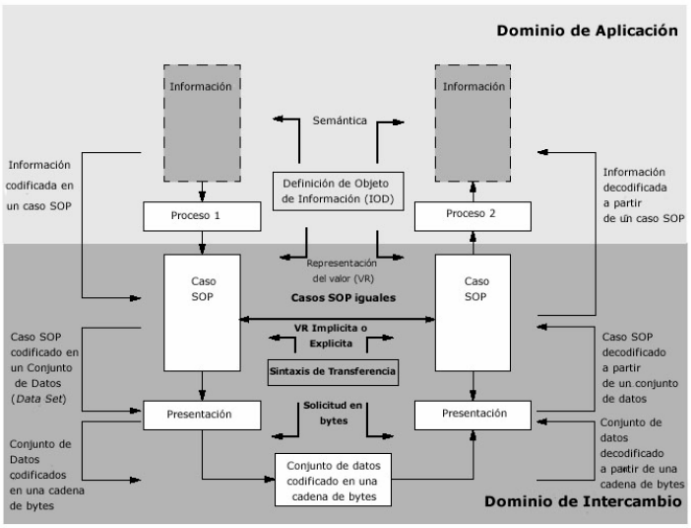


Figura 3.12 Codificación y decodificación de Instancias SOP.

Conceptos de red.

Para las aplicaciones distribuidas en red se deben establecer algunas condiciones (dominio de comunicación) para lograr la comunicación entre sí y poder dirigirse al proceso homólogo, además de acordar sobre varias cuestiones antes de poder intercambiar los objetos DICOM.

Entidad de Aplicación.

En las redes DICOM los procesos se reconocen entre sí a través de las Entidades de Aplicación, es la parte del proceso que trata con la comunicación. Contiene al Usuario de Servicio del

proceso, el cual tiene funciones de iniciación de conexiones y transferencia de información. Tiene un nombre, Título de Aplicación, que son nombres simbólicos para los procesos involucrados en la comunicación. Ver Figura 3.13.

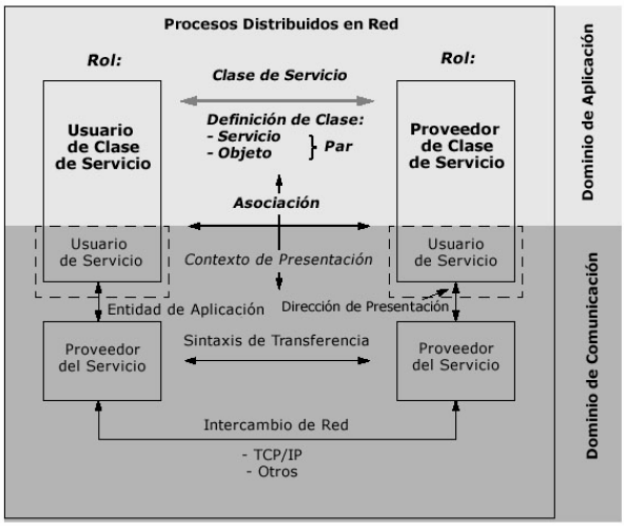


Figura 3.13 Procesos distribuidos en red.

Dirección de Presentación.

En una red se debe proporcionar una dirección, llamada Dirección de Presentación, que apunta a la Entidad de Aplicación. Se llama Dirección de Presentación porque el Usuario de Servicio es la capa de aplicación OSI, el Proveedor de Servicio es la capa de presentación OSI (y capas más bajas). El límite entre ambas capas es el punto de Acceso a Red, donde los datos se transfieren de la capa de aplicación a las capas de red. Cada punto de acceso a red tiene una dirección única.

El formato de la Dirección de Presentación depende del protocolo de red utilizado. Las redes DICOM en la mayoría de los instancias utilizan el protocolo TCP/IP. En este instancia la Dirección de Presentación se mapea a un *Socket* TCP/IP.

Negociación de Asociación.

La conexión para el intercambio de información entre dos Entidades de Aplicación se llama Asociación. Para establecer una Asociación se debe definir el contexto en que la información será intercambiada (Contexto de Aplicación DICOM) donde ambas Entidades deben acordar interactuar como lo defina este contexto.

El Contexto de Aplicación se identifica por un UID y durante la iniciación de la Asociación el UID se transfiere a la otra parte, donde se compara y se decide si es capaz de manejar la solicitud para una Asociación, aceptándola o rechazándola.

El Contexto de Aplicación cubre la funcionalidad global para el intercambio de información. El tipo de intercambio de información que tendrá lugar a través de la Asociación, lo define las Clases SOP y las Clases de Servicio. La parte que solicita la Asociación, propone las Clases SOP, los roles (Usuario o Proveedor de Clase de Servicio) para cada Clase SOP y la forma de representación de la información. Dependiendo de las capacidades de la otra parte, aceptará o rechazará cada Clase SOP. Después del proceso de negociación entre ambas partes y conociendo las capacidades y limitaciones de ambos, el intercambio de información se realiza de acuerdo a las reglas definidas por la Clase de Servicio y la Clase SOP.

Contexto de Presentación.

Para cada Clase SOP negociada durante la iniciación de una Asociación, se debe alcanzar un acuerdo acerca de la Sintaxis de Transferencia usada entre ambos procesos. La parte que inicia propone todas las Sintaxis de Transferencia que puede manejar para una cierta Clase SOP. El otro lado selecciona una de estas Sintaxis de Transferencia, fijando el Contexto de Presentación para esta Clase SOP. Después de la negociación, se define el Contexto de Presentación para cada Clase SOP aceptada.

El Contexto de Presentación se identifica por un número acordado entre ambas partes. Para una Asociación pueden existir varios Contextos de Presentación. El número del Contexto de Presentación identifica a la Clase SOP para la que se realizará el intercambio de información.

Protocolos de Red.

Los protocolos de red actuales deben cumplir con los servicios estándar como lo define el protocolo OSI. El uso de otros protocolos es posible siempre y cuando se adapten a los servicios OSI.

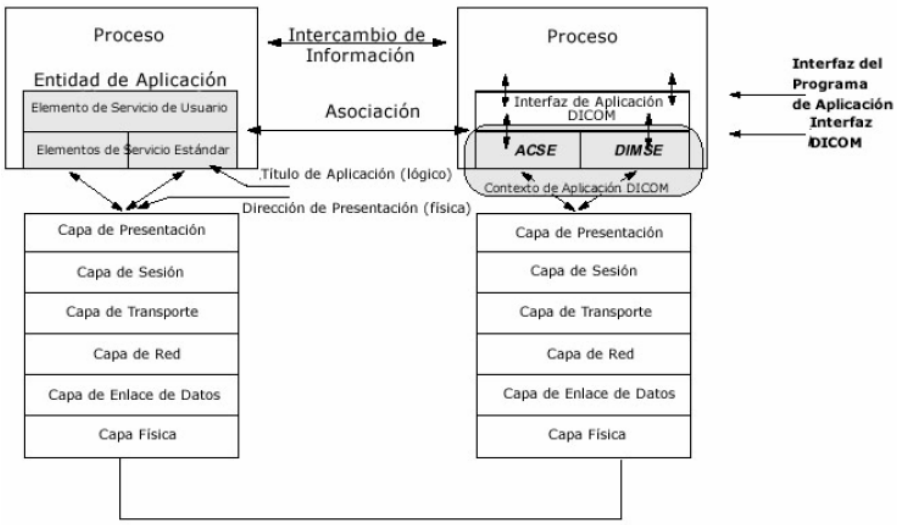


Figura 3.14. Esquema de relación entre Modelo OSI y DICOM.

En la figura 3.14, la parte izquierda del esquema muestra la Entidad de Aplicación para un proceso de comunicación en general, la parte derecha muestra la funcionalidad DICOM para la Entidad de Aplicación.

Se deben tener dos servicios operando en la capa de aplicación para una implementación con DICOM: Elemento asociado al control del servicio es el estándar del protocolo OSI (ACSE/*Association Control Service Element*) y elemento de servicio de mensajes (DIMSE/DICOM *Message Service Element*) implementa los Elementos de Servicio DICOM. La interfaz entre ACSE, DIMSE y la aplicación es la Interfaz DICOM, la cual indica que parámetros son requeridos para cada función de solicitud de ACSE y DIMSE.

La Interfaz DICOM y los protocolos ACSE y DIMSE son parte del Contexto de Aplicación DICOM. La interfaz de la aplicación (API) no se especifica en el estándar DICOM, pero depende de la implementación. En general, la API proporciona las funciones para conectarse con otras aplicaciones, construir o procesar intancia SOP y transferirlos a aplicaciones remotas.

Protocolo TCP/IP.

En la implementación de DICOM a través de una red, comúnmente se utiliza la combinación del *stack* de TCP/IP y una extensión de los servicios de aplicación OSI. En TCP/IP no hay capas altas definidas, la funcionalidad de las capas de aplicación, presentación y sesión requeridas por DICOM, se combinan en una sola capa: La Capa Superior DICOM (DUL). La Capa Superior DICOM utiliza la misma interfaz DICOM tanto para el protocolo TCP/ IP como para el protocolo OSI. A nivel más bajo la DUL tiene una interfaz con la capa de TCP. La Asociación DICOM entre las Entidades de Aplicación se mapea a una conexión TCP. La Dirección de Presentación se mapea a un número de puerto TCP, combinado con el número IP o el nombre del *Host*. Esta combinación del número de puerto TCP y el número IP se llama la Dirección de *Socket*. Una conexión TCP esta definida por la combinación de la dirección del *Socket* local y la dirección del *Socket* remoto. Manteniendo la unicidad de los números IP en toda la red y teniendo los puertos TCP únicos dentro del sistema, solo habrá una conexión TCP con esta combinación. La administración de las conexiones la hace la Interfaz de *Socket* la cual proporciona funciones para el establecimiento de conexiones, transferencia de cadenas de *bytes*, etcétera. Debe conocerse el puerto TCP de la otra Entidad solicitado durante la iniciación de la conexión. Este puede ser un número de puerto acordado entre dos aplicaciones o un número de puerto reservado para implementaciones DICOM (número de puerto 104). Ver Figura 3.15.

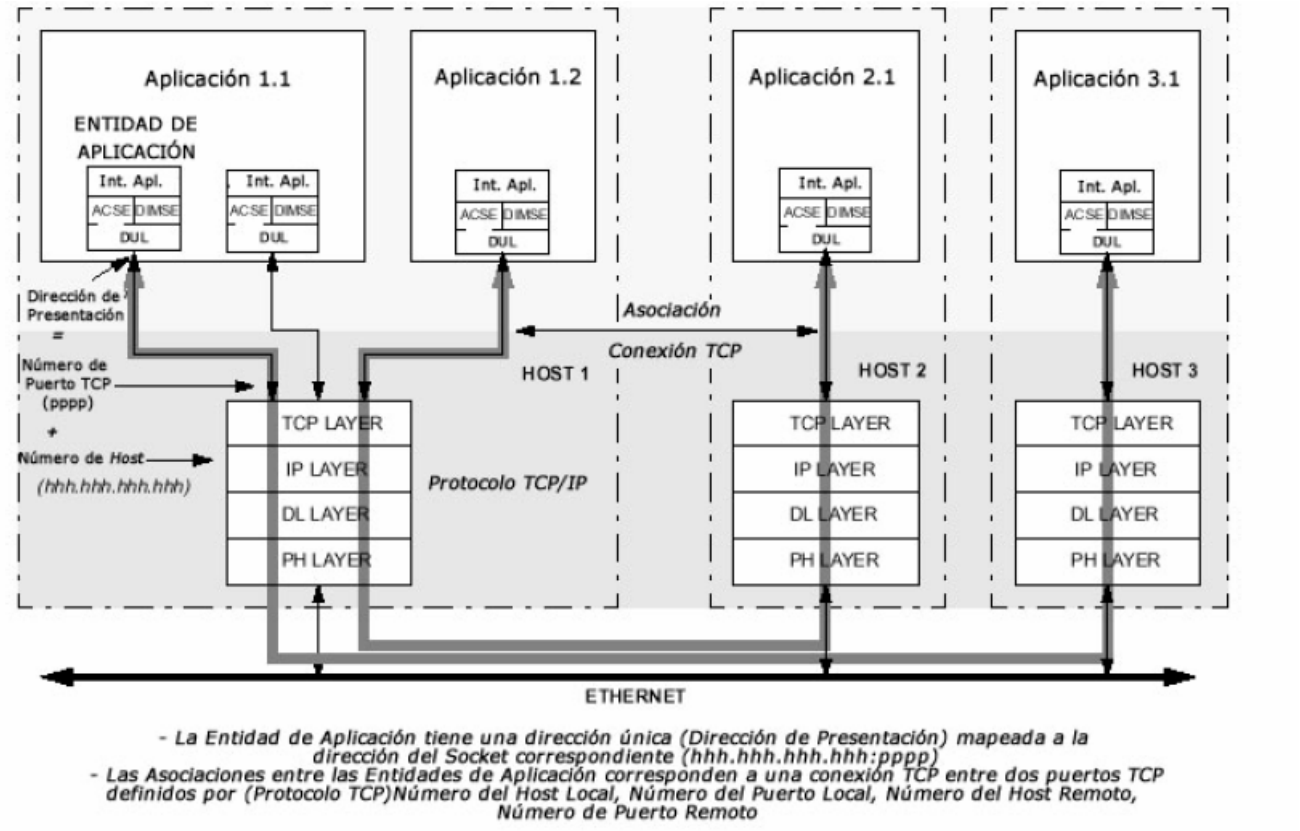


Figura 3.15 Diagrama de conexión TCP.

Modelo de Información de imágenes DICOM.

El manejo electrónico de la información requiere de un modelo para representar la forma en que se estructura la información. Esta estructuración es necesaria para tener intancias uniformes y hacer posible la descripción de las relaciones entre instancias sin ambigüedades. El Modelo de Información se deriva de la forma en que las imágenes se manejan en el departamento de radiología de un hospital. Las imágenes se coleccionan a partir de una o más modalidades en el archivo del paciente y se ordenan de acuerdo al tipo de examen médico. Los técnicos de cada tipo de modalidad utilizan su propia terminología para este ordenamiento. Los datos de una imagen que proviene de diferentes fuentes, deben juntarse en un solo ambiente, esto es posible únicamente cuando todos los datos de la imagen están estructurados de acuerdo al mismo Modelo de Información. El Modelo de Información para imágenes DICOM se basa en la forma en que se relaciona la información de diferentes modalidades. Ver figura 3.16.

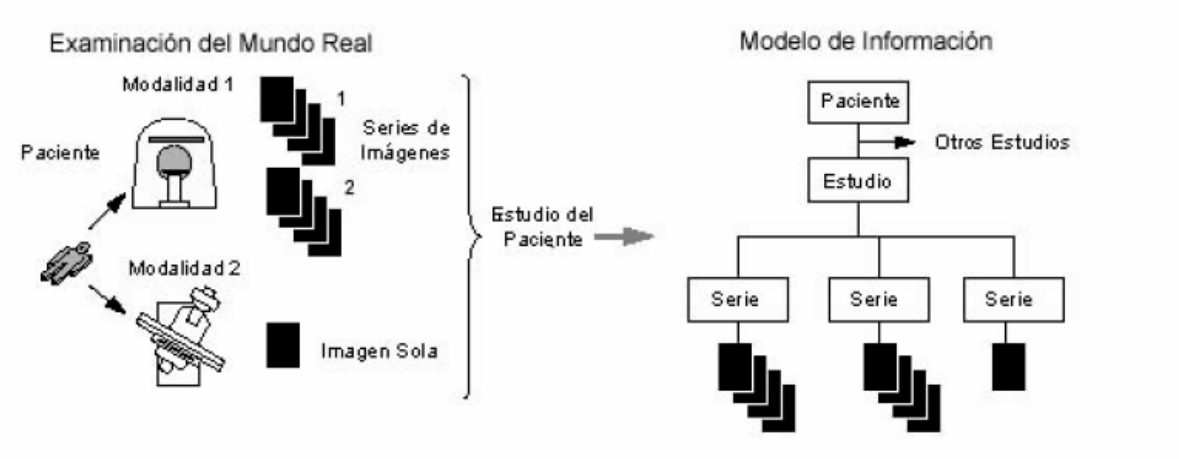


Figura 3.16 Modelo de información de imágenes DICOM.

Los cuatro niveles de este Modelo de Información son: Paciente, Estudio, Serie(s) e Imagen.

A continuación se desarrollaran estos cuatro niveles y la instancia SOP de imágenes.

Nivel de Paciente.

El nivel de paciente contiene la información de identificación y demografía del paciente al que pertenece el estudio. Un paciente puede tener uno o más estudios, por lo que este nivel se considera el más alto. Sin embargo, la práctica normal es usar el nivel de estudio para la recopilación de la información que proviene de varios sistemas de información para una solicitud de examen médico.

Nivel de Estudio.

El nivel de estudio es el más importante dentro del modelo de información. Un estudio es el resultado de una solicitud de cierto tipo de examen médico. Todas las actividades dentro del departamento de radiología giran entorno al manejo correcto del estudio.

En este nivel se guarda la identificación de la información y las referencias de la información relacionada con el estudio y el sistema general de información del hospital (HIS o RIS).

En general, una solicitud puede involucrar procedimientos de exámenes médicos de diferentes modalidades. Los resultados se tendrán en series de una o más imágenes. Todos los datos de imágenes se colocan junto con el mismo estudio como raíz. Un solo paciente puede tener múltiples estudios como resultado de otras solicitudes (estudio previo) para un procedimiento de examen médico.

Nivel de Serie.

Abajo del nivel de estudio se reúnen todas las series de imágenes. En este nivel se identifica el tipo de modalidad de las imágenes, fecha y hora de creación, detalles acerca del tipo de examen médico y equipo utilizado. Las series son un conjunto de imágenes relacionadas que provienen de una sola modalidad. La manera en que las imágenes se agrupan en series depende de su uso clínico. El modo de adquisición no es tan importante a este nivel, sin embargo, existen atributos que lo identifican y pueden mostrarse en el despliegue. Para varias modalidades, la relación entre imágenes está definida por la forma en que se realiza la adquisición. Cuando las adquisiciones de una secuencia tienen una relación temporal o espacial, las imágenes resultantes pueden ser agrupadas en series. Una nueva serie debe comenzar cuando la relación existente entre imágenes no es valida. Otro criterio para agrupar imágenes puede ser, reunir imágenes de una sola parte del cuerpo, durante un examen médico completo. Por ejemplo, cuando una modalidad produce varias imágenes del estómago de un paciente, en diferentes momentos y posiciones, las imágenes pueden reunirse en una sola serie. Algunos sistemas producen más de una imagen en una sola adquisición. Por ejemplo, en un sistema de tomografía computarizada, las imágenes reconstruidas de cada toma se reúnen en una serie y tienen una relación espacial. La siguiente toma será una nueva serie, porque en la mayoría de los instancias, cada toma se hace a partir de una posición distinta. En una serie, puede incluirse una imagen de referencia como visión de conjunto de la posición de cada "rebanada" individual. Ver figura 3.17.

Es posible almacenar re-construcciones distintas de la misma adquisición en series separadas. Para cada tipo de modalidad, se deben describir las reglas que definen el contenido de una sola serie. DICOM no define para ninguna modalidad como se debe reunir una serie.

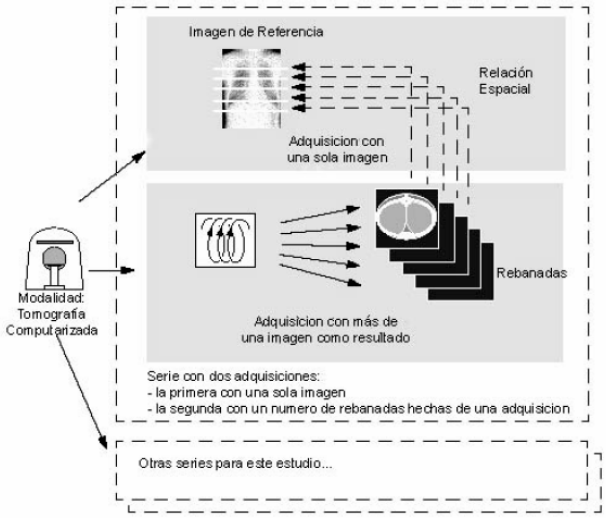


Figura 3.17. Serie creada a partir de la modalidad de tomografía computarizada.

Nivel de Imagen.

El nivel más bajo del Modelo de Información es el nivel de imagen. Cada imagen contiene información de adquisición y posición, así como también los datos propios. Dependiendo del tipo de modalidad, el nivel de imagen contiene datos para una sola imagen, dos imágenes (sistema de dos planos) o una colección de imágenes tomadas a partir de una colección de datos (imágenes) en un período corto de tiempo (*multiframe images*).

El uso de imágenes *multiframe*, evita la duplicidad de información en los niveles más altos, pero es solamente posible cuando la relación entre *frames* puede describirse en una forma sencilla. Por ejemplo, los incrementos en los movimientos del sistema y el tiempo son iguales para todas las *frames* sencillas.

La creación de imágenes *multiframe* es más compleja y consume más recursos que la creación de una imagen sencilla. La relación entre *frames*, la capacidad de la modalidad y el monto de los datos de imágenes producidos, pueden utilizarse para determinar cuando es mejor aplicar una serie de imágenes sencillas o *multiframe*.

Intancia SOP de imágenes.

En la Figura 3.18. se muestra el diagrama de flujo del modelo de información DICOM para imágenes, cada bloque representa una Entidad de Información de un IOD compuesto. Las relaciones indican las opciones para cada Entidad de Información en un Instancia SOP. Una imagen es un Instancia SOP Compuesto que contiene todo el árbol del modelo de información DICOM. Las imágenes contenidas en una serie tienen las mismas Entidades de Información del paciente, estudio y serie. Las series tienen las mismas Entidades de Información del paciente y estudio, etcétera.

Un Instancia SOP Compuesto hace el intercambio y manejo de la información más fácil pero incrementa la cantidad de datos cuando se transfiere un estudio completo. En este instancia las Entidades de Información del paciente y el estudio se multiplican en una colección de Instancias SOP. En cambio, con Instancias SOP Normalizados se utilizan referencias a otras Entidades permitiendo un protocolo más eficiente pero requiriendo un manejo más complejo.

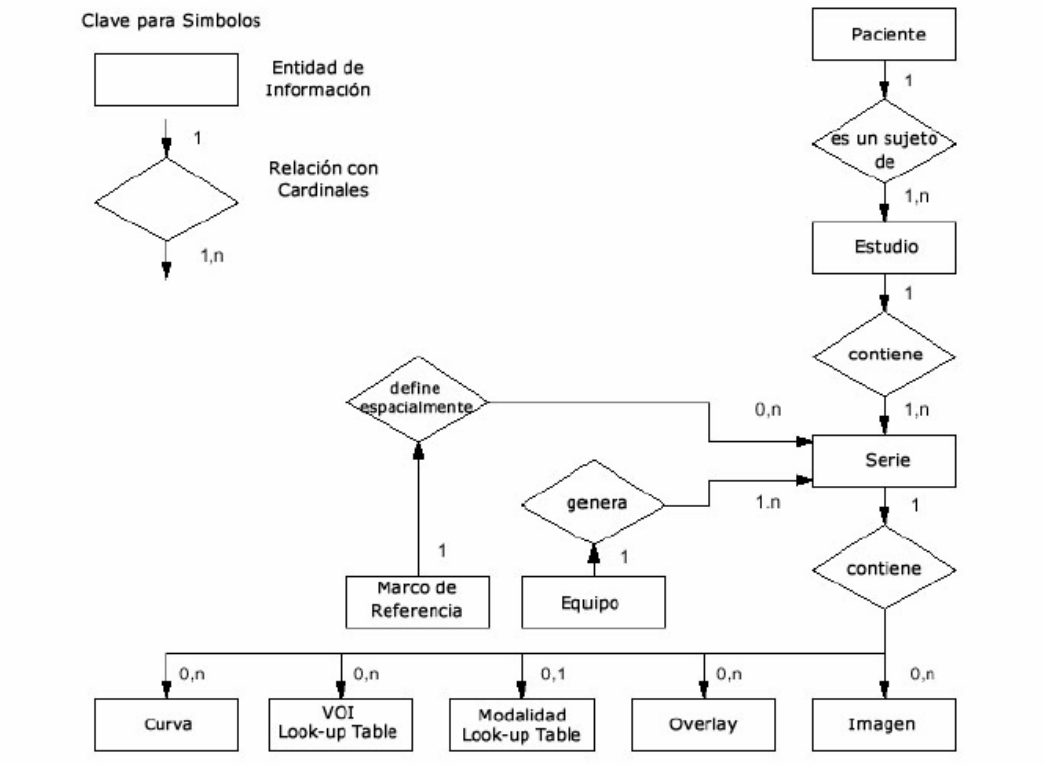


Figura 3.18 Diagrama de flujo para el modelo de información de imágenes DICOM.

Al reunir un grupo de Instancias SOP de imágenes que se relacionan entre sí pero creadas a partir de diferentes modalidades, es importante poder relacionar las Entidades de Información a diferentes niveles. Se consideran importantes dos aspectos:

- Todas las modalidades deben tener un mapeo consistente entre los datos de imagen y el Instancia SOP. Al nivel de serie e imagen, la identificación se hace a partir de la modalidad. Arriba del nivel de serie las Entidades de Información deben contener datos que pertenezcan al estudio y al paciente, los cuales pueden compararse con datos de otras modalidades.
- Cada Entidad de Información debe tener una identificación para relacionarse correctamente con las Entidades de Información equivalentes y otros Instancias SOP, permitiendo que todos los sistemas involucrados (modalidades, sistemas de almacenamiento, estaciones de trabajo, etcétera) puedan manejarlas.

Clasificación de datos de la imagen.

En la figura 3.19 se muestran las etapas de origen y clasificación de la información para crear una instancia SOP de imagen a partir de una modalidad cualquiera, cada etapa agrega Atributos al instancia SOP.

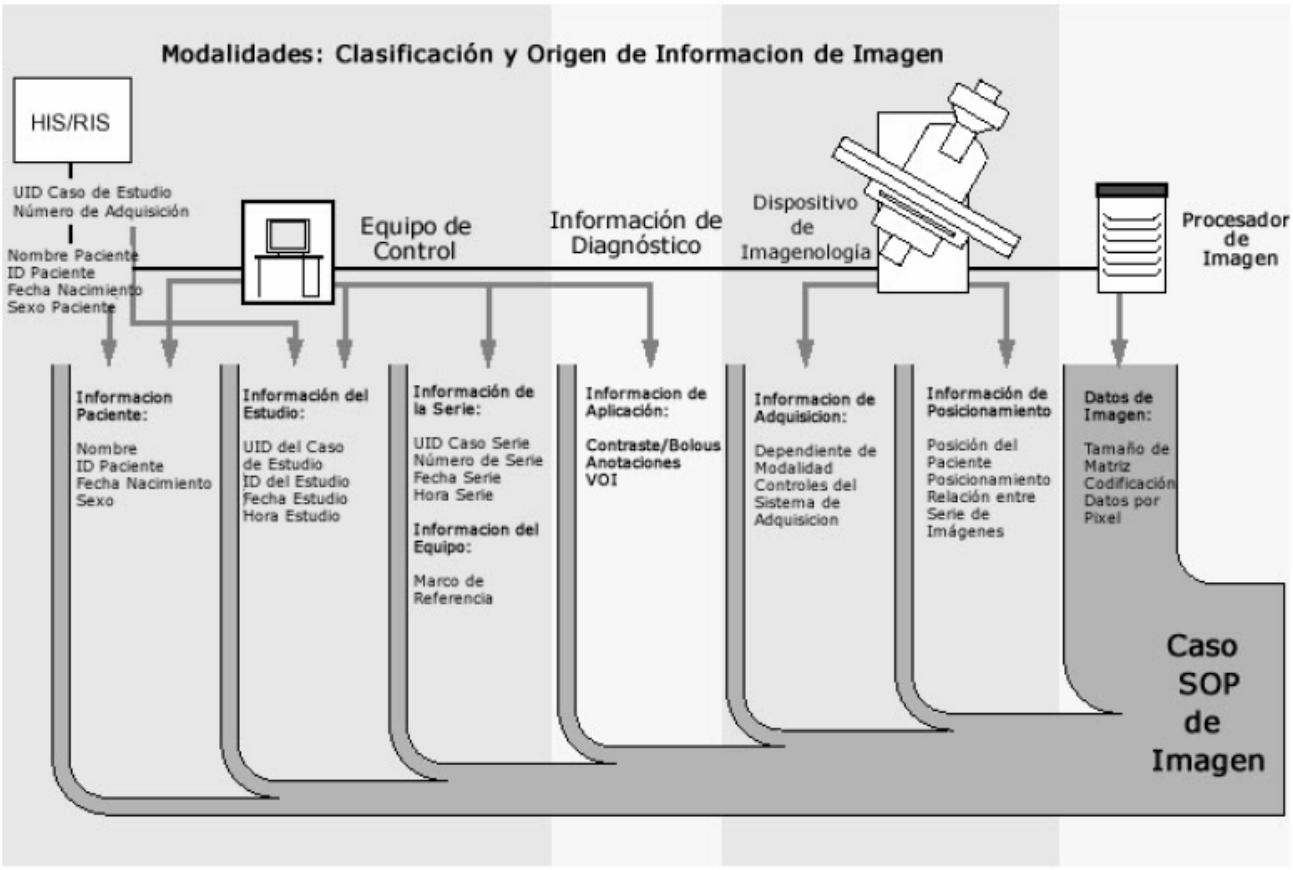


Figura 3.19. Origen y clasificación de información de una imagen.

Para la clasificación de datos de la imagen hablaremos de los siguientes puntos:

- Información del paciente.
- Información del estudio.
- Información de la serie.
- Información de aplicación.
- Información de adquisición.
- Información de posicionamiento.
- Datos de la imagen.

Información del paciente.

Generalmente la información del paciente se obtiene de fuentes externas a la modalidad, como los sistemas de información HIS o RIS. Para registrar de manera formal esta información se definen ciertos Atributos, como el nombre del paciente, ID, fecha de nacimiento, etcétera. La información que se obtiene en esta etapa es estable, excepto por corrección de errores de captura o cambio en el nombre por matrimonio.

Los Atributos en esta etapa son muy importantes porque permiten la conexión a otros sistemas que operan dentro de un hospital.

En instancias excepcionales, como un paciente de emergencia, se requieren procedimientos que permitan actualizar la información posterior a la atención del paciente.

Información del estudio.

En la etapa de recopilación de la información del estudio, se utilizan datos que provienen de los sistemas de información (RIS o HIS) y el tipo de modalidad, la cual agrega información del paciente al momento de realizarse el estudio.

La manera más eficiente de identificación del estudio es por el UID del Instancia de Estudio, aunque se puede utilizar un atributo alternativo llamado Número de Adquisición que genera el RIS, o bien, si no existe ninguno de estos, la modalidad debe generar un UID que garantice su unicidad.

Otros atributos provistos por la modalidad son los nombres de los médicos que solicitan o consultan las imágenes, la información del paciente que es dinámica como la edad, peso, etcétera, así como también el valor para el atributo ID de Estudio, hora y fecha del mismo, dicho ID es únicamente relevante para la modalidad.

Información de la serie.

La información de la serie proviene únicamente de la modalidad, consiste de Atributos como el tipo de sistema, ubicación, identificación serial, calibración, etcétera.

El UID del Instancia de la Serie la identifica como única en el ambiente de los datos de imagen. La modalidad genera un ID de serie, que puede usarse en una secuencia de series de un estudio.

Con esta información se tienen más detalles de la obtención de la serie, como el personal involucrado, posición relevante, parte del cuerpo examinado, etcétera.

Se utiliza un marco de referencia para agrupar imágenes con alguna relación temporal o espacial, permitiéndose dividir una serie en partes o en más de una serie si se aplica la misma relación. El marco de referencia se identifica por un UID y se comparte entre las imágenes involucradas.

Información de aplicación.

En esta etapa se agregan Atributos acerca de la imagen de un Instancia SOP necesarios para el diagnóstico como son, un texto como comentario de los detalles del contraste, terapia y dispositivos utilizados durante el estudio, así como el VOI que es una selección del rango de valores de pixel que son clínicamente significativos para el despliegue o impresión de la imagen, debiéndose convertir únicamente dicho rango a los niveles de gris disponibles.

Información de adquisición.

En esta etapa, se almacenan los Atributos referentes a los ajustes del equipo de adquisición, los cuales dependen del tipo de modalidad y puede variar de pocos Atributos para sistemas simples a estructuras complejas para sistemas como la Resonancia Magnética.

Los Atributos contienen detalles de los ajustes de sistemas de adquisición, como valores en kiloVolts para Rayos X, identificación de la secuencia de escaneo utilizada para resonancia magnética, etcétera.

Las imágenes obtenidas a partir de una sola adquisición tienen un Número de Adquisición para identificación de la serie, pero se debe considerar que una adquisición puede resultar en varias series con diferentes características. La adquisición no tiene relación con el Modelo de Información DICOM y no tiene UID de identificación equivalente.

Información de posicionamiento.

En la etapa referente a la información de posicionamiento se describe la forma en que la matriz de la imagen se posiciona, mediante términos simples como anterior, posterior, derecha, izquierda, etcétera. Se debe asegurar la suficiente información para tener un despliegue sin ambigüedades.

En una serie que tiene relación espacial, como la serie de imágenes de tomografía computarizada o resonancia magnética, se debe proveer de mayor detalle acerca de la posición de las imágenes en el espacio tridimensional del cuerpo del paciente. Esta información permite a los sistemas planificadores de tratamiento de radioterapia usar el posicionamiento tridimensional para el procesamiento de los datos de imagen.

Datos de la imagen.

La última etapa es la adquisición de los datos de la imagen y su procesamiento para tener una imagen visible en formato digital. Aquí se detalla la interpretación de la información en píxeles como el tamaño de matriz, representación del valor, codificación y formato.

La imagen se identifica por un UID de imagen. El UID de la imagen se utiliza también como UID del Instancia SOP. Este UID identifica al Instancia SOP cuando se transfiere o recupera de un servidor de imágenes.

Tipos de imágenes.

Todos los Instancias SOP de imagen comparten un mínimo de información que permite a los dispositivos de despliegue manejar las imágenes sin importar su tipo. Ver figura 3.20.

Existe una Clase SOP de imagen dedicada para encapsular imágenes que no están disponibles en formato digital, sino que se capturan a partir de video o formato de película.

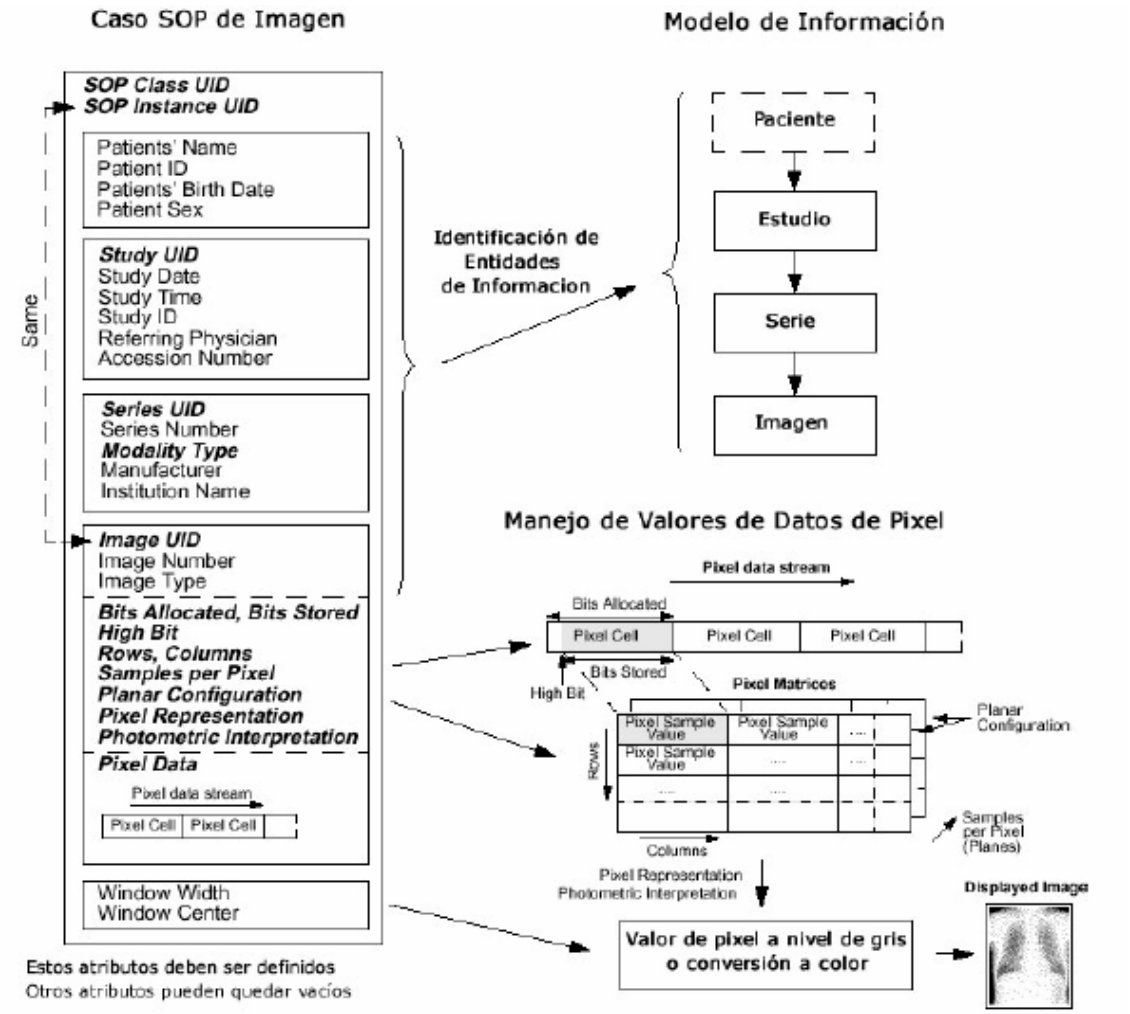


Figura 3.20. Conjunto mínimo de Atributos para un Instancia SOP de imagen

El conjunto mínimo de atributos requeridos para un Instancia SOP son los siguientes:

- Atributos de identificación: UID de Clase SOP, UID de Instancia de Estudio, UID del Instancia de Serie y UID de Instancia de Imagen (*Image Instance* UID = SOP *Instance* UID).
- Tipo de modalidad.
- Descripción de la Matriz de Píxel: Muestras por píxel, renglones, columnas.
- Interpretación del valor de píxel: Interpretación fotométrica.
- Codificación de píxel: Bits asignados, bits almacenados, bit más alto, representación.
- del píxel, configuración plana.
- Matriz de Píxel.

Este conjunto mínimo permite el despliegue de los datos de imagen y proporciona identificación a nivel de sistema, de manera que el Instancia SOP se apegue al Modelo de Información. Para tener un Instancia SOP más completo, se debe adicionar información mínima para los tres primeros niveles del Modelo de Información. Los Atributos que identifican al Instancia SOP para consulta y permiten el despliegue correcto de la imagen son:

- Nivel de paciente: Nombre del paciente, ID del paciente, fecha de nacimiento, sexo.

- Nivel de estudio: Fecha de estudio, hora de estudio, nombre del médico de referencia,ID del estudio, número de adquisición.
- Nivel de serie: Número de serie, Fabricante, nombre de la institución.
- Nivel de imagen: Número de imagen, tipo de imagen.
- Ajustes de presentación: Ancho de ventana, centro de ventana.

Estos atributos son en su mayoría tipo 2 (obligatorios u opcionales) o tipo 3 (opcionales).

Tipos de Imágenes Especializados.

El formato genérico descrito anteriormente se utiliza en cada definición de Clase SOP de imagen, pero dependiendo del tipo de modalidad se extiende con información dedicada. El número de tipos de imágenes especializadas crece constantemente por el desarrollo de nuevos tipos de modalidades. Actualmente, las siguientes modalidades tienen una definición de Clase SOP de Almacenamiento para imágenes en el estándar DICOM:

- IOD de Radiografía Computarizada (Computed Radiography IOD), utilizada por los equipos de radiografía tradicionales. Las imágenes creadas por este tipo de modalidad no contienen extensión de información acerca de la adquisición y posicionamiento. En la mayoría de los instancias no hay relación (temporal o espacial) con otras imágenes en serie.
- IOD de Tomografía Computarizada (Computed Tomography IOD), para escáneres de tomografía computarizada. Para este tipo de modalidad la información del posicionamiento es importante para el procesamiento de pilas de imagen o para crear vistas en el espacio tridimensional.
- IOD de Resonancia Magnética (Magnetic Resonante IOD), además de la misma información de posicionamiento, también necesitan extensión de información acerca del protocolo de adquisición.
- IOD de Medicina Nuclear (Nuclear Medicine IOD) contienen un formato especial de imagen e información de adquisición. Las imágenes están en formato multiframe.
- IOD de Ultrasonido (Ultrasound IOD), este tipo de modalidad contiene detalles acerca del posicionamiento y adquisición de la imagen. Las imágenes pueden estar en formato de color y usar formato multiframe.
- IOD de Angiografía por Rayos X (X-Ray Angiographic IOD) para sistemas digitales cardiovasculares. Este formato puede capturar una corrida en formato multiframe o imágenes sencillas.

3.3 El Admisntirado de Base de Datos MySQL.

MySQL es un sistema de administración de bases de datos relacional,multiusuario, con licencia bajo GNU GPL (General Public License o licencia pública general) . Su diseño multihilo (Multithread) le permite soportar una gran carga de forma muy eficiente. MySQL es una implementación cliente/servidor[36][37][38]. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este adminsitrador en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

MySQL es el más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a la existencia de librerías y otras herramientas que permiten su uso a través de lenguajes de programación distintos como: Java C, C++, Eiffel, Perl, PHP, Python y TCL, además de su fácil instalación y configuración.

Figura 3.21 Mysql bajo Linux (izquierda) y Windows (derecha).

Las principales características de este administrador de bases de datos son las siguientes:

- El principal objetivo de MySQL es velocidad y robustez.
- Escrito en C y C++, probado con GCC 2.7.2.1. Usa GNU autoconf para potabilidad. Clientes C, C++, Java, Perl, TCL, etc.
- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Puede trabajar en distintas plataformas y S.O. distintos (ver fig 3.21).
- Soporta una elevada cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Soporta hasta 32 índices por tabla.
- Administración de usuarios y claves , manteniendo un muy buen nivel de seguridad en los datos.

3.4 Acceso a la Base de Datos utilizando Java Database Connectivity (JDBC).

JDBC es un API de Java para acceder a sistemas de bases de datos. El API JDBC consiste de un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. En otras palabras, con el API JDBC no es necesario escribir un programa para accesar a Sybase, otro programa para accesar a Oracle, y otro programa para accesar a MySQL; con esta API, se puede crear un sólo programa en Java que sea capaz de enviar sentencias SQL a la base de datos apropiada.

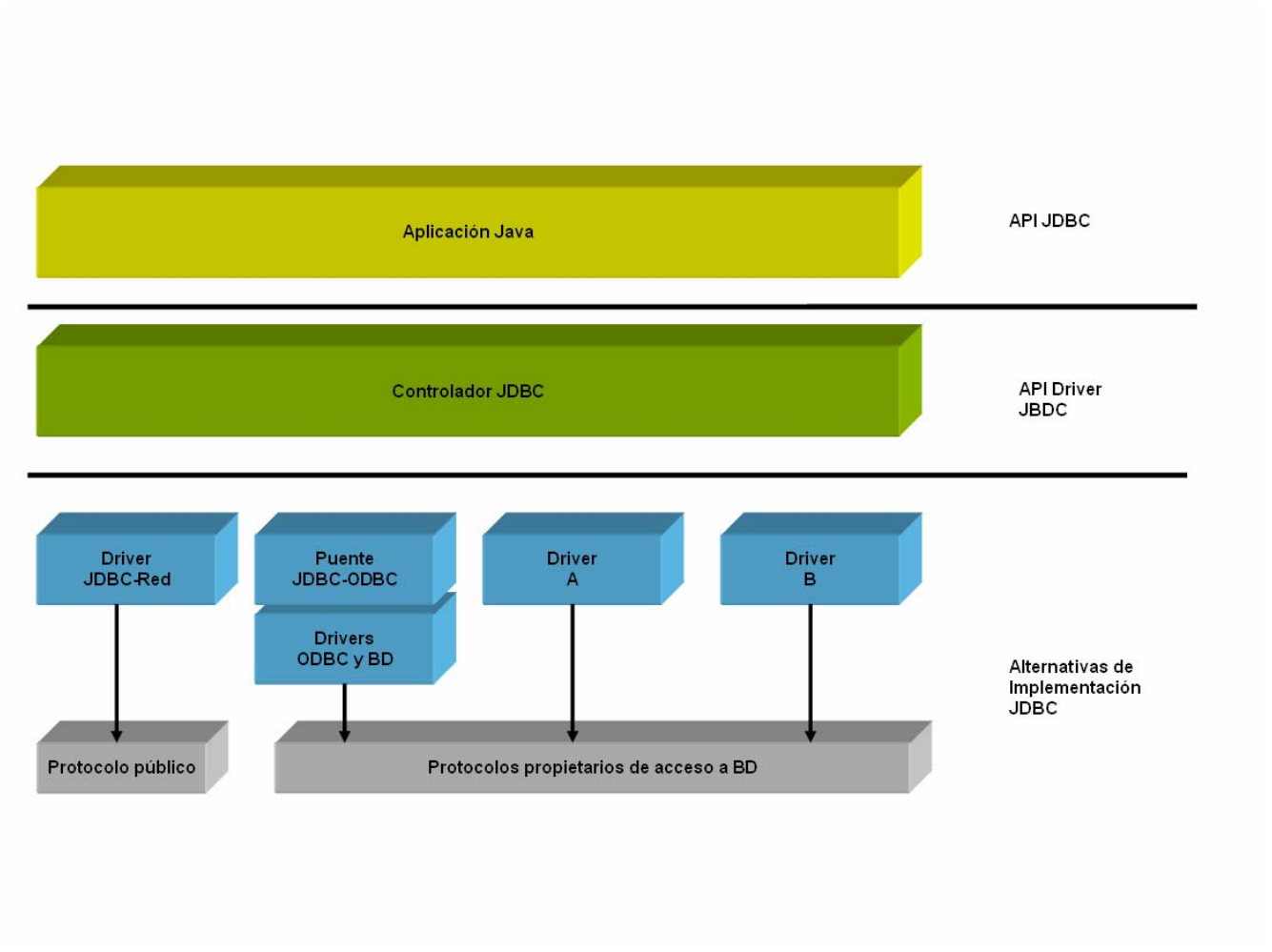


Figura 3.22 Relación entre la aplicación Java JDBC y driver de la BD.

Al igual que ODBC, la aplicación de Java debe tener acceso a un controlador (driver) JDBC adecuado. Ver figura 3.22.

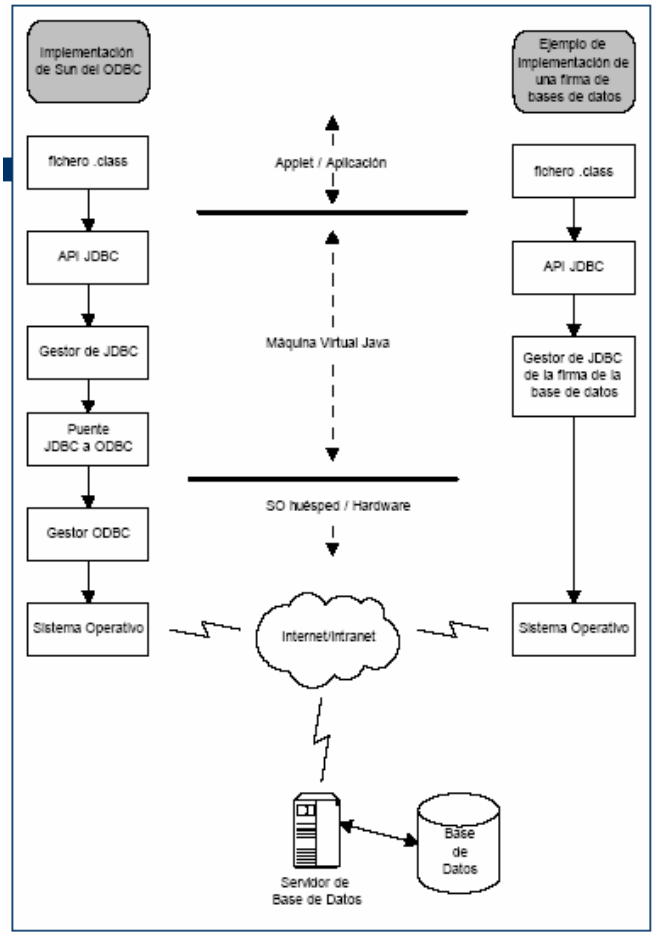


Figura 3.23 Comunicación con la base de datos.

Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real (ver Figura 3.23).

De manera muy simple, al usar JDBC se pueden el siguente proceso:

1. Establecer Driver.
2. Establecer conexión.
3. Crear sentencia.
4. Ejecutar sentencia.
5. Procesar resultados.
6. Finalizar sentencia.
7. Cerrar conexión.

Los distribuidores de bases de datos suministran los controladores que implementan el API JDBC y que permiten acceder a sus propias implementaciones de bases de datos. De esta forma JDBC proporciona a los programadores de Java una interfaz de alto nivel y les evita el tener que tratar con detalles de bajo nivel para acceder a bases de datos.

En el caso del manejador de bases de datos MySQL, Connector/J es el controlador JDBC oficial.

Cabe señalar que actualmente JDBC es el nombre de una marca registrada, y ya no más un acrónimo; es decir, JDBC ya no debe entenderse como "Java Database Conectivity".[38]

Tipo de Clases de JDBC.

Como vimos JDBC consiste de un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos. En la tabla 3.1 se muestra el tipo y la clase implementada para JDBC.

Tipo	Clase JDBC
Implementación	java.sql.Driver java.sql.DriverManager java.sql.DriverPropertyInfo
Conexión a base de datos	java.sql.Connection
Sentencias SQL	java.sql.Statement java.sql.PreparedStatement java.sql.CallableStatement
Datos	java.sql.ResulSet
Errores	java.sql.SQLException java.sql.SQLWarning

Tabla 3.1 Tabla de tipo y clase implementada para JDBC.

java.sql.DriverManager.

Lleva el control de la administración JDBC disponibles (es posible que existan varios dentro del sistema), por default carga todos los disponibles en "sql.drivers", por medio del metodo registerDriver se lleva un registro de esté.

La sintaxis utilizada es del tipo URL por ejemplo:

- jdbc:<subprotocolo>:<parámetros>
- jdbc:odbc:NOTICIAS:UID=Sistema;PWD=SistemaPW

java.sql.Driver.

Se encarga de administrar la información y configuración general. Se carga durante la inicialización mediante:

  - DriverManager.registerDriver.
  - Class.forName.

Residirá en memoria ya que se le pedirá información a lo largo del programa. Los métodos más usados son:

  - Connect.
  - getPropertyInfo.

java.sql.Connection.

Este se encarga de llevar los Punteros a la base de datos. Proporciona el contexto de trabajo para los objetos Statement y ResultSet. Soporta propiedades de transacción setAutoCommit, commit y rollback.

java.sql.Statement.

Nos sirve para la ejecución de una sentencia SQL:

- executeQuery.- Para sentencias SELECT. Y devuelve un ResultSet.
- executeUpdate.-Para sentencias INSERT, DELETE, UPDATE, CREATE. Y devuelve un entero
- execute.- Para sentencias desconocidas en tiempo de compilación o sentencias que devuelven resultados complejos. Y devuelve true/false

java.sql.PreparedStatement.

Esta clase extiende Statement para añadir sentencias precompiladas SQL (Compila la sentencia SQL la primera vez, y a sentencias que son llamadas más de una vez en el programa).

Soporta parámetros de entrada como setInt, setFloat, setLong, setString.

java.sql.ResultSet.

Esté contiene los datos resultado de una sentencia SQL. Recuperan secuencialmente en filas (next sirve para avanzar una fila).

Se puede acceder a los datos de las columnas en cualquier orden (índice de posición, nombre del campo, método wasNull(), Métodos: getString, getFloat, getInt, etc.).

Ejemplo de inserción de datos.

A contimuación se muestra un ejemplo de insertar datos en una tabla.

```
import java.sql.*;
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
final String BBDD = "jdbc:odbc:ARTICULOS";
try {
        Class.forName(DRIVER);
        Connection conexion = DriverManager.getConnection(BBDD);
        PreparedStatement insercion = conexion.prepareStatement(
        ”INSERT INTO articulos VALUES (?,?,?)");
        insercion.setString(1,” JAVA”);
        insercion.setString(2,”123”);
        insercion.setString(3,”Informática”);
        int resultado = insercion.executeUpdate();
        insercion.close(); conexion.close();
} catch (Exception e) {
        System.out.println("Error: " + e);
}
```

Acontimuación mostramos el metodo de prueba de comunicación:

```
public boolean Test(String login, String password){
      Connection  con = null;
      boolean     flag = false;
      try {
         Class.forName(DRIVER).newInstance();
         con = DriverManager.getConnection(URL, login, password );

         if(!con.isClosed()){
            System.out.println("YA ESTOY CONECTADO Y " +
                  "LA PRUEBA ES PERFECTA");
            flag = true;
         }
      } catch(Exception e) {
         System.err.println("Exception: " + e.getMessage());
      } finally {
         try {
            if(con != null) con.close();
         } catch(SQLException e) {}
         return flag;
      }
   }
```

Para consultas de la descripción y tipo de atributos, disponemos del método getMetadata() de la clase ResultSet. Éste devuelve un objeto ResultSetMetaData con el que podemos obtener información acerca de la estructura de las sentencias devueltas. Los principales métodos son:

- getColumnCount() Devuelve el número de columnas del ResultSet.
- getColumnLabel(int i) Devuelve la cabecera de la columna i.
- getColumnName(int i) Nombre de la columna situada en la posición i.
- getColumnType(int i) Devuelve un número que identifica el tipo de dato. La clase java.sql.Types contiene las constantes que definen cada tipo de dato sql.
- getColumnTypeName(int i) Muestra en un objeto cadena el nombre del tipo de dato presente en la columna i.
- getColumnDisplaySize(int i) El tamaño de visualización en caracteres de la columna i.
- getPrecision(int i) Cuantos dígitos decimales tiene la columna.
- getScale(int i) Cuantos dígitos hay a la derecha del punto decimal.
- getTableName(int i) En qué tabla está la columna.

A continuación se muestra un ejemplo de cómo utilizarlo.

```
String query = "select * from Table1";
        ResultSet rs = stmt.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();
        int columnCount = rsmd.getColumnCount();
        for (int i = 1; i <= columnCount; i++) {
                String s = rsmd.getColumnTypeName(i);
                System.out.println ("Column " + i + " is type " + s);
        }
```

3.5 Diseño de Base de Datos.

El modelo de Base de Datos Relacionales evita la redundancia de la información, poder compartir o bloquear información, permitiendo la disponibilidad de recursos, y podemos tener un mejor control de la administración de la información.

Un modelo de Datos es la representación gráfica del mundo real que nos permite: organizar la información mediante estructuras de dados, permitiendo la definición de unidades de datos y sus relaciones, por lo que podemos plantear y comunicar nuestra idea de la Base de Datos.

El modelo entidad-relación fue formalizada por C.W. Bachean[46] a fines de los 60' y sus principales caracteristicazas son: Poder esquematizar las relaciones entre entidades, manejar mucha información de manera sencilla, facilitando su entendimiento y documentación, y el mapeo es directo en la construcción de una Base de Datos Relacionales.

El método de Diseño de una base de datos es:

- Identificación y definición de los objetos principales dentro del sistema.
- Diagrama Entidad Relación.
- Resolución del Modelo Lógico.
- Normalización del modelo lógico.
- Conversión del modelo lógico a un esquema físico.

Identificación y definición de los objetos principales dentro del sistema.

En este punto hablaremos sobre que es: entidad, Instancia de las entidades y relación.

Entidad.

La declaración de Entidades es el objeto de interés principal o importante para el usuario dentro del sistema. Para este caso tenemos que son:

- Diccionario.
- Paciente.
- Estudio.
- Serie.
- Imagen.

Instancia de las entidades.

Es Información que almacenamos de la entidad.

Relación

Una relación esta asociada a un verbo o una preposición que establece la conexión entre dos entidades. Las relaciones entre entidades se describen en tres términos que son: conectividad, cardinalidad y dependencia de existencia.

EN UN SENTIDO	EN SENTIDO CONTRARIO	RESULTANTE
1:1	1:1	1:1
1:1	1:M	1:M
M:1	1:1	1:M
1:M	M:1	M:M

Tabla 3.2 La regla para determinar la conectividad entre entidades.

Conectividad.

La conectividad describe el número de instancias de una entidad asociada a otra entidad. Existen tres tipo de conectividad los cuales son: Uno a Uno (1:1); Uno a Muchos (1:M); Muchos a Muchos (M:M ó N:M).

La regla utilizada para determinar la conectividad entre entidades se muestra en la tabla 3.2.

Se obtiene lo siguiente:

Paciente	← 1:1	Estudio
	→1:M	
	1:M	
Estudio	← 1:1	Serie
	→1:M	
	1:M	
Serie	← 1:1	Imagen
	→1:M	
	1:M	

Tabla 3.3 Conectividad.

Cardinalidad.

La Cardinalidad definen las reglas del negocio o políticas

Paciente.

- Número Mínimo de pacientes 5.
- El paciente será examinado por un máximo de 100 usuarios.

Usuario.

- Número máximo de usuarios 100.

Estudio.

- Cada paciente tendrá al menos un estudio. En el caso contrario el administrador dará de baja al paciente.

Imagen.

- El contenido de la imagen deberá ser menor a 1Mega.

Dependencia de Existencia.

Es la descripción de la una entidad dentro de una relación esta puedes ser: obligatoria u opcional. Las relaciones tiene una dependencia de existencia de:

- Obligatoria / Opcional.
- Obligatoria / Obligatoria.
- La relación opcional / opcional no existe.

Obligatoria.

Se considera obligatoria si la existencia de una instancia dentro de una entidad es requerida para que exista la relación.

Opcional.

Se considera opcional cuando la existencia de una instancia dentro de una entidad no es necesaria para que exista la relación.

Paciente	Obligatoria	Estudio
Estudio	Obligatoria	Serie
Serie	Obligatoria	Imagen

Tabla 3.4 Dependencia de existencia.

Atributos.

Los atributos son las características o calificadores que proporcionan la información detallada de la entidad. Se clasifican en identificadores y descriptores.

Identificadores.

El identificador define de manera única una instancia dentro de una entidad. No permiten el almacenamiento de valores nulos.

Identificador ▶ llaves → relacionan las entidades.

Descriptores.

Describen todas las características de la entidad.

Atributos derivados.

Son los atributos cuyo valor depende de otros atributos. Estos no deben de almacenarse a menos que sean prescindibles para los procesos finales.

Dominio.

Es el conjunto de valores válidos asociados a un atributo.

Diagrama Entidad Relación.

El Diagrama Entidad-Relación nos permite modelar la información en base a las necesidades de una empresa o proyecto. C.W.Bachman formalizo los diagramas utilizados en los modelos de datos Entidad-Relación en los 60's.

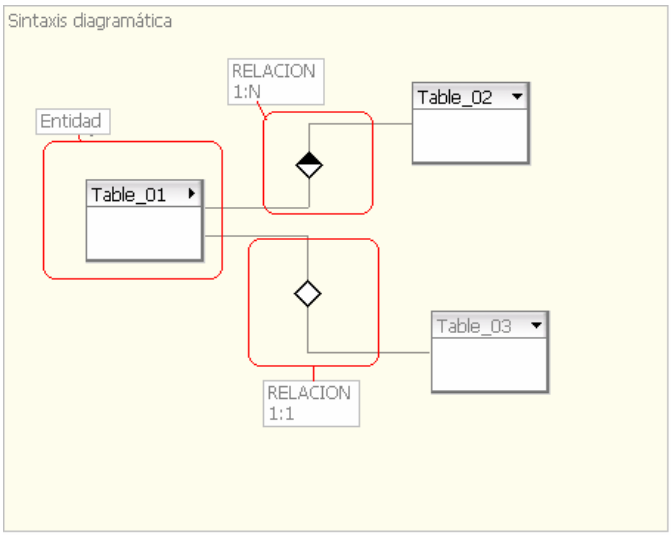


Figura 3.24 Diagrama Entidad-Relación.

Resolución del Modelo Lógico.

En este punto hablaremos como eliminar el problema que se presentan en los diagramas entidad relación, llaves primarias y las llaves secundarias.

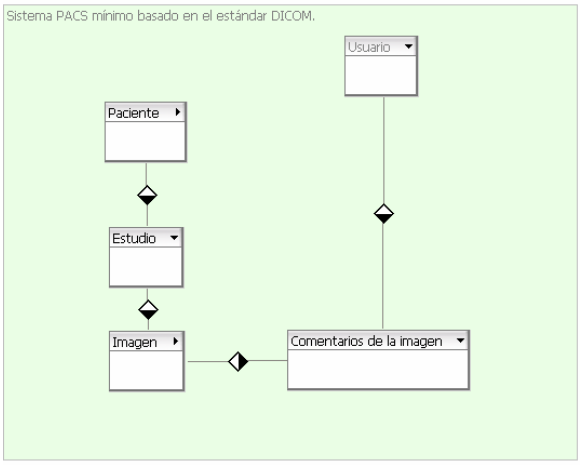


Figura 3.25 Diagrama Entidad–Relación, de nuestro sistema.

Eliminación de problemas que se presentan en el diagrama Entidad-Relación.

En el diagrama Entidad-Relación se pueden presentar las siguientes relaciones:

- Relación Uno a Uno.
- Relaciones Mucho a Mucho.

Para el "Sistema PACS mínimo basado en el estándar DICOM" no presento estos casos pero es importante tenerlo en cuenta para otros sistemas.

En las siguientes secciones se describe como debe tratarse esta situación.

Relaciones Uno a Uno

Este tipo de relaciones se presentan pocas veces y al presentarse se debido a que una entidad es un subconjunto de la otra.

Para eliminar esta relación 1:1, es necesario unir las entidades involucradas en una sola. Ejemplo:

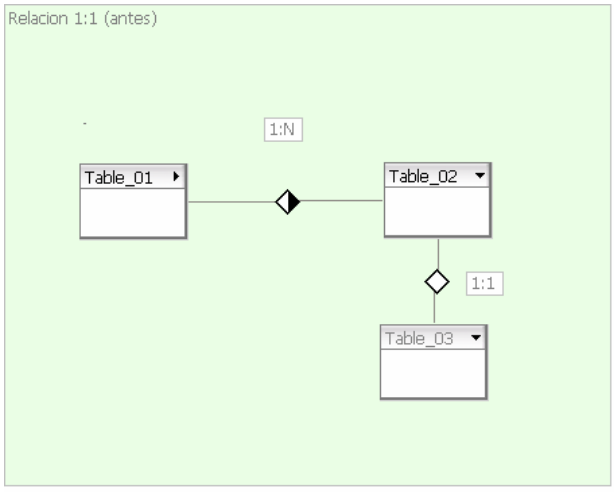


Figura 3.26 Eliminar esta relación 1:1, es necesario unir las entidades involucradas en una sola (antes).

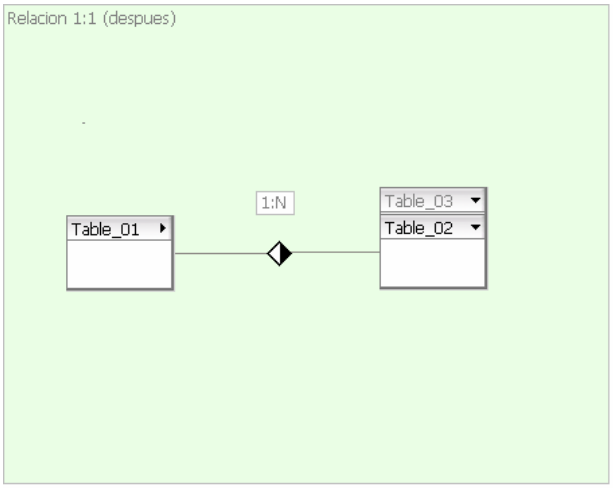


Figura 3.27 Eliminar esta relación 1:1, es necesario unir las entidades involucradas en una sola (después).

Relaciones Muchos a Mucho.

Para resolver este tipo de relaciones, se debe crear una nueva entidad que separe las entidades donde se presenta la relación M:M.

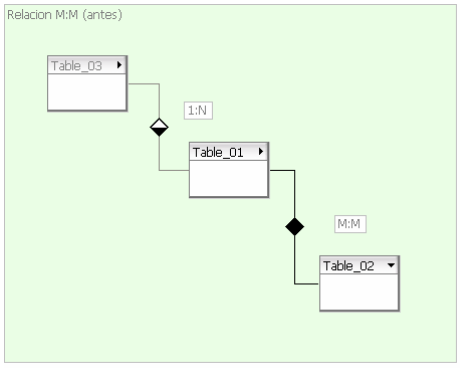


Figura 3.28 Relaciones Muchos a Mucho (antes).

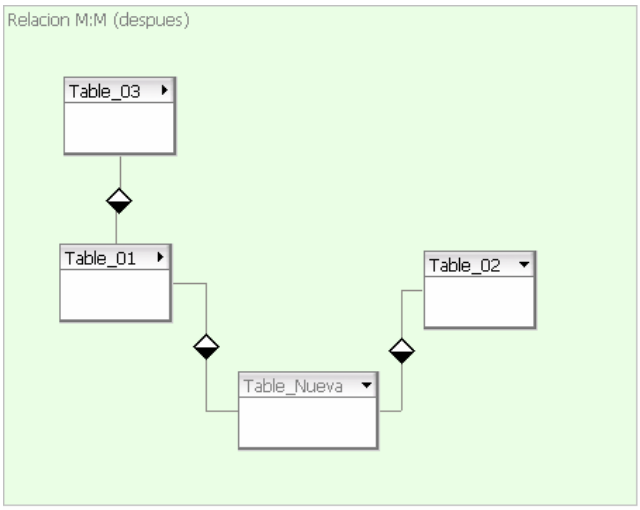


Figura 3.29 Relaciones Muchos a Mucho (después).

Llaves primarias.

Las llaves primarias están formadas por un atributo o combinación de atributo que identifica de manera única una instancia de una entidad.

La entidad debe tener una llave primaria sin valores nulos. Es recomendable que sean valores numéricos o cadenas cortas de caracteres. Pueden ser simples o compuestas (con dos o más atributos). Las características son tres: Primera.- no acepta valores nulos; Segunda.-Deben almacenar valores únicos; Tercero.-No depende del orden de almacenamiento.

Llaves foráneas o secundarías.

Estas llaves son utilizadas para establecer las relaciones entre entidades. Las características de estas llaves son: Primero.- Son llaves primarias de otra entidad; Segundo Aceptan valores nulos; Tercero.- Pueden modificarse; Cuarto.- Una entidad puede tener múltiples llaves secundarias; Quinto.- Aceptan valores duplicados.

Normalización del modelo lógico.

En la normalización nosotros solo utilizaremos hasta la tercera forma que son:

- Primera forma normal (1nf).
- Segunda forma normal (2nf).
- Tercera forma normal (3nf).

Primera forma normal (1nf).

Una entidad cumple con la 1er forma normal si no hay repetición de grupos o dominios.

No debe haber columnas repetidas (que almacenan información equivalente).

Segunda forma normal (2nf).

Una entidad cumple con la segunda forma normal si:

- Cumple con la 1NF.
- Todos sus atributos dependen de la llave primaria.

Tercera forma normal (3nf).

Una entidad cumple con la tercera forma normal si:

- Cumple con la 2NF.
- Ningún atributo de la entidad debe tener "Dependencia Transitiva" con la llave primaria.

En otras palabras, ningún atributo de una entidad depende de otro atributo que no sea la llave primaria.

Conversión del modelo lógico a un esquema físico.

El mapeo del modelo lógico al modelo físico es de la siguiente forma

Modelo Lógico.	Modelo físico.
Modelo E-R.	Esquema de la BD.
Entidad.	Tabla.
Atributo.	Columna.
Relación.	Índices, llaves.
Instancia de Entrada.	Registro o renglón Que es la información cargada.

Tabla 3.5 Mapeo del modelo lógico al modelo físico.

3.6 Criptografía.

La criptografía es la técnica que permite codificar un objeto de forma que su significado no sea obvio. Es un medio para mantener datos seguros en un entorno inseguro, en el cual un objeto original (O) se puede convertir en un objeto cifrado(C) aplicando una función de cifrado (E). Obviamente, es necesario que se pueda descifrar el mensaje cifrado, para volver a obtener el mensaje original aplicando una función de descifrado (D), que puede ser, o no, la inversa de E. Esta técnica permite que la información no sea inteligible para los usuarios que no conocen la forma de descifrar la información.

C=E (O)

O=D(C)

Existen dos conceptos básicos en criptografía: clave y algoritmos de cifrado. Ambos conceptos no están indisolublemente ligados, puesto que, habitualmente, un algoritmo de cifrado admite muchas claves y una clave podría servir para muchas funciones de cifrado.

Claves.

La clave es el patrón que usan los algoritmos de cifrado y descifrado para manipular los mensajes en uno u otro sentido. El uso de claves tiene varias ventajas, ya que permite que las funciones de cifrado y descifrado puedan ser públicas, que se puedan usar las mismas funciones para generar distintos cifrados y que el resultado cifrado no dependa únicamente del diseñador del algoritmo, sino también de una clave fija por el dueño del objeto cifrado.

Sistemas de Claves privadas y sistemas de clave publica.

Los sistemas de claves privadas el emisor y el receptor comparten una clave de codificación que conocen solo ellos. EL problema generado es que se basa en la ocultación de claves, por lo que si se quiere que alguien descifré el objeto cifrado debe conocer la clave con que está cifrado, podemos dar dos soluciones: propagar la clave o usar claves nuevas. Pero cuando existen muchas claves es muy difícil mantener claves confidenciales y con buenas características

El problema se puede resolver usando sistemas de cifrado con clave publica, en el que cada usuario tiene una clave de cifrado para que cualquiera pueda enviar un mensaje cifrado de dicha clave. Sin embargo, cada usuario tiene también una clave de descifrado publica, que es secreta.

La clave pública y la privada cumple con dos propiedades:

- Se puede descodificar con la clave privada algo codificado con la clave pública.
- Se puede descodificar algo codificado con la clave privada solo si se dispone de la clave pública.

Esto implica que ambas claves se pueden aplicar en cualquier orden, lo que significa que se pueden descifrar aplicando la clave privada y luego la publica. Cualquier proceso A puede enviar mensajes a B cifrados con la clave pública de B. Sin embargo, el receptor sólo los podrá descifrar si tiene la clave privada de B. Este método asegura la privacidad.

Dando solución a que:

- No es necesario intercambiar claves para poder comunicarse con un servidor de forma segura.
- Muestran lo más posible al exterior, evitando que los intrusos tengan curiosidad por conocer las claves de los servidores.

3.7 Algoritmo de encriptamiento Data Encryption Standard (DES).

DES (Data Encryption Standard) es un esquema de encriptación simétrico desarrollado en 1977 por el Departamento de Comercio y la Oficina Nacional de Estándares de EEUU en colaboración con la empresa IBM, que se creó con objeto de proporcionar al público en general un algoritmo de cifrado normalizado para redes de computo. Estaba basado en la aplicación de todas las teorías criptográficas existentes hasta el momento, y fué sometido a las leyes de USA.

Posteriormente se sacó una versión de DES implementada por hardware, que entró a formar parte de los estándares de la ISO con el nombre de DEA.

Se basa en un sistema monoalfabético, con un algoritmo de cifrado consistente en la aplicación sucesiva de varias permutaciones y sustituciones. Inicialmente el texto en claro a cifrar se somete a una permutación, con bloque de entrada de 64 bits (o múltiplo de 64), para posteriormente ser sometido a la acción de dos funciones principales, una función de permutación con entrada de 8 bits y otra de sustitución con entrada de 5 bits, en un proceso que consta de 16 etapas de cifrado.

En general, DES utiliza una clave simétrica de 64 bits, de los cuales 56 son usados para la encriptación, mientras que los 8 restantes son de paridad, y se usan para la detección de errores en el proceso.

Como la clave efectiva es de 56 bits, son posible un total de 2 elevado a 56 = 72.057.594.037.927.936 claves posibles, es decir, unos 72.000 billones de claves, por lo que la ruptura del sistema por fuerza bruta o diccionario es súmamente improbable, aunque no imposible si se dispone de suerte y una grán potencia de cálculo.

Los principales inconvenientes que presenta DES son:

- Se considera un secreto nacional de EEUU, por lo que está protegido por leyes específicas, y no se puede comercializar ni en hardware ni en sofware fuera de ese país sin permiso específico del Departamento de Estado.
- La clave es corta, tanto que no asegura una fortaleza adecuada. Hasta ahora había resultado suficiente, y nunca había sido roto el sistema. Pero con la potencia de cálculo actual y venidera de los computadores y con el trabajo en equipo por Internet se crée que se puede violar el algoritmo, como ya ha ocurrido una vez, aunque eso sí, en un plazo de tiempo que no resultó peligroso para la información cifrada.
- No permite longitud de clave variable, con lo que sus posibilidades de configuración son muy limitadas, además de permitirse con ello la creación de limitaciones legales.
- La seguridad del sistema se ve reducida considerablemente si se conoce un número suficiente textos elegidos, ya que existe un sistema matemático, llamado Criptoanálisis Diferencial, que puede en ese caso romper el sistema en 2 elevado a 47 iteraciones.

Entre sus ventajas cabe citar:

- Es el sistema más extendido del mundo, el que más máquinas usan, el más barato y el más probado.
- Es muy rápido y fácil de implementar.

DES trabajaba sobre bloques de 128 bits, teniendo la clave igual longitud. Se basa en operaciones lógicas booleanas y podía ser implementado fácilmente, tanto en software como en hardware. A continuación describiremos como funciona.

Consiste básicamente en la reducción de la longitud de clave y de los bloques, DES cifra bloques de 64 bits, mediante permutación y sustitución y usando una clave de 64 bits, de los que 8 son de paridad (esto es, en realidad usa 56 bits), produciendo así 64 bits cifrados.

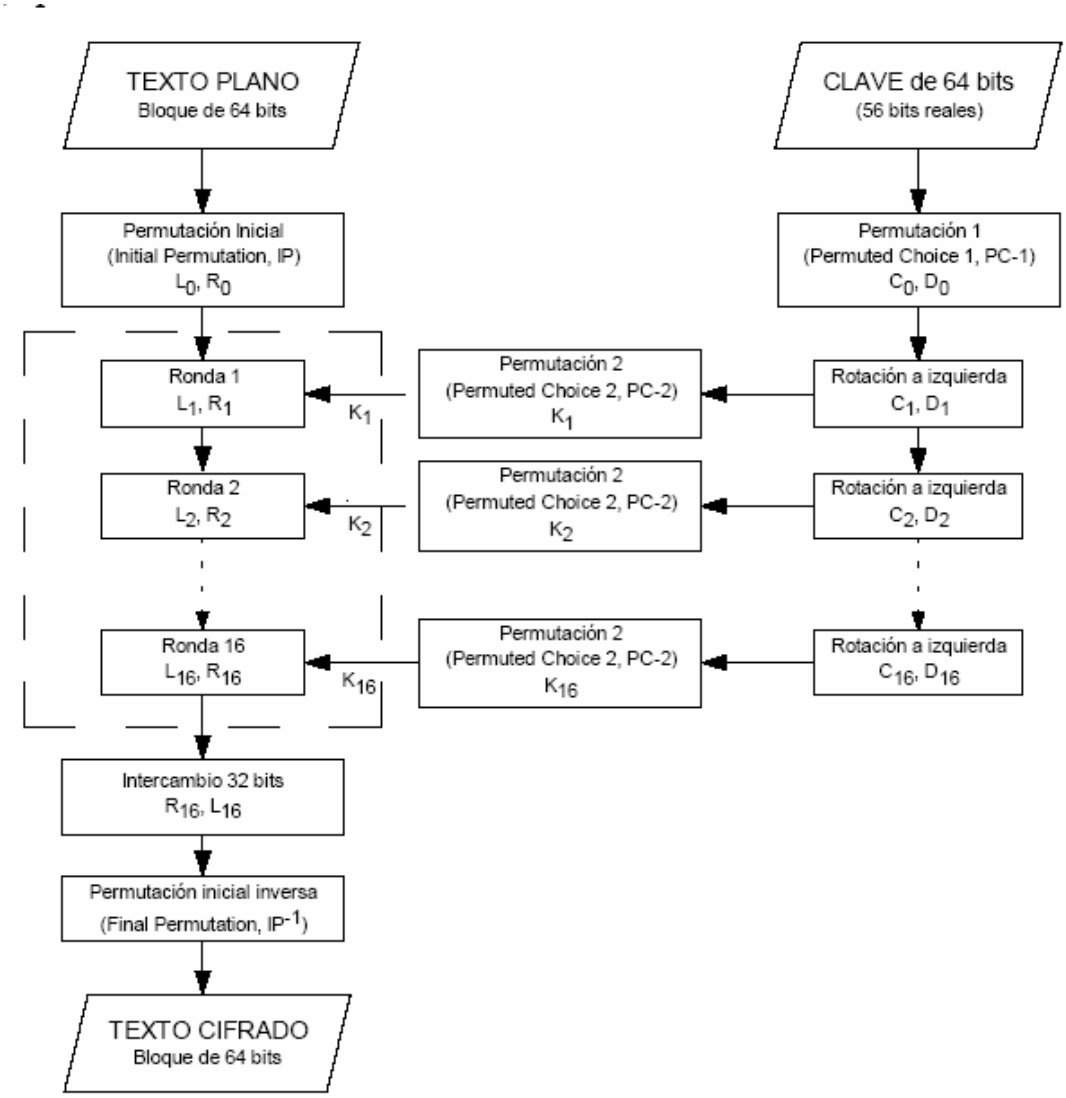


Figura 3.30 Esquema general del algoritmo DES.

DES tiene 19 etapas diferentes. La primera etapa es una transposición, una permutación inicial (IP) del texto plano de 64 bits, independientemente de la clave. La última etapa es otra transposición (IP-1), exactamente la inversa de la primera. La penúltima etapa intercambia los 32 bits de la izquierda y los 32 de la derecha. Las 16 etapas restantes son una Red de Feistel de 16 rondas. En cada una de las 16 iteraciones se emplea un valor, Ki, obtenido a partir de la clave de 56 bits y distinto en cada iteración.

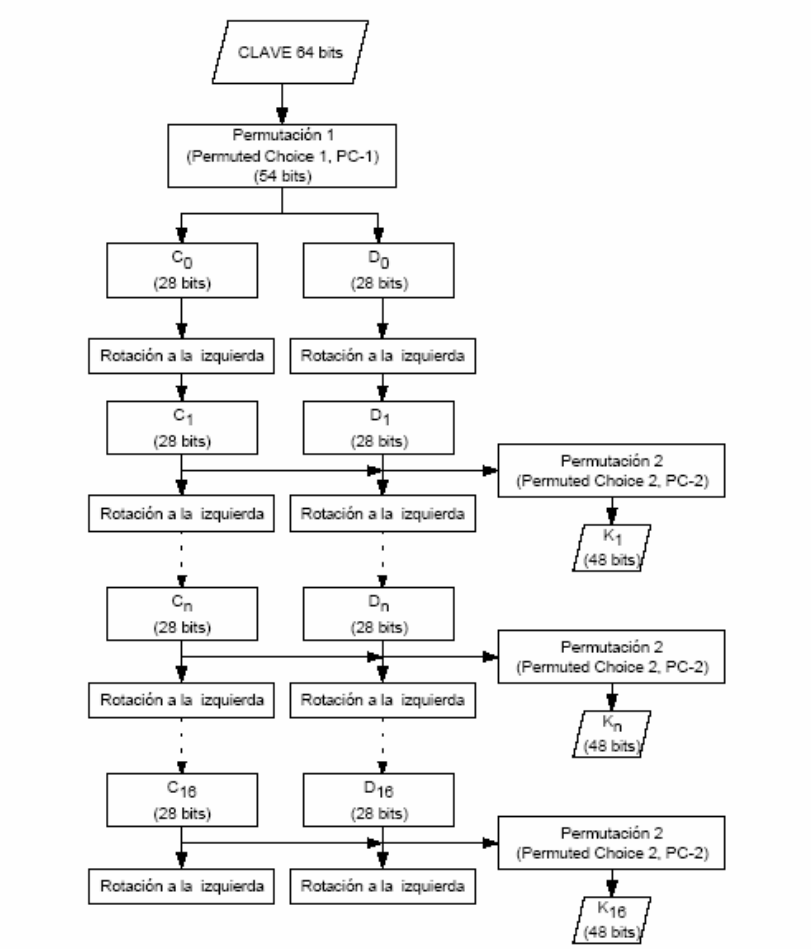


Figura 3.31 Cálculo de las subclaves, Ki.

Se realiza una permutación inicial (PC-1) sobre la clave, y luego la clave obtenida se divide en dos mitades de 28 bits, cada una de las cuales se rota a izquierda un número de bits determinado que no siempre es el mismo. Ki se deriva de la elección permutada (PC-2) de 48 de los 56 bits de estas dos mitades rotadas.

La función f de la red de Feistel se compone de una permutación de expansión (E), que convierte el bloque correspondiente de 32 bits en uno de 48. Después realiza una or-exclusiva con el valor Ki, también de 48 bits, aplica ocho S-Cajas de 6*4 bits, y efectúa una nueva permutación (P).

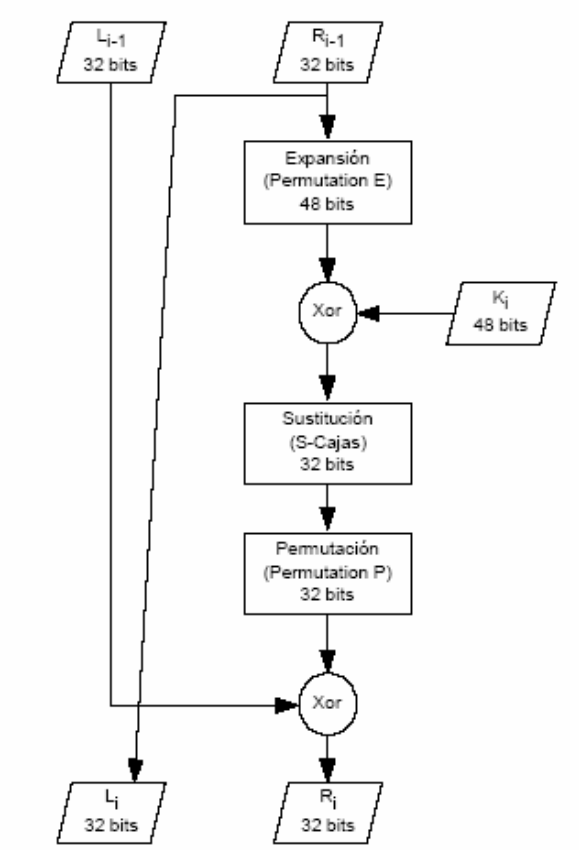


Figura 3.32 Ronda del algoritmo DES.

La forma para descifrar se el mismo algoritmo empleando las Ki en orden inverso. Está descrito oficialmente en FIPS PUB 46.

A continuación se describen los pasos necesarios para implementar este algoritmo.

(i) Encriptación.

1.- Procesar la clave.

1.1.- Solicitar una clave de 64 bits al usuario. La clave se puede introducir directamente o puede ser el resultado de alguna operación anterior, ya que no hay ninguna especificación al respecto.

De cada uno de los ocho bytes se elimina el octavo bit (el menos significativo).

1.2.- Calcular las subclaves.

1.2.1.- Realizar la siguiente permutación en la clave de 64 bits reduciéndose la misma a 56 bits (El bit 1, el más significativo, de la clave transformada es el bit 57 de la clave original, el bit 2 pasa a ser el bit 49, etc.).Ver tabla 3.6.

Permuted Choice 1 (PC-1)

57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Tabla 3.6 Permutación en la clave.

1.2.2.- Dividir la clave permutada en dos mitades de 28 bits cada una. C(0) el bloque que contiene los 28 bits de mayor peso y D(0) los 28 bits restantes.

1.2.3.- Calcular las 16 subclaves (Empezar con i=1)

1.2.3.1.- Rotar uno o dos bits a la izquierda C(i-1) y D(i-1) para obtener C(i) y D(i), respectivamente. El número de bits de desplazamiento está dado por la tabla 3.7.

Ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits despl.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabla 3.7 Número de bits a rotar.

1.2.3.2.- Concatenar C(i) y D(i) y permutar como se indica a continuación(ver Tabla 1.3.). Así se obtiene K(i), que tiene una longitud

Permuted Choice 2 (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

de 48 bits.

Tabla 3.8 Permutación.

1.2.3.3.- Ir a 1.2.3.1. hasta que se haya calculado K(16).

2.- Procesar el bloque de datos de 64 bits.

2.1.- Obtener un bloque de datos de 64 bits. Si el bloque contiene menos de 64 bits debe ser completado para poder continuar con el siguiente paso.
2.2.- Realizar la siguiente permutación del bloque. Ver Tabla 3.9.

Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tabla 3.9 Permutación del bloque.

2.3.- Dividir el bloque resultante en dos mitades de 32 bits cada una. L(0) el bloque que contiene los 32 bits de mayor peso y R(0) el resto.

2.4.- Aplicar las 16 subclaves obtenidas en el paso 1.

2.4.1.- E(R(i)). Expandir R(i) de 32 a 48 bits de acuerdo con la tabla 3.10.

Expansion (E)

32	1	2	3	4	5	4	5
6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27
28	29	28	29	30	31	32	1

Tabla 3.10 Expansión.

2.4.2.- E(R(i-1)) Xor K(i). Or-exclusiva del resultado del paso 2.4.1. con K(i)

2.4.3.- B(1), B(2),..., B(8). Partir E(R(i-1)) Xor K(i) en ocho bloques de seis bits. B(1) representa a los bits 1-6, B(2) representa a los bits 7-12,..., B(8) representa a los bits 43-48.

2.4.4.- S(1)(B(1)), S(2)(B(2)),..., S(8)(B(8)). Sustituir todos los B(j) por los valores correspondientes de las S-Cajas o tablas de sustitución (Substitution Boxes, S-Boxes) de 6*4 bits, según se indica en los subapartados que siguen. Todos los valores de las S-Cajas se consideran de 4 bits de longitud. (Ver S-cajas del algoritmo DES, Tabla 3.12).

2.4.4.1.- Tomar los bits 1º y 6º de B(j) y formar un número de 2 bits que llamaremos m. Este valor nos indicará la fila en la tabla de sustitución correspondiente. S(j). Obsérvese que m=0 representa la 1ª fila y m=3 la última.

2.4.4.2.- Con los bits 2º a 5º de B(j) formar otro número, n, de cuatro bits que indicará la columna de S(j) en la que buscar el valor de sustitución. En esta ocasión n=0 representa la 1ª columna y n=15 la última columna.

2.4.4.3.- Reemplazar B(j) con S(j)(m,n), m fila y n columna.

2.4.4.4.- Ejemplo. Sea B(3)=42, en binario B(3)=101010.

Buscaremos el nuevo valor de B(3) en S(3). Fila m y columna n, según lo expuesto anteriormente m=10, n=0101, y en decimal m=2 y n=5. Por tanto, B(3) será S(3)(2,5)=15

2.4.4.5.- Volver a 2.4.4.1. hasta que todos los bloques B(j) hayan sido reemplazados por el valor de S(j) adecuado.

2.4.5.- P[S(1)(B(1))... S(2)(B(8))]. Concatenar los bloques B(1) a B(8) y permutar los 32 bits (cuatro bits cada B(j)) en función de la Tabla 3.11.

Permutation P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tabla 3.11 Permutación de los 32 bits.

2.4.5.1.- Volver a 2.4.4.1. hasta que todos los bloques B(j) hayan sido reemplazados por el valor de S(j) adecuado.

Fila	Columna															S-Caja
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Tabla 3.12 S; cajas del Algoritmo DES.

2.4.6.- P[S(1)(B(1))... S(2)(B(8))]. Concatenar los bloques B(1) a B(8) y permutar los 32 bits (cuatro bits cada B(j)) en función de la Tabla 3.13.

Permutation P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tabla 3.13 Permutación de los 32 bits.

2.4.7.- R(i). Realizar una or-exclusiva entre el valor resultante y L(i-1). Este valor será R(i). Por tanto, R(i)=L(i-1) Xor P[S(1)(B(1))... S(2)(B(8))]

2.4.8.- L(i). L(i)=R(i-1)

2.4.9.- Repetir desde 2.4.1. hasta que se hayan aplicado las 16 subclaves.

2.5.- Hacer la siguiente permutación del bloque R(16)L(16). Obsérvese que esta vez R(16) precede a L(16). Ver Tabla 3.14.

Final Permutation (IP**-1)

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tabla 3.14. Permutación del bloque R(16)L(16).

(ii) Desencriptación:

Usar el mismo proceso descrito con anterioridad pero empleando las subclaves en orden inverso, esto es, en lugar de aplicar K(1) para la primera iteración aplicar K(16), K(15) para la segunda y así hasta K(1).

Después de implementar este algoritmo podemos verificarlo por medio de los vectores de prueba para DES. A continuación en la Tabla 3.15 se muestran algunos de los valores que se pueden emplear para comprobarlo.

Bloque a cifrar en hexadecimal	**Clave**	**Cifrado**
0123456789abcdef	133457799BBCDFF1	85E813540F0AB405
8787878787878787	0E329232EA6D0D73	0000000000000000
"Your lips are smoother than vaseline" (36 caracteres + Chr(13) + Chr(10)) En hexadecimal: 596f7572206c6970 732061726520736d 6f6f746865722074 68616e2076617365 6c696e650d0a0000	0E329232EA6D0D73	C0999FDDE378D7ED 727DA00BCA5A84EE 47F269A4D6438190 D9D52F78F5358499 828AC9B453E0E653
"Now is the time for all " (24 Caracteres -hay un espacio en blanco del final-) En hexadecimal: 4e6f772069732074 68652074696d6520 666f7220616c6c20	0123456789ABCDEF	3FA40E8A984D4815 6A271787AB8883F9 893D51EC4B563B53

Tabla 3.15 Vectores de prueba para DES.

3.8 Algoritmo de encriptamiento Triple DES.

Como hemos visto, el sistema DES se considera en la actualidad poco práctico, debido a la corta longitud de su clave. Para solventar este problema y continuar utilizando DES se creó el sistema Triple DES (TDES), basado en tres iteraciones sucesivas del algoritmo DES, con lo que se consigue una longitud de clave de 128 bits, y que es compatible con DES simple.

Este hecho se basa en que DES tiene la característica matemática de no ser un grupo, lo que implica que si se encripta el mismo bloque dos veces con dos llaves diferentes se aumenta el tamaño efectivo de la llave.

Para implementarlo, se toma una clave de 128 bits y se divide en 2 diferentes de 64 bits, aplicándose el siguiente proceso al documento en claro:

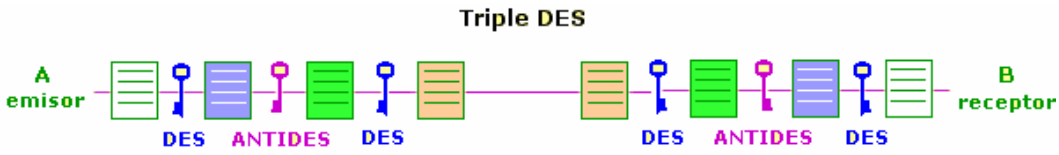


Figura 3.33 Esquema de cifrado TDES.

1. Se le aplica al documento a cifrar un primer cifrado mediante la primera clave, C1.
2. Al resultado (denominado ANTIDES) se le aplica un segundo cifrado con la segunda clave, C2.
3. Y al resultado se le vuelve a aplicar un tercer cifrado con la primera clave, C1.

Si la clave de 128 bits está formada por dos claves iguales de 64 bits (C1=C2), entonces el sistema se comporta como un DES simple.

Trás un proceso inicial de búsqueda de compatibilidad con DES, que ha durado 3 años, actualmente TDES usa 3 claves diferentes, lo que hace el sistema mucho más robusto, al conseguirse longitudes de clave de 192 bits (de los cuales son efectivos 168), mientras que el uso de DES simple no está aconsejado.

3.9 Seguridad.

Para los sistemas conectados en red, es muy importante evitar el acceso de intrusos que usen dichos sistemas para realizar ataques a la seguridad de terceros, debido a las posibles responsabilidades asociadas a dichos ataques. Aquí hablaremos sobre los mecanismos de protección de la privacidad que se utilizan en la WEB.

Seguridad y protección.

La seguridad de un sistema tiene múltiples facetas como aspectos de protección ante posibles daños físicos. La seguridad debe tener en cuenta eventos externos provenientes del entorno en que opera el sistema. La conexión de los sistemas a la red ha creado problemas de seguridad nuevos que van más allá del simple aspecto técnico y que incluyen aspectos éticos, legales, etc.

La protección consiste en evitar que se haga un uso indebido a los sistemas cuando se esta dentro del ámbito de uso del mimo. El sistema operativo, sus componentes y las aplicaciones que se usan para trabajar en red, deben proporcionar medios para implementar las políticas de protección deseada por el usuario.

La seguridad de un sistema conectado a una red se basa principalmente en tres aspectos:

- Controlar la privacidad de los datos.
- Controlar el acceso a los sistemas, sus datos, y recursos.
- Evitar la perdida de datos. Este aspecto cabe dentro de los mecanismos y políticas de administración de un equipo de cómputo y esté tema no lo tocaremos.

Algoritmo de cifrado.

Los algoritmos de cifrado son muchos y variados, en general, todo diseñador de un sistema de criptografía busca algoritmos nuevos y mejorados que los existentes.

Actualmente se usan algoritmos de cifrado muy complejos basados en la manipulación de números muy grandes y claves muy complicadas. Dos muy populares son:

- RSA que fue inventado en 1978 y ha sido seguro hasta la fecha. Incorpora resultados de la teoría de grandes números y con la determinación de números primos. Su implementación software proporciona un rendimiento muy pobre, por lo es muy frecuente el uso de hardware que lleva a cabo el cifrado con este tipo de algoritmo. Se usa en sistemas de claves públicas [27].
- DES (Data Encryption Standard), desarrollado por el US Nacional Institute of Standards and Technology para usarlo en aplicaciones comerciales. Para ello usa únicamente aritmética estándar y operaciones lógicas que permiten su implementación en cualquier computadora. A pesar de la complejidad del algoritmo, DES ya no es seguro. Además, ha sido cuestionado por tener debilidad en su diseño, clave muy cortas (56bits) y posibilitar la existencia de puertas traseras que permitirían a la Nationa Security Agency (NSA) descifrar cualquier objeto cifrado con este algoritmo. A pesar de todo, es un algoritmo muy popular y varias versiones del mismo son ampliamente usadas en la actualidad. Además para reforzar la seguridad del algoritmo, se ha incrementado la longitud de la clave [27].

Autentificar usuario.

La autentificación consiste en determinar si un usuario es quien dice ser. Para ello, la mayoría de los sistemas solicitan al usuario que previamente al acceso de los recursos del sistema, proporcione algún tipo de información que se supone que únicamente es conocido por él y que debe ser suficiente par su identificación.

Conexiones seguras en Internet.

Netscape propuso un protocolo de propósito general para enviar información a través de Internet. Este protocolo, denominado SSL se definió en 1994 por Netscape. La Familia de protocolos seguros en Internet se muestra en la Tabla 3.16.

Nombre	Función
https	http protegido con SSL
ssmtp	SMTP protegido con SSL para enviar correos
snews	News Usenet protegido con SSL.
Spop3	POP3 protegido con SSL para recibir correos
Ssl-ldap	LDAP protegido con SSL

Tabla 3.16 Familia de protocolos seguros en Internet.

3.10 Secure Socket Layer (SSL).

SSL permite la Confidencialidad y la Autentificación en las transacciones por Internet, siendo usado principalmente en aquellas transacciones en la que se intercambian datos sensibles, como números de tarjetas de crédito o contraseñas de acceso a sistemas privados. SSL es una de las formas base para la implementación de soluciones PKI (Infraestructuras de Clave Pública).

Historia.

Secure Socket Layer es un sistema de protocolos de carácter general diseñado en 1994 por la empresa Nestcape Communcations Corporation[27], y está basado en la aplicación conjunta de Criptografía Simétrica, Criptografía Asimétrica (de llave pública), certificados digitales y firmas digitales para conseguir un canal o medio seguro de comunicación a través de Internet. De los sistemas criptográficos simétricos, motor principal de la encriptación de datos transferidos en la comunicación, se aprovecha la rapidez de operación, mientras que los sistemas asimétricos se usan para el intercambio seguro de las claves simétricas, consiguiendo con ello resolver el problema de la Confidencialidad en la transmisión de datos.

SSL implementa un protocolo de negociación para establecer una comunicación segura a nivel de socked (nombre de máquina más puerto), de forma transparente al usuario y a las aplicaciones que lo usan.

Actualmente es el estándar de comunicación segura en los navegadores web más importantes (protocolo HTTP), como Nestcape Navigator e Internet Explorer, y se espera que pronto se saquen versiones para otras otros protocolos de la capa de Aplicación (correo, FTP, etc.).

La identidad del servidor Web seguro (y a veces también del usuario cliente) se consigue mediante el Certificado Digital correspondiente, del que se comprueba su validez antes de iniciar el intercambio de datos sensibles (Autenticación), mientras que de la seguridad de Integridad de los datos intercambiados se encarga la Firma Digital mediante funciones hash y la comprobación de resúmenes de todos los datos enviados y recibidos.

Desde el punto de vista de su implementación en los modelos de referencia OSI y TCP/IP, SSL se introduce como una especie de nivel o capa adicional, situada entre la capa de Aplicación y la capa de Transporte, sustituyendo los sockets del sistema operativo, lo que hace que sea independiente de la aplicación que lo utilice, y se implementa generalmente en el puerto 443.

(NOTA: Los puertos son las interfases que hay entre las aplicaciones y la pila de protocolos TCP/IP del sistema operativo).

SSL proporciona servicios de seguridad a la pila de protocolos, encriptando los datos salientes de la capa de Aplicación antes de que estos sean segmentados en la capa de Transporte y encapsulados y enviados por las capas inferiores. Es más, también puede aplicar algoritmos de compresión a los datos a enviar y fragmentar los bloques de tamaño mayor a 214 bytes, volviéndolos a reensamblarlos en el receptor.

Algoritmos Simétricos y asimétricos de encriptación.

La versión más actual de SSL es la 3.0. que usa los algoritmos simétricos de encriptación DES, TRIPLE DES, RC2, RC4 e IDEA, el asimétrico RSA, la función hash MD5 y el algoritmo de firma SHA-1.

Los algoritmos, longitudes de clave y funciones hash de resumen usados en SSL.

Los algoritmos, longitudes de clave y funciones hash de resumen usados en SSL dependen del nivel de seguridad que se busque o se permita, siendo los más habituales los siguientes:

- RSA + Triple DES de 168 bits + SHA-1: soportado por las versiones 2.0 y 3.0 de SSL, es uno de los conjuntos más fuertes en cuanto a seguridad, ya que son posibles $3.7 * 10^{50}$ claves simétricas diferentes, por lo que es muy difícil de romper. Por ahora sólo está permitido su uso en Estados Unidos, aplicándose sobre todo en transacciones bancarias.
- RSA + RC4 de 128 bits + MD5: soportado por las versiones 2.0 y 3.0 de SSL, permite $3.4 * 10^{38}$ claves simétricas diferentes que, aunque es un número inferior que el del caso anterior, da la misma fortaleza al sistema. Análogamente, en teoría sólo se permite su uso comercial en Estados Unidos, aunque actualmente ya es posible su implementación en los navegadores más comunes, siendo usado por organismos gubernamentales, grandes empresas y entidades bancarias.
- RSA + RC2 de 128 bits + MD5: soportado sólo por SSL 2.0, permite $3.4 * 10^{38}$ claves simétricas diferentes, y es de fortaleza similar a los anteriores, aunque es más lento a la hora de operar. Sólo se permite su uso comercial en Estados Unidos, aunque actualmente ya es posible su implementación en los navegadores más comunes.
- RSA + DES de 56 bits + SHA-1: soportado por las versiones 2.0 y 3.0 de SSL, aunque es el caso de la versión 2.0 se suele usar MD5 en vez de SHA-1. Es un sistema menos seguro que los anteriores, permitiendo $7.2 * 10^{16}$ claves simétricas diferentes, y es el que suelen traer por defecto los navegadores Web en la actualidad (en realidad son 48 bits para clave y 8 para comprobación de errores).
- RSA + RC4 de 40 bits + MD5: soportado por las versiones 2.0 y 3.0 de SSL, ha sido el sistema más común permitido para exportaciones fuera de Estados Unidos. Permite aproximadamente $1.1 * 10^{12}$ claves simétricas diferentes, y una velocidad de proceso muy elevada, aunque su seguridad es ya cuestionable con las técnicas de Criptoanálisis actuales.
- RSA + RC2 de 40 bits + MD5: en todo análogo al sistema anterior, aunque de velocidad de proceso bastante inferior.
- Sólo MD5: usado solamente para autentificar mensajes y descubrir ataques a la integridad de los mismos. Se usa cuando el navegador cliente y el servidor no tienen ningún sistema SSL común, lo que hace imposible el establecimiento de una comunicación cifrada. No es soportado por SSL 2.0, pero si por la versión 3.0.

La clave de encriptación simétrica es única y diferente para cada sesión, por lo que si la comunicación falla y se debe establecer una nueva sesión SSL, la contraseña simétrica se generará de nuevo.

SSL proporciona cifrado de alto nivel de los datos intercambiados (se cifran incluso las cabeceras HTTP), autenticación del servidor (y si es necesario también del cliente) e integridad de los datos recibidos.

Durante el proceso de comunicación segura SSL existen dos estados fundamentales, el estado de sesión y el estado de conexión. A cada sesión se le asigna un número identificador arbitrario, elegido por el servidor, un método de compresión de datos, una serie de algoritmos de encriptación y funciones hash, una clave secreta maestra de 48 bytes y un flag de nuevas conexiones, que indica si desde la sesión actual se pueden establecer nuevas conexiones. Cada conexión incluye un número secreto para el cliente y otro para el servidor, usados para calcular los MAC de sus mensajes, una clave secreta de encriptación particular para el cliente y otra para el servidor, unos vectores iniciales en el caso de cifrado de datos en bloque y unos números de secuencia asociados a cada mensaje.

Forma de saber si una conexión se está realizando mediante SSL.

Generalmente los navegadores disponen de un icono que lo indica, generalmente un candado en la parte inferior de la ventana. Si el candado está abierto se trata de una conexión normal, y si está cerrado de una conexión segura. Si hacemos debe hacer click sobre el candado cerrado nos aparecerá el Certificado Digital del servidor Web seguro.

Además, las páginas que proceden de un servidor SSL vienen implementadas mediante protocolo HTTP seguro, por lo que su dirección, que veremos en la barra de direcciones del navegador, empezará siempre por https, como por ejemplo:

https://www.htmlweb.net

Por último, cuando estamos en una conexión segura podemos ver el certificado del servidor acudiendo al menú "Archivo" del navegador y hacer un click en "Propiedades". En la parte inferior tenemos una opción "Certificados", que nos mostrará el del servidor actual.

Posibilidades para configurar SSL.

Podemos ver que hay varias posibilidades para configurar SSL. Por ejemplo:

- Cualquier cliente puede conectarse a un URL determinado, usando https. En este caso el servidor enviará su certificado al cliente para que este pueda descifrar la información que le llega del servidor y cifrar la que envía hacia el servidor.
- También podríamos hacer que sólo los clientes que tengan un determinado certificado puedan conectarse a una determinada URL.
- Otra posibilidad es combinar el primer ejemplo con las técnicas de autenticación que hemos visto antes. De forma que cuando intentemos acceder a una determinada URL usando https tendremos que autentificarnos primero.

Creando nuestra propia CA (Certification Authority)

Crear una llave privada de RSA para su servidor (sea Triple-DES cifrado):

$ openssl genrsa -des3 -out server.key 1024

Crear un certificado uno mismo-firmado (estructura X509) con la llave de RSA que usted acaba de crear:

$ openssl req -new -x509 -nodes -sha1 -days 365 -key server.key -out server.crt

Esto firma el CSR del servidor y los resultado es un archivo server.crt. Podemos utilizar el siguiente comando para ver los detalles del certificado:

$ openssl x509 -noout -text -in server.crt

En la figura 3.34 muestra el contenido de la pantalla de nuestro servidor para generar el Certificado.

```
[root@loky ssl.crt]# ls
respa  server.crt  server.key
[root@loky ssl.crt]# rm ser*
rm: remove regular file `server.crt'? y
rm: remove regular file `server.key'? y
[root@loky ssl.crt]# ls
respa
[root@loky ssl.crt]# openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.........++++++
.................................++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
[root@loky ssl.crt]# echo "sistemapacs"
sistemapacs
[root@loky ssl.crt]# echo "sistemapacs -como phrase:-"
sistemapacs -como phrase:-
[root@loky ssl.crt]# openssl req -new -x509 -nodes -sha1 -days 365 -key server.key -out server.crt
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:MX
State or Province Name (full name) [Berkshire]:DistritoFederal
Locality Name (eg, city) [Newbury]:Mexico
Organization Name (eg, company) [My Company Ltd]:UAM
Organizational Unit Name (eg, section) []:AZC
Common Name (eg, your name or your server's hostname) []:loky.uam.mx
Email Address []:armjih@hotmail.com
[root@loky ssl.crt]# openssl x509 -noout -text -in server.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=MX, ST=DistritoFederal, L=Mexico, O=UAM, OU=AZC, CN=loky.uam.mx/emailAddress=armjih@hotmail.com
        Validity
            Not Before: Mar 17 18:32:04 2006 GMT
            Not After : Mar 17 18:32:04 2007 GMT
        Subject: C=MX, ST=DistritoFederal, L=Mexico, O=UAM, OU=AZC, CN=loky.uam.mx/emailAddress=armjih@hotmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b6:92:cc:80:29:99:c6:53:ae:98:70:72:49:eb:
                    72:ab:cd:42:c8:65:dd:a4:8d:00:45:f9:a0:37:87:
                    b9:9b:12:4a:fd:05:73:c1:f0:cc:3f:85:f7:a7:5c:
                    e7:65:54:37:25:1b:0a:aa:aa:6a:b6:4f:af:0c:94:
                    c6:d3:cf:d8:39:7d:01:47:d0:54:34:0b:c7:62:16:
                    2f:d8:0b:ae:e9:3a:83:f9:7d:a2:95:e4:a9:5b:4f:
                    52:af:2b:15:b1:bf:e6:77:aa:9b:ab:dc:f4:a3:89:
                    6f:10:c8:3d:23:e9:7e:12:08:00:89:a5:1d:a9:d8:
                    de:6b:e2:56:2d:f5:92:63:09
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
            90:43:B4:E2:C6:E0:B8:3D:96:C7:FC:5A:8E:0D:FD:F5:BA:A3:9C:C7
            X509v3 Authority Key Identifier:
            keyid:90:43:B4:E2:C6:E0:B8:3D:96:C7:FC:5A:8E:0D:FD:F5:BA:A3:9C:C7
            DirName:/C=MX/ST=DistritoFederal/L=Mexico/O=UAM/OU=AZC/CN=loky.uam.mx/emailAddress=armjih@hotmail.com
            serial:00
            X509v3 Basic Constraints:
            CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
        27:e5:49:41:8a:a5:a2:af:30:4f:71:34:13:10:fe:48:fb:08:
        f2:e5:27:ee:c5:71:f7:e6:6e:fb:f3:b5:2f:81:df:31:3f:18:
        16:6d:09:60:08:0e:27:e8:67:c2:70:0d:60:4f:8b:19:e8:5e:
        ca:31:1d:bd:a0:90:c3:5a:4f:54:8f:42:de:e1:ad:1e:ee:0e:
        28:06:e3:92:a2:78:0b:40:27:b1:d6:2b:70:6d:d2:7c:8e:46:
        ae:84:f8:3b:56:07:6c:dd:b5:25:e0:a0:ea:75:81:d3:5e:f2:
        0c:b0:ee:cc:02:95:c9:47:34:8e:9e:7a:69:ce:6a:ea:6d:a4:
        71:84
[root@loky ssl.crt]# ls
respa  server.crt  server.key
[root@loky ssl.crt]#
```

Figura 3.34 Contenido de la pantalla del servidor para la generación del certificado.

4 Metodología y resultados.

Para poder alcanzar los objetivos para este proyecto contamos con cuatro etapas:

- Análisis.
- Diseño.
- Implementación.
- Pruebas.

En figura 4.1. Se muestra como están relacionadas cada etapa para el desarrollo de este proyecto.

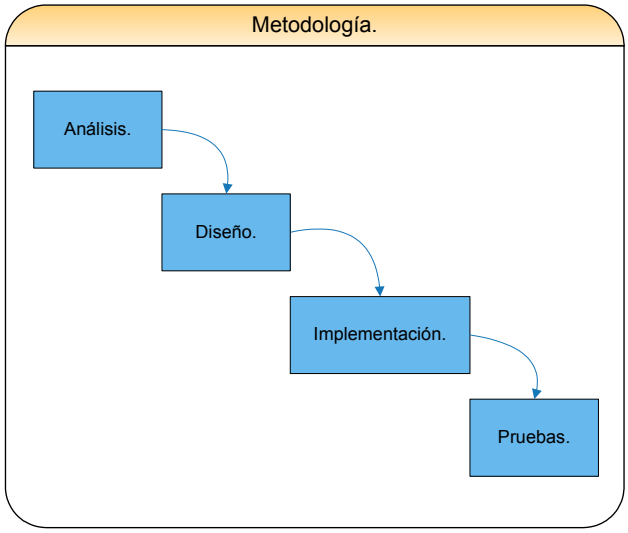


Figura 4.1 Metodología.

En la figura 4.2 se muestra el diagrama de flujo del desarrollo de este proyecto, los temas que se tuvieron que abordar en general. Por lo que a continuación desarrollaresmos cual fue su relación con la metodología.

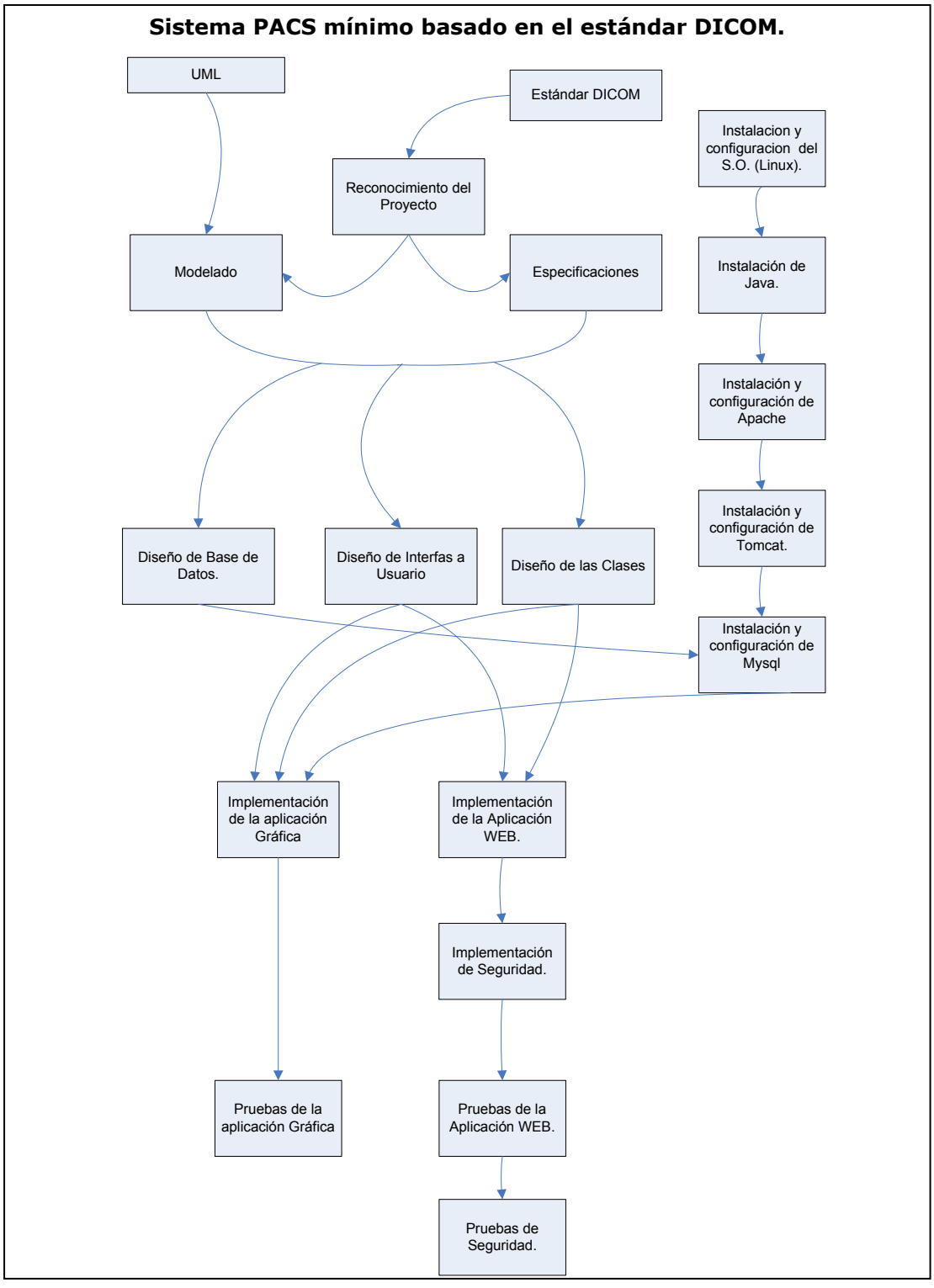


Figura 4.2 Diagrama de flujo del desarrollo del proyecto.

4.1 Análisis.

Esta etapa sirve para identificar las necesidades y las características que debe contar nuestro sistema. Ver figura 4.3. En base a los siguientes puntos:

- Reconocimiento del problema.
- Modelado.
- Especificaciones.

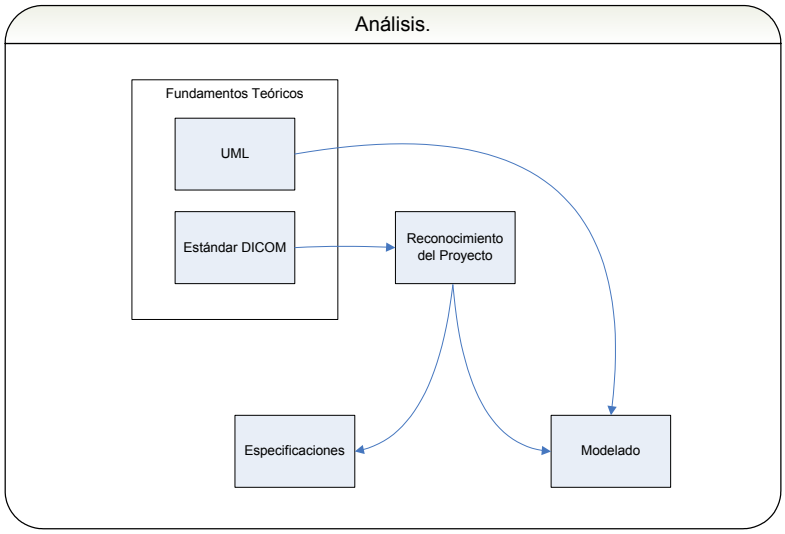


Figura 4.3 Diagrama de flujo para la etapa de Analisis.

Reconocimiento del problema.

En el punto 1.1 Fundamentos de un PACS hablamos de las características y sus componentes. Por lo que es importante el análisis de los modelos existente de visualizadores nos servira para poder entender que se esta haciendo y como se esta haciendo.

Análisis de los modelos existente de visualizadores.

Es necesario pode aprender de las experiencias relacionadas al tema por lo que es necesario poder hacer el ejercicio del método de Análisis de Decisiones. Enunciamos en que consiste el método y sus resultados.

Método de Análisis de Decisiones [44].

Se presenta un resumen del método para el Análisis de Decisiones. El cual utilizamos para obtener el mejor resultado

Definición del enunciado de la decisión.

Hay que tomar encuesta el propósito, que puede ser:

- Seleccionar/Determinar/Desarrollar.
- Mejor /optimo.
- Breve y simple.

- verdadera decisión/Efecto neto (no si/no).

El Nivel que es el precedente (¿hecho o supuesto?).

Enunciado de la Decisión es:

"Seleccionar el modelo base del visualizador que servirá de guía para nuestro Visualizador Simple".

Establecimiento de Objetivos

Los tipos de objetivos pueden ser:

- Resultados o productos.
- Recursos o insumos.
- Lista de control.
- Variable Única.

Nota: Solo colocar los objetivos necesarios en el caso que se requiera agregar o quitar hacerlo.

En este caso nuestros objetivos los podemos ver es la tabla 4.1

Num.	Objetivos
1	El Lenguaje de programación es Java
2	Tememos Acceso a los programas fuentes
3	Licencia Libre

Tabla 4.1 Objetivos establecidos.

Evaluación de Objetivos.

La evaluación de los objetivos consiste en ponderarlos de 10 a 1 según su importancia relativa. En caso de tener un objetivo Imperativo ponerlo como límite Absoluto. En la tabla 4.2 podemos ver los valores de ponderación de cada objetivo.

Num	Objetivos	Ponderación
2	Tememos Acceso a los programas fuentes	10
1	El Lenguaje de programación es Java	9
3	Licencia Libre	8

Tabla 4.2 Ponderación de los objetivos.

Generación de Opciones.

Generar nuestras opciones en a la selección de estándar; en desarrollo si es creación o diseño, tener dos o más opciones a evaluar.

Comparación y Selección.

Comparación y Selección en base a la Escala de satisfacción la Puntuación es de 10 a 0 ó eliminador/Imperativo; Valores ponderados por puntuaciones: Puntuaciones ponderadas Totales ¿Segunda Evaluación? ¿Es Real el imperativo? Relatividad.

Nota: Para la mejor opción debe tener mas del 90 % de laPuntuación Pondera Total

Resultado de la decisión.

En base a estos resultados nos apegaremos dicomviewer054 para adquirir su conocimiento y experiencia sobre la forma de visualizar un IOD-compuesto. Ver tabla 4.3 y 4.4

Enunciado de la decisión:		
Seleccionar el modelo base del visualizador que servirá de guía para nuestro Visualizador Simple		
Opciones		
A	dicomviewer054	
B	DCMJK	
C	EviewBox	
D	OSIRIS	
E	ImageJ	
Objetivo:	10	Tememos Acceso a los programas fuentes
Escala de Satisfacción		PUNTUACIÓN
documentación en español o ingles		8
documentación en otro idioma		6
ejemplo de compilación y sencillo		2
Objetivo:	9	El Lenguaje de programación es Java
Escala de Satisfacción		PUNTUACIÓN
java		10
Objetivo:	8	Licencia Libre
Escala de Satisfacción		PUNTUACIÓN
Licencia Libre		10
con licencia para utilizarlo		0

Tabla 4.3 Síntesis de este análisis.

OBJETIVOS	VP	OPCIONES									
		dicomviewer054		DCMJK		EviewBox		OSIRIS		ImageJ	
		ES	ESP ES	ES	ESP ES	ES	ESP ES	ES	ESP ES	ES	ESP ES
Tememos Acceso a los programas fuentes	10	8	80	8	80	0	0	8	80	6	60
El Lenguaje de programación es Java	9	10	90	10	90	10	90	0	0	10	90
Licencia Libre	8	10	80	0	0	10	80	0	0	10	80
Puntuación Ponderada TOTAL		**250**			**170**		**170**		**80**		**230**
Puntuación Ponderada TOTAL (%)		**92.59%**			**62.96%**		**62.96%**		**29.63%**		**85.19%**
Puntuación Ponderada TOTAL (ideal)		**270**									

Tabla 4.4 Comparación y selección.

Modelado.

En este punto desarrollamos:

- Los casos de usos.
- La arquitectura propuesta para este sistema.

Casos de uso

Para su modelado utilizamos casos de uso.

Los actores son:

- Archivo IOD Compuesto. Archivo bajo el estándar DICOM.
- Archivo diccionario de Tags DICOM.- Generará el diccionario de Tags y este contiene toda el tag, VR, VM y versión .
- Usuario.

Los Casos de uso son:

- Comun_Mysql.
- DicomData.
- DicomDic.
- DicomFile.
- DicomvalueData.
- DicomValueDic.
- ImageData.
- ImageProcessor.
- JFrameSobreSistema.
- JPanelErrores.
- jtable_3.
- Principal.
- Vectores.
- Visualizador.
- Consultar.
- Comunicación.
- Sesion.

En la figura 4.4 se muestra el diagrama de Casos de uso para nuestro sistema. Y son la base para generar las clases de la tabla 4.5.

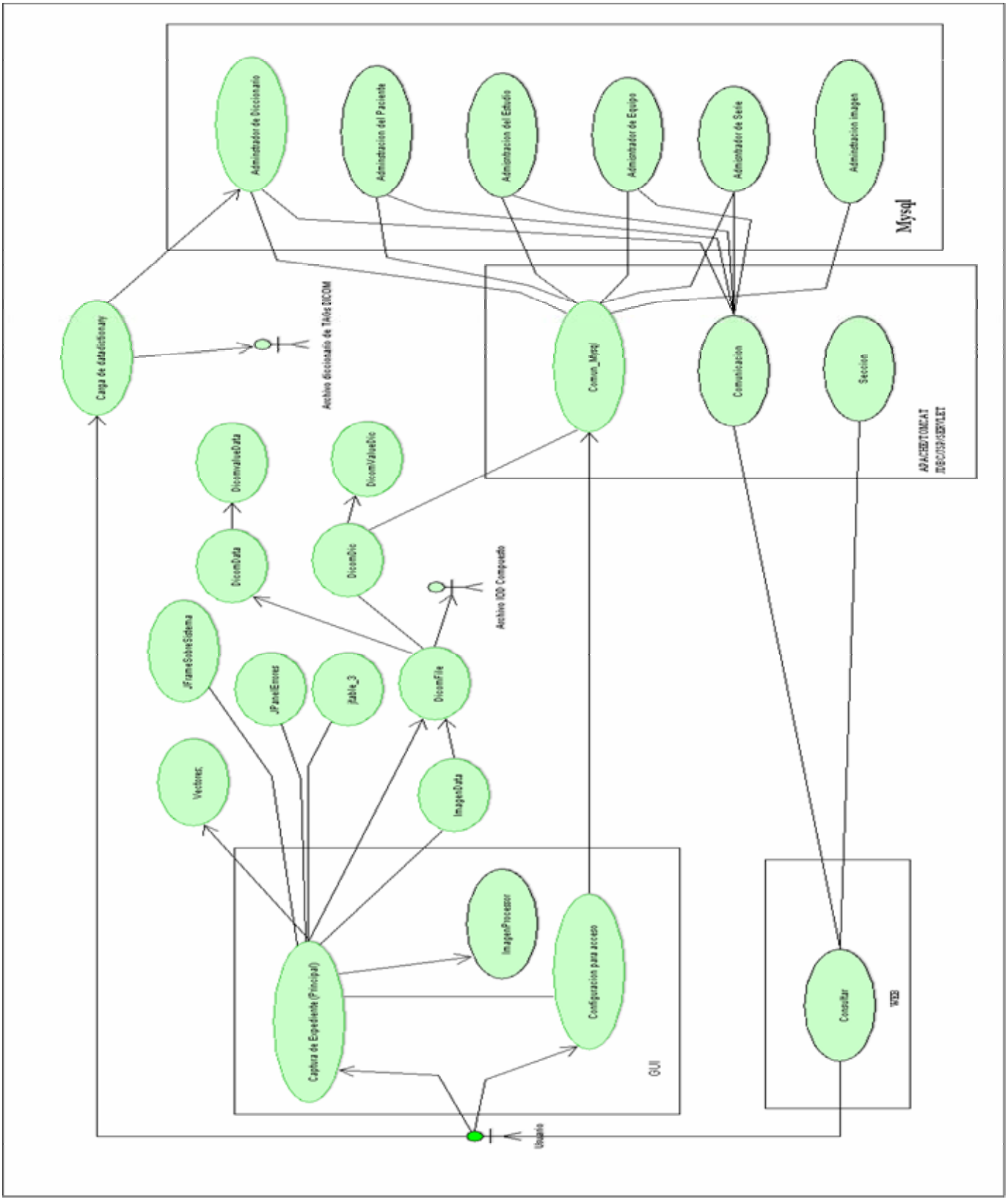


Figura 4.4 Diagrama de Casos de Uso de nuestro sistema.

Las clases para la adquisición de imágenes bajo el estándar DICOM.

En la tabla 4.5 mostramos las clases para la adquisición de imágenes bajo el estándar DICOM que cresponden al diagrama de la figura 4.4.

Clase	**Descripcion**
Comun_Mysql	Comunica con MYSQL para pedir datos y sacarlos.
DicomData	Crea una tabla para que contenga: tag, name, vr, vm, versión, value, valueLength, analyzedValue.
DicomDic	Crea una tabla del diccionario de tags.
DicomFile	Se encarga de manejar lo relacionado al archivo del IOD Compuesto.
DicomValueData	Estuctura del objeto.
DicomValueDic	Estructura del objeto para el Diccionario DICOM.
ImageData	Esta clase es para poder tener la información de los datos de la Imagen.
ImageProcessor	Esta clase controla el procesamiento de la imagen.
JFrameSobreSistema	Esta clase nos permite mostrar información de quien creo este sistema y poderlo controlar como Jframe.
JPanelErrores	Esta clase genera un JPanel para los posibles errores.
Jtable_3	Esta clase maneja la información para la tabla y la construye.
Principal	La clase Principal controla la presentación a usuario y contiene la estructura general lo que se muestra.
Vectores	Esta clase es de manejo de vectores se considera que el vector es de la forma vector[][] Nota: se asume que el valor del tag es único.
Visualizador	Esta clase genera un JFrame para poder ver la imagen.

Tabla 4.5 Clases para la adquisición de imagen y visualización.

En consulta y visualización de la imagen utilizaremos un navegador y el servidor de Web se encargara de realizar la comunicación con la base de datos y control de sesión.

Arquitectura del Sistema.

En este punto hablaremos de: Arquitectura de software; Arquitectura Cliente-Servidor; Arquitectura del Sistema.

Arquitectura de software.

La arquitectura software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura software debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea de computación. -La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.- ***Kruchten, Philippe***.[3]

Toda arquitectura software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Las vistas o modelos de una arquitectura pueden expresarse mediante uno o varios lenguajes. El más obvio es el lenguaje natural, pero existen otros lenguajes tales como los diagramas de estado, los diagramas de flujo de datos, etc. Estos lenguajes son apropiados únicamente para un modelo o vista. Afortunadamente existe cierto consenso en adoptar UML (*Unified Modeling Language*, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas.

Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

- Monolítica. Donde el software se estructura en grupos funcionales muy acoplados.
- Cliente-servidor. Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- Arquitectura de tres niveles. Generalización de la arquitectura cliente-servidor donde la carga se divide en tres partes con un reparto claro de funciones: una capa para la presentación, otra para el cálculo y otra para el almacenamiento. Una capa solamente tiene relación con la siguiente.

Otras arquitecturas menos conocidas son:

- En pipeline.
- Entre pares.
- En pizarra.
- Orientada a servicios.

Arquitectura Cliente-Servidor.

La arquitectura cliente-servidor llamado modelo cliente-servidor o servidor-cliente es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permita simplificar las actualizaciones y mantenimiento del sistema.

En esta arquitectura la capacidad de proceso está repartida entre el servidor y los clientes.

En la funcionalidad de un programa distribuido se pueden distinguir 3 capas o niveles:

- Manejador de Base de Datos (Nivel de almacenamiento).

- Procesador de aplicaciones o reglas del negocio (Nivel lógico).
- Interfaz del usuario (Nivel de presentación).

En una arquitectura monolítica no hay distribución; los tres niveles tienen lugar en el mismo equipo.

En un comienzo, los mainframes concentraban la funcionalidad de almacenamiento y lógica y a ellos se conectaban terminales tontas, posiblemente ubicadas en sitios remotos.

De acuerdo con la distribución de la lógica de la aplicación hay dos posibilidades:

- Cliente delgado.- si el cliente solo se hace cargo de la presentación.
- Cliente pesado.- si el cliente asume también la lógica del negocio.

Las ventajas de la arquitectura cliente-servidor son:

- El servidor no necesita tanta potencia de procesamiento, parte del proceso se reparte con los clientes.
- Se reduce el tráfico de red considerablemente. Idealmente, el cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando la red libre.

Arquitectura de nuestro Sistema .

La Arquitectura de este sistema es del tipo cliente/servidor. El servidor dará servicio de HTML utilizando SSL/Apache con Tomcat, se crearon las paginas Web Dinámicas por lo que se utiliza JSP/Servlet y para el manejo de datos se utiliza JDBC y Mysql. Ver figura 4.5.

El cliente tiene la aplicación el navegador, el visualizador, y la aplicación de carga de imagen a expediente.

El equipo de adquisición de imagen (DICOM) entregara la información bajo el estándar DICOM. Utilizando IOD (Information object definition) compuesto.

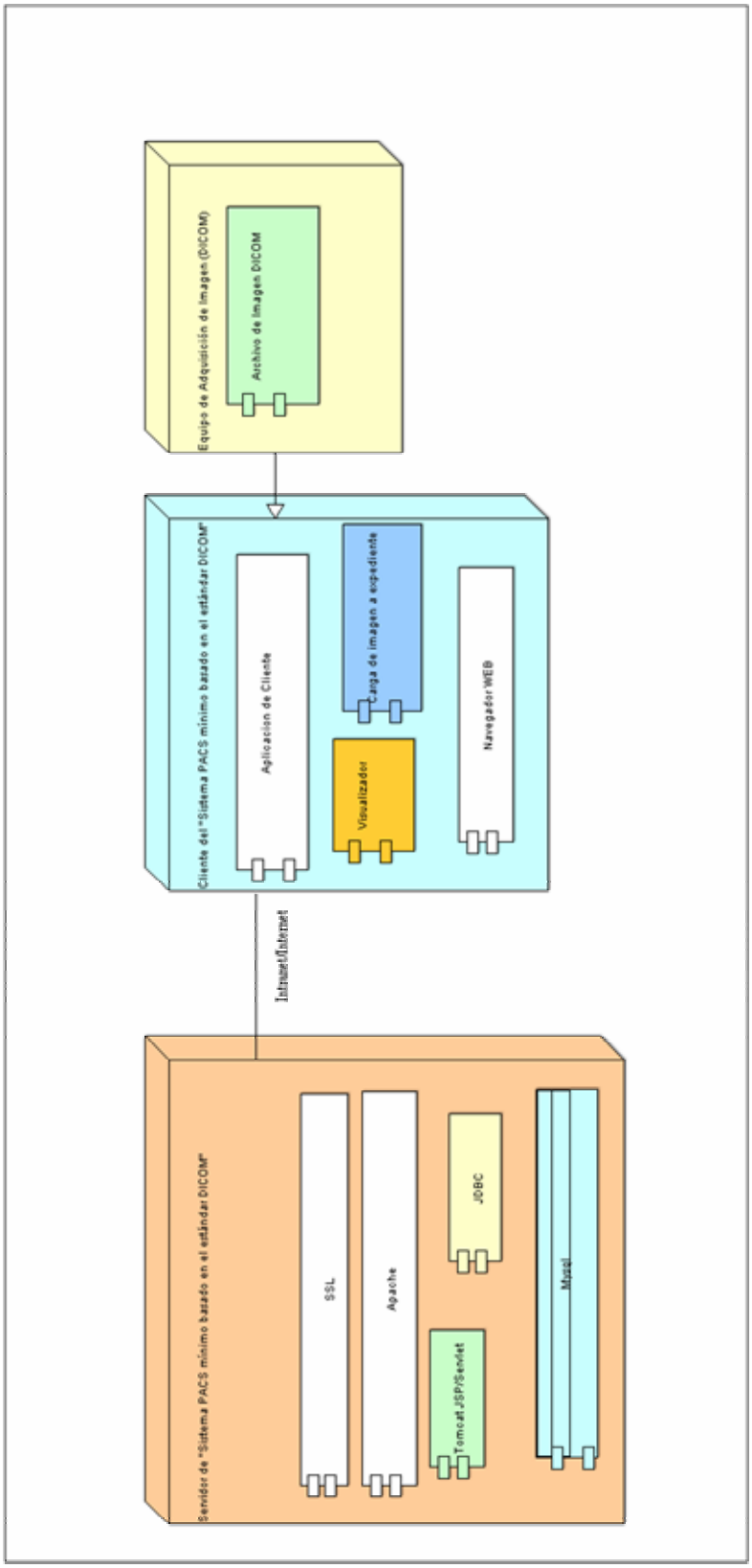


Figura 4.5 Arquitectura del sistema.

Especificaciones.

En esta parte hablaremos de las especificaciones de nuestro sistema. Para ello desarrollaremos los siguientes puntos:

- Adquisición de imágenes.
- Red de comunicación.
- Bases de datos y Sistemas de Almacenamiento.
- Estaciones de Diagnóstico y Visualización.
- Especificaciones de hardware y software del Sistema.

Adquisición de imágenes.

El sistema adquirirá la información e imagen por medio de un archivo que contiene el IOD Compuesto. Por lo que se debe crear una Clase (Principal) que puede leer esta información y poderla subir a la Base de datos. Dado que contará con un diccionario de tags utilizados por DICOM podremos abrir y ver cualquier tipo de estudio.

Red de comunicación.

Nuestra comunicación se apegará a las características de comunicación que cuenta nuestra universidad y al equipo que está designado para el desarrollo del proyecto. La programación de nuestro sistema estará enfocada a Internet red Ethernet 10/100 en UTP.

Bases de datos y Sistemas de Almacenamiento.

Utilizaremos Mysql para administrar nuestra base de datos, la capacidad de almacenamiento dependerá de la características del equipo que se designa para este proyecto, y utilizamos JDBC que permite a cualquier programa Java acceder al sistema de Base de Datos de forma homogénea.

Estaciones de Diagnóstico y Visualización.

El usuario podrá utilizar el Navegador por lo que esta aplicación deberá soportar páginas dinámicas. Y se implementará SSL para permitir la Confidencialidad y la Autentificación en las transacciones por Internet hechas por nuestro sistema.

Especificaciones de hardware y software del Sistema.

El quipo a utilizar para el desarrollo de este proyecto deberá apegar a las siguientes características (ya que es el equipo que se cuenta para el desarrollo de este proyecto):

Para realizar la documentación del proyecto será una Computadora Personal con sistema operativo Windows XP y Office.

Sistema Operativo: Windows XP.

Paquetería y Herramientas de desarrollo de Software: Office, WinMysql, Mysql Administrador, NetBeans, Java, DB Designer 4, ArgoUML (ver figura 4.6).

El quipo donde se deberá desarrollar el proyecto:

Sistema Operativo: Centos 4.2 (Linux).

Paquetería: Apache, Tomcat, Mysql, NetBeans, Java.

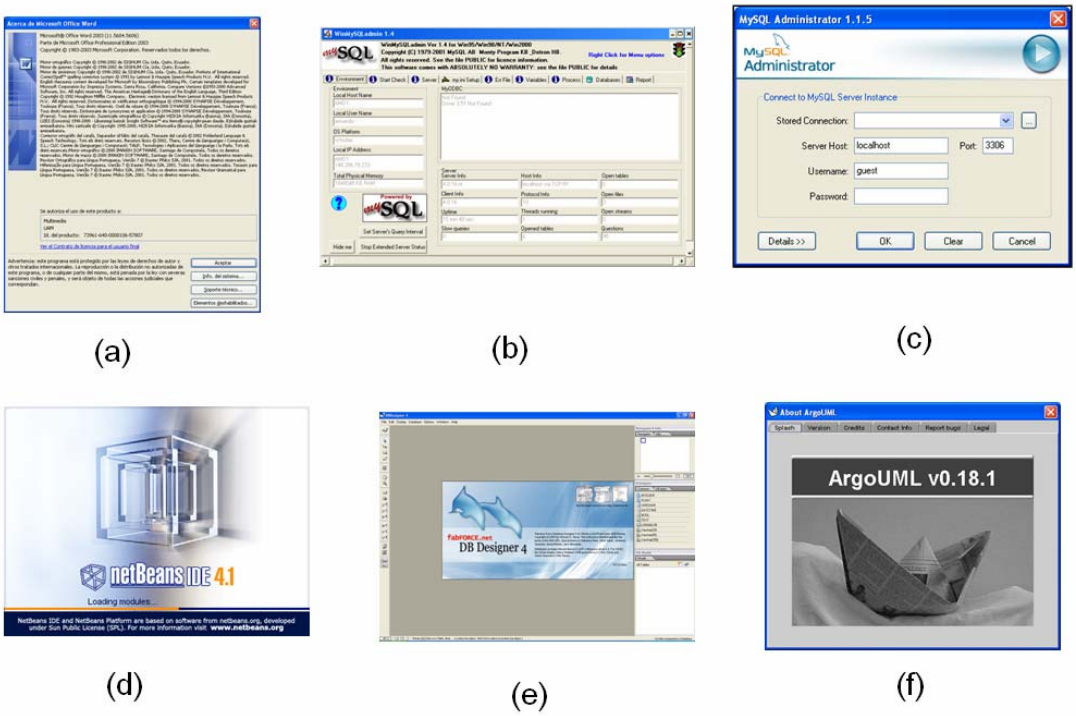


Figura 4.6 Paquetería y Herramientas de desarrollo de Software: (a) Office, (b)Mysql (c)MySQLAdministrator (d) netBeans, (e) DB Designer 4, (f) ArgoUML.

4.2 Diseño.

En esta etapa tiene el propósito de definir al sistema, con suficiente detalles para su interpretación y realización física. Ver figura 4.7.

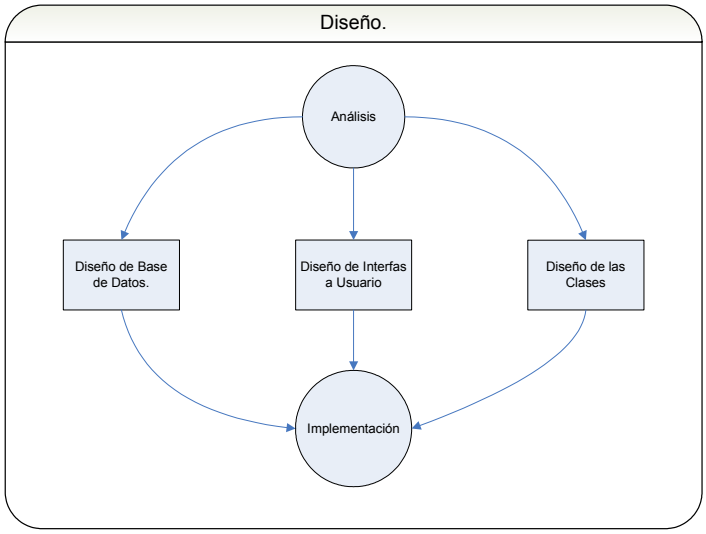


Figura 4.7 Diagrama de flujo de Diseño del sistema.

Describiremos el diseño de base de datos, diseño de la interfaz a usuarios y el diseño de las clases.

Diseño de Base de Datos[46].

El modelo de Base de Datos Relacionales evita la redundancia de la información, poder compartir o bloquear información, permitiendo la disponibilidad de recursos, y podemos tener un mejor control de la administración de la información.

Un modelo de Datos es la representación gráfica del mundo real que nos permite: organizar la información mediante estructuras de dados, permitiendo la definición de unidades de datos y sus relaciones, por lo que podemos plantear y comunicar nuestra idea de la Base de Datos.

El modelo entidad-relación fue formalizada por C.W. Bachean a fines de los 60' y sus principales caracteristicazas son: Poder esquematizar las relaciones entre entidades, manejar mucha información de manera sencilla, facilitando su entendimiento y documentación, y el mapeo es directo en la construcción de una Base de Datos Relacionales.

El método de Diseño de una base de datos es:

- Identificación y definición de los objetos principales dentro del sistema.
- Diagrama Entidad Relación
- Resolución del Modelo Lógico.
- Normalización del modelo lógico.
- Conversión del modelo lógico a un esquema físico.

Identificación y definición de los objetos principales dentro del sistema.

En este punto hablaremos sobre que es: entidad, Instancia de las entidades y relación.

Entidad.

La declaración de Entidades es el objeto de interés principal o importante para el usuario dentro del sistema. Para este caso tenemos que son:

- Diccionario.
- Paciente.
- Estudio.
- Serie.
- Imagen.

Instancia de las entidades.

Es Información que almacenamos de la entidad. Podemos ver ejemplos en las siguientes tablas: 4.7 – 4.11.

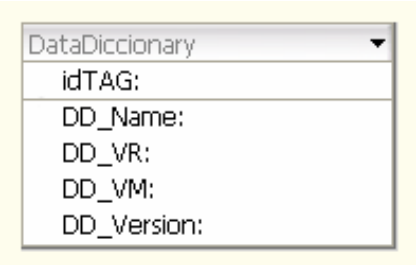


Tabla 4.6 Diccionario DICOM.

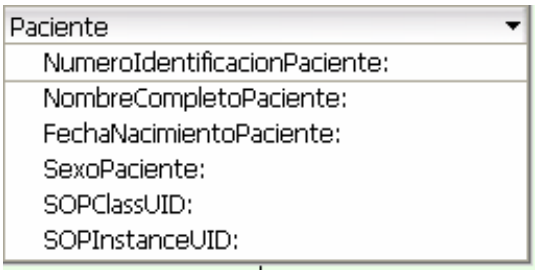


Tabla 4.7 Paciente.

Estudio
IdentificadorInstanciaEstudio:
FechaEstudioComenzo:
HoraEstudioComenzo:
NombreMedico:
IdentificadorEstudio:
IdentificadorEstudiGeneradorPorRIS:
IdentificadorClaseSOP:
IdentificadorInstanciaClaseSOP:

Tabla 4.8 Estudio.

Equipo
FabricanteEquipo:

Tabla 4.9 Equipo.

Serie
IdentificadorSerie:
TipoEquipoUsadoTomarEstudio:
NumeroSerie:
LateralidadParteCuerpoExaminada:
IdentificadorClaseSOP:
IdentificadorInstanciaClaseSOP:
PosiciónPacienteRespectoEquipo:
IdProcedRequeridoServicioImagen:
IdentificadorProcedimientoProgramado:
IdProcedAplicadoInstanciaSOPRelacionadaSeri...
IdentificaPropósitoIimágenesDentroSerie:
IdentificaCuadroReferenciaSerie:
ParteAnatomíaPacienteUsadaReferencia:

Tabla 4.10 Serie.

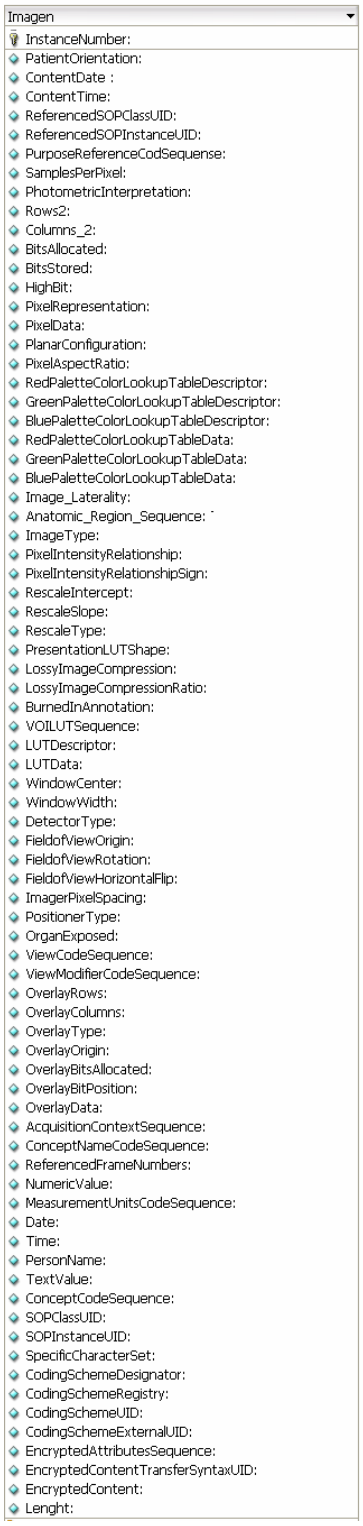


Tabla 4.11 Imagen.

Conversión del modelo lógico a un esquema físico.

El mapeo del modelo lógico al modelo físico se utilizó la información de la Tabla 3.5.

En la figura 4.8 muestra el modelo de las tablas para el control de usuarios para aplicaciones Web. La tabla 4.12 sirve para poder implementar nuestra base de datos y el modelo que utilizamos se muestra en la figura 4.9 .

Figura 4.8 Modelo de la base de datos para el control de sesión.

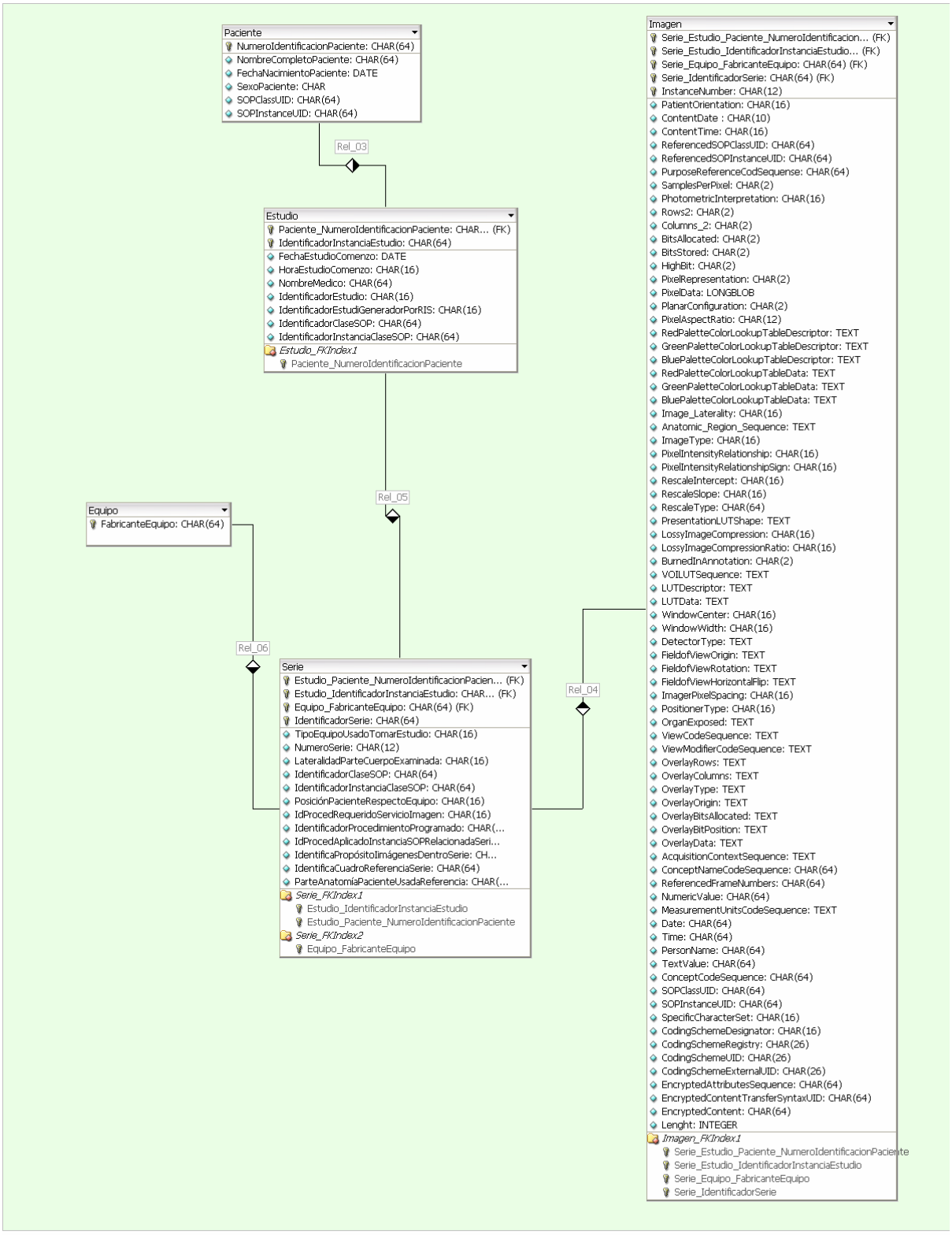


Figura 4.9 Modelo del Sistema.

Diseño de la interfaz a usuario.

La interfaz a usuario es:

- Gráfica para adquisición de imágenes.
- Web para visualización de imágenes.

Interfaz gráfica

En la etapa de adquisición de imagen se utiliza una interfaz gráfica la cual esta formada por tres partes. Ver figura 4.10.

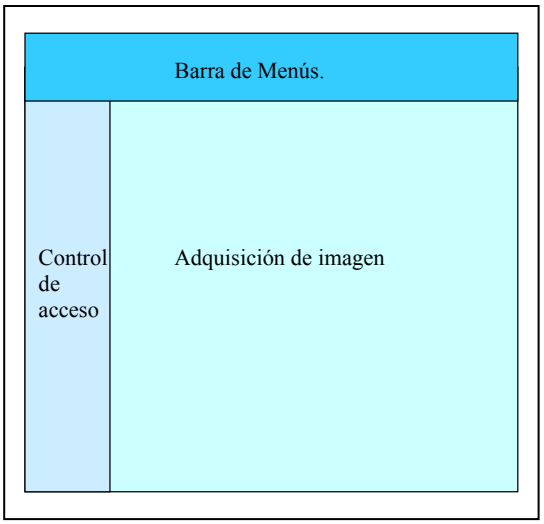


Figura 4.10 Interfaz gráfica.

- Barra de menú. La barra tiene dos menús: el primero tiene dos opciones el de inició que inicializará toda la aplicación y el segundo de salir; El Segundo dará la información de l titulo del sistema, quien lo hizo y el nombre del Asesor.
- Control de Acceso Este panel pondremos el login y password para poder acceder a la base de datos.
- Despliegue de información.- Estar formado por tres paneles el primero seleccionaremos el archivo a procesar; el segundo procesara la información este mostrara en forma de tabla los tags que contiene el archivo y se habilita la opción de editar el contenido del tag y al mostrara la imagen esta tiene la opción darle un poco mas de brillo, hacerla mas oscura, poderla girar y poderla verla a una escala de reducción; y el tercero guardara la información en la base de datos.

Interfaz WEB

Para la etapa de visualización y consulta de la imagen la pantalla esta dividida en tres partes (Ver figura 4.11):

Cabecera.- Esta muestra el título del proyecto, alumno y nombre del Asesor.

Menú de Opciones. Este muestra lo que podemos hacer desde la Web que es iniciar sesión, consulta de la tabla de imágenes, buscar un todas las imágenes relacionadas al paciente, visualizar la imagen así como la información asociada a la misma; y finalizar sesión.

Despliegue de información.- En esta parte de la pantalla mostrara todo lo relacionado a lo que mande a ejecutar del menú.

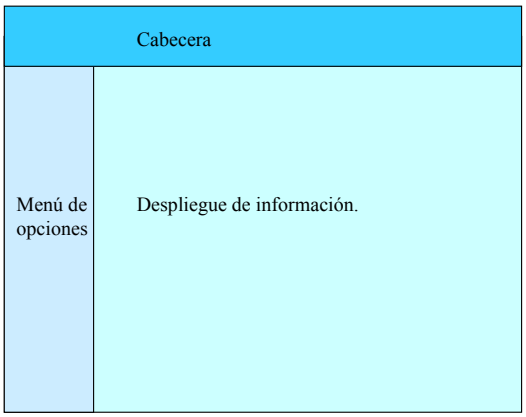


Figura 4.11 Interfaz WEB Visualización.

Diseño de las Clases.

En el anexo A podemos ver las clases para la interfaz gráfica, en este podemos ver las características de cada clase así como los métodos utilizados.

4.3 Implementación.

En esta parte hablaremos de: Implementar la aplicación Cliente/Servidor; Implementar la aplicación para el almacenamiento de información en un dispositivo de almacenamiento secundario; Técnicas de encriptamiento; Implementar las pruebas de validación; Implementación del API. Ver figura 4.12.

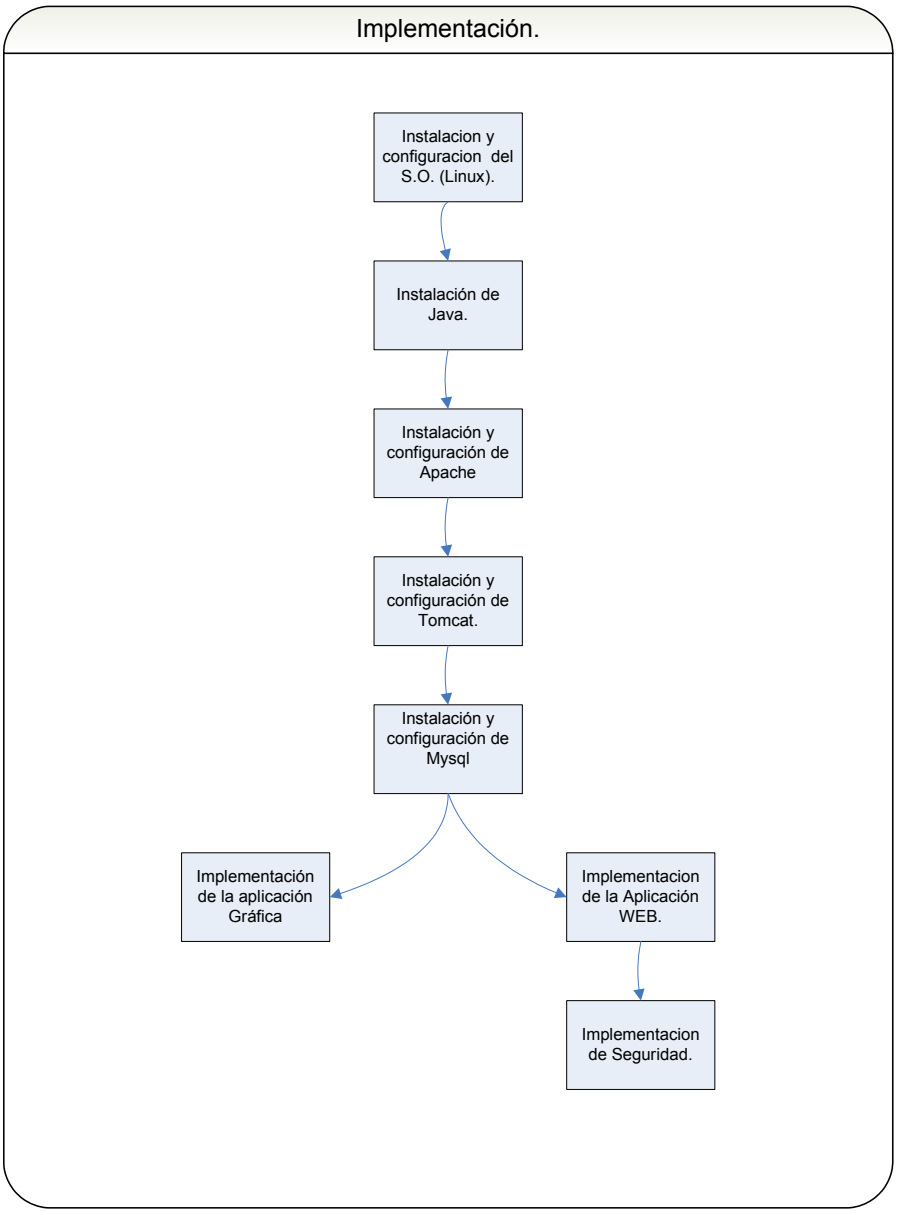


Figura 4.12 Diagrama de flujo de Implementación.

Implementar la aplicación Cliente/Servidor.

En este punto desarrollaremos: Instalación de CentOS para el Servidor; Instalación de JDK, Apache y Tomcat.

Instalación de CentOS para el Servidor.

CentOS (**C**ommunity **ENT**erprise **O**perating **S**ystem) es un clon a nivel binario de la distribución Red Hat Enterprise Linux, compilado por voluntarios a partir del código fuente liberado por Red Hat.

Requerimientos de Computo.

Este información es en base al equipo que tenemos a nuestra disposición para desarrollar este proyecto.

- Hardware recomendado para operar:
- Memoria RAM: 192 MB (Mínimo).
- Espacio en Disco Duro: 850 MB (Mínimo) - 2 GB (Recomendado).
- Procesador: Intel Pentium I/II/III/IV/Celeron, AMD K6/II/III, AMD Duron, AMD Athlon/XP/MP .

Proceso de Instalación.

Arranque de CD-ROM : Primeramente debe asegurarse que su BIOS se encuentre configurado para leer el CD-ROM al momento de inicializarse su PC o Servidor, esto es necesario para que se ejecute el "shell" de instalación CentOS en lugar del sistema operativo que tiene instalado en su disco duro.

NOTA: El acceso al BIOS varía dependiendo del Hardware que utilice, sin embargo, las opciones más comunes son las teclas ESC, F2 o DEL.

Arranque de CentOS : Inicialice su sistema con el Disco 1 de CentOS colocado en su lector de CD-ROM. Si configuró correctamente su BIOS para leer CD-ROM's al arranque, debe observar una pantalla como la siguiente:

Arranque.

Proceso inicial y detección de Hardware: Estando en la pantalla anterior, simplemente esperando u oprimiendo Enter, iniciará el proceso para detección de Hardware y proceso de instalación a través de una pantalla gráfica , esta secuencia puede durar entre 10 o 15 segundos, mientras observa mensajes descriptivos sobre su sistema. Si desea realizar su proceso de instalación mediante línea de comandos, entonces deberá introducir la secuencia linux text.

Verificación de medios (CD-ROM's) : Posteriormente será presentado con la opción de realizar una prueba de integridad sobre los CD-ROM's de instalación CentOS, esta prueba dura entre 10 y 15 minutos para los 4 CD's de instalación. Si no desea realizar esta prueba, seleccione la opción Skip.

La pantalla es la siguiente:

Verificación CDs.

Bienvenida: Es desplegado un mensaje de bienvenida de CentOS con instrucciones de navegación, simplemente oprima Next para continuar con el proceso.

Lenguaje del Sistema: La siguiente selección que debe hacer es sobre el idioma que desea para su proceso de instalación y sistema, puede elegir español (Spanish):

Idioma.

Configuración de Teclado: Seleccione el tipo de teclado en base a su sistema; español si posee un teclado con tecla eñe y caracteres latinos, o bien, estadounidense de lo contrario.

Tipo de Instalación (Actualización, Escritorio personal, Estación de Trabajo, Servidor, Personalizada): Seguido, será realizado un proceso de auto-detección para determinar si posee

una instalación pre-existente de CentOS, de encontrarse será presentado con una opción de actualización, de lo contrario, podrá elegir entre 4 modalidades -- Escritorio Personal, Estación de Trabajo, Servidor o Personalizada -- cada una de éstas presenta una breve descripción de su funcionamiento. Para el caso de esta guía, se asumirá una configuración de Servidor, ya que CentOS se caracteriza por su estabilidad en esta área. A continuación se muestra esta pantalla:

Selección de Aplicaciones.

Partición de Disco Duro: Posteriormente, debe realizar el particionamiento de su disco duro, CentOS le ofrece dos alternativas para llevar acabo este proceso. (NOTA: Es recomendable lea el proceso de Pre-Instalación en caso que no este familiarizado con el concepto de particionar discos duros).

o Particionamiento Automático: Como su nombre lo implica, CentOS realiza el particionamiento del disco duro a dimensiones pre-determinadas, sin embargo, esto implica generalmente que debe borrar toda la información existente en su disco duro.

o Partición manual con Disk Druid: Para usuarios con amplio conocimiento del proceso de partición, pueden optar por hacer su propia distribución de espacio con esta opción.

La pantalla es la siguiente:

Particionar.

Se recomienda seleccionar Particionamiento Automático puesto que implica el método más directo y sencillo para instalar CentOS, no obstante, si utiliza la opción de Disk Druid tome en cuenta los factores de redimensionamiento que utilice para sus particiones. La principal ventaja de utilizar la opción automática, se debe a que las particiones son creadas en el orden y tamaño recomendado por CentOS para operación óptima.

Administrador de Arranque ("Boot Loader") : Posteriormente debe confirmar la instalación del administrador de arranque ("Boot Loader") GRUB; si CentOS será el único sistema operativo instalado en su equipo, este paso no debe ser de mayor trascendencia. Sin embargo, si posee más de un disco duro, o bien, además de CentOS existirá otro sistema operativo, esta configuración tiene implicaciones en la manera que es inicializado su sistema. La pantalla es la siguiente:

GRUB Instalación.

Configuración de Red: Seguido debe indicar los parámetros para acceso a red, ya sea manualmente con información de nodos IP y DNS, o bien, indicando una configuración automática vía DHCP.

Configuración Cortafuegos ("Firewall"): Aquí debe especificar si desea instalar un mecanismo de "Firewall" para proteger su sistema. De ser así, también tiene la opción de habilitar determinados servicios para que éstos no sean afectados por el "Firewall", tales como: SSH, Servidores Web, Servidores de Correo y FTP. La pantalla es la siguiente:

Firewall.

Idiomas adicionales para Sistema: Esta pantalla le permitirá definir idiomas alternos para su sistema, además del seleccionado en primeras instancias:

Idiomas Adicionales.

Zona Horaria del Sistema: Aquí debe definir la zona horaria para su instalación. La pantalla es la siguiente:

Firewall.

Definición de adminsitrador (root): Posteriormente debe indicar una contraseña para el administrador ("root") del sistema, como su nombre lo indica, éste será el usuario maestro de

la instalación y tendrá control absoluto de acceso sobre toda aplicación en CentOS. La pantalla se muestra a continuación:

Root.

Selección de Aplicaciones/ Paquetes : En esta pantalla tiene la opción de elegir la serie de aplicaciones que serán instalados, por "default" se encontraran seleccionados una serie de paquetes que corresponden a una configuración típica de servidor, esto debido a que con anterioridad fue seleccionado este tipo de instalación. No obstante, puede agregar aplicaciones a su discreción, tales como un ambiente gráfico u otra función necesaria para cumplir con sus requerimientos.

Aplicaciones.

Selección de Aplicaciones/ Paquetes: Seleccionadas las aplicaciones, al oprimir el botón Siguiente iniciará la instalación de aplicaciones, dependiendo de su Hardware, este paso puede demorar entre 20 o 40 minutos. Asegúrese también de tener cerca de su PC los discos de instalación y no dejarla desatendida, ya que durante este proceso necesitará colocar los distintos CD-ROM's conforme los requiera su instalación. Se ha terminado de instalar satisfactoriamente CentOS Linux. Ahora sólo debe reinicializar su sistema para entrar a su ambiente Linux . La pantalla se muestra a continuación (figura 4.13.):

Figura 4.13 Pantalla de instalación de CentOS.

Instalación de APACHE.

Descargar Apache 2.0.55 de
Desempaquetar este en /usr/local/src
Vamos a instalar Apache en /usr/local/apache
Nos vamos al directorio donde están los fuente de apache y lo instalaremos:

cd /usr/local/src/httpd-2.0.55

./buildconf

./configure --prefix=/usr/local/apache --enable-ssl --enable-so –with-mod_jk

Make

Make install

Verificamos que esta correcta nuestra instalación y la respuesta será de tipo –Systax OK-

/usr/local/apache/bin/apachectl configtest

El comando para activar apache es:

apachectl start

NOTA: startssl es para que pueda correr sobre SSL.

Verificación que esta corriendo apache por medio del navegador podemos visualizar la página

Shutdown de apache por medio de:

apachectl stop

Instalación de Java.

Descargar el jdk-1_5_0_06-nb-4_1-linux-ml que se encuentra en www.java.sun.com
y dejarlo en el directorio correspondiente,
Revisar los permiso (755) para después ejecutarlo
Ejecutamos el instalador

./jdk-1_5_0_06-linux-i586.bin

Nos pedira información que debemos darle y dejamos que installe Java y NetBeans
En nuestro caso el directorio de java es /opt/

Instalación de TOMCAT.

Utilizaremos la versión 5.0.28 de Tomcat que la podemos bajar de:

http://apache.xmundo.com.ar/tomcat/...t-5.0.28.tar.gz

Al igual que apache y java creo un directorio en donde guardar el .tar.gz.

$:cd $HOME

$:mkdir tomcat

La instalación de tomcat la realizaremos en /usr/local

$:cd tomcat

Copiamos el .tar.gz a /usr/local

$:cp jakarta-tomcat-5.0.28.tar.gz /usr/local

$:cd /usr/local

Descomprimimos

$:tar zxvf jakarta-tomcat-5.0.28.tar.gz

Borramos jakarta-tomcat-5.0.28.tar.gz de /usr/local ya que lo tenemos en $HOME/tomcat

$:rm -f jakarta-tomcat-5.0.28.tar.gz

Creamos un enlace simbólico que nos facilitará la administración de tomcat.

$:ln -s jakarta-tomcat-5.0.28 tomcat5

Instalación de conector mod_jk.

Se utiliza la versión 1.2.15 del conector mod_jk que se puede bajar de http://www.apache.org/dist/tomcat/t...2.15-src.tar.gz que guardaremos $HOME/mod_jk.

$:cd $HOME

$:mkdir mod_jk

Requisitos para la instalar mod_jk necesitamos los siguientes:
libtool:
Para corroborar si nuestro sistema tiene libtool:

$:which libtool

Si tenemos como respuesta algo del tipo /usr/local/bin/libtool el sistema alberga libtool.
En caso contrario deberemos bajar libtool de ftp:/ftp.gnu.org/gnu/libtool e instalamos:

$:./configure

$:make

$:make install

autoconf.
Corroboramos:

$:which autoconf

Si tenemos como respuesta algo del tipo /usr/bin/autoconf el sistema alberga autoconf
En caso contrario deberemos bajar autoconf de ftp://ftp.gnu.org/autoconf e instalamos:

$:./configure

$:make

$:make install

Ant:
Para esta guía se utilizó la versión 1.6.5 de Ant, que se puede obtener de http://apache.mesi.com.ar/ant/binar....6.5-bin.tar.gz que la guardaremos en $HOME/ant.

$:cd $HOME

$:mkdir ant

$:cd ant

$:cp apache-ant-1.6.5-bin.tar.gz /usr/local

$:cd /usr/local

$:tar zxvf apache-ant-1.6.5-bin.tar.gz

$:rm -f apache-ant-1.6.5-bin.tar.gz

$:ln -s apache-ant-1.6.2 ant

Configuración de las variables globales.
Ahora deberemos crear la variable global JAVA_HOME y agregar al PATH la ruta del ejecutable de java. Editamos /etc/profile agregando las siguientes líneas al final del archivo:

$:vi /etc/profile

En este archivo le agregaremos las siguientes líneas:

JAVA_HOME=/opt/java....

ANT_HOME=/usr/local/ant

PATH=$PATH:$JAVA_HOME/bin:$ANT_HOME

export PATH JAVA_HOME ANT_HOME JAVA_HOME

Para que el sistema tome los nuevos valores de /etc/profile:

$:source /etc/profile

Procedemos a la instalación de mod_jk:

$:cd $HOME/mod_jk

$:tar zxvf jakarta-tomcat-connectors-1.2.15-src.tar.gz

$:cd jakarta-tomcat-connectors-1.2.15-src/jk/native/

$:./configure --with-apxs=/usr/local/apache2/bin/apxs --enable-EAPI

$:make

$:cp ./apache-2.0/mod_jk.so /usr/local/apache/modules

Configuración de mod_jk para Apache2.

Creamos el archico workers.properties en /usr/local/apache2/conf

$: vi /usr/local/apache2/conf/workers.properties

Y copiamos lo siguiente:

workers.tomcat_home=/usr/local/tomcat5

workers.java_home=$JAVA_HOME

ps=/

worker.list=default

worker.default.port=8009

worker.default.host=localhost

worker.default.type=ajp13

worker.default.lbfactor=1

Configuramos Apache:

Editaremos el archivo de httpd.conf

$:vi /usr/local/apache2/conf/httpd.conf

En la secciòn de de "LoadModules" agregamos al final:

LoadModule jk_module modules/mod_jk.so

Y al final del archivo :

#

 # Opciones de mod_jk

#

JkWorkersFile "conf/workers.properties"

JkLogFile "logs/mod_jk.log"

JkLogLevel error

JkMount /jsp-examples default

JkMount /jsp-examples/ default*

 # Fin de opciones de mod_jk

Vereficamos la correcta sintaxis de http.conf (mostrara un OK):

$:cd /usr/local/apache/bin

$:./apachectl configtest

Pruebas SSL + Apache + Tomcat.

Revisaremos que todo este funcionando correctamente. Primero levantaremos el servicio de Tomcat dejamos pasar 1/2 min. Y levantamos Apache

$:cd $CATALINA_HOME/bin

$:./startup.sh

Deberemos verificar que se levantaron los procesos java:

$ps -ef | grep java

###Y deberìamos tener como resultado algo del estilo:

###root 9038 1 4 17:54 pts/0 00:00:07 /usr/java/jdk1.5.0_06/bin/java -###Djava.endorsed.dirs=/usr/local/tomcat5/common/endorsed -classpath /usr/java/jdk1.5.0_06/lib/###tools.jar:/usr/local/tomcat5/bin/bootstrap.jar:/usr/local/tomcat5/bin/commons-logging-api.jar -###Dcatalina.base=/usr/local/tomcat5 -Dcatalina.home=/usr/local/tomcat5 -Djava.io.tmpdir=/usr/###local/tomcat5/temp org.apache.catalina.startup.Bootstrap start

Levantaremosos Apache:

$:cd ../../apache/bin

$:./apachectl start

Podemos vereficar Apache ingresando a:

Tomcat:

Y el conectos mod_jk:

http://localhost/jsp-examples

Bajar los servicios.

Para bajar los servicio de Tomcat y Apache hay hacer de la sigiente forma:

Primero Tomcat.

$:cd /usr/local/tomcat5/bin

$:./shutdown.sh

Después Apache.

$:cd ../../apache/bin

$:./apachectl stop

Instalación y configuración de mysql.

Se instaló y configuró Mysql así como software de configuración y administración. Ver figura 4.14.

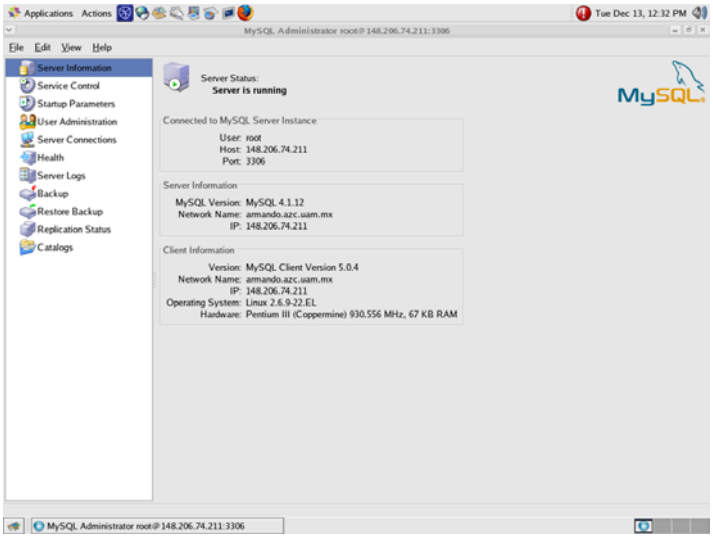


Figura 4.14 Pantalla de administración de Mysql.

Posteriormente se procedió a generar la base de datos de nuestro sistema junto con sus tablas. Y se ralizó la carga de la información del diccionario de tags.

Implementación del programa de Adquisición de imágenes.

Se realizo un API (Application Programming Interface - Interfaz de Programación de Aplicaciones) el cual manda a traer información del diccionario DICOM de la base de datos por lo que se requiere de un login y un password. Al abrir un archivo bajo el formato DICOM extrae la información de este y muestra la imagen. Ver figura 4.15.

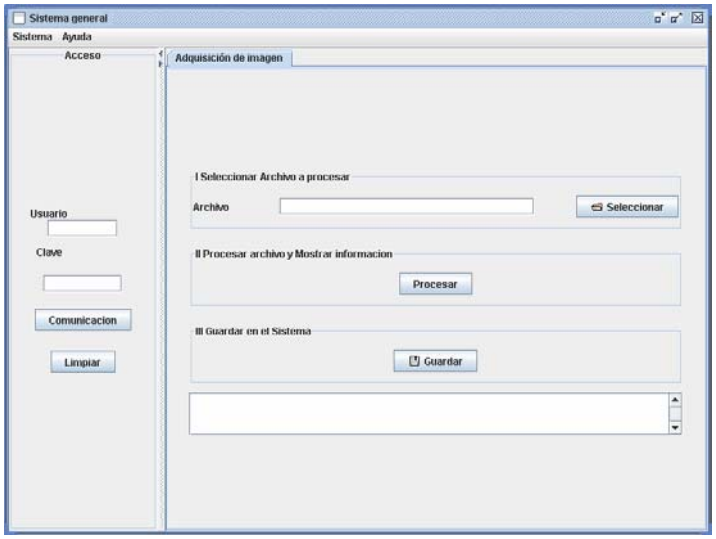


Figura 4.15 Pantalla de la interfaz gráfica de Adquisición de imagen.

Las clases que utilizamos para la adquisición de imagen es:

- Comun_Mysql.java
- DicomDic.java
- DicomFile.java
- DicomvalueData.java
- DicomValueDic.java
- ImageData.java
- ImageProcessor.java
- JFrameSobreSistema.java
- JPanelErrores.java
- jtable_3.java
- Principal.java
- Vectores.java
- Visualizador.java

El listado de los programas para la adquisición de las imágenes se muestra en el anexo B.

Programas para visualizar la información por medio de una navegador WEB.

En esta parte podemos dividirla en dos partes:

- Programas JSP y HTML.
- Programas en Java.

Programas JSP y HTML.

Los programas JSP y HTML que se utilizan para la visualización de la información de la imagen son:

- Alta.jsp
- Bienvenido.jsp
- cabecera.html
- CatalogImagenes.jsp
- CatalogoImagen.jsp
- CuentaSuspendida.jsp
- cuerpo.html
- FallaInSes2.jsp
- Info_E_Imagen.html
- InicioSesion2.jsp
- Intro.html
- listPaciente.jsp
- MarcoMenu.html
- MostrarImagen.jsp
- MuestraImagen.jsp
- MuestraInfo.jsp
- NoSesionInic.jsp
- Paciente.jsp
- Pasaje3.jsp
- PrimeroFinSes.jsp
- principal.html
- Salir.jsp
- SesionDuplicada2.jsp
- SolicitudInSes.jsp
- ValidarUsuario2.jsp
- VerificarFinSes.jsp

En el Anexo C podemos ver el contenido de estos programas.

Programas en Java.

En el Anexo D podemos ver las Clases para visualizar la información por medio del navegador WEB que son llamadas por los programas JSP, HTML del Anexo C.

4.4 Pruebas.

Son tres las pruebas para nuestro sistema las cuales se aplican a la interfaz gráfica para la adquisición de imagen; interfaz WEB para la visualización de la información. Y seguridad para la aplicación WEB. Ver la figura 4.16 donde se muestra un diagrama de flujo de estas pruebas.

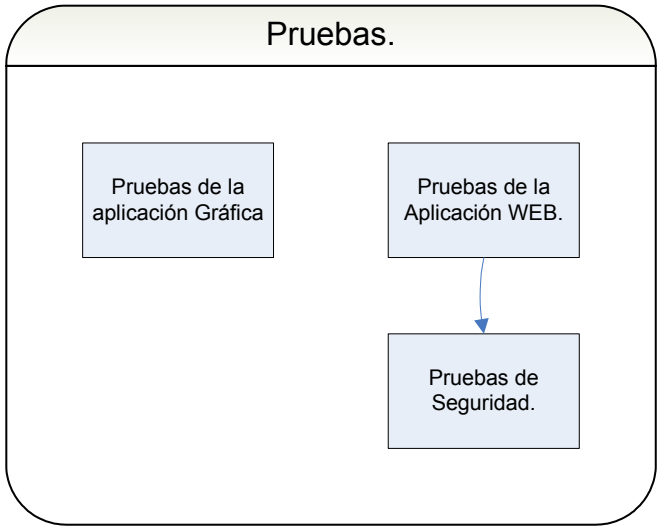


Figura 4.16 Pruebas.

Pruebas de la aplicación Gráfica.

Esta prueba se desarrollaron los siguientes puntos:

- La base de datos.
- Adquisición de imágenes.

La Base de datos.

Se comprobó la existencia de la base de datos y sus tablas del sistema por lo que:

Se instaló y configuró Mysql en Linux y Windows XP. Para el Sistema de Datos de las imágenes se crearon la tablas y se cargó la información para el diccionario. Nuestro servidor es loky.uam.mx el cual contiene la base de datos del control de sesión.

En la figura 4.17 se muestra las bases de datos que se encuentran en el servidor.

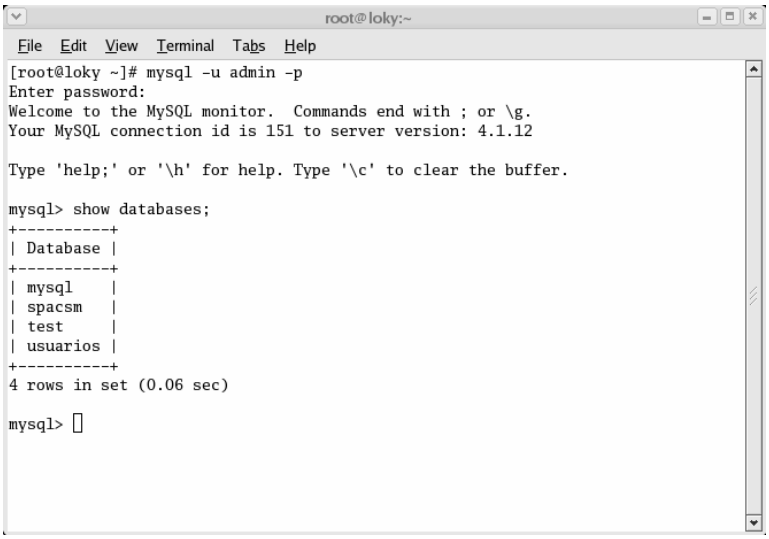


Figura 4.17 Base de datos en el servidor.

En la figura 4.18 se muestra las tablas donde se guarda la información de la imagen así como la información asociada a la misma.

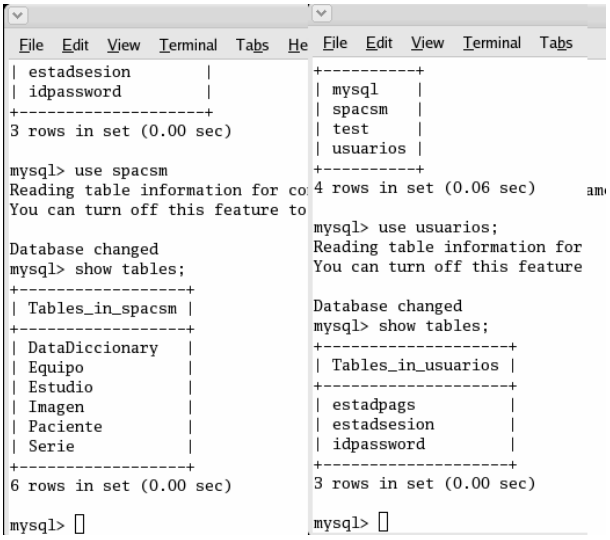


Figura 4.18 Las tablas del sistema contenidas en el servidor.

Carga de la imagen.

La aplicación es una interfaz gráfica la cual esta formada por tres parte una barra de menús, control de acceso, Adquisición de imagen. Ver figura 4.19.

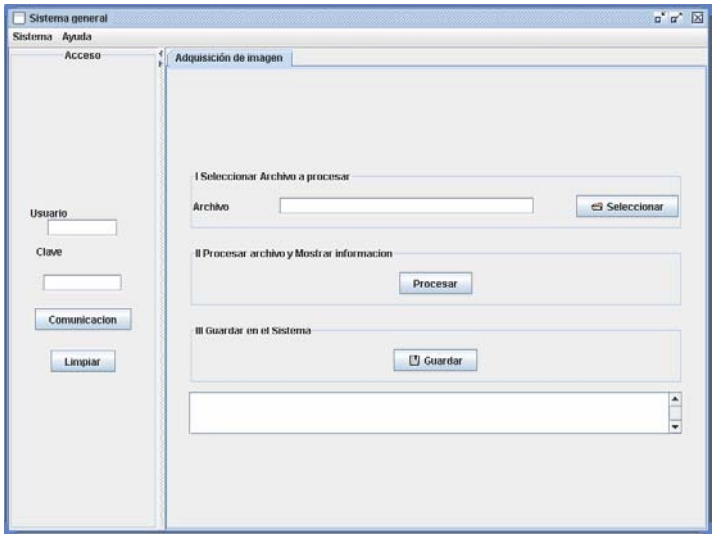


Figura 4.19 Interfaz Gráfica.

Control de acceso.- En esta parte revisamos que al darle usuario y la clave (login, password) se pueda confirmar que sean correctos, por medio un click en comunicación nosotros recibimos un mensaje confirmado, que sean correctos estos datos. Ver figura 4.20

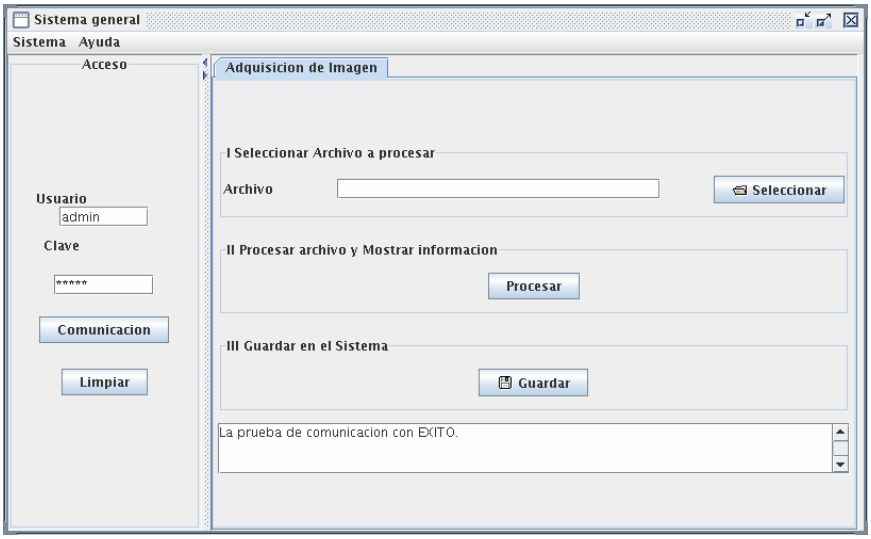


Figura 4.20 Confirmación del usuario y clave correctos.

Al dar un click en el botón de Procesar, esté despliega la información contenida en el IOD-compuesto y a la imagen en la ventana donde muestra la tabla IOD- compuesto podemos editar esta información. Ver Figura 4.21.

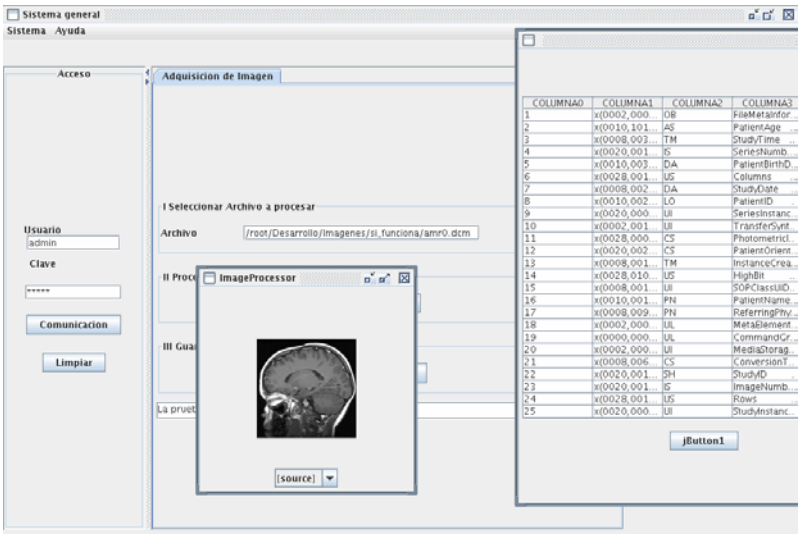


Figura 4.21 Se muestra las ventanas generadas por esta interfaz

En esta figura podemos ver a la imagen el cual tiene un menú para poder ver mas clara, mas obscura, una reduccion, o moverla. Y la otra ventana podemos ver el número, tipo de dato, descripción, contenido para cada Tag.

Pruebas de la aplicación WEB.

Esta interfaz es WEB. Es necesario abrir el navegador y colocar la dirección: .

Esto dará la bienvenida y posteriormente hay que ir al menú de opciones y dar un click a "inició de sesión" para poder entrar y ver la información solicitada. En el caso que no corresponda la autentificación del usuario manda un mensaje y después de tres intentos el usuario queda deshabilitado y solo lo pobra habilitar la gente que tenga permiso para esto (Se generó una aplicarción WEB para el control de los Usuarios). Ver figura 4.22.

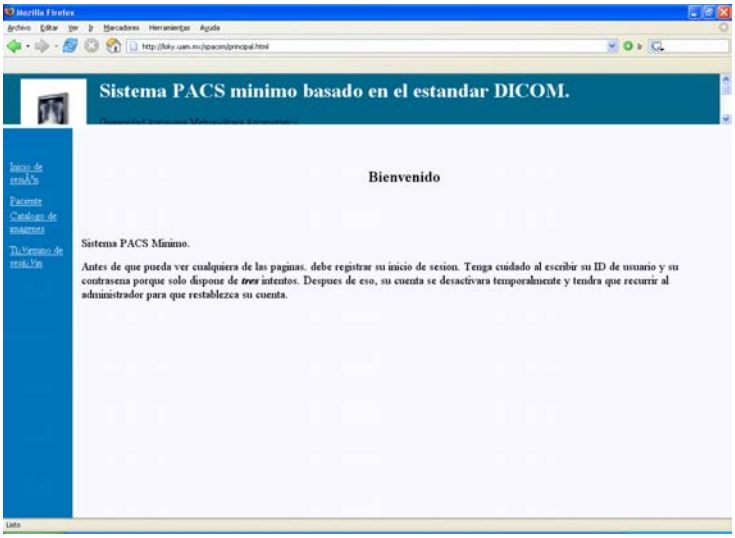


Figura 4.22 Primera pantalla de la interfaz Web.

En la figura 4.23 se muestra la pantalla en donde hay que escribir el usuario y la contraseña para entrar (En donde usuario y password es super).

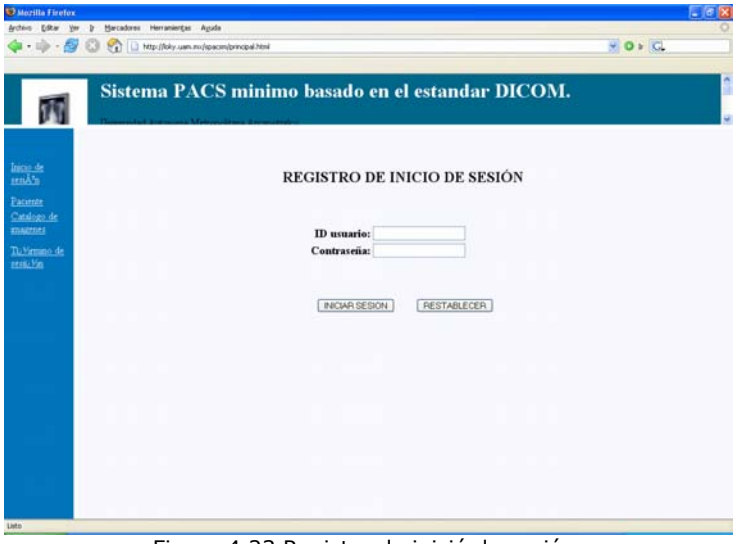


Figura 4.23 Registro de inició de sesión.

Al darle el usuario y clave automáticamente muestra la pantalla de éxito de aceptación. Ver figura 4.24.

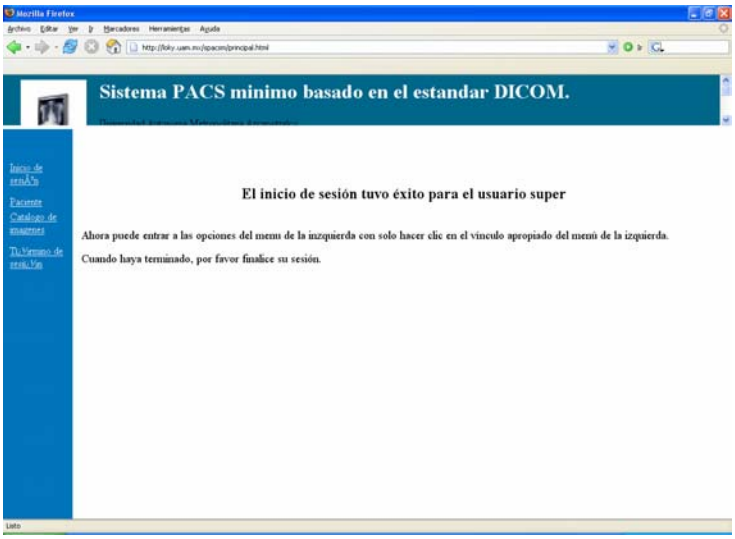


Figura 4.24 Aceptación de inició de sesión.

En el caso que la clave este mal e intentó más de tres veces entrar con el mismo usuario, este usuario será suspendido y mandara a pantalla un mensaje. Ver figura 4.25.

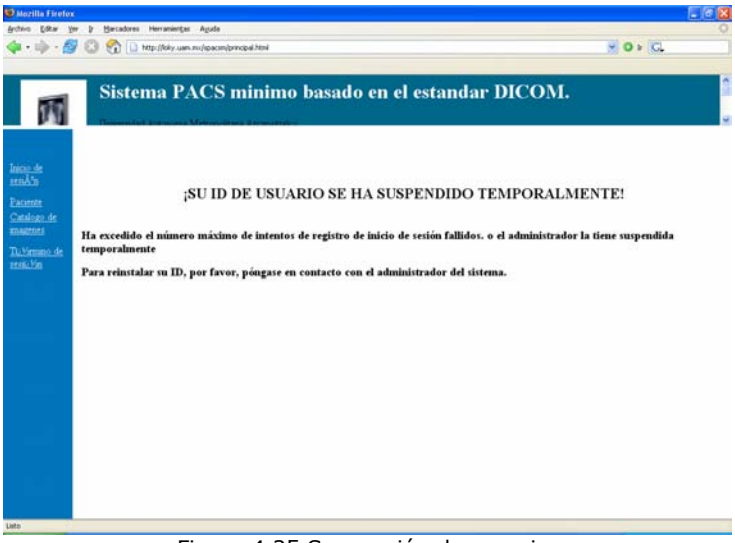


Figura 4.25 Suspensión de usuario.

Podemos ir al menú y seleccionar Catálogo de imágenes y en este mostrará todas las imágenes dando información de paciente, estudio, equipo , serie, para poder ver esta información solo hay que dar un click en la imagen de interés. Ver figura 4.26.

Figura 4.26 Catálogo de imagen

Después de darle el click a la imagen, nos mostrará la imagen y toda la información asociado a la misma. Ver figura 4.27.

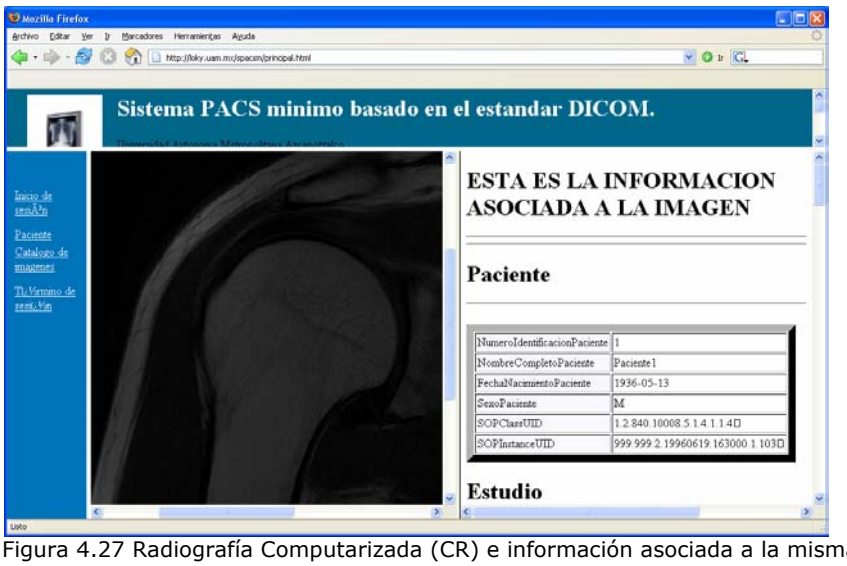


Figura 4.27 Radiografía Computarizada (CR) e información asociada a la misma.

También cuenta con un mecanismo que al no utilizar el navegador, automáticamente en el servidor cierra la sesión. Y al intentar entrar por ejemplo otra persona manda un mensaje para que inicie sesión, ver figura 4.28.

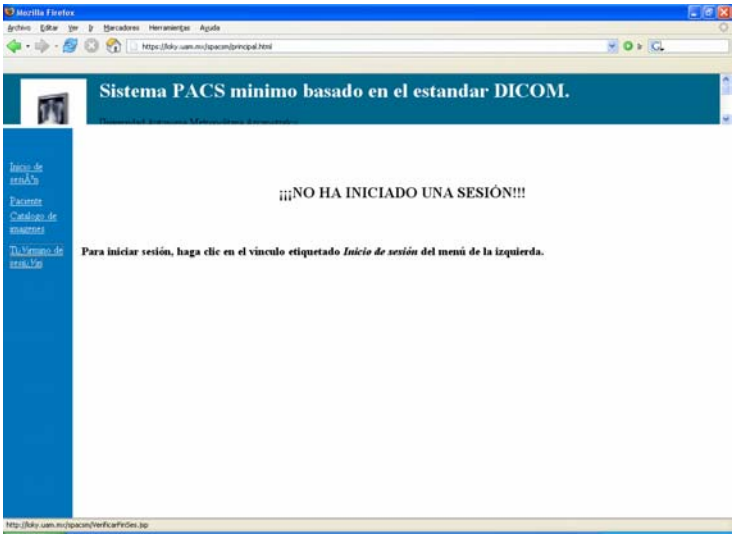


Figura 4.28 Mensaje que no inició sesión, no utilizó el navegador (por 10 min.configurable) o fue cerrada la sesión.

Para salir de la sesión hay que dar un click en "terminar sesión" y mostrará un mensaje. Ver figura 4.29.

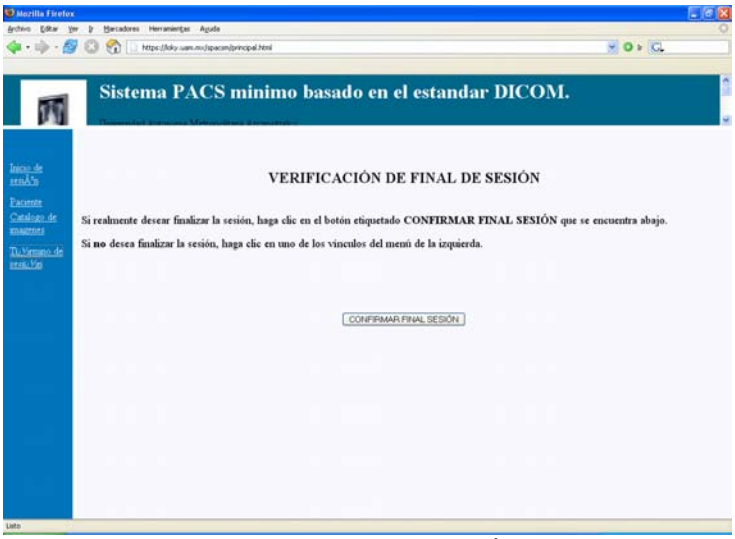


Figura 4.29 Terminar sesión.

Pruebas de Seguridad.

Nuestro sistema maneja el controlar la privacidad de los datos y controlar el acceso al sistema. Esto se aplica para la Adquisición de Imagen y Visualización de la siguiente forma.

Adquisición de Imágenes.

En esta interfaz aplicamos el control de acceso al sistema utilizando un login y un password para poder tener acceso a la base de datos en caso de que no se autentifique esta información no podrá acceder a la base de datos para grabar la información y manda un mensaje. Ver figura 4.20.

Visualización interfaz WEB.

Aquí manejamos el control de la privacidad de la información por medio de SSL y controlamos el acceso al sistema por medio de la sesión donde deberemos dar un login y un password para poder tener acceso a la información.

En la figura 4.30 se muestra el control de acceso por medio de un login y password.

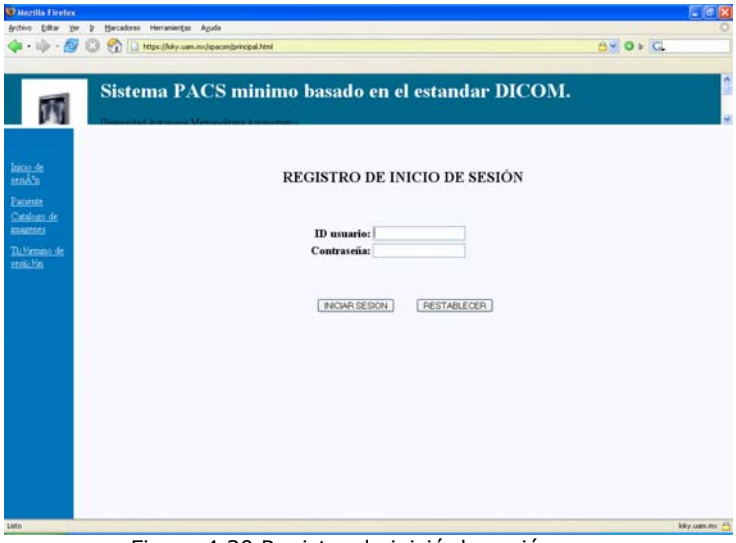


Figura 4.30 Registro de inició de sesión seguro.

En la figura 4.31 se muestra que está activo el control de privacidad de los datos por medio de SSL.

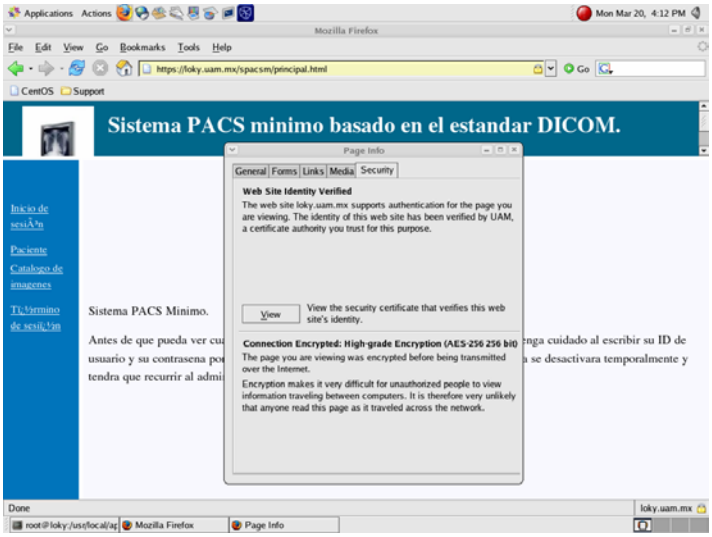


Figura 4.31 Seguridad en el navegador.

5 Conclusiones y Futuros Trabajos.

En este punto hablaremos de las conclusiones de este proyecto y plantearemos los posibles trabajos futuros sobre este proyecto.

5.1 Conclusiones

En este proyecto se integraron los conceptos y técnicas de computación para resolver el problema de control de imágenes e información de estudios médicos. El Sistema PACS mínimo basado en el estándar DICOM que es un sistema complejo que describe un sistema multiplataforma (ya que utilizamos como lenguaje de programación Java) desarrollado para el manejo de imágenes de diagnóstico médico a través de la red (Intranet o Internet) el cual incorpora una capa de seguridad, por lo que se tuvieron que evaluar diferentes alternativas computacionales.

En base a los objetivos propuestos se realizó: El análisis, diseño, implementación, y pruebas del Sistema PACS mínimo basado en el estándar DICOM.Ver figura 5.1.

El cual tiene las siguientes características:

- Cuenta con las librerías para los distintos tipos de imágenes (ya que se cuenta con un diccionario de los TAGs del estándar DICOM).
- Maneja la Base de Datos de la información del paciente.
- Tiene un servidor de Base de Dato e Imágenes que es capas de atender a múltiples clientes y que soporte el protocolo DICOM para adquirir imágenes médicas cargando Objetos de Información compuesto (IOD-Compuesto).
- Cuenta con una interfaz gráfica para la adquisición de imágenes donde muestra la información y podemos editar la información para cada tag si es requerido. Muestra la imagen teniendo la posibilidad de darle más brillo, o ponerlo más oscuro, reducir la imagen y rotarla.
- El sistema muestre la información del paciente, estudio, equipo, serie, e imagen según lo solicite el usuario (Visualizador) utilizando una interfaz WEB.
- Tiene una comunicación segura ya que se implementaron mecanismos de protección de la privacidad del sistema controlando el acceso y privacidad de los datos para la interfaz WEB utilizando SSL. Para la interfaz gráfica de adquisición de imagen se controla el acceso.

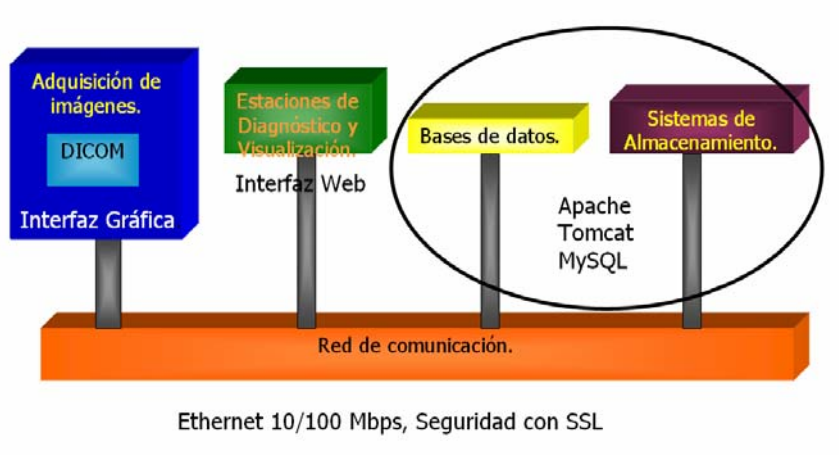


Figura 5.1 Sistema PACS mínimo basado en el estándar DICOM.

5.2 Futuros trabajos.

Sin duda todo trabajo software es perfeccionable. Y para este proyecto podemos realizar trabajos futuros en:

- En el Visualizador. Aquí podemos desarrollar herramientas visuales que permitan hacer un acercamiento, control de brillo, contraste, hacer notación dentro de la imagen entre otros.
- En los sistemas de almacenamiento de imágenes hacer una estructura jerárquica que dependa de la probabilidad de la demanda de la imagen. En general las imagen recientemente adquirid se consultan con mas frecuencia en los minutos siguientes y su frecuencia de consulta disminuye rápidamente con el tiempo.
- Negociar el préstamo de equipo médico bajo el estándar DICOM para poder establecer una comunicación directa con este.
- Poder implantar este sistema en producción.

Referencias

[1] Alicia Morales Reyes
Manejo de imágenes DICOM (Digital Imaging and Communications in Medicine) Mediante un sistema en Internet (Tesis).
http://ccc.inaoep.mx/~amoralesr/dicom/tesisDicomAliciaMoralesReyes.pdf
2002

[2] ArgoUML
http://argouml.tigris.org

[3] Arquitectura de Software.
http://es.wikipedia.org/wiki/Arquitectura_software

[4] Azpiroz Leehan, J. Martínez Martínez, M.
Instalación y operación de Sistemas PACS (Almacenamiento y Comunicación de Imágenes): características fundamentales
Revista Mexicana de Ingeniería Biomédica • Vol XIX • No.3 • ISSN 0188-9532 •noviembre 1998

[5] Josefina Gutiérrez, Martínez
Sistemas PACS-CNR: Una propuesta tecnológica.
Revista Mexicana de Ingenieria Biomedica
Marzo, 2003; Volumen 24: Número 1
Rev Mex Ing Biomed 2003; 24 (1)

[6] Estándar DICOM (Organización NEMA)
http://medical.nema.org/dicom/2003.html

[7] OMG
www.omg.org/uml/

[8] Ivar Jacobson et al
El proceso Unificado de Desarrollo de Software.
Addison Wesley
464 pag.
ISBN-7829-036-2

[9] Jason Hunter
Java Servlet Programming.
Editorial O'Reilly
753 pag.
Año 2001
ISBN 0-596-00040-5

[10] Javier González Sánchez
UML y el proceso unificado de desarrollo trabajando con objetos
http://javiergs.dgproject.com/pdf/uml-diapositivas.pdf

[11] Joaquín Azpiroz Leehan
PACS Estado Actual y Tendencias Futuras
UAM-Iztapalapa
Febrero 2001

[12] Joaquín Azpiroz Leehan et al
Instalación y operación de Sistemas PACS (Almacenamiento y comunicación de imágenes) en México: características fundamentales.
1999.

[14] Juan Carlos Espinosa H.
Linux 7.0 Instalación y configuración básica.
Alfaomega
192 pág.
ISBN 970-15-0725-8.

[15] Kapil Sharma et al
Professional Red Hat Enterprise Linux3
Wrox (September 3, 2004)
744 pág.
ISBN 0-7645-7283-0

[16] Kathy Walrath, et al
The JFC Swing Tutorial
http://java.sun.com/docs/books/tutorial/uiswing/

[17] Kris Jamsa
Aprenda y practique JAVA
474 pág.
Oxford University Press (February 15, 2000)
ISBN 970-613-525-1

[18] M. Sadiku and C. Sadiku
Writing a research report
IEEE Potentials, May 1988, pp 41-44.
http://ccc.inaoep.mx/~ariasm/seminario/doc/reporte.html

[19] Martínez M.A. , Jiménez A.J.R, Medina B.V., Azpiroz L.J.
Los Sistemas PACS
2003
http://itzamna.uam.mx/alfonso/pacs.html

[20] Marty Hall
Servlets y Java Server Pages guía practica.
Prentice Hall
580 pág.
ISBN 970-26-0118-5

[21] NEMA
Estándar DICOM (Organización NEMA)
http://medical.nema.org/dicom/2003.html

[22] Patricio Letelier Torres
Desarrollo de Software Orientado a Objeto usando UML
Universidad Politécnica de Valencia (UPV) – España
http://www.dsic.upv.es/~letelier/

[23] Paul Tremblett
Superutilidades para JavaServerPages
Osborne/McGraw Hill
525 pág.
ISBN: 970-10-3828-2

[24] Popkin Software and Systems. Modelado de Sistemas con UML
www.popkin.com

[25] Reiner, Bruce I., Siegel, Eliot L., Smith, Edward M. Great Falls, Va.
Archiving Issues in the Medical Enterprise Reiner
Society for Computer Applications in Radiology,
2001
Capitulo 1.

[26] Ricardo Hernández
Guía para la Elaboración del Reporte de Investigación
Facultad de Ciencias – ULA
http://webdelprofesor.ula.ve/ciencias/ymartin/index_archivos/Guia%20para%20la%20Elaboracion%20del%20Reporte%20de%20Investigacion.pdf

[27] Santiago Rodrígues de la Fuente et al
Programación de Aplicaciones Web.
Thomson-Paraninfo
586 p
ISBN: 84-9732-181-2

[28] Sherif M. Yacoub, Hany H. Ammar
The Development of a Client/Server Architecture for Standardized Medical Application Network Services
Universidad De Virginia Occidental
http://csdl.computer.org/comp/proceedings/asset/1999/0122/00/01220002abs.htm
1999

[29] UML
http://www.umlderby.org/compare/tool/

[30] Uri Shani, et al
Real-time teleconsulting solution for teleradiology based on PACS
http://www.haifa.il.ibm.com/projects/software/idmr/papers/teleradiology.htm
1998

[31] Vivek Chopra
Professional Apache Tomcat 5
Wiley,
Año 2004.
598 pág
ISBN 0-7645-5902-8

[32] Edgar M. Smith.
Introducción a la adquisición, manejador de archivos y almacenamiento de imagenes (Introduction to Image Acquisition, Archive Managers, and Storege).
Publicado en el libro "Archiving Issues in the Medical Enterprise" Reiner, Bruce I., Siegel, Eliot L., Smith, Edward M. Great Falls, Va. : Society for Computer Applications in Radiology, c 2001 Capitulo 1.

[33] Uri Shani, et al
Teleconsulta en Tiempo real solución para teleradiografía basado en PACS (Real-time teleconsulting solution for teleradiology based on PACS)
Publicado en Israel en el Laboratorio de investigación de IBM Haifa

[34] Sherif M. Yacoub, et al
El desarrollo de arquitectura Cliente/Servidor para estandarización médica de servicios de aplicaciones en red (The Development of a Client/Server Architecture for Standardized Medical Application Network Services).
Publicado en la Universidad De Virginia Occidental y IEEE

[35] Luciano Moreno
Transacciones seguras
http://www.htmlweb.net/seguridad/ssl/ssl_5.html

[36] PostGreSQL vs. MySQL
http://www.netpecos.org/docs/mysql_postgres/index.html

[37] MySQL vs. PostgreSQL
http://www.mmlabx.ua.es/mysql-postgres.html

[38] MySQL con Java en MS Windows
www.mysql-hispano.org

[39] JDBC
UNIVERSIDAD DE CASTILLA-LA MANCHA
alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/jdbc_2xh.pdf

[40] Acceso a BD desde Java. JDBC
Prácticas de Modelos Avanzados de Bases de Datos
J. M. Medina, O. Pons, M.A.Vila
http://www-etsi2.ugr.es/depar/ccia/mabd/material/practicas/transparencias/Curso2002.2003/AccesoBDJava.pdf

[41] Matthew Fischer
The DES Algorithm Illustrated
http://www.aci.net/kalliste/des.htm

[42] Luciano Moreno
Criptografía
http://www.htmlweb.net/seguridad/cripto/cripto_7.html

[43] DICOM Cook Book for Implementations in Modalities
ftp://ftp.philips.com/pub/pms-3003/DICOM_Information/CookBook.pdf

[44] William J. Altier
Instrumentos intelectuales del gerente
Oxfod, 232 pag, 2000
ISBN 970 613 559 6

[45] Fernando Ballesteros Herranz.
Desarrollo de aplicaciones DICOM para la gestión de imágenes biomédicas
Universidad Politécnica de Madrid.
Octubre 2003

[46] Rafaela Blanca Silva López
Diseño y optimizacion de Base de Datos Relacionales.
Universidad Autónoma Metropolitana.
Departamento de Sistemas.
Julio 1997.

Glosario

AAPM	American Association of Phisicists in Medicine
Angi-C	Cardiología angiografía
Angi-R	Radiología angiografía
API	Acrónimo de Application Programmer's Interface
CR	Radiografía computarizada
CT	Tomografía computarizada
DF	Fluoroscopia Digital
DICOM	Estandar de Comunicación e Imágenes Médicas (Digital Imaging and Communications in Medicine).
DR	Radiografía Directa
DUL	Acrónimo de DICOM Upper Layer
FD	Película digitalizada
FDDI	Fiber Distributed Data Interface; Interfaz de datos distribuidos por fibra es un estándar para transmisión de datos en LAN que opera sobre fibra óptica a 100 Mbps.
Gigabit Ethernet	También conocida como GigE, es una ampliación del estándar Ethernet (concretamente la versión 802.3ab y 802.3z del IEEE) que consigue una capacidad de transmisión de 1 gigabit por segundo, correspondientes a unos 1000 megabytes por segundo de rendimiento contra unos 100 de Fast Ethernet.
HIS	Sistema de Información Hospitalario (Hospital Information Systems)
IE	Acrónimo de Information Entity
IOD	Definición de objeto de Información (Information object definition)
IOM	Acrónimo de Information Object Module
MRI	Resonancia Magnética
NEMA	Nacional Electrical Manufacturers Association.
NM	Medicina Nuclear
OSI	Open Systems Interconnection
PACS	Sistema de almacenamiento y comunicación de imágenes (Picture Archiving and Communications System).
RIS	Sistema de información de Radiología (Radiologic Information Systems)
RSNA	Sociedad Radiologica de Norte America (Radiological Society of North America)
SCP	Acrónimo de Service Class Provider
SCU	Acrónimo de Service Class User
SOP	Acrónimo de Service Objetc-Pair
UID	Acrónimo de Unique Identifier
US	Ultrasonido
VOI	Acrónicmo de Value Of Interest
VR	Acrónimo de Value Representation

Anexo A

Documentacion de las Clases del Sistema.

Principal
Class Visualizador

public class **Visualizador**
extends javax.swing.JFrame

Esta clase genera unJFrame para poder ver la imagen
Author:
armando
See Also:

Constructor Detail

Visualizador

public **Visualizador**(java.lang.String titulo,
java.awt.Image tmpimage)
Creates a new instance of Visualizador

Method Detail

paint

public void **paint**(java.awt.Graphics graphics)
Overrides:
`paint` in class `java.awt.Container`

Principal
Class Comun_Mysql

java.lang.Object
Principal.Comun_Mysql

public class **Comun_Mysql**
extends java.lang.Object

Comunica con MYSQL para pedir datos y sacarlos.
Author:
Armando Jimïénez Herrera

Constructor Detail

Comun_Mysql

public **Comun_Mysql**()

Method Detail

Test

public boolean **Test**(java.lang.String login,
java.lang.String password)
Prueba la comunicación de con el servidor Mysql
Parameters:
`login` -
`password` -
Returns:
boolean

comun_DicomDic

public java.util.Hashtable **comun_DicomDic**(java.lang.String login,
java.lang.String password)
Se comunica con Mysql y trae toda la tabla de los datos del diccionario
Parameters:
`login` -
`password` -
Returns:
table Que es un Hashtable con la llave de TAG

isContainsPaciente

public int **isContainsPaciente**(java.lang.String login,
java.lang.String password,
java.lang.String numIDPaciente)
Revisa la base de datos si exite Numero_de_identificación_del_paciente dado. Valores 1 Si existe 0 No existe -1 Existe mas de 1 registro

insertPaciente

public int **insertPaciente**(java.lang.String login,
java.lang.String password,
java.lang.String datos)
Inserta un registro de Paciente 1 exito 0 error

infPaciente

public java.util.Vector **infPaciente**(java.lang.String login,
java.lang.String password,
boolean flagTodo,
java.lang.String numIDPaciente)
Busca toda la información del paciente y regresa el vector Si flagTodo = tru & numidpaciente null regresa el vector de toda la tabla flagTodo = false regresa el vetor del paciente dado

isContainsEstudio

public int **isContainsEstudio**(java.lang.String login,
java.lang.String password,
java.lang.String IdentificadorInstanciaEstudio)
Revisa la base de datos si exite IdentificadorInstanciaEstudio dado. Valores 1 Si existe 0 No existe -1 Existe mas de 1 registro

insertEstudio

public int **insertEstudio**(java.lang.String login,
java.lang.String password,
java.lang.String datos)
Inserta un registro de Paciente 1 exito 0 error

isContainsEquipo

public int **isContainsEquipo**(java.lang.String login,
java.lang.String password,
java.lang.String FabricanteEquipo)
Revisa la base de datos si exite Numero_de_identificación_del_Equipo dado. Valores 1 Si existe 0 No existe -1 Existe mas de 1 registro

insertEquipo

public int **insertEquipo**(java.lang.String login,
java.lang.String password,
java.lang.String datos)
Inserta un registro de Equipo 1 exito 0 error

isContainsSerie

public int **isContainsSerie**(java.lang.String login,
java.lang.String password,
java.lang.String IdentificadorInstanciaSerie)
Revisa la base de datos si exite IdentificadorInstanciaEstudio dado. Valores 1 Si existe 0 No existe -1 Existe mas de 1 registro

insertSerie

public int **insertSerie**(java.lang.String login,
java.lang.String password,
java.lang.String datos)
Inserta un registro de Paciente 1 exito 0 error

isContainsImagen

public int **isContainsImagen**(java.lang.String login,
java.lang.String password,
java.lang.String IdImagen)
Revisa la base de datos si exite IntanceNumbre (imagen) dado. Valores 1 Si existe 0 No existe -1 Existe mas de 1 registro

insertImagen

public int **insertImagen**(java.lang.String login,
java.lang.String password,
java.lang.String datos,
java.awt.Image imagenori)
Inserta un registro de Imagen 1 exito 0 error

CargaImagen

public boolean **CargaImagen**(java.lang.String nomArch)

Carga los datos de un archivo regresa true con exito regresa false cuando ocurre un error

---

MuestraVentana

public void **MuestraVentana**(java.lang.String cadena,
java.awt.Image ima)
Muestra en una venta una imagen

---

creaArchJPG

public boolean **creaArchJPG**(java.lang.String nombArch)

---

saveMySqlPhoto

public static void **saveMySqlPhoto**(java.lang.String empID,
java.awt.Image image)

---

saveImage

public boolean **saveImage**(java.lang.String login,
java.lang.String password,
java.lang.String IdImagenes,
java.awt.Image image)

---

toBufferedImage

public static java.awt.image.BufferedImage
toBufferedImage(java.awt.Image image)

---

hasAlpha

public static boolean **hasAlpha**(java.awt.Image image)

---

getImageFile

public java.awt.Image **getImageFile**(java.lang.String login,
java.lang.String password,
java.lang.String fileName)
throws java.lang.Exception
Throws:
`java.lang.Exception`

---

getImagen

public java.awt.Image **getImagen**()

Principal

Class DicomData

java.lang.Object
Principal.DicomData
All Implemented Interfaces:
java.io.Serializable

---

public class **DicomData**
extends java.lang.Object
implements java.io.Serializable

Crea una tabla para que contenga:tag, name, vr, vm, version, value, valueLength, analyzedValue.

Constructor Detail

DicomData

public **DicomData**()

Method Detail

setTag

public void **setTag**(java.lang.String tag)
Agrega a la tabla un nuevo elemento con llave TAG
Parameters:
`tag` -

---

setValue

public void **setValue**(java.lang.String tag,
byte[] argValue)
Coloca la información de Value del tag proporciando
Parameters:
`tag` -
`argValue` -

---

setName

public void **setName**(java.lang.String tag,
java.lang.String argName)
Coloca el Name en el TAG proporcionado
Parameters:
`tag` -
`argName` -

---

setVR

public void **setVR**(java.lang.String tag,
java.lang.String argVR)
Coloca el VR en el tag proporcionado
Parameters:
`tag` -
`argVR` -

---

setVM

public void **setVM**(java.lang.String tag,
java.lang.String argVM)
Coloca el VM en el tag proporcionado
Parameters:
`tag` -
`argVM` -

---

setVersion

public void **setVersion**(java.lang.String tag,
java.lang.String argVersion)
Coloca el Version en el tag proporcionado
Parameters:
`tag` -
`argVersion` -

---

setAnalyzedValue

public void **setAnalyzedValue**(java.lang.String tag,
java.lang.String argAnalyzed)
Coloca el AnalyzedValue en el tag proporcionado
Parameters:
`tag` -
`argAnalyzed` -

---

getValue

public byte[] **getValue**(java.lang.String tag)
Consige el Value de tag solicitado
Parameters:
`tag` -
Returns:
value

---

getValueLength

public int **getValueLength**(java.lang.String tag)
Consige el Valuelength de tag solicitado
Parameters:
`tag` -
Returns:
valueLength

---

getName

public java.lang.String **getName**(java.lang.String tag)
Consige el Name de tag solicitado
Parameters:
`tag` -
Returns:
name

---

getVR

public java.lang.String **getVR**(java.lang.String tag)
Consige el VR de tag solicitado
Parameters:
`tag` -

Returns:
vr

---

getVM

public java.lang.String **getVM**(java.lang.String tag)
Consige el VM de tag solicitado
Parameters:
`tag` -
Returns:
vm

---

getVersion

public java.lang.String **getVersion**(java.lang.String tag)
Consige el Version de tag solicitado
Parameters:
`tag` -
Returns:
version

---

getAnalyzedValue

public java.lang.String **getAnalyzedValue**(java.lang.String tag)
Consige el AnalyzedValue de tag solicitado
Parameters:
`tag` -
Returns:
AnalyzedValue

---

keys

public java.util.Enumeration **keys**()
Regrasa el numero de llaves de la tabla
Returns:
enumeration

---

removeAll

public void **removeAll**()
Remueve todo el contenido de la tabla

---

isContain

public boolean **isContain**(java.lang.String tag)
Revisa si esta contenido el tag proporcionado
Parameters:
`tag` -
Returns:
boolean

---

gethashtable

public java.util.Hashtable **gethashtable**()

---

getVectorDicomData

public java.util.Vector **getVectorDicomData**()

---

getPaciente

public java.util.Vector **getPaciente**()
Selaccióna los datos : 1.PatientName x(0010,0010)x 2.PatientID x(0010,0020)x 3.PatientBirthDate x(0010,0030)x 4.PatientSex x(0010,0040)x 5.SOPClassUIDSOPClassUID x(0008,0016)x 6.SOPInstanceUID x(0008,0018)x de la tabla regresa un vector

---

Principal
Class DicomDic

java.lang.Object
  Principal.DicomDic

---

public class **DicomDic**
extends java.lang.Object

CREA UNA TABLA DEL DICCIONARIO DE TAGs

Constructor Detail

DicomDic

public **DicomDic**()

Method Detail

loadDicomDic

public void **loadDicomDic**(java.lang.String login,
java.lang.String password)
Llama a comunicación para que le entrege la tabla completa de Diccionario Dicom en Mysql

---

getName

public java.lang.String **getName**(java.lang.String tag)
Entraga la descripcion del TAG solicitado
Parameters:
`tag` -
Returns:
name

---

getVR

public java.lang.String **getVR**(java.lang.String tag)
Entrega el VR del TAG solicitadp
Parameters:
`tag` -
Returns:
vr

---

getVM

public java.lang.String **getVM**(java.lang.String tag)
Entrega el VM del TAG solicitado
Parameters:
`tag` -
Returns:
vm

---

getVersion

public java.lang.String **getVersion**(java.lang.String tag)
Entrega la version del TAG solicitado
Parameters:
`tag` -
Returns:
version

---

isContain

public boolean **isContain**(java.lang.String tag)
Busca si esta contenido en la tabla de un TAG
Parameters:
`tag` -

Principal
Class DicomFile

---

public class **DicomFile**
extends java.lang.Object

Se encarga de manejar lo relacionado al archivo del IOD Compuesto. Para generar una tabla de tipo DicomData y DicomDIC

Constructor Detail

DicomFile

public **DicomFile**()

DicomFile

public **DicomFile**(boolean argIsLtlEndian,
boolean argVRType,
boolean privacy,
argDicomDic)
Da los valores a las banderas
Parameters:
`argIsLtlEndian` - bandera
`argVRType` - bandera

privacy - bandera
argDicomDic - DicomDic

DicomFile

public **DicomFile**(boolean argIsLtlEndian,
boolean argVRType,
argDicomDic)

DicomFile

public **DicomFile**(argDicomDic)

Method Detail

getDicomDic

public **getDicomDic**()

---

getDicomData

public **getDicomData**()

---

gethashtable

public java.util.Hashtable **gethashtable**()

---

getVectorDicomData

public java.util.Vector **getVectorDicomData**()

---

getPaciente

public java.util.Vector **getPaciente**()

---

load

public **load**(java.lang.String imgURL)
Esta pensado para URL pero si no tiene la forma se configura para carga el archivo
Parameters:
imgURL -
Returns:
DicomData

Principal
Class DicomvalueData

java.lang.Object
Principal.DicomvalueData

---

public class **DicomvalueData**
extends java.lang.Object

Estuctura del objeto
Author:
armando

Constructor Detail

DicomvalueData

public **DicomvalueData**()

Principal
Class DicomValueDic

java.lang.Object
Principal.DicomValueDic

---

public class **DicomValueDic**
extends java.lang.Object

Estructura del objeto para el Diccionario DICOM
Author:
armando

Constructor Detail

DicomValueDic

public **DicomValueDic**()
Inicializa todos los elementos del objeto

Principal
Class FrameTabla

---

public class **FrameTabla**
extends javax.swing.JFrame

Author:
armando
See Also:

Constructor Detail

FrameTabla

public **FrameTabla**(java.util.Hashtable hashtable)
Creates new form FrameTabla

Principal
Class ImageData

---

public class **ImageData**
extends java.lang.Object

Esta clase es para poder tener la información de los datos de la Imagen

---

Constructor Detail

ImageData

public **ImageData**()

Method Detail

setData

public void **setData**(dicomData)

---

asdf

public java.awt.Image **asdf**()

---

getDefaultImage

public java.awt.Image **getDefaultImage**()

---

color

public boolean **color**()

---

wwANDwl

public java.awt.Image **wwANDwl**(int argWW,
int argWL)

---

setWwWl

public void **setWwWl**(int argWW,
int argWL)

---

getImageWWWL2Current

public java.awt.Image **getImageWWWL2Current**(int argWW,
int argWL)

---

getWW

public int **getWW**()

---

getWL

public int **getWL**()

getDefaultWW

public int **getDefaultWW**()

getDefaultWL

public int **getDefaultWL**()

getWidth

public int **getWidth**()

getHeight

public int **getHeight**()

getPixelMin

public int **getPixelMin**()

getPixelMax

public int **getPixelMax**()

getpixLength

public int **getpixLength**()

inverse

public void **inverse**()

setInverse

public void **setInverse**(boolean flag)

setColor

public void **setColor**(boolean flag)

changeColor

public void **changeColor**()

setDefaultPixel

public void **setDefaultPixel**()

rotateL

public void **rotateL**()

rotateR

public void **rotateR**()

flipLR

public void **flipLR**()

flipUD

public void **flipUD**()

getHistogram

public int[] **getHistogram**()

getHistMax

public int **getHistMax**()

reviseHistogram

public java.awt.Image **reviseHistogram**()

Principal
Class ImageProcessor

java.lang.Object

public class **ImageProcessor**
extends javax.swing.JComponent

Esta clase controla el procesamiento de la imagen
See Also:

Constructor Detail

ImageProcessor

public **ImageProcessor**(java.awt.image.BufferedImage image)

Method Detail

paintComponent

public void **paintComponent**(java.awt.Graphics g)
Overrides:
`paintComponent` in class `javax.swing.JComponent`

main

public static void **main**(java.lang.String[] args)

Principal
Class JFrameSobreSistema

public class **JFrameSobreSistema**
extends javax.swing.JFrame

Esta clase nos permite mostrar información de quien creo este sistema y poderlo controlar como JFrame
Author:
Armando
See Also:

Constructor Detail

JFrameSobreSistema

public **JFrameSobreSistema**()
Creates new form JFrameSobreSistema

Method Detail

main

public static void **main**(java.lang.String[] args)
Parameters:
`args` - the command line arguments

Principal

public class **JPanelErrores**
extends javax.swing.JPanel

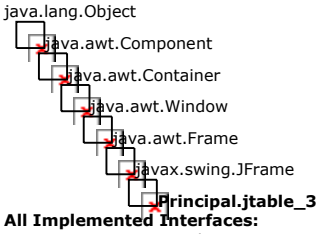
Esta clase genera un JPanel para los posibles errores
Author:
Armando
See Also:

Constructor Detail

JPanelErrores

public **JPanelErrores**()
Creates new form JPanelErrores

Principal
Class jtable_3

java.lang.Object


All Implemented Interfaces:
java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

public class **jtable_3**
extends javax.swing.JFrame

Esta clase maneja la información para la tabla y la construye
See Also:

Constructor Detail

jtable_3

public **jtable_3**()
inicializa los vetores y componentes

---

jtable_3

public **jtable_3**(java.util.Vector vecto_Datos)

Method Detail

inicializar

public void **inicializar**()

---

initVectoresDicom

public void **initVectoresDicom**(java.util.Vector vecto_Datos)

---

clear_vectRowData

public void **clear_vectRowData**()
limpia el contenido del vector de datos

---

inser_Fila

public void **inser_Fila**(java.util.Vector dato)
Inserta un contenido en el vector de datps

---

inser_ColumnNames

public void **inser_ColumnNames**(java.util.Vector dato)
Inserta un contenido en el vector de nombre de columnas

---

clear_ColumnNames

public void **clear_ColumnNames**()
Limpia el contenido del nombre de columnas

---

getvectRowData

public java.util.Vector **getvectRowData**()
Regresa el vetor de datos

---

getvectColumnNames

public java.util.Vector **getvectColumnNames**()
Regresa el vector de los nombres de las columnas

---

cargaColumnNames

public void **cargaColumnNames**(java.util.Vector Datos)

---

cargaVectordDatos

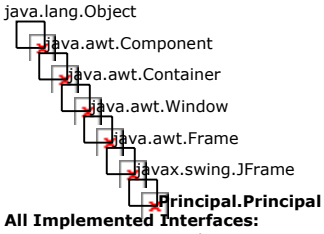
public void **cargaVectordDatos**(java.util.Vector Datos)

---

main

public static void **main**(java.lang.String[] args)
Parameters:
`args` - the command line arguments

Principal
Class Principal

java.lang.Object


All Implemented Interfaces:
java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

public class **Principal**
extends javax.swing.JFrame

La clase Principal controla la presentación a usuario y contiene la estructura general lo que se muestra
Author:
Armando
See Also:

Constructor Detail

Principal

public **Principal**()
Creates new form Principal

Method Detail

limpiar

public void **limpiar**()

---

capturaExped

public void **capturaExped**()
Carga la información del archivo carga diccionarrio carga dato

---

main

public static void **main**(java.lang.String[] args)
Parameters:
`args` - the command line arguments

Principal
Class Vectores

java.lang.Object


---

public class **Vectores**
extends java.lang.Object

Esta clase es de manejo de vectores se considera que el vector es de la forma vector[][] Nota: se asume que el valor del tag es unico
Author:
armando

Constructor Detail

Vectores

public **Vectores**(java.util.Vector vectordato)
Inicializa el V ector de datos a procesar

Method Detail

isContains

public boolean **isContains**(java.lang.String Tag)
Revisa si esta contenido en el vector el tag

---

getfilatag

public java.util.Vector **getfilatag**(java.lang.String Tag)
Busca la fila del tag solicitado y entrega el vector de los datos de la fila

---

getNumeroIdentificacionPaciente

public java.lang.String **getNumeroIdentificacionPaciente**()
Regresa el valor de Si encontro: Numero de identificación del paciente No Encontro: NULL

---

getPaciente

public java.lang.String **getPaciente**()
Busca la informción del paciente en el vector y entregalo en un vector

---

creaDatosPaciente

public void **creaDatosPaciente**()
Regresa solo los valores para el vector de paciente y quedan registrados en el vator de datos del paciente

---

getDatosPaciente

public java.util.Vector **getDatosPaciente**()

---

creaDatosEstudio

public void **creaDatosEstudio**()
Crea vector de los datos de Estudio en base a vector general

---

getDatosEstudio

public java.util.Vector **getDatosEstudio**()
Regresa el vector de datos del Estudio

---

getEstudio

public java.lang.String **getEstudio**()
Busca la información del vector de estudio y entrega un string de toda la información

---

getIdentificadorInstanciaEstudio

public java.lang.String **getIdentificadorInstanciaEstudio**()

---

creaDatosEquipo

public void **creaDatosEquipo**()
Crea el Vector de los datos del equipo en vase a vector general

---

getDatosEquipo

public java.util.Vector **getDatosEquipo**()
Regresa el vector de datos del equipo

---

getEquipo

public java.lang.String **getEquipo**()
Busca la información del vector de equipo y entrega el string de toda la información

---

getIdEquipo

public java.lang.String **getIdEquipo**()
Entrega el el fabricante del equipo

---

creaDatosSerie

public void **creaDatosSerie**()
Crea el Vector de los datos del equipo en vase a vector general

---

getDatosSerie

public java.util.Vector **getDatosSerie**()
Regresa el vector de datos del equipo

---

getSerie

public java.lang.String **getSerie**()
Busca la información del vector de equipo y entrega el string de toda la información

---

getIdSerie

public java.lang.String **getIdSerie**()
Entrega el el fabricante del equipo

---

creaDatosImagen

public void **creaDatosImagen**()
Crea el Vector de los datos del equipo en vase a vector general

---

getDatosImagen

public java.util.Vector **getDatosImagen**()
Regresa el vector de datos del Imagen

---

getImagen

public java.lang.String **getImagen**()
Busca la información del vector de equipo y entrega el string de toda la información

---

getIdImagen

public java.lang.String **getIdImagen**()
Entrega el el fabricante del equipo

---

main

public static void **main**(java.lang.String[] args)
Prueba esta Clase
Parameters:
`args` - the command line arguments

Anexo B

Acontinuación se muestran los programas hechos para la Adquisición de Imagenes.

/root/Desarrollo/spacsm/Princi_2/src/Principal/Comun_Mysql.java

```
/*
 * Comun_Mysql.java
 *
 * Created on 27 de septiembre de 2005, 11:06 AM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package Principal;

import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGEncodeParam;
import com.sun.image.codec.jpeg.JPEGImageEncoder;
import java.awt.*;
import java.awt.image.*;
import java.awt.Image;
import java.io.*;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.sql.*;
import java.sql.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

/**
 * Comunica con MYSQL para pedir datos
 * y sacarlos.
 *
 * @author Armando Jim□nez Herrera
 */
public class Comun_Mysql {

//    /** Creates a new instance of Comun_Mysql */
//
//  1. Establecer Driver
//  2. Establecer conexi□n
//  3. Crear sentencia
//  4. Ejecutar sentencia
//  5. Procesar resultados
//  6. Finalizar sentencia
//  7. Cerrar conexi□n
//

// Valores Constantes
    private static final String LOGIN = "guest";
    private static final String PASSWORD = "guest";
    private static final String URL = "jdbc:mysql://localhost/spacsm";
    private static final String DRIVER = "com.mysql.jdbc.Driver";

//    private static final String INSERTAR_PACIENTE =
//            "INSERT INTO paciente" +
//            "(Numero_de_identificación_del_paciente," +
//            "Nombre_completo_del_paciente," +
//            "Fecha_de_nacimiento_del_paciente" +
//            "Sexo_del_paciente," +
//            "SOPClassUID," +
//            "SOPInstanceUID) VALUES (";

    // Insertar en tablas

    private static final String INSERTAR_PACIENTE =
            "INSERT INTO Paciente " +
            "VALUES (";

    private static final String INSERTAR_ESTUDIO =
            "INSERT INTO Estudio " +
            "VALUES (";


    private static final String INSERTAR_EQUIPO =
            "INSERT INTO Equipo " +
            "VALUES (";

    private static final String INSERTAR_SERIE =
            "INSERT INTO Serie " +
            "VALUES (";

    private static final String INSERTAR_IMAGEN =
            "INSERT INTO Imagen " +
            "VALUES (";



    // □Esta contenido?

    private static final String ISCONTAINS_PACIENTE =
            "SELECT COUNT(*) FROM Paciente WHERE " +
            "NumeroIdentificacionPaciente =";

    private static final String ISCONTAINS_ESTUDIO =
            "SELECT COUNT(*) FROM Estudio WHERE " +
            "IdentificadorInstanciaEstudio =";

    private static final String ISCONTAINS_EQUIPO =
            "SELECT COUNT(*) FROM Equipo WHERE " +
            "FabricanteEquipo =";
    private static final String ISCONTAINS_SERIE =
            "SELECT COUNT(*) FROM Serie WHERE " +
            "IdentificadorSerie =";


    private static final String ISCONTAINS_IMAGEN =
            "SELECT COUNT(*) FROM Imagen WHERE " +
            "InstanceNumber =";



    private static final String INF_PACIENTE =
            "SELECT * FROM Paciente WHERE " +
            "NumeroIdentificacionPaciente =";

    private static final String INF_PACIENTE_TODOS =
            "SELECT * FROM Paciente";



    private static final String RPAREN = ")";
    private static final String COMMA = ",";
    private static final String QUOTE = "'";


    int len;
    int width;
    int height;
    Image imagen;



    public Comun_Mysql() {
        //con     =   null;
        len     =   0;
        width   =   0;
        height  =   0;
        imagen  =   null;
    }

   /**
    *  Prueba la comunicación de con el servidor Mysql
    *
    *  @param login
    *  @param password
    *  @return boolean
    */
    public boolean Test(String login, String password){
        Connection  con = null;
        boolean     flag = false;
        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password );

            if(!con.isClosed()){
                System.out.println("YA ESTOY CONECTADO Y " +
                        "LA PRUEBA ES PERFECTA");
                flag = true;
            }


        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
            //Principal.mandaError ( e.getMessage());
```

```
            //jTextAreaErrores.append("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
            return flag;
        }
    }


/**
 * Se comunica con Mysql y trae toda la tabla de los datos del
 * diccionario
 *  @param  login
 *  @param  password
 *  @return  table   Que es un Hashtable con la llave de TAG
 */
    public Hashtable comun_DicomDic(String login, String password) {
        Connection  con = null;
        Hashtable   table = new Hashtable();

        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();
                ResultSet res = stmt.executeQuery("SELECT * FROM " +
                        "DataDiccionary");
                String tag = null;


                while(res.next()) {
                    tag             = ("("+res.getString("idTAG")+")");
                    //tag = ("("+tag+")");
                    DicomValueDic   dicomvalue = new DicomValueDic();
                    dicomvalue.vr       = res.getString("DD_VR");
                    dicomvalue.name     = res.getString("DD_Name");
                    dicomvalue.vm       = res.getString("DD_VM");
                    dicomvalue.version  = res.getString("DD_Version");
                    System.out.println (tag + " " + dicomvalue.vr
                            + dicomvalue.name);

                    table.put(tag, dicomvalue);
                 }
            res.close();
            stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
            return table;
        }

    }


    /**
     * Revisa la base de datos si exite
Numero_de_identificacion_del_paciente
     * dado.
     * Valores
     *   1  Si existe
     *   0  No existe
     *   -1 Existe mas de 1 registro
     */

    public int isContainsPaciente (String login, String password,
            String numIDPaciente){


        Connection  con = null;
        int  tf = -1;

        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(ISCONTAINS_PACIENTE +
                        QUOTE + numIDPaciente + QUOTE);
                int numRegistros = getCount(rs);
                System.out.println ("esp"+ numRegistros);

                //nota revisar que compruebe el valor t/f segun el valor esperado
                switch (numRegistros){
                    case  0: tf =0;
                            break;
                    case  1: tf = 1;
                            break;
                    default: tf = -1;
                            break;
                }
                System.out.println ("aqui enbtro en estasuspendido<" + tf +
">");
                System.out.flush();
                rs.close();
                stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
        return tf;
        }

    }

    private int getCount(ResultSet rs) throws SQLException {
        rs.next();
        return rs.getInt(1);
    }



/**
 * Inserta un registro de Paciente
 *    1 exito
 *    0 error
 *
 */
    public int insertPaciente (String login, String password,
            String datos){
//     0 login
//     1 password
//     2 Numero de identificación del paciente  char[64]
//     3 Nombre completo del paciente char[64]
//     4 fecha de nacimiento yyyy-mm-dd
//     5 Sexo de paciente  char
//     6 sopClassUID    char 64
//     7 SOPInstanceUID char 64

        Connection      con     =   null;
        int             insertedRows = 0;
        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();

                String queryString = INSERTAR_PACIENTE +
                        datos + RPAREN;

                System.out.println (queryString);

                insertedRows = stmt.executeUpdate(queryString);
                if (insertedRows != 1) {
                    throw new SQLException( "problems when inserting !!!!");
                }

                stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
        return insertedRows;

        }
        }


    /**
     * Busca toda la información del paciente
     * y regresa el vector
     * Si flagTodo = tru  & numidpaciente  null regresa el vector de toda la
tabla
     *    flagTodo = false regresa el vetor del paciente dado
     *
     *
     */
```

```
    public Vector infPaciente (String login, String password,
      boolean flagTodo, String numIDPaciente){


    Connection  con = null;
    Vector result = new Vector();
    Statement stmt;
    ResultSet rs;

    try {
       Class.forName(DRIVER).newInstance();
       con = DriverManager.getConnection(URL, login, password);
       if (con != null) {
          stmt = con.createStatement();
          if (flagTodo)
             rs = stmt.executeQuery(INF_PACIENTE_TODOS);
          else
             rs = stmt.executeQuery(INF_PACIENTE +
                QUOTE + numIDPaciente + QUOTE);


          while (rs.next()) {
             int i = 1;
             Vector fila = new Vector();
//    2 Numero de identificación del paciente  char[64]
             fila.addElement(rs.getString(i++));
//    3 Nombre completo del paciente char[64]
             fila.addElement(rs.getString(i++));
//    4 fecha de nacimiento yyyy-mm-dd
             fila.addElement(rs.getString(i++));
//    5 Sexo de paciente  char
             fila.addElement(rs.getString(i++));
//    6 sopClassUID    char 64
             fila.addElement(rs.getString(i++));
//    7 SOPInstanceUID char 64
             fila.addElement(rs.getString(i++));

             System.out.println("este es el valor de la fila: " + fila);

             result.addElement (fila);

          }
          rs.close();
          stmt.close();
       }
    } catch(Exception e) {
       System.err.println("Exception: " + e.getMessage());
    } finally {
       try {
          if(con != null) con.close();
       } catch(SQLException e) {}
    return result;
    }

  }


  /**
  * Revisa la base de datos si exite IdentificadorInstanciaEstudio
  * dado.
  * Valores
  *   1  Si existe
  *   0  No existe
  *   -1 Existe mas de 1 registro
  */



  public int isContainsEstudio (String login, String password,
        String IdentificadorInstanciaEstudio){


    Connection  con = null;
    int  tf = -1;

    try {
       Class.forName(DRIVER).newInstance();
       con = DriverManager.getConnection(URL, login, password);
       if (con != null) {
          Statement stmt = con.createStatement();
          ResultSet rs = stmt.executeQuery(ISCONTAINS_ESTUDIO +
                QUOTE + IdentificadorInstanciaEstudio + QUOTE);
          int numRegistros = getCount(rs);

          switch (numRegistros){
             case  0: tf =0;
                   break;
             case  1: tf = 1;
                   break;
             default: tf = -1;
                   break;
          }
          rs.close();
          stmt.close();
       }
    } catch(Exception e) {
       System.err.println("Exception: " + e.getMessage());
    } finally {
       try {
          if(con != null) con.close();
       } catch(SQLException e) {}
    return tf;
    }

  }



/**
* Inserta un registro de Paciente
*    1 exito
*    0 error
*
*/
  public int insertEstudio(String login, String password,
        String datos){

    Connection      con     =    null;
    int             insertedRows = 0;

    try {
       Class.forName(DRIVER).newInstance();
       con = DriverManager.getConnection(URL, login, password);
       if (con != null) {
          Statement stmt = con.createStatement();

          String queryString = INSERTAR_ESTUDIO +
                datos + RPAREN;

          System.out.println (queryString);

          insertedRows = stmt.executeUpdate(queryString);
          if (insertedRows != 1) {
             throw new SQLException( "PROBLEMAS EN INSERCCION !!!!");
          }

          stmt.close();
       }
    } catch(Exception e) {
       System.err.println("Exception: " + e.getMessage());
    } finally {
       try {
          if(con != null) con.close();
       } catch(SQLException e) {}
    return insertedRows;

    }
    }

  /**
  * Revisa la base de datos si exite Numero_de_identificación_del_Equipo
  * dado.
  * Valores
  *   1  Si existe
  *   0  No existe
  *   -1 Existe mas de 1 registro
  */

  public int isContainsEquipo(String login, String password,
        String FabricanteEquipo){


    Connection  con = null;
    int  tf = -1;

    try {
       Class.forName(DRIVER).newInstance();
       con = DriverManager.getConnection(URL, login, password);
       if (con != null) {
          Statement stmt = con.createStatement();
          ResultSet rs = stmt.executeQuery(ISCONTAINS_EQUIPO +
                QUOTE + FabricanteEquipo + QUOTE);
          int numRegistros = getCount(rs);
          System.out.println ("esp"+ numRegistros);

          //nota revisar que compruebe el valor t/f segun el valor esperado
          switch (numRegistros){
             case  0: tf =0;
                   break;
```

```
                case  1: tf = 1;
                        break;
                default: tf = -1;
                        break;
                }
                System.out.println ("isContainsEquipo <" + tf + ">");
                System.out.flush();
                rs.close();
                stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
        return tf;
        }

    }




 /**
  * Inserta un registro de Equipo
  *    1 exito
  *    0 error
  *
  */
    public int insertEquipo (String login, String password,
            String datos){
//     0 FabricanteEquipo

        Connection      con     =   null;

        int             insertedRows = 0;
        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();

                String queryString = INSERTAR_EQUIPO +
                        datos + RPAREN;

                System.out.println (queryString);

                insertedRows = stmt.executeUpdate(queryString);
                if (insertedRows != 1) {
                    throw new SQLException( "problemas al insertar en tabla de
equipo");
                }

              stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
        return insertedRows;

        }
        }

    /**
    * Revisa la base de datos si exite IdentificadorInstanciaEstudio
    * dado.
    * Valores
    *   1  Si existe
    *   0  No existe
    *   -1 Existe mas de 1 registro
    */

    public int isContainsSerie (String login, String password,
            String IdentificadorInstanciaSerie){


        Connection  con = null;
        int  tf = -1;

        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(ISCONTAINS_SERIE +
                        QUOTE + IdentificadorInstanciaSerie + QUOTE);
                int numRegistros = getCount(rs);

                switch (numRegistros){
                    case  0: tf =0;
                            break;
                    case  1: tf = 1;
                            break;
                    default: tf = -1;
                            break;
                }
                rs.close();
                stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}

        System.out.println ("-Serie- el valor de resultdo de encontrado >" +
                tf + "< el valor de id es <" +
                IdentificadorInstanciaSerie + ">")    ;
        return tf;
        }

    }



/**
* Inserta un registro de Paciente
*    1 exito
*    0 error
*
*/
    public int insertSerie(String login, String password,
            String datos){

        Connection      con     =   null;
        int             insertedRows = 0;

        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();

                String queryString = INSERTAR_SERIE +
                        datos + RPAREN;

                System.out.println (queryString);

                insertedRows = stmt.executeUpdate(queryString);
                if (insertedRows != 1) {
                    throw new SQLException( "PROBLEMAS EN INSERCCION Serie");
                }

              stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
        return insertedRows;

        }
        }


    /**
    * Revisa la base de datos si exite IntanceNumbre (imagen)
    * dado.
    * Valores
    *   1  Si existe
    *   0  No existe
    *   -1 Existe mas de 1 registro
    */

    public int isContainsImagen (String login, String password,
            String IdImagen){


        Connection  con = null;
```

```
        int  tf = -1;

        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(ISCONTAINS_IMAGEN +
                        QUOTE + IdImagen + QUOTE);
                int numRegistros = getCount(rs);

                switch (numRegistros){
                    case  0: tf =0;
                             break;
                    case  1: tf = 1;
                             break;
                    default: tf = -1;
                             break;
                }
                rs.close();
                stmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}

        System.out.println ("-Imagen- el valor de resultdo de encontrado >" +
                tf + "< el valor de id es <" +
                IdImagen + ">")    ;
        return tf;
        }

    }



    /**
    * Inserta un registro de Imagen
    *    1 exito
    *    0 error
    *
    */
      public int insertImagen(String login, String password,
             String datos, Image imagenori){

        FileInputStream fis = null;
        Connection      con     =   null;
        int             insertedRows = 0;
        String  nombreArch = "imagentmp.jpg";

        imagen = imagenori;

        width  = imagen.getWidth(null);
        height = imagen.getHeight(null);
        // Crea archivo jpeg

        creaArchJPG(nombreArch);

       // habre el archivo y preparalo para mandarlo mysql

        try{
           fis = new FileInputStream(nombreArch);
           len = fis.available();
        }

        catch(IOException ioe) {
            System.err.println("Error en el archivo al abrirlo");
            return (-1);
        }




        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                PreparedStatement pStmt =
con.prepareStatement(INSERTAR_IMAGEN +
                        datos + COMMA + QUOTE+
                        len + QUOTE +
                        RPAREN);

                /*
                  private static final String RPAREN = ")";
    private static final String COMMA = ",";
    private static final String QUOTE = "'";

                */
                pStmt.setBinaryStream(1, fis, len);
                System.out.println("manda a ejecutar sentencia");
                pStmt.executeUpdate();
                System.out.println("Almacenado imagen " + fis
                        + ", tamano: " + fis.available());
                pStmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
        } finally {
            try {
                if(con != null) con.close();
            } catch(SQLException e) {}
        return insertedRows;

        }
        }


/*****************************************/
/*****************************************/



    /**
     * Carga los datos de un archivo
     * regresa true con exito
     * regresa false cuando ocurre un error
     */
    public boolean CargaImagen (String nomArch){
        try {
        File file = new File(nomArch);
        imagen = ImageIO.read(file);
        } catch (IOException e) {
            return false;
        }
        width  = imagen.getWidth(null);
        height = imagen.getHeight(null);
        return true;
    }


    /**
     * Muestra en una venta una imagen
     */
    public void MuestraVentana(String cadena, Image ima) {


        int w = ima.getWidth(null),
            h = ima.getHeight(null);

        BufferedImage buffImage = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);
        Graphics2D imageGraphics = buffImage.createGraphics();
        imageGraphics.drawImage(ima, 0, 0, null);

        JFrame jFrame1 = new JFrame(cadena);
        jFrame1.getContentPane().add(new ImageProcessor(buffImage));
        jFrame1.setSize(buffImage.getWidth(), buffImage.getHeight());
        jFrame1.setDefaultCloseOperation( JFrame.DISPOSE_ON_CLOSE );
        jFrame1.setVisible(true);
    }


    /**
     */
    public boolean creaArchJPG(String nombArch) {
        try {
            // Create an image buffer
            BufferedImage outImage = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
            // Paint image.
            Graphics bg = outImage.getGraphics();
            bg.drawImage(imagen,0,0,null);
            bg.dispose();
            /*
            Graphics2D g2d = outImage.createGraphics();
            g2d.drawImage(imagen, new AffineTransform(), null);
            g2d.dispose();
             *
             */
            // JPEG-encode the image and write to file.
            System.out.println("Image2jpg: writing jpg encoded file " +
nombArch );
            OutputStream os = new FileOutputStream(nombArch);
```

```
            JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(os);
            JPEGEncodeParam oParam =
encoder.getDefaultJPEGEncodeParam(outImage);
            oParam.setQuality(1f, false);
            oParam.setDensityUnit(1);
            oParam.setXDensity(300);
            oParam.setYDensity(300);/**/
            encoder.setJPEGEncodeParam(oParam);
            encoder.encode(outImage);
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
            return(false);
        }
    return(true);
    }




public static void saveMySqlPhoto(String empID, Image image) {
  try {

    BufferedImage bImage = new
BufferedImage(image.getWidth(null),image.getHeight(null),BufferedImage.TYPE
_INT_RGB);
    Graphics bg = bImage.getGraphics();
    bg.drawImage(image,0,0,null);
    bg.dispose();
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    ImageIO.write(bImage,"jpeg",out);
    byte[] buf = out.toByteArray();
    // setup stream for blob
    ByteArrayInputStream inStream = new ByteArrayInputStream(buf);
    // get or create a connection here

  }
  catch (Exception exc) {
    // process error
  }
}



    public boolean saveImage (String login, String password,
           String IdImagenes, Image image){

        Connection      con     =   null;
        int insertedRows = 0;
        FileInputStream fis = null;
        int len = 0;

        try{
           fis = new FileInputStream(IdImagenes + ".jpg");
           len = fis.available();
        }

        catch(IOException ioe) {
            System.err.println("Error en el archivo al abrirlo");
            return (false);
        }

        try {
            Class.forName(DRIVER).newInstance();
            con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                PreparedStatement pStmt = con.prepareStatement("insert " +
                        "into imagenes values (?,?,?)");
                pStmt.setString(1, IdImagenes);
                pStmt.setInt(2, len);
                pStmt.setBinaryStream(3, fis, len);
                pStmt.executeUpdate();
                System.out.println("Stored: " + fis
                        + ", length: " + len);
                pStmt.close();
            }
        } catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
            return (false);
        } finally {

            try {
                fis.close();
            } catch(IOException CloseError){
                System.err.println("Error al cerrar el archivo");
                return (false);
            }
            try {
                if(con != null) con.close();
            } catch(SQLException e) {
                return false;
            }
        return true;

        }
        }

// This method returns a buffered image with the contents of an image
    public static BufferedImage toBufferedImage(Image image) {
        if (image instanceof BufferedImage) {
            return (BufferedImage)image;
        }
        // This code ensures that all the pixels in the image are loaded
        image = new ImageIcon(image).getImage();
        // Determine if the image has transparent pixels; for this method"s
        // implementation, see e661 Determining If an Image Has Transparent
Pixels
        boolean hasAlpha = hasAlpha(image);
        // Create a buffered image with a format that"s compatible with the
screen
        BufferedImage bimage = null;
        GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        try {
            // Determine the type of transparency of the new buffered image
            int transparency = Transparency.OPAQUE;
            if (hasAlpha) {
                transparency = Transparency.BITMASK;
            }
            // Create the buffered image
            GraphicsDevice gs = ge.getDefaultScreenDevice();
            GraphicsConfiguration gc = gs.getDefaultConfiguration();
            bimage = gc.createCompatibleImage(
                image.getWidth(null), image.getHeight(null), transparency);
        } catch (HeadlessException e) {
            // The system does not have a screen
        }

        if (bimage == null) {
            // Create a buffered image using the default color model
            int type = BufferedImage.TYPE_INT_RGB;
            if (hasAlpha) {
                type = BufferedImage.TYPE_INT_ARGB;
            }
            bimage = new BufferedImage(image.getWidth(null),
image.getHeight(null), type);
        }
        // Copy image to buffered image
        Graphics g = bimage.createGraphics();
        // Paint the image onto the buffered image
        g.drawImage(image, 0, 0, null);
        g.dispose();
        return bimage;
    }


    // This method returns true if the specified image has transparent pixels
    public static boolean hasAlpha(Image image) {
        // If buffered image, the color model is readily available
        if (image instanceof BufferedImage) {
            BufferedImage bimage = (BufferedImage)image;
            return bimage.getColorModel().hasAlpha();
        }
        // Use a pixel grabber to retrieve the image"s color model;
        // grabbing a single pixel is usually sufficient
         PixelGrabber pg = new PixelGrabber(image, 0, 0, 1, 1, false);
        try {
            pg.grabPixels();
        } catch (InterruptedException e) {
        }

        // Get the image"s color model
        ColorModel cm = pg.getColorModel();
        return cm.hasAlpha();
    }

      public Image getImageFile(String login, String password,
          String fileName) throws Exception {
        Connection      con     =   null;
        Image img = null;
        //String baseName=StoreBinary.getBaseName(fileName);
        Class.forName(DRIVER).newInstance();
        con = DriverManager.getConnection(URL, login, password);
            if (con != null) {
                Statement stmt = con.createStatement();
```

```
            ResultSet rs=stmt.executeQuery("select * from " +
                    "Imagenes where IdImagenes=""
                    +fileName+""");
        if (!rs.next()) {
            System.out.println("Image:"+fileName+" not found");
            return null;
        }
        len=rs.getInt(2);
        byte [] b=new byte[len];
        InputStream in = rs.getBinaryStream(3);
        in.read(b);
        in.close();
        ImageIcon icon = new ImageIcon(b);
        img = icon.getImage();
        //img=Toolkit.getDefaultToolkit().createImage(b);
        System.out.println("Image: "+fileName+" retrieved ok, size: "+ len);
        rs.close();
        stmt.close();
            }
        return img;
    }


      public Image getImagen (){
          return imagen;
      }



/*******************************************/
/*******************************************/









///////////////**
/////////////// *  Prueba esta clase
/////////////// */
///////////////    public static void main(String[] args) {
///////////////        // TODO code application logic here
///////////////        Comun_Mysql    comun_mysql= new Comun_Mysql();
///////////////        //comun_mysql.Test(LOGIN, PASSWORD);
///////////////        System.out.println("el que se busca es : 00001 "+ "El
resultado es:" + comun_mysql.isContainsPaciente(LOGIN, PASSWORD,
"00001"));
///////////////        System.out.println("el que se busca es : 0001 "+ "El
resultado es:" + comun_mysql.isContainsPaciente(LOGIN, PASSWORD,
"0001"));
///////////////
///////////////
////////////////     0 login
////////////////     1 password
////////////////     2 Numero de identificación del paciente  char[64]
////////////////     3 Nombre completo del paciente char[64]
////////////////     4 fecha de nacimiento yyyy-mm-dd
////////////////     5 Sexo de paciente  char
////////////////     6 sopClassUID    char 64
////////////////     7 SOPInstanceUID char 64
///////////////
///////////////        String[] nuevoPaciente= {"0011",
///////////////                                 "Nombre completo del paciente
char[64]",
///////////////                                 "1970-10-21",
///////////////                                 "F",
///////////////                                 "6 sopClassUID    char 64",
///////////////                                 "7 SOPInstanceUID char 64"};
///////////////
///////////////
///////////////
///////////////        System.out.println( comun_mysql.insertPaciente (LOGIN,
PASSWORD, nuevoPaciente) );
///////////////        System.out.println( "*************************" );
///////////////        System.out.println( "*************************" );
///////////////        System.out.println( comun_mysql.infPaciente (LOGIN,
PASSWORD, false, "0001") );
///////////////        System.out.println( comun_mysql.infPaciente (LOGIN,
PASSWORD, true, null) );
///////////////
///////////////
///////////////        //Hashtable  tabla =
comun_mysql.comun_DicomDic(LOGIN, PASSWORD);
///////////////        System.out.println ("Ya termino");
///////////////
///////////////    }
///////////////
///////////////

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/DicomDic.java

```
package Principal;

import java.io.*;
import java.util.*;
import java.io.Serializable;
import java.util.Hashtable;


//import java.net.*;

/**
 *  CREA UNA TABLA DEL DICCIÓNARIO DE TAGs
 */

public class DicomDic {
Hashtable   table ;


  public  DicomDic () {
    table = new Hashtable();
}

/**
 *  Llama a comunicación para que le entrege la tabla completa
 *  de Diccionario Dicom en Mysql
 *
 */
  public void loadDicomDic(String login, String password) {
        Comun_Mysql    comun_mysql= new Comun_Mysql();
        this.table = comun_mysql.comun_DicomDic(login, password);


  }

/**
 * Entraga la descripción del TAG solicitado
 * @param tag
 * @return name
 */
  public String getName(String tag) {
    return ((DicomValueDic)table.get(tag)).name;
  }


  /**
   * Entrega el VR del TAG solicitadp
   * @param tag
   * @return vr
   */
  public String getVR(String tag) {
//    return ((DicomValueDic)table.get(tag)).vr;
//      Dicomvalue temp = new Dicomvalue();
//      temp = (Dicomvalue)table.get(tag);
//      String a = temp.vr;
//
//      return a;
    return ((DicomValueDic)table.get(tag)).vr;
  }

  /**
   * Entrega el VM del TAG solicitado
   * @param tag
   * @return vm
   */
  public String getVM(String tag) {
    return ((DicomValueDic)table.get(tag)).vm;
  }


  /**
   * Entrega la version del TAG solicitado
   * @param tag
   * @return version
   */
```

```
  public String getVersion(String tag) {
   //return ((DicomvalueDic)table.get(tag)).version;
    return ((DicomValueDic)this.table.get(tag)).version;
  }

  /**
   * Busca si esta contenido en la tabla de un TAG
   * @param tag
   */
  public boolean isContain(String tag) {
    return table.containsKey(tag);
  }



//  public static void main(String[] args) {
//
//    String LOGIN     = "guest";
//    String PASSWORD  = "guest";
//
//        System.out.printf ("Inicia Carga diccionario de Tag");
//        DicomDic dicomDiccc = new DicomDic();
//        dicomDiccc.loadDicomDic(LOGIN, PASSWORD);
//        System.out.printf ("Termino Carga diccionario de Tag");
//        String  result =  dicomDiccc.getVersion("0028,0010") ;
//        System.out.printf (result);
//
//  }

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/DicomFile.java

```
package Principal;

import java.io.*;
import java.text.*;
import java.net.*;
import java.util.*;

/**
 * Se encarga de manejar lo relacionado al archivo
 * del IOD Compuesto. Para generar una tabla de tipo
 * DicomData y DicomDIC
 *
 */

public class DicomFile {

  int             debug_level = 3;

  boolean         isLtlEndian;
  boolean         vrType;
  boolean         patientPrivacy;
  boolean         VReqSQ = false;
  boolean         containDic;
  boolean         dicomPart10;
  boolean         Part10Endian;
  boolean         Part10Endian2;
  boolean         Part10vr;
  boolean         Part10vr2;
  boolean         Part10flag;
  DicomDic        dicomDic;
  DicomData       dicomData;

  public DicomFile(){
     dicomDic  =   new DicomDic();
     dicomData =   new DicomData();
  }

/**
 *  Da los valores a las banderas
 *
 *  @param argIsLtlEndian   bandera
 *  @param argVRType        bandera
 *  @param privacy          bandera
 *  @param argDicomDic      DicomDic
 */
  public DicomFile(boolean argIsLtlEndian, boolean argVRType, boolean privacy,
DicomDic argDicomDic) {
    patientPrivacy = privacy;
    isLtlEndian = Part10Endian = Part10Endian2 = argIsLtlEndian;
    vrType = Part10vr = Part10vr2 = argVRType;
    dicomDic = argDicomDic;
    dicomPart10 = Part10flag = false;
  }

  public DicomFile(boolean argIsLtlEndian, boolean argVRType, DicomDic
argDicomDic) {
    this(argIsLtlEndian, argVRType, false, argDicomDic);
  }
  public DicomFile(DicomDic argDicomDic) {
    this(true, false, false, argDicomDic);
  }


  public DicomDic getDicomDic(){
     return dicomDic;
  }


  public DicomData getDicomData(){
     return dicomData;
  }


  public Hashtable gethashtable (){
     return dicomData.gethashtable();
  }


  public Vector getVectorDicomData(){
     return dicomData.getVectorDicomData();
  }

   public Vector getPaciente(){
       return dicomData.getPaciente();
   }


/**
 * Esta pensado para URL pero si no tiene la forma se configura para carga
 * el archivo
 * @param   imgURL
 * @return  DicomData
 */


  public DicomData load(String imgURL){

    dicomData = new DicomData();

    try {

      URL urlConn;
      FileInputStream FinS;
      BufferedInputStream inS;
      DataInputStream din;

      if(imgURL.matches("https*:\\p{ASCII}*")){
        System.out.println("http");
        urlConn = new URL(imgURL);
        // InputStream inS = urlConn.openStream();
        inS = new BufferedInputStream(urlConn.openStream());
        din = new DataInputStream(inS);
        System.out.println("http");
      }else{
        System.out.println("file");
        FinS = new FileInputStream(imgURL);
        inS = new BufferedInputStream(FinS);
        din = new DataInputStream(inS);
        System.out.println("file");
      }

      int tempInt;
      int part10sum = 0;
      int part10length = 0;
      byte[] buff2 = new byte[2];
      byte[] buff4 = new byte[4];

      String group;
      String number;
      String tag;
      String vr;
      int length;
      byte[] value;

      while (din.read(buff2) != -1) {
        tempInt = readInt2(buff2);
        group  = Integer.toString((tempInt&0x0000f000)>>12,16);
        group += Integer.toString((tempInt&0x00000f00)>>8,16);
        group += Integer.toString((tempInt&0x000000f0)>>4,16);
        group += Integer.toString((tempInt&0x0000000f),16);
        din.readFully(buff2);
        tempInt = readInt2(buff2);
```

```
        number  = Integer.toString((tempInt&0x0000f000)>>12,16);
        number += Integer.toString((tempInt&0x00000f00)>>8,16);
        number += Integer.toString((tempInt&0x000000f0)>>4,16);
        number += Integer.toString((tempInt&0x0000000f),16);
        tag = ("("+group+","+number+")");
//        tag = (group+","+number);

        System.out.println ("<" + tag + ">");


        if (debug_level > 3) System.out.println("currentTag is : " + tag);
        dicomData.setTag(tag);
        containDic = dicomDic.isContain(tag);

        if(vrType && !VReqSQ){
          StringBuffer sbuff = new StringBuffer(2);
          din.readFully(buff2);
          for(int i=0; i<2; i++)
                sbuff.append((char)buff2[i]);
          dicomData.setVR(tag, sbuff.toString());
          if(sbuff.toString().equals("OB") ||
             sbuff.toString().equals("OW") ||
             sbuff.toString().equals("SQ")) {

            din.skip(2);

            din.readFully(buff4);
            length = readInt4(buff4);

            if (Part10flag) part10sum += 4;
          } else {

            din.readFully(buff2);
            length = readInt2(buff2);
          }
        } else{



          if(containDic) dicomData.setVR(tag, dicomDic.getVR(tag));
          else dicomData.setVR(tag, "na");
          din.readFully(buff4);
          length = readInt4(buff4);
        }

        if(tag.equals("(fffe,e0dd)")) VReqSQ = false;

        vr = dicomData.getVR(tag);

        if (debug_level > 3) System.out.println("currentVR is : " + vr);
        if (debug_level > 3) System.out.println("currentLength: " + length);

        if(length == -1) {
          VReqSQ = true;
          length = 0;
          if(tag.equals("(fffe,e000)")) VReqSQ = false;
        }


        value = new byte[length];
        din.readFully(value);
        dicomData.setValue(tag, value);


        if(containDic) {
          dicomData.setName(tag, dicomDic.getName(tag));
          dicomData.setVM(tag, dicomDic.getVM(tag));
          dicomData.setVersion(tag, dicomDic.getVersion(tag));
        }else {
          dicomData.setName(tag, "NotContainedInDICOMDictionary");
          dicomData.setVM(tag, "na");
          dicomData.setVersion(tag, "na");
        }


        if (debug_level > 3) System.out.println("currentName is : " +
dicomData.getName(tag));

        this.analyzer(tag, vr);


        if(Part10flag){
          if(tag.equals("(0002,0000)")){
            part10length = Integer.parseInt(dicomData.getAnalyzedValue(tag));
          }else{
            part10sum += length;
            part10sum += 8;
            if(part10length == part10sum){
              isLtlEndian = Part10Endian;
              vrType = Part10vr;
              Part10flag = false;
              part10length = 0;
              part10sum = 0;
            }
          }

          if(tag.equals("(0002,0010)")){
            String TransferSyntax = dicomData.getAnalyzedValue(tag);
            if(TransferSyntax.matches("1.2.840.10008.1.2\\p{ASCII}??")){
              if (debug_level > 3) System.out.println("Implicit VR Little
Endian");
              Part10Endian = true;
              Part10vr = false;
            }else
if(TransferSyntax.matches("1.2.840.10008.1.2.2\\p{ASCII}??")){
              if (debug_level > 3) System.out.println("Explicit VR Big Endian");
              Part10Endian = false;
              Part10vr = true;
            }else{
              if (debug_level > 3) System.out.println("Explicit VR Big Endian:");
              Part10Endian = true;
              Part10vr = true;
            }
          }

        }
        if(!dicomPart10 && tag.equals("(0000,0000)") && length == 0){

          din.skip(124);
          if (debug_level > 3) System.out.println("skip");
          isLtlEndian = true;
          vrType = true;
          dicomPart10 = Part10flag = true;
          if (debug_level > 3) System.out.println("part10 ");
        }

        if (debug_level > 5){
          StringBuffer buffer=new StringBuffer();
          for(int i=0; i<dicomData.getValue(tag).length; i++){
            String tmp=Integer.toHexString(dicomData.getValue(tag)[i] & 0xff);
            if(tmp.length()==1){buffer.append("0").append(tmp);
            }else{
              buffer.append(tmp);
            }
          }
          System.out.println("Value = " + buffer);
        }


      }


      if(dicomPart10){
        Part10flag = dicomPart10 = false;
        isLtlEndian = Part10Endian2;
        vrType = Part10vr2;
      }

      din.close();
      inS.close();
    }
    catch(EOFException eof){
      System.out.println("DicomFile.EOFException: " + eof.getMessage() );
    }
    catch(IOException ioe){
      System.out.println("DicomFile.IOException: " + ioe.getMessage() );
    }
    catch(Exception e){
      System.out.println("DicomFile.Exception: " + e.getMessage() );
    }

    if(patientPrivacy) {
      String patientName;

      patientName = dicomData.getAnalyzedValue("(0010,0010)");
      StringBuffer patientBuf = new StringBuffer(patientName);


      for(int i=0; i < patientName.length(); i++) {
        if(i % 2 == 1) patientBuf.setCharAt(i, "*");
      }

      dicomData.setAnalyzedValue("(0010,0010)", patientBuf.toString());
    }

    return dicomData;
  }


  private int readInt2(byte[] argtmp){
```

```
    int tmp;
    if(isLtlEndian) {
      tmp = ((0x000000ff & argtmp[1]) << 8 | (0x000000ff & argtmp[0]));
    } else {
      tmp = ((0x000000ff & argtmp[0]) << 8 | (0x000000ff & argtmp[1]));
    }
    return tmp;
  }


  private int readInt4(byte[] argtmp){
    int tmp;
    if(isLtlEndian) {
      tmp = ((0x000000ff & argtmp[3]) << 24 | (0x000000ff & argtmp[2])
<< 16
          | (0x000000ff & argtmp[1]) << 8  | (0x000000ff & argtmp[0]));
    } else {
      tmp = ((0x000000ff & argtmp[0]) << 24 | (0x000000ff & argtmp[1])
<< 16
          | (0x000000ff & argtmp[2]) << 8  | (0x000000ff & argtmp[3]));
    }
    return tmp;
  }


  private void analyzer(String currentTag, String currentVR) {

    if(currentVR==null){

      dicomData.setAnalyzedValue(currentTag, "Not contain VR.");
    }
    else if(dicomData.getValueLength(currentTag)==0){

      dicomData.setAnalyzedValue(currentTag, "");
    }
    else if(currentVR.equals("PN") | currentVR.equals("LO")
         |     currentVR.equals("SH") | currentVR.equals("LT")
         |     currentVR.equals("ST") | currentVR.equals("UI")
         |     currentVR.equals("DS") | currentVR.equals("CS")
         |     currentVR.equals("IS") | currentVR.equals("AS")){

      for(int j=0; j<dicomData.getValueLength(currentTag); j++)
        if((dicomData.getValue(currentTag))[j] == 0)
          (dicomData.getValue(currentTag))[j] = 20;
      dicomData.setAnalyzedValue(currentTag, new
String(dicomData.getValue(currentTag)));

    }
    else if(currentVR.equals("SS")){
      int tmp;

      if(isLtlEndian){
        tmp = (((int)(dicomData.getValue(currentTag))[1] & 0x000000ff) << 8)
            | ((int)(dicomData.getValue(currentTag))[0] & 0x000000ff);
      } else {
        tmp = (((int)(dicomData.getValue(currentTag))[0] & 0x000000ff) << 8)
            | ((int)(dicomData.getValue(currentTag))[1] & 0x000000ff);
      }
      if((tmp & 0x00008000)==0x00008000)
                              tmp |= 0xffff0000;
      dicomData.setAnalyzedValue(currentTag, Integer.toString(tmp));

    }
    else if(currentVR.equals("US")){
      int tmp;

      if(isLtlEndian){
        tmp = (((int)(dicomData.getValue(currentTag))[1] & 0x000000ff) << 8)
            | ((int)(dicomData.getValue(currentTag))[0] & 0x000000ff);
      } else {
        tmp = (((int)(dicomData.getValue(currentTag))[0] & 0x000000ff) << 8)
            | ((int)(dicomData.getValue(currentTag))[1] & 0x000000ff);
      }
      dicomData.setAnalyzedValue(currentTag, Integer.toString(tmp));
    }
    else if(currentVR.equals("UL")){
      int tmp;

      if (isLtlEndian){
        tmp = (((int)(dicomData.getValue(currentTag))[3] & 0x000000ff) <<
24)
          | (((int)(dicomData.getValue(currentTag))[2] & 0x000000ff) << 16)
          | (((int)(dicomData.getValue(currentTag))[1] & 0x000000ff) <<  8)
          |  ((int)(dicomData.getValue(currentTag))[0] & 0x000000ff);
      } else {
        tmp = (((int)(dicomData.getValue(currentTag))[0] & 0x000000ff) <<
24)
          | (((int)(dicomData.getValue(currentTag))[1] & 0x000000ff) << 16)
          | (((int)(dicomData.getValue(currentTag))[2] & 0x000000ff) <<  8)
          |  ((int)(dicomData.getValue(currentTag))[3] & 0x000000ff);
      }
      dicomData.setAnalyzedValue(currentTag, Integer.toString(tmp));

    }
    else if(currentVR.equals("TM")){
      dicomData.setAnalyzedValue(currentTag, new
String(dicomData.getValue(currentTag)));
      StringBuffer buffer = new
StringBuffer(dicomData.getAnalyzedValue(currentTag));
      buffer.insert(2, ":");
      buffer.insert(5, ":");
      dicomData.setAnalyzedValue(currentTag, buffer.toString());
    }
    else if(currentVR.equals("DA")){
      dicomData.setAnalyzedValue(currentTag, new
String(dicomData.getValue(currentTag)));

      if(dicomData.getValueLength(currentTag)==8){
        StringBuffer buffer = new
StringBuffer(dicomData.getAnalyzedValue(currentTag));
        buffer.insert(4, "-");
        buffer.insert(7, "-");
        dicomData.setAnalyzedValue(currentTag, buffer.toString());
      }else if(dicomData.getValueLength(currentTag) == 10){

        StringTokenizer st = new
StringTokenizer(dicomData.getAnalyzedValue(currentTag), ".");
        String temp  = st.nextToken();
        temp += "-" + st.nextToken();
        temp += "-" + st.nextToken();
        dicomData.setAnalyzedValue(currentTag, temp);
      }
    }
    else

      dicomData.setAnalyzedValue(currentTag, "Unknown VR");

    if (debug_level > 3) System.out.println("AnalyzedValue :" +
dicomData.getAnalyzedValue(currentTag));
  }
}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/DicomvalueData.java

```
/*
 * DicomvalueData.java
 *
 * Created on 5 de octubre de 2005, 03:35 PM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package Principal;

/**
 * Estuctura del objeto
 * @author armando
 */
public class DicomvalueData {


    String      name;
    String      vr;
    String      vm;
    String      version;
    byte[]      value;
    int         valueLength;
    String      analyzedValue;


    /**
     *
     */
    public DicomvalueData() {
    name  = null;
    vr    = null;
    vm    = null;
    version    = null;
    value      = null;
    valueLength    = 0;
    analyzedValue    = null;
    }
```

```
}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/DicomValueDic.java

```
/*
 * DicomValueDic.java
 *
 * Created on 5 de octubre de 2005, 03:12 PM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package Principal;
/**
 * Estructura del objeto para el Diccionario DICOM
 * @author armando
 */
public class DicomValueDic {

    String name;                //Descripción
    String vr;                 //VR
    String vm;                  //VM
    String version;             //Version Dicom
 /**
  * Inicializa todos los elementos del objeto
  */
    public DicomValueDic() {
    name    = null;             //Descripción
    vr      = null;            //VR
    vm      = null;             //VM
    version = null;             //Version Dicom
    }

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/ImageData.java

```
package Principal;

import java.awt.*;
import java.awt.image.*;
import java.io.*;
import java.lang.*;
import com.sun.image.codec.jpeg.*;
import java.io.ByteArrayInputStream;
import javax.imageio.ImageIO;
/**
 * Esta clase es para poder tener la información de los datos de la
 * Imagen
 */
public class ImageData {

  int     debug_level = 5;

  boolean       blackANDwhite = true;
  boolean       rgbMode = false;
  boolean       inv = false;
  int           pixel[];
  int           orgPixel[];
  int           pixLength;
  int           pixelMin;
  int           pixelMax;
  int           width;
  int           height;
  int           histogram[] = new int[256];
  int           histMax;
  int           ww,wl;



  int                 defaultPixel[];
  int                 defaultWidth;
  int                 defaultHeight;
  int     defaultWW, defaultWL;

  boolean jpeg_flag = false;
  BufferedImage img;

  Image   image;
  Toolkit toolkit;
  MemoryImageSource source;
  ImageIO imageio;


  public void setData(DicomData dicomData){
    if (debug_level > 3) System.out.println("Now set width and height....");
    width               =
Integer.parseInt(dicomData.getAnalyzedValue("(0028,0011)"));
    height      = Integer.parseInt(dicomData.getAnalyzedValue("(0028,0010)"));
    defaultWidth = width;
    defaultHeight = height;
    if (debug_level > 3) System.out.println("Image width  : " + width);
    if (debug_level > 3) System.out.println("Image heigth : " + height);
    if (debug_level > 3) System.out.print("Now set byte[] to int[]....");

    orgPixel        = new int[width * height];
    pixLength       = orgPixel.length;
    pixel           = new int[pixLength];
    defaultPixel    = new int[pixLength];

    if(dicomData.getValue("(7fe0,0010)").length <= 10){jpeg_flag = true;}
    byte[] tmpValue;

    if(!jpeg_flag){
      tmpValue = new byte[dicomData.getValue("(7fe0,0010)").length];
      System.arraycopy(dicomData.getValue("(7fe0,0010)"), 0, tmpValue, 0,
tmpValue.length);
    }else{
      tmpValue = new byte[dicomData.getValue("(fffe,e000)").length];
      System.arraycopy(dicomData.getValue("(fffe,e000)"), 0, tmpValue, 0,
tmpValue.length);

      try{
        ByteArrayInputStream bainS = new ByteArrayInputStream(tmpValue);
        //JPEGImageDecoder decoder =
JPEGCodec.createJPEGDecoder(bainS);
        //img = decoder.decodeAsBufferedImage();
        img = imageio.read(bainS);
      }catch(EOFException eof){
        System.out.println("ImageData.EOFException: " + eof.getMessage() );
      }
      catch(IOException ioe){
        System.out.println("ImageData.IOException: " + ioe.getMessage() );
      }
      catch(Exception e){
        System.out.println("ImageData.Exception: " + e.getMessage() );
      }
    }

   if(jpeg_flag){
     int k = 0;
     if(dicomData.isContain("(0028,0004)") &&
dicomData.getAnalyzedValue("(0028,0004)").trim().equals("RGB")) {
       rgbMode = true;
       for(int i=0;i<height;i++){
         for(int j=0;j<width;j++){
           orgPixel[k] = img.getRGB(j,i)&0x00ffffff;
           k++;
         }
       }
     }else{
       if(dicomData.getAnalyzedValue("(0028,0100)").trim().equals("16")) {
         for(int i=0;i<height;i++){
           for(int j=0;j<width;j++){
             orgPixel[k] = img.getRGB(j,i)&0x0000ffff;
             k++;
           }
         }
       }else{
         for(int i=0;i<height;i++){
           for(int j=0;j<width;j++){
             orgPixel[k] = img.getRGB(j,i)&0x000000ff;
             k++;
           }
         }
       }
     }
   }else{


    // (0028,0004)Photometric Interpretation RGB
    if(dicomData.isContain("(0028,0004)") &&
       dicomData.getAnalyzedValue("(0028,0004)").trim().equals("RGB")) {
      rgbMode = true;
      for(int i=0; i<pixLength; i++){
        orgPixel[i] = ((255 << 24) |
                       (0xff & tmpValue[3*i]) << 16 |
                       (0xff & tmpValue[(3*i)+1]) << 8 |
                       (0xff & tmpValue[(3*i)+2]) );
      }
```

```
    } else {

      if(dicomData.getAnalyzedValue("(0028,0100)").trim().equals("16")) {
        short shValue = 0;
        for(int i=0; i<pixLength; i++){
          shValue = (short)((0xff & tmpValue[(2*i)+1]) << 8 |
                            (0xff & tmpValue[2*i]) );
          orgPixel[i] = (int)shValue;

        }

      }else {
        for(int i=0; i<pixLength; i++)
          orgPixel[i] = (int)(0xff & tmpValue[i]);
      }

      int bit_stored =
Integer.parseInt(dicomData.getAnalyzedValue("(0028,0101)"));
      int bit_hight  =
Integer.parseInt(dicomData.getAnalyzedValue("(0028,0102)"));
      int bit_gap    = bit_hight - bit_stored +1;
      if(bit_gap > 0) {
        for(int i=0; i<pixLength; i++) orgPixel[i] = (orgPixel[i] >> bit_gap);
      }
    }
   }

    System.arraycopy(orgPixel, 0, defaultPixel, 0, pixLength);

    tmpValue = null;


    if (debug_level > 3) System.out.print("Now set pixelMin and
pixelMax....");

    pixelMin = 0;
    pixelMax = 0;
    for(int i=0; i<pixLength; i++){
      if(pixelMin > orgPixel[i])
        pixelMin = orgPixel[i];
      if(pixelMax < orgPixel[i])
        pixelMax = orgPixel[i];
    }

    if (debug_level > 3) System.out.println(" OK!");


    if (debug_level > 3) System.out.print("Now set WW/WL....");

    if(dicomData.isContain("(0028,1051)") && !jpeg_flag) {
      try {
        ww =
Integer.parseInt(dicomData.getAnalyzedValue("(0028,1051)").replace("+", "
").trim());
      }catch(NumberFormatException e) {

        ww = pixelMax - pixelMin;
      }
    } else {
      ww = pixelMax - pixelMin;
    }
    if(dicomData.isContain("(0028,1050)") && !jpeg_flag) {
      try {
        wl =
Integer.parseInt(dicomData.getAnalyzedValue("(0028,1050)").replace("+", "
").trim());
      }catch(NumberFormatException e) {

        wl = (ww >> 1) + pixelMin;
      }
    } else {
      wl = (ww >> 1) + pixelMin;
    }
    defaultWW = ww;
    defaultWL = wl;

    if (debug_level > 3) System.out.println(" OK!");
    if (debug_level > 3) System.out.println("WW :" + ww + " WL :" + wl);


    for(int i=0; i < pixel.length; i++) pixel[i] = 0xff000000;

    image = CreaImagen (width, height, pixel);

    /*
    source = new MemoryImageSource(width, height, pixel, 0, width);
    source.setAnimated(true);
    toolkit = Toolkit.getDefaultToolkit();
    image = toolkit.createImage(source);
     */

  }



  private Image CreaImagen(int ancho, int altura, int[] pixeles ){
    Image imagen = null;
    source = new MemoryImageSource(ancho, altura, pixeles, 0, ancho);
    source.setAnimated(true);
    toolkit = Toolkit.getDefaultToolkit();
    imagen = toolkit.createImage(source);
    return imagen;
  }




  private void contrast() {


   if(!rgbMode){
    //int tmp = (int)(ww * 0.5);
    int tmp = ww >> 1;
    int contrastMin = wl - tmp;
    int contrastMax = wl + tmp;

    if(blackANDwhite) {

      double invWW = 255d / (double)ww;
      int pix;

      for(int i=0; i<pixLength; i++){
        pix = orgPixel[i];
        if(pix <= contrastMin) pix = 0;
        else if(pix >= contrastMax) pix = 255;
        else

          // pix = (int)Math.round((255*(pix - contrastMin))/ww);
              pix = (int)((pix - contrastMin) * invWW);
        pixel[i] = (0xff000000 | (pix << 16) | (pix << 8) | pix);

        if(inv) pixel[i] = ((~pixel[i] & 0x00ffffff) | (pixel[i] & 0xff000000));
      }
    }else {

      float invWW = 0.67f / (float)ww;
      int pminWW = ww + contrastMin;
      int pix;
      float hue;

      for(int i=0; i<pixLength; i++){
        pix = orgPixel[i];
        if(pix <= contrastMin) hue = 0.67f;
        else if(pix >= contrastMax) hue = 0.0f;
        else

          // hue = (1.0f - (pix - contrastMin)/ww) * 0.67f;
          hue = (float)(pminWW - pix) * invWW;
        pixel[i] = hue2RGB(hue);

        if(inv) pixel[i] = ((~pixel[i] & 0x00ffffff) | (pixel[i] & 0xff000000));
      }
    }
   }else {
    // Dicom RGB Mode
    System.arraycopy(orgPixel, 0, pixel, 0, pixel.length);
   }
  }


  private int hue2RGB(float hue) {
        int r = 0, g = 0, b = 0;

   float h = (hue - (float)Math.floor(hue)) * 6.0f;
   float f = h - (float)java.lang.Math.floor(h);
   float q = 1.0f - f;

   switch ((int) h) {
          case 0:
     r = 255;
                g = (int) (f * 255.0f + 0.5f);
                b = 0;
                break;
          case 1:
                r = (int) (q * 255.0f + 0.5f);
                g = 255;
```

```
                b = 0;
                break;
            case 2:
                r = 0;
                g = 255;
                b = (int) (f * 255.0f + 0.5f);
                break;
            case 3:
                r = 0;
                g = (int) (q * 255.0f + 0.5f);
                b = 255;
                break;
            case 4:
                r = (int) (f * 255.0f + 0.5f);
                g = 0;
                b = 255;
                break;
            case 5:
                r = 255;
                g = 0;
                b = (int) (q * 255.0f + 0.5f);
                break;
        }
        return 0xff000000 | (r << 16) | (g << 8) | (b << 0);
    }

    private Image getImage() {

        source.newPixels();
        return image;
/*

        Toolkit toolkit = Toolkit.getDefaultToolkit();
        return toolkit.createImage(new MemoryImageSource(width, height,
pixel, 0, width));
*/
    }

     public Image asdf(){
       return img;
     }


    public Image getDefaultImage() {
        contrast();
        return getImage();
    }


    public boolean color() {
        return rgbMode;
    }


    public Image wwANDwl(int argWW, int argWL) {
        ww = argWW;
        wl = argWL;
        contrast();
        return getImage();
    }


    public void setWwWl(int argWW, int argWL) {
        ww = argWW;
        wl = argWL;
    }


    public Image getImageWWWL2Current(int argWW, int argWL) {
        ww = defaultWW + argWW;
        wl = defaultWL + argWL;
        contrast();
        return getImage();
    }


    public int getWW() {
        return ww;
    }
    public int getWL() {
        return wl;
    }


    public int getDefaultWW() {
        return defaultWW;
    }
    public int getDefaultWL() {
        return defaultWL;
    }
```

```
    public int getWidth() {
        return width;
    }
    public int getHeight() {
        return height;
    }


    public int getPixelMin() {
        return pixelMin;
    }
    public int getPixelMax() {
        return pixelMax;
    }

    public int getpixLength(){
       return pixLength;
    }


    public void inverse() {
        inv = !inv;
    }
    public void setInverse(boolean flag) {
        inv = flag;
    }
/*
    public Image inverse() {
        inv = !inv;
        contrast();
        return getImage();
    }
*/


    public void setColor(boolean flag) {
        blackANDwhite = !flag;
    }
    public void changeColor() {
        blackANDwhite = !blackANDwhite;
    }


    public void setDefaultPixel() {
        System.arraycopy(defaultPixel, 0, orgPixel, 0, pixLength);
        width =  defaultWidth;
        height = defaultHeight;

        blackANDwhite = true;                       // □□?□□□□□true
        inv = false;                                // □l□K□|□W□□□]
    }


    public void rotateL() {
        int[] tmpPixel = new int[orgPixel.length];
        int temp;

        System.arraycopy(orgPixel, 0, tmpPixel, 0, tmpPixel.length);
        for(int i=0; i < height; i++) {
            for(int j=0; j < width; j++) {
                orgPixel[(width - j -1) * height + i] = tmpPixel[i * width + j];
            }
        }
        temp = width;
        width = height;
        height = temp;
    }
    public void rotateR() {
        int[] tmpPixel = new int[orgPixel.length];
        int temp;

        System.arraycopy(orgPixel, 0, tmpPixel, 0, tmpPixel.length);
        for(int i=0; i < height; i++) {
            for(int j=0; j < width; j++) {
                orgPixel[j * height + (height - i -1)] = tmpPixel[i * width + j];
            }
        }
        temp = width;
        width = height;
        height = temp;
    }


    public void flipLR() {
        int[] tmpPixel = new int[orgPixel.length];
        int temp1, temp2;

        System.arraycopy(orgPixel, 0, tmpPixel, 0, tmpPixel.length);
```

```
  for(int i=0; i < height; i++) {
    temp1 = i * width;
    temp2 = temp1 + width -1;
    for(int j=0; j < width; j++) {
      orgPixel[temp2 - j] = tmpPixel[temp1 + j];
    }
  }
}


public void flipUD() {
  int[] tmpPixel = new int[orgPixel.length];
  int temp1, temp2;

  System.arraycopy(orgPixel, 0, tmpPixel, 0, tmpPixel.length);
  for(int i=0; i < height; i++) {
    temp1 = (height - i -1) * width;
    temp2 = i * width;
    for(int j=0; j < width; j++) {
      orgPixel[temp1 + j] = tmpPixel[temp2 + j];
    }
  }
}


private void makeHistogram()
{

  for(int i=0; i<256; i++)
     histogram[i] = 0;
  histMax = 0;


  for(int i=0; i<pixel.length; i++){
    int data = (0x000000ff & pixel[i]);
    histogram[data]++;
  }

  for(int i=0; i<256; i++){
    if(histMax < histogram[i])
      histMax = histogram[i];
  }
}

public int[] getHistogram()
{
  return histogram;
}

public int getHistMax()
{
  return histMax;
}

public Image reviseHistogram()
{
  calcRevisedHistogram();
  return getImage();
}

private void calcRevisedHistogram(){
  double[]    aFreq   = new double[256];
  int         sum     = 0;
  double      allPixelsInv;
  double      vmin;
  double      vminInv;
  int[]       fixed   = new int[256];

  makeHistogram();

  for(int i=0; i<pixel.length; i++){
    pixel[i] = (0x000000ff & pixel[i]);
  }


  allPixelsInv = 1. / (height * width);
  for(int i=0; i<256; i++){
    sum += histogram[i];
    aFreq[i] = (double)sum * allPixelsInv;
  }


  vmin = aFreq[0];
  vminInv = (double)255. / (1.- vmin);
  for(int i=0; i<256; i++){
    fixed[i] = (int)((aFreq[i] - vmin) * vminInv);
  }
  for(int i=0; i<pixel.length; i++){
    pixel[i] = ((255 << 24)
                | (fixed[pixel[i]] << 16)
                | (fixed[pixel[i]] << 8)
                | fixed[pixel[i]]);
  }
 }
}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/ImageProcessor.java

```
/*
 * ImageProcessor.java
 *
 * Created on 21 de octubre de 2005, 06:41 PM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package Principal;

//file: ImageProcessor.java
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.awt.image.*;
import javax.swing.*;

/**
 * Esta clase controla el procesamiento de la imagen
 *
 */

public class ImageProcessor extends JComponent {
  private BufferedImage source, destination;
  private JComboBox options;

  public ImageProcessor( BufferedImage image ) {
    source = destination = image;
    setBackground(Color.white);
    setLayout(new BorderLayout());
    // create a panel to hold the combo box
    JPanel controls = new JPanel();
    // create the combo box with the names of the area operators
    options = new JComboBox(
      new String[] { "[source]", "brighten", "darken", "rotate", "scale" }
    );
    // perform some processing when the selection changes
    options.addItemListener(new ItemListener() {
      public void itemStateChanged(ItemEvent ie) {
        // retrieve the selection option from the combo box
        String option = (String)options.getSelectedItem();
        // process the image according to the selected option
        BufferedImageOp op = null;
        if (option.equals("[source]"))
          destination = source;
        else if (option.equals("brighten"))
          op = new RescaleOp(2.0f, 0, null);
        else if (option.equals("darken"))
          op = new RescaleOp(.5f, 0, null);
        else if (option.equals("rotate"))
          op = new AffineTransformOp(
              AffineTransform.getRotateInstance(Math.PI / 6), null);
        else if (option.equals("scale"))
          op = new AffineTransformOp(
              AffineTransform.getScaleInstance(.5, .5), null);
        if (op != null) destination = op.filter(source, null);
        repaint();
      }
    });
    controls.add(options);
    add(controls, BorderLayout.SOUTH);
  }

  public void paintComponent(Graphics g) {
    int imageWidth = destination.getWidth();
    int imageHeight = destination.getHeight();
    int width = getSize().width;
    int height = getSize().height;
    g.drawImage(destination,
        (width - imageWidth) / 2, (height - imageHeight) / 2, null);
  }

  public static void main(String[] args) {
    //String filename = args[0];

    String filename = "C:\\imagenes\\gif_jpeg\\m1.gif";
```

```
        ImageIcon icon = new ImageIcon(filename);
        Image i = icon.getImage();

    // draw the Image into a BufferedImage
    int w = i.getWidth(null), h = i.getHeight(null);
    BufferedImage buffImage = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_RGB);
//    Graphics2D imageGraphics = buffImage.createGraphics();
//    imageGraphics.drawImage(i, 0, 0, null);

    JFrame frame = new JFrame("ImageProcessor");
    frame.getContentPane().add(new ImageProcessor(buffImage));
    frame.setSize(buffImage.getWidth(), buffImage.getHeight());
        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    frame.setVisible(true);
  }
}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/JFrameSobreSistema.java

```
package Principal;
/*
 * JFrameSobreSistema.java
 *
 * Created on 13 de octubre de 2005, 01:47 PM
 */
import javax.swing.*;
/**
 * Esta clase nos permite mostrar información de quien creo este sistema
 * y poderlo controlar como JFrame
 *
 * @author  Armando
 */
public class JFrameSobreSistema extends JFrame {

    /** Creates new form JFrameSobreSistema */
    public JFrameSobreSistema() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        jLabel1 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jTextPane1 = new javax.swing.JTextPane();
        jTextPane2 = new javax.swing.JTextPane();
        jButtonCerrar = new javax.swing.JButton();

        getContentPane().setLayout(new java.awt.GridBagLayout());


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Sobre el Sistema");
        setAlwaysOnTop(true);
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setUndecorated(true);
        jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagen/spacsm_v.jpg")));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth =
java.awt.GridBagConstraints.RELATIVE;
        gridBagConstraints.gridheight =
java.awt.GridBagConstraints.RELATIVE;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.insets = new java.awt.Insets(9, 0, 6, 0);
        getContentPane().add(jLabel1, gridBagConstraints);

        jPanel1.setLayout(new java.awt.GridBagLayout());

        jTextPane1.setBackground(getBackground());
        jTextPane1.setEditable(false);
        jTextPane1.setFont(new java.awt.Font("Arial Black", 0, 10));
        jTextPane1.setText("Universidad Autonoma Metropolitana\n
Azcapotzalco\n");
        jTextPane1.setVerifyInputWhenFocusTarget(false);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 0;
        jPanel1.add(jTextPane1, gridBagConstraints);

        jTextPane2.setBackground(getBackground());
        jTextPane2.setEditable(false);
        jTextPane2.setText("Maestria en Ciencias de la
Computaci\u00f3n\nProyecto:\nSistema PACS m\u00ednimo basado en el
est\u00e1ndar DICOM\nPor:  Armando Jim\u00e9nez Herrera.\nAsesor:
Dr.Carlos Aviles\n2005");
        jTextPane2.setMaximumSize(getPreferredSize());
        jTextPane2.setMinimumSize(getPreferredSize());
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
        jPanel1.add(jTextPane2, gridBagConstraints);

        jButtonCerrar.setText("Cerrar");
        jButtonCerrar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonCerrarActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth =
java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.ipadx = 19;
        gridBagConstraints.ipady = 8;
        jPanel1.add(jButtonCerrar, gridBagConstraints);

        getContentPane().add(jPanel1, new java.awt.GridBagConstraints());

        pack();
    }
    // </editor-fold>

    private void jButtonCerrarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        setVisible(false);
         dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
////////              try {
////////                    UIManager.setLookAndFeel(
////////
UIManager.getCrossPlatformLookAndFeelClassName());
////////                } catch (Exception e) { }
////////
////////              //Make sure we have nice window decorations.
////////              JFrame.setDefaultLookAndFeelDecorated(true);
////////              JDialog.setDefaultLookAndFeelDecorated(true);
                new JFrameSobreSistema().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButtonCerrar;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextPane jTextPane1;
    private javax.swing.JTextPane jTextPane2;
    // End of variables declaration

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/JPanelErrores.java

```
/*
 * JPanelErrores.java
 *
 * Created on 8 de febrero de 2006, 01:53 PM
 */

package Principal;

/**
 * Esta clase genera un JPanel para
 * los posibles errores
```

```
 *
 * @author  Armando
 */
public class JPanelErrores extends javax.swing.JPanel {

    /** Creates new form JPanelErrores */
    public JPanelErrores() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {

        setLayout(new java.awt.BorderLayout());

    }
    // </editor-fold>


    // Variables declaration - do not modify
    // End of variables declaration

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/jtable_3.java

```
package Principal;
/*
 * jtable_3.java
 *
 * Created on 24 de enero de 2006, 05:21 PM
 */

/**
 *
 * @author  Armando Jimenez Herrera
 *
 */
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;

/**
 * Esta clase maneja la información para la tabla y la construye
 *
 */
public class jtable_3 extends javax.swing.JFrame {

    private boolean DEBUG = true;
//Nota hay que revisar lo de la celda de edición  su configuración

    private int TAM_TABLA_COL = 10;
    private int TAM_TABLA_FILA = 1000;


    //Variables no modificar son para JtablevectRowDatavectColumnNames
    private Vector vectRowData;
    private Vector vectColumnNames;



    /**
     * inicializa los vetores y componentes
     */




    public jtable_3() {
        initVectores();
        initComponents();
    }


    public jtable_3(Vector vecto_Datos) {
        initVectoresDicom(vecto_Datos);
        initComponents();
    }

    public void  inicializar(){

        vectColumnNames = new Vector();
        vectRowData = new Vector();
    }

    public void initVectoresDicom(Vector vecto_Datos){
        inicializar();
        for (int i = 0; i <5 ; i++){
                    vectColumnNames.addElement("COLUMNA"+i);
                }
        cargaVectordDatos(vecto_Datos);

    }


    private void initVectores(){
                this.inicializar();

                for (int i = 0; i <TAM_TABLA_COL; i++){
                    vectColumnNames.addElement("COLUMNA"+i);
                }


                 for (int i = 0; i <TAM_TABLA_FILA; i++){
                    Vector temp = new Vector();
                    for (int j = 0; j <TAM_TABLA_COL; j++){
                         temp.addElement( "El_" + j+","+i);
                    }
                    vectRowData.addElement (temp);
                 }
    }



    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        jButton1 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();

        getContentPane().setLayout(new java.awt.GridBagLayout());


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        jButton1.setText("jButton1");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.insets = new java.awt.Insets(10, 2, 10, 2);
        getContentPane().add(jButton1, gridBagConstraints);

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
vectRowData,vectColumnNames) {

            boolean[] canEdit = new boolean [] {
                false, false, true, true, true
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jScrollPane1.setViewportView(jTable1);

        jPanel1.add(jScrollPane1);

        getContentPane().add(jPanel1, new java.awt.GridBagConstraints());

        pack();
    }
    // </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        setVisible(false);
        dispose();
    }

    /**
```

```
     * limpia el contenido del vector de datos
     *
     */
    public void clear_vectRowData (){
        vectRowData.clear();
    }

    /**
     *Inserta un contenido en el vector de datps
     */
    public void inser_Fila (Vector dato){
        vectRowData.addElement(dato);
    }
    /**
     *Inserta un contenido en el vector de nombre de columnas
     */
    public void inser_ColumnNames(Vector dato){
        vectColumnNames.addElement(dato);
    }
    /**
     *Limpia el contenido del nombre de columnas
     */
    public void clear_ColumnNames(){
     vectColumnNames.clear();
    }



    /**
     * Regresa el vetor de datos
     */
    public Vector getvectRowData (){
        return vectRowData;

    }

    /**
     * Regresa el vector de los nombres de las columnas
     */
    public Vector getvectColumnNames(){
        return vectColumnNames;

    }

     public void cargaColumnNames (Vector Datos){
       this.clear_ColumnNames();
       vectColumnNames = (Vector)Datos.clone();

    }



   public void cargaVectordDatos (Vector Datos){
       this.clear_vectRowData();
       vectRowData = (Vector)Datos.clone();

    }



    /**
     *
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new jtable_3().setVisible(true);
            }
        });
    }



    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    // End of variables declaration

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/Principal.java

```
package Principal;
/*
 * Principal.java
 *
 * Created on 20 de octubre de 2005, 10:15 AM
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.image.*;
import java.util.*;

/**
 * La clase Principal controla la presentación a usuario
 * y contiene la estructura general lo que se muestra
 *
 * @author  Armando
 */
public class Principal extends javax.swing.JFrame {



static DicomFile    dicomFile;
static DicomDic     dicomDic;
static Image    imagePrinc;
static Vector vecto_Datos;
    /**
     * Creates new form Principal
     */
    public Principal() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        jFileChooser = new javax.swing.JFileChooser();
        jPanel1 = new javax.swing.JPanel();
        jSplitPane1 = new javax.swing.JSplitPane();
        jPanelAcceso = new javax.swing.JPanel();
        jLabelaccesoClave = new javax.swing.JLabel();
        jButtonaccesoLimpiar = new javax.swing.JButton();
        jButtonComunicacion = new javax.swing.JButton();
        jLabelaccesoUsuario = new javax.swing.JLabel();
        jPasswordFieldaccesoClave = new javax.swing.JPasswordField();
        jTextFieldaccesoUsuario = new javax.swing.JTextField();
        jPanel3 = new javax.swing.JPanel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanelCaptura = new javax.swing.JPanel();
        jPanel6 = new javax.swing.JPanel();
        jTextFieldSeleccionar = new javax.swing.JTextField();
        jLabelSeleccionar = new javax.swing.JLabel();
        jButtonSeleccionar = new javax.swing.JButton();
        jPanel7 = new javax.swing.JPanel();
        jButton3 = new javax.swing.JButton();
        jPanel8 = new javax.swing.JPanel();
        jButton7 = new javax.swing.JButton();
        jPanelMensajes = new javax.swing.JPanel();
        jScrollPaneMensajes = new javax.swing.JScrollPane();
        jTextAreaMensajes = new javax.swing.JTextArea();
        menuBar = new javax.swing.JMenuBar();
        Menu = new javax.swing.JMenu();
        sistemaIniciar = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        sistemaSalir = new javax.swing.JMenuItem();
        Ayuda = new javax.swing.JMenu();
        contentMenuItem = new javax.swing.JMenuItem();
        Sobre = new javax.swing.JMenuItem();

        jFileChooser.setMaximumSize(new java.awt.Dimension(800, 600));
        jFileChooser.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jFileChooserActionPerformed(evt);
            }
        });

        getContentPane().setLayout(new javax.swing.BoxLayout(getContentPane(),
javax.swing.BoxLayout.X_AXIS));


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Sistema general");
        jPanel1.setLayout(new java.awt.BorderLayout(2, 0));
```

```
        jPanel1.setMaximumSize(new java.awt.Dimension(800, 600));
        jSplitPane1.setMaximumSize(new java.awt.Dimension(1024, 768));
        jSplitPane1.setOneTouchExpandable(true);
        jPanelAcceso.setLayout(new java.awt.GridBagLayout());

        jPanelAcceso.setBorder(new javax.swing.border.TitledBorder(new
javax.swing.border.TitledBorder(null, "Acceso",
javax.swing.border.TitledBorder.CENTER,
javax.swing.border.TitledBorder.DEFAULT_POSITION)));
        jPanelAcceso.setMaximumSize(jPanelAcceso.getPreferredSize());
        jPanelAcceso.setPreferredSize(new java.awt.Dimension(25, 115));
        jLabelaccesoClave.setText("Clave");
        jLabelaccesoClave.setAlignmentX(10.0F);
        jLabelaccesoClave.setAlignmentY(10.0F);

jLabelaccesoClave.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.ipadx = 80;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanelAcceso.add(jLabelaccesoClave, gridBagConstraints);

        jButtonaccesoLimpiar.setText("Limpiar");
        jButtonaccesoLimpiar.setAlignmentX(10.0F);
        jButtonaccesoLimpiar.setAlignmentY(10.0F);

jButtonaccesoLimpiar.setHorizontalTextPosition(javax.swing.SwingConstants.C
ENTER);
        jButtonaccesoLimpiar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonaccesoLimpiarActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 5;
        gridBagConstraints.insets = new java.awt.Insets(14, 41, 14, 41);
        jPanelAcceso.add(jButtonaccesoLimpiar, gridBagConstraints);

        jButtonComunicacion.setText("Comunicacion");
        jButtonComunicacion.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonComunicacionActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanelAcceso.add(jButtonComunicacion, gridBagConstraints);

        jLabelaccesoUsuario.setText("Usuario");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.ipadx = 80;
        jPanelAcceso.add(jLabelaccesoUsuario, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.gridwidth =
java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.ipadx = 90;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanelAcceso.add(jPasswordFieldaccesoClave, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.ipadx = 80;
        gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 3);
        jPanelAcceso.add(jTextFieldaccesoUsuario, gridBagConstraints);

        jSplitPane1.setLeftComponent(jPanelAcceso);

        jPanel3.setLayout(new java.awt.BorderLayout());

        jPanelCaptura.setLayout(new java.awt.GridBagLayout());

        jPanel6.setLayout(new java.awt.GridBagLayout());

        jPanel6.setBorder(new javax.swing.border.TitledBorder("I Seleccionar
Archivo a procesar"));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.ipadx = 300;
        gridBagConstraints.insets = new java.awt.Insets(10, 10, 10, 0);
        jPanel6.add(jTextFieldSeleccionar, gridBagConstraints);

        jLabelSeleccionar.setText("Archivo");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.ipadx = 50;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanel6.add(jLabelSeleccionar, gridBagConstraints);

        jButtonSeleccionar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagen/Open16.gif")));
        jButtonSeleccionar.setText("Seleccionar");
        jButtonSeleccionar.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonSeleccionarActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        gridBagConstraints.insets = new java.awt.Insets(10, 50, 10, 0);
        jPanel6.add(jButtonSeleccionar, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanelCaptura.add(jPanel6, gridBagConstraints);

        jPanel7.setLayout(new java.awt.GridBagLayout());

        jPanel7.setBorder(new javax.swing.border.TitledBorder("II Procesar
archivo y Mostrar informacion"));
        jButton3.setText("Procesar");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanel7.add(jButton3, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanelCaptura.add(jPanel7, gridBagConstraints);

        jPanel8.setLayout(new java.awt.GridBagLayout());

        jPanel8.setBorder(new javax.swing.border.TitledBorder("III Guardar en el
Sistema"));
        jButton7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagen/Save16.gif")));
        jButton7.setText("Guardar");
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanel8.add(jButton7, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 10, 0);
        jPanelCaptura.add(jPanel8, gridBagConstraints);

        jPanelMensajes.setLayout(new java.awt.BorderLayout());
```

```
jScrollPaneMensajes.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstant
s.VERTICAL_SCROLLBAR_ALWAYS);
        jTextAreaMensajes.setEditable(false);
        jTextAreaMensajes.setRows(3);
        jScrollPaneMensajes.setViewportView(jTextAreaMensajes);

        jPanelMensajes.add(jScrollPaneMensajes,
java.awt.BorderLayout.CENTER);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        jPanelCaptura.add(jPanelMensajes, gridBagConstraints);

        jTabbedPane1.addTab("Captura", jPanelCaptura);

        jPanel3.add(jTabbedPane1, java.awt.BorderLayout.CENTER);

        jSplitPane1.setRightComponent(jPanel3);

        jPanel1.add(jSplitPane1, java.awt.BorderLayout.CENTER);

        getContentPane().add(jPanel1);

        Menu.setText("Sistema");
        sistemaIniciar.setText("Iniciar");
        sistemaIniciar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                sistemaIniciarActionPerformed(evt);
            }
        });

        Menu.add(sistemaIniciar);

        Menu.add(jSeparator1);

        sistemaSalir.setText("Salir");
        sistemaSalir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                sistemaSalirActionPerformed(evt);
            }
        });

        Menu.add(sistemaSalir);

        menuBar.add(Menu);

        Ayuda.setText("Ayuda");
        contentMenuItem.setText("Contents");
        Ayuda.add(contentMenuItem);

        Sobre.setText("Sobre el Sistema");
        Sobre.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                SobreActionPerformed(evt);
            }
        });

        Ayuda.add(Sobre);

        menuBar.add(Ayuda);

        setJMenuBar(menuBar);

        pack();
    }
    // </editor-fold>

    private void jFileChooserActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {


        // variables para comunicacion con mysql
        Comun_Mysql    comun_mysql= new Comun_Mysql();
        // crea variable de vector e inicializa.
        Vectores prueba = new Vectores(vecto_Datos);
        // crea vector de paciente
        prueba.creaDatosPaciente();
        // Si no existe: graba vector dato de paciente con mensaje
        //       exite:  Manda mensaje de que existe registro
        if ( 0 ==
comun_mysql.isContainsPaciente(jTextFieldaccesoUsuario.getText(),
                jPasswordFieldaccesoClave.getText(),
                prueba.getNumeroIdentificacionPaciente() ) ){
            jTextAreaMensajes.append( "Se grabo
Paciente"+comun_mysql.insertPaciente
                ( jTextFieldaccesoUsuario.getText(),
                  jPasswordFieldaccesoClave.getText(),
                  prueba.getPaciente() ) + "registro\n");

        }
        else{
            jTextAreaMensajes.append ( "Error: Ya se tiene grabado la
informacion de paciente.\n " );
        }
        //
        // Estudio
        //
        prueba.creaDatosEstudio();
        System.out.println ("\n\n\nesta es el vector de Estudio"+
prueba.getEstudio());

        // Si no existe: graba vector dato de paciente con mensaje
        //       exite:  Manda mensaje de que existe registro
        if ( 0 ==
comun_mysql.isContainsEstudio(jTextFieldaccesoUsuario.getText(),
                jPasswordFieldaccesoClave.getText(),
                prueba.getIdentificadorInstanciaEstudio() ) ){
            jTextAreaMensajes.append( "Se grabo
Estudio"+comun_mysql.insertEstudio
                ( jTextFieldaccesoUsuario.getText(),
                  jPasswordFieldaccesoClave.getText(),
                  prueba.getEstudio() ) + "registro\n");
        }
        else{
            jTextAreaMensajes.append ( "Error: Ya se tiene grabado la
informacion de Estudio.\n " );
        }

        //
        //
        // Equipo
        //
        prueba.creaDatosEquipo();
        System.out.println ("\n\n\nesta es el vector de Equipo <"
                + prueba.getEquipo() + ">");

        // Si no existe: graba vector dato de paciente con mensaje
        //       exite:  Manda mensaje de que existe registro
        if ( 0 ==
                comun_mysql.isContainsEquipo(jTextFieldaccesoUsuario.getText(),
                 jPasswordFieldaccesoClave.getText(),
                 prueba.getIdEquipo() ) ){
            jTextAreaMensajes.append( "Se grabo Equipo" +
                    comun_mysql.insertEquipo
                     ( jTextFieldaccesoUsuario.getText(),
                       jPasswordFieldaccesoClave.getText(),
                       prueba.getEquipo() ) + "registro\n");
        }
        else{
            jTextAreaMensajes.append ( "Error: Ya se tiene grabado la
informacion de Equipo.\n " );
        }


        //
        //
        // SERIE
        //
        prueba.creaDatosSerie();
        System.out.println ("\n\n\nesta es el vector de Serie <"
                + prueba.getSerie() + ">");

        // Si no existe: graba vector dato de paciente con mensaje
        //       exite:  Manda mensaje de que existe registro
        if ( 0 ==
                comun_mysql.isContainsSerie(jTextFieldaccesoUsuario.getText(),
                 jPasswordFieldaccesoClave.getText(),
                 prueba.getIdSerie() ) ){
            jTextAreaMensajes.append( "Se grabo Serie" +
                    comun_mysql.insertSerie
                     ( jTextFieldaccesoUsuario.getText(),
                       jPasswordFieldaccesoClave.getText(),
                       prueba.getSerie() ) + "registro\n");
        }
        else{
            jTextAreaMensajes.append ( "Error: Ya se tiene grabado la
informacion de Serie.\n " );
        }



        //
        // IMAGEN
```

```
        //
        prueba.creaDatosImagen();
        System.out.println ("\n\n\nesta es el vector de Imagen <"
                + prueba.getImagen() + ">");

        // Si no existe: graba vector dato de paciente con mensaje
        //       exite:  Manda mensaje de que existe registro
        if ( 0 ==
                comun_mysql.isContainsImagen(jTextFieldaccesoUsuario.getText(),
                 jPasswordFieldaccesoClave.getText(),
                 prueba.getIdImagen() ) ){
            jTextAreaMensajes.append( "Se grabo Imagen" +
                    comun_mysql.insertImagen
                     ( jTextFieldaccesoUsuario.getText(),
                       jPasswordFieldaccesoClave.getText(),
                       prueba.getImagen(), imagePrinc) + "registro\n");
        }
        else{
            jTextAreaMensajes.append ( "Error: Ya se tiene grabado la
informacion de Imagen.\n " );
        }



    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        capturaExped ();


    }

    private void jButtonaccesoLimpiarActionPerformed(java.awt.event.ActionEvent
evt) {
// TODO add your handling code here:
        limpiar ();
    }



    private void sistemaIniciarActionPerformed(java.awt.event.ActionEvent evt) {
// abre la ventana para poner usuario y clave
        jSplitPane1.resetToPreferredSizes();
    }

    private void jButtonSeleccionarActionPerformed(java.awt.event.ActionEvent
evt) {
// TODO add your handling code here:
            int returnVal = jFileChooser.showOpenDialog(this);
           if (returnVal == JFileChooser.APPROVE_OPTION)
               jTextFieldSeleccionar.setText(
jFileChooser.getSelectedFile().toString() );



            else {
               jTextFieldSeleccionar.setText( null );
               System.out.printf ("Open command cancelled by user.");
            }

    }

    private void jButtonComunicacionActionPerformed(java.awt.event.ActionEvent
evt) {
// TODO add your handling code here:

        Comun_Mysql comun_Mysql = new Comun_Mysql();
        boolean a = comun_Mysql.Test( jTextFieldaccesoUsuario.getText(),
jPasswordFieldaccesoClave.getText());

        if (a == true) {
            jTextAreaMensajes.append("La prueba de comunicacion con
EXITO.\n");
        }
        else{
            jTextAreaMensajes.append("ERROR: En la prueba de
comunicacion.\n");
        }

    }

    private void SobreActionPerformed(java.awt.event.ActionEvent evt) {
        JFrame frame = new JFrameSobreSistema();
        frame.setLocationRelativeTo (null);
        frame.setVisible (true);

    }

    private void sistemaSalirActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }



    public void limpiar (){
        jTextFieldaccesoUsuario.setText( null );
        jPasswordFieldaccesoClave.setText( null );
        jTextFieldSeleccionar.setText( null );
    }

    /**
     *  Carga la informacion del archivo
     *   carga diccionarrio
     *   carga dato
     *
     */
    public void capturaExped (){


        dicomFile   = new DicomFile();
        dicomDic    = new DicomDic ();
        dicomDic.loadDicomDic (jTextFieldaccesoUsuario.getText(),
jPasswordFieldaccesoClave.getText());

        dicomFile = new DicomFile ( dicomDic ) ;

       /* DicomData Temp1 = new DicomData();
        Temp1 =*/
        dicomFile.load (jTextFieldSeleccionar.getText());

        ImageData tmpimageData = new ImageData();
        tmpimageData.setData( dicomFile.getDicomData() );

        imagePrinc = tmpimageData.getDefaultImage ();
        //muestraimagen ();



        Image i = imagePrinc;

    // draw the Image into a BufferedImage
    int w = i.getWidth(null), h = i.getHeight(null);
    BufferedImage buffImage = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);

    Graphics2D imageGraphics = buffImage.createGraphics();
    imageGraphics.drawImage(i, 0, 0, null);

//    JFrame frame = new JFrame("ImageProcessor");
//    frame.getContentPane().add(new ImageProcessor(buffImage));
//    frame.setSize(buffImage.getWidth(), buffImage.getHeight());
//    frame.setDefaultCloseOperation( JFrame.DISPOSE_ON_CLOSE );



    JFrame jFrame11 = new JFrame("ImageProcessor");
    jFrame11.getContentPane().add(new ImageProcessor(buffImage));
    jFrame11.setSize(buffImage.getWidth(), buffImage.getHeight());
    jFrame11.setDefaultCloseOperation( JFrame.DISPOSE_ON_CLOSE );
    jFrame11.setVisible(true);

    vecto_Datos = dicomFile.getVectorDicomData();
    JFrame jFrame22 = new jtable_3(vecto_Datos);
    jFrame22.setDefaultCloseOperation( JFrame.DISPOSE_ON_CLOSE );
    jFrame22.setVisible(true);
     }



//////////     public void muestraimagen(){
//////////         JFrame frame = new Visualizador("Imagen", image);
//////////
frame.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_O
N_CLOSE);
//////////         frame.setLocationRelativeTo (null);
//////////         frame.setVisible (true);
//////////     }

    private static void createAndShowGUI() {
        //Use the Java look and feel.
        try {
            UIManager.setLookAndFeel(
                UIManager.getCrossPlatformLookAndFeelClassName());
        } catch (Exception e) { }
```

```
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        Principal frame = new Principal();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Display the window.
        frame.setMaximumSize(new java.awt.Dimension(800,800));
        frame.setPreferredSize(new java.awt.Dimension(724, 768));
        frame.setMinimumSize (new java.awt.Dimension(500, 500));

        frame.setSize(frame.getPreferredSize());
        frame.setLocationRelativeTo(null); //center it
        //frame.setMaximumSize(new java.awt.Dimension(724, 768));
///rsetMaximumSize(new java.awt.Dimension(1024, 768))
        frame.pack();
        frame.setVisible(true);
    }



    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {

        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });

    }

    // Variables declaration - do not modify
    private javax.swing.JMenu Ayuda;
    private javax.swing.JMenu Menu;
    private javax.swing.JMenuItem Sobre;
    private javax.swing.JMenuItem contentMenuItem;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButtonComunicacion;
    private javax.swing.JButton jButtonSeleccionar;
    private javax.swing.JButton jButtonaccesoLimpiar;
    private javax.swing.JFileChooser jFileChooser;
    private javax.swing.JLabel jLabelSeleccionar;
    private javax.swing.JLabel jLabelaccesoClave;
    private javax.swing.JLabel jLabelaccesoUsuario;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanelAcceso;
    private javax.swing.JPanel jPanelCaptura;
    private javax.swing.JPanel jPanelMensajes;
    private javax.swing.JPasswordField jPasswordFieldaccesoClave;
    private javax.swing.JScrollPane jScrollPaneMensajes;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSplitPane jSplitPane1;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTextArea jTextAreaMensajes;
    private javax.swing.JTextField jTextFieldSeleccionar;
    private javax.swing.JTextField jTextFieldaccesoUsuario;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JMenuItem sistemaIniciar;
    private javax.swing.JMenuItem sistemaSalir;
    // End of variables declaration

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/Vectores.java

```
/*
 * Vectores.java
 *
 * Created on 27 de enero de 2006, 02:24 PM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package Principal;

import java.util.*;


/**
 * Esta clase es de manejo de vectores
 * se considera que el vector es de la forma vector[][]
 * Nota: se asume que el valor del tag es unico
 * @author armando
 */
public class Vectores {
    private Vector  vectorDatos;
    private Vector  VectorDatosPaciente;
    private Vector  VectorDatosEstudio;
    private Vector  VectorDatosEquipo;
    private Vector  VectorDatosSerie;
    private Vector  VectorDatosImagen;
    private boolean DEBUG = true;
    /**
     * Tag de paciente
     */
    private String  TAGPAC10= "x(0010,0010)x";
    private String  TAGPAC20= "x(0010,0020)x";
    private String  TAGPAC30= "x(0010,0030)x";
    private String  TAGPAC40= "x(0010,0040)x";
    private String  TAGPAC16= "x(0008,0016)x";
    private String  TAGPAC18= "x(0008,0018)x";

    /**
     * Tags de Estudio
     */
    private String  TAGEST0D    = "x(0020,000d)x";
    private String  TAGPAC_EST20= "x(0010,0020)x";
    private String  TAGEST20    = "x(0008,0020)x";
    private String  TAGEST30    = "x(0008,0030)x";
    private String  TAGEST90    = "x(0008,0090)x";
    private String  TAGEST10    = "x(0020,0010)x";
    private String  TAGEST50    = "x(0008,0050)x";
    private String  TAGEST1150  = "x(0008,1150)x";
    private String  TAGEST1155  = "x(0008,1155)x";

    /**
     * Tags de Equipo
     */
    private String  TAGEQU0070  = "x(0008,0070)x";

    /**
     * Tags de Serie
     **/
     private String  TAGSERIE000E      = "x(0020,000e)x";
     private String  TAGSERIE08_0060   = "x(0008,0060)x";
     private String  TAGSERIE0011      = "x(0020,0011)x";
     private String  TAGSERIE0060      = "x(0020,0060)x";
     private String  TAGSERIE1150      = "x(0008,1150)x";
     private String  TAGSERIE1155      = "x(0008,1155)x";
     private String  TAGSERIE5100      = "x(0018,5100)x";
     private String  TAGSERIE1001      = "x(0040,1001)x";
     private String  TAGSERIE0009      = "x(0040,0009)x";
     private String  TAGSERIE1111      = "x(0008,1111)x";
     private String  TAGSERIE0068      = "x(0008,0068)x";
     private String  TAGSERIE0052      = "x(0020,0052)x";
     private String  TAGSERIE1040      = "x(0020,1040)x";


     /**
      * Tag Imagen
      */

       private String  TAGIMG00200013  = "x(0020,0013)x";
       private String  TAGIMG00200020  = "x(0020,0020)x";
       private String  TAGIMG00080023  = "x(0008,0023)x";
       private String  TAGIMG00080033  = "x(0008,0033)x";
       private String  TAGIMG00081150  = "x(0008,1150)x";
       private String  TAGIMG00081155  = "x(0008,1155)x";
       private String  TAGIMG0040A170  = "x(0040,a170)x";
       private String  TAGIMG00280002  = "x(0028,0002)x";
       private String  TAGIMG00280004  = "x(0028,0004)x";
       private String  TAGIMG00280010  = "x(0028,0010)x";
       private String  TAGIMG00280011  = "x(0028,0011)x";
       private String  TAGIMG00280100  = "x(0028,0100)x";
       private String  TAGIMG00280101  = "x(0028,0101)x";
       private String  TAGIMG00280102  = "x(0028,0102)x";
       private String  TAGIMG00280103  = "x(0028,0103)x";
       private String  TAGIMG7FE00010  = "x(7fe0,0010)x";
       private String  TAGIMG00280006  = "x(0028,0006)x";
       private String  TAGIMG00280034  = "x(0028,0034)x";
       private String  TAGIMG00281101  = "x(0028,1101)x";
       private String  TAGIMG00281102  = "x(0028,1102)x";
       private String  TAGIMG00281103  = "x(0028,1103)x";
       private String  TAGIMG00281201  = "x(0028,1201)x";
```

```
    private String  TAGIMG00281202  = "x(0028,1202)x";
    private String  TAGIMG00281203  = "x(0028,1203)x";
    private String  TAGIMG00200062  = "x(0020,0062)x";
    private String  TAGIMG00082218  = "x(0008,2218)x";
    private String  TAGIMG00080008  = "x(0008,0008)x";
    private String  TAGIMG00281040  = "x(0028,1040)x";
    private String  TAGIMG00281041  = "x(0028,1041)x";
    private String  TAGIMG00281052  = "x(0028,1052)x";
    private String  TAGIMG00281053  = "x(0028,1053)x";
    private String  TAGIMG00281054  = "x(0028,1054)x";
    private String  TAGIMG20500020  = "x(2050,0020)x";
    private String  TAGIMG00282110  = "x(0028,2110)x";
    private String  TAGIMG00282112  = "x(0028,2112)x";
    private String  TAGIMG00280301  = "x(0028,0301)x";
    private String  TAGIMG00283010  = "x(0028,3010)x";
    private String  TAGIMG00283002  = "x(0028,3002)x";
    private String  TAGIMG00283006  = "x(0028,3006)x";
    private String  TAGIMG00281050  = "x(0028,1050)x";
    private String  TAGIMG00281051  = "x(0028,1051)x";
    private String  TAGIMG00187004  = "x(0018,7004)x";
    private String  TAGIMG00187030  = "x(0018,7030)x";
    private String  TAGIMG00187032  = "x(0018,7032)x";
    private String  TAGIMG00187034  = "x(0018,7034)x";
    private String  TAGIMG00181164  = "x(0018,1164)x";
    private String  TAGIMG00181508  = "x(0018,1508)x";
    private String  TAGIMG00400318  = "x(0040,0318)x";
    private String  TAGIMG00540220  = "x(0054,0220)x";
    private String  TAGIMG00540222  = "x(0054,0222)x";
    private String  TAGIMG60xx0010  = "x(60xx,0010)x";
    private String  TAGIMG60xx0011  = "x(60xx,0011)x";
    private String  TAGIMG60xx0040  = "x(60xx,0040)x";
    private String  TAGIMG60xx0050  = "x(60xx,0050)x";
    private String  TAGIMG60xx0100  = "x(60xx,0100)x";
    private String  TAGIMG60xx0102  = "x(60xx,0102)x";
    private String  TAGIMG60xx3000  = "x(60xx,3000)x";
    private String  TAGIMG00400555  = "x(0040,0555)x";
    private String  TAGIMG0040A043  = "x(0040,a043)x";
    private String  TAGIMG0040A136  = "x(0040,a136)x";
    private String  TAGIMG0040A30A  = "x(0040,a30a)x";
    private String  TAGIMG004008EA  = "x(0040,08ea)x";
    private String  TAGIMG0040A121  = "x(0040,a121)x";
    private String  TAGIMG0040A122  = "x(0040,a122)x";
    private String  TAGIMG0040A123  = "x(0040,a123)x";
    private String  TAGIMG0040A160  = "x(0040,a160)x";
    private String  TAGIMG0040A168  = "x(0040,a168)x";
    private String  TAGIMG00080016  = "x(0008,0016)x";
    private String  TAGIMG00080018  = "x(0008,0018)x";
    private String  TAGIMG00080005  = "x(0008,0005)x";
    private String  TAGIMG00080102  = "x(0008,0102)x";
    private String  TAGIMG00080112  = "x(0008,0112)x";
    private String  TAGIMG0008010C  = "x(0008,010c)x";
    private String  TAGIMG00080114  = "x(0008,0114)x";
    private String  TAGIMG04000500  = "x(0400,0500)x";
    private String  TAGIMG04000510  = "x(0400,0510)x";
    private String  TAGIMG04000520  = "x(0400,0520)x";


   /**
    *
    */



/**
 * Inicializa el V
 * ector de datos a procesar
 */
public Vectores(Vector vectordato) {
    vectorDatos = (Vector)vectordato.clone();
    VectorDatosPaciente = new Vector();
    VectorDatosEstudio = new Vector();
    VectorDatosEquipo = new Vector();
    VectorDatosSerie = new Vector();
    VectorDatosImagen = new Vector();
  }


/**
 * Revisa si esta contenido en el vector el tag
 */
public boolean isContains(String Tag){
    boolean result = false;
    Vector vectorFila = new Vector();

     for (Enumeration e = vectorDatos.elements() ; e.hasMoreElements() ;) {
        vectorFila = (Vector)e.nextElement();
        if (DEBUG) System.out.println(vectorFila);
        if ( vectorFila.contains(Tag) ) {
            if (DEBUG) {
                    System.out.println("El Tag esta contenido en el vector");
                    System.out.println(vectorFila);
            }
            result = true;
            break;

        }
      }
    return result;
}

/**
 * Busca la fila del tag solicitado y entrega el vector de los datos de la fila
 */

public Vector getfilatag(String Tag){
    Vector vectorFila = new Vector();
    Vector result  = new Vector();
     for (Enumeration e = vectorDatos.elements() ; e.hasMoreElements() ;) {
        vectorFila = (Vector)e.nextElement();
        if (DEBUG) System.out.println(vectorFila);
        if ( vectorFila.contains(Tag) ) {
            if (DEBUG) {
                    System.out.println("El Tag esta contenido en el vector");
                    System.out.println(vectorFila);
            }
            result = (Vector)vectorFila.clone();
            break;
        }
      }
    return result;
}

/**
 * Regresa el valor de
 * Si encontro:  Numero de identificacion del paciente
 * No Encontro:  NULL
 */
public String getNumeroIdentificacionPaciente(){

    String tag= TAGPAC20;


    if (DEBUG) System.out.println("getNumeroIdentificacionPaciente es:>"
+
            (String)VectorDatosPaciente.get(0) + "<");

    return (String)VectorDatosPaciente.get(0);

}

/**
 * Busca la informcion del paciente
 * en el vector y entregalo en un vector
 *
 */
public String getPaciente(){
    String result= null;
    String Tempo = null;
    boolean  flag = true;

     for (Enumeration e = VectorDatosPaciente.elements() ;
e.hasMoreElements() ;) {
        Tempo = """+(String)e.nextElement() + """;
        if (flag) {
            result = Tempo;
            flag = false;
        }
        else result = result.concat(Tempo);
        if( e.hasMoreElements()){
            result = result.concat(",");
        }
     }

    System.out.println ("este es el valor de getpaciente>" + result + "<");
    return result;
}

/**
 * Regresa solo los valores para el
 * vector de paciente y quedan registrados
 *en el vator de datos del paciente
 *
 */
public void creaDatosPaciente(){

    int i=1;
    String    tag="";

   for (int j=0; j<6;j++){
      Vector    filadatos = new Vector();
```

```
        switch (j){
            case 0 : tag = TAGPAC20;
                     break;
            case 1 : tag = TAGPAC10;
                     break;
            case 2 : tag = TAGPAC30;
                     break;
            case 3 : tag = TAGPAC40;
                     break;
            case 4 : tag = TAGPAC16;
                     break;
            case 5 : tag = TAGPAC18;
                     break;
            default: tag = "";

        }

        // Esta contenido si: sacar datos y agregarlo al vector
        //              no: continuar con el siguiete
        if (DEBUG) System.out.println("El valor de TAG es: >" + tag +
                                      "<");
        if ( isContains( tag ) ){
            if (DEBUG) System.out.println("Si lo encontro a " + tag );

            VectorDatosPaciente.addElement(getfilatag(tag).get(4));
        }

        else
            if (DEBUG) System.out.println("No encontro al tag >" + tag +
                                      "<");
     }



    }

    public Vector getDatosPaciente(){
        return VectorDatosPaciente;
    }



    /**
     * Crea vector de los datos de Estudio en base a vector general
     *
     */

     public void creaDatosEstudio (){

                int i=1;
       String    tag="";

       for (int j=0; j<9;j++){
         Vector    filadatos = new Vector();

         switch (j){
            case 0 : tag = TAGPAC20;
                     break;
            case 1 : tag = TAGEST0D;
                     break;
            case 2 : tag = TAGPAC_EST20;
                     break;
            case 3 : tag = TAGEST20;
                     break;
            case 4 : tag = TAGEST30;
                     break;
            case 5 : tag = TAGEST90;
                     break;
            case 6 : tag = TAGEST10;
                     break;
            case 7 : tag = TAGEST50;
                     break;
            case 8 : tag = TAGEST1150;
                     break;
            case 9 : tag = TAGEST1155;
                     break;
            default: tag = "";

         }

         // Esta contenido si: sacar datos y agregarlo al vector
         //              no: continuar con el siguiete
         if (DEBUG) System.out.println("-Estudio-El valor de TAG es: >" + tag
+
                                       "<");
         if ( isContains( tag ) ){
            if (DEBUG) System.out.println("-Estudio-Si lo encontro a " + tag );

            VectorDatosEstudio.addElement(getfilatag(tag).get(4));
         }

         else{
            VectorDatosEstudio.addElement("");
            if (DEBUG) System.out.println("-Estudio-No encontro al tag >"
                  + tag + "<");
         }
       }
       }

      /**
       * Regresa el vector de datos del Estudio
       *
       */
       public Vector getDatosEstudio(){
          return VectorDatosEstudio;
       }

       /**
        * Busca la informacion del vector de estudio y
        *entrega un string de toda la informacion
        */
       public String getEstudio(){
          String result= null;
          String Tempo = null;
          boolean  flag = true;

          for (Enumeration e = VectorDatosEstudio.elements() ;
e.hasMoreElements() ;) {
             Tempo = """+(String)e.nextElement() + """;
             if (flag) {
                result = Tempo;
                flag = false;
             }
             else result = result.concat(Tempo);
             if( e.hasMoreElements()){
                result = result.concat(",");
             }
          }

          System.out.println ("este es el valor de getpaciente>" + result + "<");
          return result;



       }


          public String getIdentificadorInstanciaEstudio(){

             if (DEBUG) System.out.println("getNumeroIdentificacionPaciente
es:>" +
                   (String)VectorDatosEstudio.get(0) + "<");

             return (String)VectorDatosEstudio.get(0);
          }

       //
       // EQUIPO
       //

        /**
        * Crea el Vector de los datos del equipo en vase a vector general
        */
        public void creaDatosEquipo() {
           int i=1;
           String    tag="";

           tag = TAGEQU0070;

           // Esta contenido si: sacar datos y agregarlo al vector
           //              no: continuar con el siguiete
           if (DEBUG) System.out.println("-Equipo-El valor de TAG es: >" + tag +
                                         "<");
           if ( isContains( tag ) ){
              if (DEBUG) System.out.println("-Equipo-Si lo encontro a " + tag );
              VectorDatosEquipo.addElement(getfilatag(tag).get(4));
           }

           else{
              VectorDatosEstudio.addElement("");
              if (DEBUG) System.out.println("-Equipo-No encontro al tag >"
                    + tag + "<");
           }
        }

        /**
         * Regresa el vector de datos del equipo
         */
```

```
    public Vector getDatosEquipo(){
        return VectorDatosEquipo;

    }

    /**
     * Busca la informacion del vector de equipo y entrega el string de toda
la informacion
     *
     */
     public String getEquipo(){

        String result= null;
        String Tempo = null;
        boolean  flag = true;

         for (Enumeration e = VectorDatosEquipo.elements() ;
e.hasMoreElements() ;) {
            Tempo = """+(String)e.nextElement() + """;
            if (flag) {
                result = Tempo;
                flag = false;
            }
            else result = result.concat(Tempo);
            if( e.hasMoreElements()){
                result = result.concat(",");
            }
         }

        System.out.println ("este es el valor de getEquipo>" + result + "<");
        return result;
     }

    /**
     *  Entrega el  el fabricante del equipo
     */

    public String getIdEquipo(){
            if (DEBUG) System.out.println("getIdEquipo es:>" +
                (String)VectorDatosEquipo.get(0) + "<");

            return (String)VectorDatosEquipo.get(0);

    }


  //
  // SERIE
  //

    /**
    * Crea el Vector de los datos del equipo en vase a vector general
    */
    public void creaDatosSerie() {

      int i=1;
      String    tag="";

      for (int j=0; j<16;j++){
          Vector    filadatos = new Vector();

          switch (j){
              case 0 : tag =TAGPAC20;
                       break;
              case 1 : tag =TAGEST0D;
                       break;
              case 2 : tag =TAGEQU0070;
                       break;
              case 3 : tag =TAGSERIE000E;
                       break;
              case 4 : tag =TAGSERIE08_0060;
                       break;
              case 5 : tag =TAGSERIE0011;
                       break;
              case 6 : tag =TAGSERIE0060;
                       break;
              case 7 : tag =TAGSERIE1150;
                       break;
              case 8 : tag =TAGSERIE1155;
                       break;
              case 9 : tag =TAGSERIE5100;
                       break;
              case 10 : tag =TAGSERIE1001;
                       break;
              case 11 : tag =TAGSERIE0009;
                       break;
              case 12 : tag =TAGSERIE1111;
                       break;
              case 13 : tag =TAGSERIE0068;
                       break;
              case 14 : tag =TAGSERIE0052;
                       break;
              case 15 : tag =TAGSERIE1040;
                       break;
              default: tag = "";

           }

           // Esta contenido si: sacar datos y agregarlo al vector
           //                no: continuar con el siguiete
          if (DEBUG) System.out.println("El valor de TAG es: >" + tag +
                                        "<");
          if ( isContains( tag ) ){
              if (DEBUG) System.out.println("Si lo encontro a " + tag );

              VectorDatosSerie.addElement(getfilatag(tag).get(4));
          }

          else{
              if (DEBUG) System.out.println("No encontro al tag >" + tag +
                                            "<");
              VectorDatosSerie.addElement("");
          }
       }


    }

    /**
     * Regresa el vector de datos del equipo
     */
    public Vector getDatosSerie(){
        return VectorDatosSerie;

    }

    /**
     * Busca la informacion del vector de equipo y entrega el string de toda
la informacion
     *
     */
     public String getSerie(){

        String result= null;
        String Tempo = null;
        boolean  flag = true;

         for (Enumeration e = VectorDatosSerie.elements() ; e.hasMoreElements()
;) {
            Tempo = """+(String)e.nextElement() + """;
            if (flag) {
                result = Tempo;
                flag = false;
            }
            else result = result.concat(Tempo);
            if( e.hasMoreElements()){
                result = result.concat(",");
            }
         }

        System.out.println ("este es el valor de getSerie>" + result + "<");
        return result;
     }

    /**
     *  Entrega el  el fabricante del equipo
     */

    public String getIdSerie(){
        int COLUMNA = 3;
            if (DEBUG) System.out.println("getIdSerie es:>" +
                (String)VectorDatosSerie.get(COLUMNA) + "<");

            return (String)VectorDatosSerie.get(COLUMNA);

    }




  //
  // IMAGEN
  //

    /**
    * Crea el Vector de los datos del equipo en vase a vector general
    */
    public void creaDatosImagen () {
```

```
int i=1;
String    tag="";

for (int j=0; j<81;j++){
    Vector    filadatos = new Vector();

    switch (j){

          case 0 : tag =TAGPAC20;
                break;
        case 1 : tag =TAGEST0D;
                break;
        case 2 : tag =TAGEQU0070;
                break;
        case 3 : tag =TAGSERIE000E;
                break;
        case 4 : tag = TAGIMG00200013;
                break;
        case 5 : tag =TAGIMG00200020;
                break;
        case 6 : tag =TAGIMG00080023;
                break;
        case 7 : tag =TAGIMG00080033;
                break;
        case 8 : tag =TAGIMG00081150;
                break;
        case 9 : tag =TAGIMG00081155;
                break;
        case 10 : tag =TAGIMG0040A170;
                break;
        case 11 : tag =TAGIMG00280002;
                break;
        case 12 : tag =TAGIMG00280004;
                break;
        case 13 : tag =TAGIMG00280010;
                break;
        case 14 : tag =TAGIMG00280011;
                break;
        case 15 : tag =TAGIMG00280100;
                break;
        case 16 : tag =TAGIMG00280101;
                break;
        case 17 : tag = TAGIMG00280102;
                break;
        case 18 : tag = TAGIMG00280103;
                break;
        case 19 : tag = TAGIMG7FE00010;
                break;
        case 20 : tag = TAGIMG00280006;
                break;
        case 21 : tag = TAGIMG00280034;
                break;
        case 22 : tag = TAGIMG00281101;
                break;
        case 23 : tag = TAGIMG00281102;
                break;
        case 24 : tag = TAGIMG00281103;
                break;
        case 25 : tag = TAGIMG00281201;
                break;
        case 26 : tag = TAGIMG00281202;
                break;
        case 27 : tag = TAGIMG00281203;
                break;
        case 28 : tag = TAGIMG00200062;
                break;
        case 29 : tag = TAGIMG00082218;
                break;
        case 30 : tag = TAGIMG00080008;
                break;
        case 31 : tag = TAGIMG00281040;
                break;
        case 32 : tag = TAGIMG00281041;
                break;
        case 33 : tag = TAGIMG00281052;
                break;
        case 34 : tag = TAGIMG00281053;
                break;
        case 35 : tag = TAGIMG00281054;
                break;
        case 36 : tag = TAGIMG20500020;
                break;
        case 37 : tag = TAGIMG00282110;
                break;
        case 38 : tag = TAGIMG00282112;
                break;
        case 39 : tag = TAGIMG00280301;
                break;
        case 40 : tag = TAGIMG00283010;
                break;
        case 41 : tag = TAGIMG00283002;
                break;
        case 42 : tag = TAGIMG00283006;
                break;
        case 43 : tag = TAGIMG00281050;
                break;
        case 44 : tag = TAGIMG00281051;
                break;
        case 45 : tag = TAGIMG00187004;
                break;
        case 46 : tag = TAGIMG00187030;
                break;
        case 47 : tag = TAGIMG00187032;
                break;
        case 48 : tag = TAGIMG00187034;
                break;
        case 49 : tag = TAGIMG00181164;
                break;
        case 50 : tag = TAGIMG00181508;
                break;
        case 51 : tag = TAGIMG00400318;
                break;
        case 52 : tag = TAGIMG00540220;
                break;
        case 53 : tag = TAGIMG00540222;
                break;
        case 54 : tag = TAGIMG60xx0010;
                break;
        case 55 : tag = TAGIMG60xx0011;
                break;
        case 56 : tag = TAGIMG60xx0040;
                break;
        case 57 : tag = TAGIMG60xx0050;
                break;
        case 58 : tag = TAGIMG60xx0100;
                break;
        case 59 : tag = TAGIMG60xx0102;
                break;
        case 60 : tag = TAGIMG60xx3000;
                break;
        case 61 : tag = TAGIMG00400555;
                break;
        case 62 : tag = TAGIMG0040A043;
                break;
        case 63 : tag = TAGIMG0040A136;
                break;
        case 64 : tag = TAGIMG0040A30A;
                break;
        case 65 : tag = TAGIMG004008EA;
                break;
        case 66 : tag = TAGIMG0040A121;
                break;
        case 67 : tag = TAGIMG0040A122;
                break;
        case 68 : tag = TAGIMG0040A123;
                break;
        case 69 : tag = TAGIMG0040A160;
                break;
        case 70 : tag = TAGIMG0040A168;
                break;
        case 71 : tag = TAGIMG00080016;
                break;
        case 72 : tag = TAGIMG00080018;
                break;
        case 73 : tag = TAGIMG00080005;
                break;
        case 74 : tag = TAGIMG00080102;
                break;
        case 75 : tag = TAGIMG00080112;
                break;
        case 76 : tag = TAGIMG0008010C;
                break;
        case 77 : tag = TAGIMG00080114;
                break;
        case 78 : tag = TAGIMG04000500;
                break;
        case 79 : tag = TAGIMG04000510;
                break;
        case 80 : tag = TAGIMG04000520;
                break;
        default: tag = "";

    }

    // Esta contenido si: sacar datos y agregarlo al vector
    //                no: continuar con el siguiete
    if (DEBUG) System.out.println(" -Imagen-El valor de TAG es: >" + tag
+
                                  "<");
```

```
        if ( isContains( tag ) && tag!=TAGIMG7FE00010 ){
            if (DEBUG) System.out.println("-Imagen-Si lo encontro a " + tag );

            VectorDatosImagen.addElement(getfilatag(tag).get(4));
        }

        else{
            if (DEBUG) System.out.println("-Imagen-No encontro al tag >" +
tag +
                                          "<");
            if (tag!=TAGIMG7FE00010)   VectorDatosImagen.addElement("");
            else VectorDatosImagen.addElement("?");
        }
     }

   if (DEBUG) System.out.println("El numero de elementos de este vector
es: <"
           + VectorDatosImagen.size() + ">");

   }

   /**
    * Regresa el vector de datos del Imagen
    */
   public Vector getDatosImagen(){
       return VectorDatosImagen;

   }

   /**
    * Busca la informacion del vector de equipo y entrega el string de toda
la informacion
    *
    */
    public String getImagen(){

       String result= null;
       String Tempo = null;

       boolean  flag = true;

        for (Enumeration e = VectorDatosImagen.elements() ;
e.hasMoreElements() ;) {

           Tempo =(String)e.nextElement();

           if (!Tempo.equals("?"))  Tempo ="""" + Tempo + """ ;
           /*
           Tempo = """+(String)e.nextElement() + """;
           */
           if (flag) {
               result = Tempo;
               flag = false;
           }
           else result = result.concat(Tempo);
           if( e.hasMoreElements()){
               result = result.concat(",");
           }
        }
       System.out.println ("\n\n*******-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
       System.out.println ("este es el valor de getImagen>" + result + "<");
       System.out.println ("\n\n*******-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
       return result;
    }

   /**
    *  Entrega el  el fabricante del equipo
    */

   public String getIdImagen(){
       int COLUMNA = 3;
           if (DEBUG) System.out.println("getIdImagen es:>" +
               (String)VectorDatosImagen.get(COLUMNA) + "<");

           return (String)VectorDatosImagen.get(COLUMNA);

   }
```

```
   /**
    * Prueba esta Clase
    *
    * @param args the command line arguments
    */
   public static void main(String[] args) {
       Vector vectordato = new Vector();
       int     NUM_FILASmas =20;
       String tag;

        String  TAGPAC10y= "x(0010,0010)x";
String  TAGPAC20y= "x(0010,0020)x";
String  TAGPAC30y= "x(0010,0030)x";
String  TAGPAC40y= "x(0010,0040)x";
String  TAGPAC16y= "x(0008,0016)x";
String  TAGPAC18y= "x(0008,0018)x";

        // genera el vector de datos
       for (int k=0; k < NUM_FILASmas; k++){
           Vector fila = new Vector();
           fila.addElement(k);
           fila.addElement("(0010,00" + k + ")");
           fila.addElement("VR" + k);
           fila.addElement("valor" + k);
           fila.addElement("otro" + k);
           vectordato.add(fila);

       }
        for (int j=0; j<6;j++){
         Vector    filadatos = new Vector();

         switch (j){
            case 1 : tag = TAGPAC20y;
                     break;
            case 0 : tag = TAGPAC10y;
                     break;
            case 2 : tag = TAGPAC30y;
                     break;
            case 3 : tag = TAGPAC40y;
                     break;
            case 4 : tag = TAGPAC16y;
                     break;
            case 5 : tag = TAGPAC18y;
                     break;
            default: tag = "";
         }
           filadatos.addElement(j);
           filadatos.addElement(tag);
           filadatos.addElement("VR" + j);
           filadatos.addElement("valor" + j);
           filadatos.addElement("otro" + j);
           vectordato.add(filadatos);

        }


       Vectores prueba = new Vectores(vectordato);
       // ejecuita isContains
          // con valor true y false
       System.out.println (prueba.isContains(TAGPAC18y));
       System.out.println (prueba.isContains(TAGPAC18y));



       // ejecuta getfilatag
            // estara un dato si no regresa el vector vacio
       if (prueba.isContains(TAGPAC18y)){
           System.out.println ("el contenido de la fila es");
           System.out.println(prueba.getfilatag(TAGPAC18y));

       }


       // ejecuta getpaciente
            // estara un dato si no regresa el vector vacio
       System.out.println ("El contenido del vecot de paciente es");
      // System.out.println(prueba.getPaciente());
       System.out.println ("***************");
       prueba.creaDatosPaciente();
```

```
        System.out.println(prueba.getDatosPaciente());
        System.out.println ("**ento a get paciente***********");

        System.out.println(prueba.getPaciente());
//        System.out.println ("*******salio*******");

    }

}
```

/root/Desarrollo/spacsm/Princi_2/src/Principal/Visualizador.java

```
/*
 * Visualizador.java
 *
 * Created on 21 de octubre de 2005, 01:39 PM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package Principal;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;

/**
 *  Esta clase genera unJFrame para poder ver la imagen
 * @author armando
 */
public class Visualizador extends JFrame{
    Image image;

    /** Creates a new instance of Visualizador */
    public Visualizador(String titulo, Image tmpimage) {
            image = tmpimage;
            Toolkit toolkit = Toolkit.getDefaultToolkit();

             MediaTracker mediaTracker = new MediaTracker(this);
             mediaTracker.addImage(image, 0);
             try
             {
                    mediaTracker.waitForID(0);
             }
             catch (InterruptedException ie)
             {
                    System.err.println(ie);
                    System.exit(1);
             }
             addWindowListener(new WindowAdapter() {
             public void windowClosing(WindowEvent e) {
                    System.exit(0);
             }
             });

             setSize(image.getWidth(null), image.getHeight(null));
             setTitle(titulo);

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
             show();
       }

       public void paint(Graphics graphics) {
             graphics.drawImage(image, 0, 0, null);
       }


}
```

Anexo C

Programas para visualizar la información por medio de una navegador WEB Programas para Browser.

/root/Desarrollo/spacsm/SPACSM_1/web/VerificarFinSes.jsp

```
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  if (!administradorSesion.yaRegistrado(credenciales)) {
%>
    <jsp:forward page="NoSesionInic.jsp" />
<%
  }
%>
<html>
<body bgcolor="#F8F8FF" link="#999999"
   vlink="#999999" alink="#999999" onLoad="giveFocus()">
<center>
<font size=+2>
<BR>
<BR>
<B>VERIFICACIÓN DE FINAL DE SESIÓN</B>
</font>
</center>
<font size=+1>
<br>
<br>
Si realmente desear finalizar la sesión, haga clic en el botón etiquetado
<b>CONFIRMAR FINAL SESIÓN</b> que se encuentra abajo.
<p>
Si <b>no</b> desea finalizar la sesión, haga clic en uno de los vínculos
del menú de la izquierda.
<br>
<br>
</font>
<center>
<FORM NAME="logout" METHOD=POST ACTION="Salir.jsp">
<BR>
<BR>
<BR>
<font size=+1>
<b>
<INPUT TYPE="submit" VALUE="CONFIRMAR FINAL SESIÓN">
</b>
</font>
</FORM>
</CENTER>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/Alta.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.io.*,
                java.util.*,
                java.net.*,
                java.lang.*,
                java.sql.*" %>
<%--
The taglib directive below imports the JSTL library. If you uncomment it,
you must also add the JSTL library to the project. The Add Library... action
on Libraries node in Projects view can be used to add the JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Alta Usuario</title>
    </head>
    <body>
    <%


    String Usuario = "Usuariox";
    String Password = "Password";
    String Status = "off";
    String Admin = "off";


    if(request.getParameter("Usuario") != null){
        Usuario = request.getParameter("Usuario");
    }
    if(request.getParameter("Password") != null)
        Password = request.getParameter("Password");
    if(request.getParameter("Status") != null)
        Status = request.getParameter("Status");

    if(request.getParameter("Admin") != null)
        Admin = request.getParameter("Admin");

    /*
    String Usuario = "Usuario1420000";
    String Password = "Password";
    String Status = "on";
    String Admin = "off";
    */

    int renglonInsertado=0;

    int status = 0;
    if (Status.equals("on")) status=1;

    int admin = 0;
    if (Admin.equals("on")) admin=1;


    Connection connection = null;
    Statement statement = null;
    try{
/* Get a connection. */
        String driverClassName = "com.mysql.jdbc.Driver";
        String driverUrl = "jdbc:mysql://loky.uam.mx/usuarios";
        String user = "admin";
        String password = "qwert";
        Class.forName(driverClassName);
        connection = DriverManager.getConnection(driverUrl, user, password);

/* Create "statement". */
        statement = connection.createStatement();
/* Insert the accounts in database. */

/* Execute query. */
          String queryString = "INSERT INTO idpassword " +
"(idusuario,password,status,admin) VALUES (" +
                        """ + Usuario +""","+
                        """ + Password +""","+
                        """ + status +""","+
                        """ + admin +"""+  ")";

          int insertedRows = statement.executeUpdate(queryString);
          if (insertedRows != 1)  {
             out.println(" <h1>Error</h1>");
             throw new SQLException(Usuario + ": problems when inserting
!!!!");

          }
          else   out.println(" <h1>Exito al insertar "+ Usuario +"</h1>");


    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    finally {
          try {
             if (statement != null) {
                statement.close();
                }
             if (connection != null) {
                connection.close();
                }
          } catch (SQLException e) {
                e.printStackTrace();
             }
    } // try


    %>

    </body>
```

```
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/Bienvenido.jsp

```
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<html>
<body>
<center>
<font size="+2">
<b>
<br>
<br>
<br>
El inicio de sesión tuvo éxito para el usuario
<%= credenciales.getUsuario() %>
</b>
</font>
</center>
<font size="+1">
<br>
<br>
Ahora puede entrar a las opciones del menu de la inzquierda con solo
hacer clic en el vínculo apropiado del menú de la izquierda.
<p>
Cuando haya terminado, por favor finalice su sesión.
</font>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/cabecera.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
  <head>
    <title></title>
  </head>
  <meta name="description" content="descripcion de la cabezera">
  <meta name="author" content="Armando Jimenez Herrera">
  <meta name="keywords" content="Sistema PACS Minimo">
  <meta name="date" content="20 enero del 2006">
  <!-- ... otras especificaciones en la cabeza del archivo ... -->

    <body bgcolor=#00678A> <!-- color de fondo steelblue -->
    <!--<body bgcolor=#4682B4> <!-- color de fondo steelblue -->
    <!--<table bgcolor=#00C0C0> <!-- el color de la tabla es az□l-verde -
->
    <!--<hr color=#CC00CC>  <!-- l□nea de separaci□n violeta -->

    <h1 style="color:white" >
       <img src="Imagenes/spacsm_v.jpg" width="100" height="230"
border="2"  align="left" vspace="0" hspace="20" alt="SPACSM_V">
       Sistema PACS minimo basado en el estandar DICOM.</h1>
       <p>Universidad Autonoma Metropolitana  Azcapotzalco</p>
       <p>Maestria en Ciencias de la Computacion.</p>
       <p> Tesis presentada por:JIMENEZ HERRERA Armando.</p>
       <p> Para obtener el grado de:Maestria en Ciencias de la
Computacion. </p>
       <p> Asesor:Dr. Carlos Aviles. </p>

  </body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/CatalogImagenes.jsp

```
<%@ page info="Favorito #2 de Paul" %>
<%@ page errorPage="/comun/Excepcion.jsp" %>
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  if (!administradorSesion.mantenerViva(credenciales,getServletInfo())) {
%>
    <jsp:forward page="SolicitudInSes.jsp" />
<%
  }
%>
    <jsp:forward page="CatalogoImagen.jsp" />
```

/root/Desarrollo/spacsm/SPACSM_1/web/CatalogoImagen.jsp

```
<%@ page import="java.io.*,java.util.*,java.net.*,java.sql.*" %>
<h1>Catalogo de Imagenes</h1>
<hr>

<%
// declarando y creando objetos globales
Connection canal = null;
ResultSet tabla= null;
Statement instruccion=null;
String strcon =
"jdbc:mysql://loky.uam.mx/spacsm?user=admin&password=qwert";


// abriendo canal o enlace en su propio try-catch

  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion

    String q="select " +
          "Serie_Estudio_Paciente_NumeroIdentificacionPaciente," +
          "Serie_Estudio_IdentificadorInstanciaEstudio," +
          "Serie_Equipo_FabricanteEquipo," +
          "Serie_IdentificadorSerie," +
          "InstanceNumber from Imagen";


// mandando el sql a la base de datos

    try {
       tabla = instruccion.executeQuery(q);

// mandando resultset a tabla html

       out.println("<TABLE Border=10 CellPadding=5><TR>");

       out.println("<th bgcolor=#F8F8FF>PACIENTE</th><th" +
             " bgcolor=#F8F8FF>ESTUDIO</th><th " +
             " bgcolor=#F8F8FF>EQUIPO</th><th " +
             " bgcolor=#F8F8FF>SERIE</th><th " +
            "bgcolor=#F8F8FF>IMAGEN</th></TR>");

       while(tabla.next()) {

          out.println("<TR>");

          out.println("<TD>"+tabla.getString(1)+"</TD>");
          out.println("<TD>"+tabla.getString(2)+"</TD>");
          out.println("<TD>"+tabla.getString(3)+"</TD>");
          out.println("<TD>"+tabla.getString(4)+"</TD>");
          String href = "<a href=\"MostrarImagen.jsp?" +
                    "IdPaciente=" + tabla.getString(1) + "&"+
                    "IdEstudio="  + tabla.getString(2) + "&"+
                    "IdEquipo="   + tabla.getString(3) + "&"+
                    "IdSerie="    + tabla.getString(4) + "&"+
                    "IdImagen="   + tabla.getString(5) + "\">";

          out.println("<TD>" + href + tabla.getString(5) + "</a></TD>");

          out.println("</TR>");
       }; // fin while

       out.println("</TABLE></CENTER></DIV></HTML>");

       tabla.close();
       } //fin try no usar ; al final de dos o mas catchs

       catch(SQLException e) {};

       try {canal.close();} catch(SQLException e) {};

%>
```

/root/Desarrollo/spacsm/SPACSM_1/web/CuentaSuspendida.jsp

```
<%
  session.invalidate();
%>
<html>
<body>
<center>
<font size="+2">
```

```
<b>
<br>
<br>
<br>¡SU ID DE USUARIO SE HA SUSPENDIDO TEMPORALMENTE!
</font>
</center>
<font size="+1">
<br>
<br>
Ha excedido el número máximo de intentos de registro de inicio de sesión
fallidos.
o el administrador la tiene suspendida temporalmente
<p>
Para reinstalar su ID, por favor, póngase en contacto con el administrador
del sistema.
</font>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/cuerpo.html

```
<html>
<head>
  <title>C□digo de ejemplo del cap□tulo 3</title>
</head>
<frameset cols="110,*">>

  <frame src="MarcoMenu.html"
   frameborder="0" noresize="noresize" name="marcomenuprincipal">
  <frame src="Intro.html"
   frameborder="0" name="marcoderecha">
</frameset>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/FallaInSes2.jsp

```
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<html>
<head>
<title>FALLA EN INICIO DE SESIÓN</title>
</head>
<body bgcolor="#FFFFFF">
<center>
<br>
<h1>El inicio de sesión no ha tenido éxito para el usuario
<%= credenciales.getUsuario() %>
</h1>
<br>
<b>
<%
  if (administradorSesion.estaSuspendido(credenciales)) {
%>
<jsp:forward page="CuentaSuspendida.jsp" />


<%
  }
  else {
    administradorSesion.aumentarIntentosInSes(credenciales);
    if (administradorSesion.getIntentosInSes(credenciales) >= 3) {
      administradorSesion.suspenderCuenta(credenciales);
%>
<jsp:forward page="CuentaSuspendida.jsp" />
<%
    }
  }
%>
Haga clic en el botón de abajo para hacer una nueva solicitud.
</b>
<form method=get action="InicioSesion2.jsp">
<input type=submit name="submit" value="OK">
</form>
</center>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/Info_E_Imagen.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>C□digo de ejemplo del cap□tulo 3</title>
</head>
<frameset cols="50%,50%">>

  <frame src="CatalogImagenes.jsp"
   frameborder="0" noresize="noresize" name="marcomenuprincipal">
  <frame src="MuestraInfo.jsp"
   frameborder="0" name="marcoderecha">
</frameset>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/InicioSesion2.jsp

```
 <jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  if (administradorSesion.yaRegistrado(credenciales)) {
%>
      <jsp:forward page="PrimeroFinSes.jsp" />
<%
  }
%>
<html>
<head>
  <title>Inicio de sesión</title>

<script language="JavaScript">

   function darEnfoque() {
     document.login.usuario.focus()
   }

   function remitirForm() {
     document.login.submit()
   }

   function restablecerForm() {
     document.login.reset()
     document.login.usuario.focus()
 }
</script>
</head>

<body bgcolor="#F8F8FF" link="#999999"
  vlink="#999999" alink="#999999" onLoad="darEnfoque()">
<center>
<font size=+2>
<BR>
<BR>
<B>REGISTRO DE INICIO DE SESIÓN</B>
</font>
<br>
<br>
<br>
<br>
<FORM NAME="login" METHOD=POST ACTION=
  "ValidarUsuario2.jsp">
<TABLE WIDTH="50%">
<TR>
<TD align=right>
<font size=+1>
<B>
ID usuario:
</B>
</font>
</TD>
<TD>
<font size=+1><INPUT NAME="usuario" TYPE="TEXT" LENGTH="9"
  MAXLENGTH="9">
</font>
</TD>
</TR>
<TR>
<TD align=right>
<font size=+1>
<B>
Contraseña:
</B>
</font>
</TD>
<TD>
<font size=+1><INPUT NAME="password" TYPE="PASSWORD"
LENGTH="8"
  MAXLENGTH="8">
</font>
</TD>
```

```
</TR>
</TABLE>
<BR>
<BR>
<BR>
<font size=+1>
<b>
<INPUT TYPE="button" VALUE="INICIAR SESION"
onClick="remitirForm()">
     
<INPUT TYPE="button" VALUE="RESTABLECER"
onClick="restablecerForm()">
</b>
</font>
</FORM>
</CENTER>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/Intro.html

```
<html>
<body bgcolor="#F8F8FF" link="#999999"
   vlink="#999999" alink="#999999">
<center>
<font size=+2>
<br><br>
<b>Bienvenido</B>
</font>
</center>
<br>
<br>
<br>
<br>
<font size=+1>
Sistema PACS Minimo.
<p>
Antes de que pueda ver cualquiera de las paginas. debe registrar su inicio
de
sesion. Tenga cuidado al escribir su ID de usuario y su contrasena porque
solo
dispone de <i><b>tres</b></i> intentos. Despues de eso, su cuenta se
desactivara temporalmente y tendra que recurrir al administrador para que
restablezca su cuenta.
<p>

</font>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/listPaciente.jsp

```
<%@ page import="java.io.*,java.util.*,java.net.*,java.sql.*" %>

<html>
  <head>
    <title>Busqueda de pacientes</title>
  </head>
  <body>

  <h1>Busqueda de Paciente</h1>
    <hr>
<%

// declarando y creando objetos globales

Connection canal = null;

ResultSet tabla= null;

Statement instruccion=null;

String strcon =
"jdbc:mysql://loky.uam.mx/spacsm?user=admin&password=qwert";

if(request.getParameter("OK") != null)

{

// abriendo canal o enlace en su propio try-catch

try {

Class.forName("com.mysql.jdbc.Driver").newInstance();

canal=DriverManager.getConnection(strcon);

instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,

ResultSet.CONCUR_UPDATABLE);

} catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};

// preparando condicion de busqueda

int clave = Integer.parseInt(request.getParameter("CLAVEB"));

// construyendo select con condicion

String q="select * from Paciente where
NumeroIdentificacionPaciente="+clave;

// mandando el sql a la base de datos

try { tabla = instruccion.executeQuery(q);

// mandando resultset a tabla html

out.println("<TABLE Border=10 CellPadding=5><TR>");

out.println("<th bgcolor=#F8F8FF>IdPaciente</th><th" +
        " bgcolor=#F8F8FF>Nombre</th><th " +
        " bgcolor=#F8F8FF>Fecha de nacimiento</th><th " +
        " bgcolor=#F8F8FF>Sexo</th><th " +
        " bgcolor=#F8F8FF>SOPClassUID</th><th " +
        "bgcolor=#F8F8FF>SOPinstanceUID</th></TR>");

while(tabla.next()) {

out.println("<TR>");

out.println("<TD>"+tabla.getString(1)+"</TD>");
out.println("<TD>"+tabla.getString(2)+"</TD>");
out.println("<TD>"+tabla.getString(3)+"</TD>");
out.println("<TD>"+tabla.getString(4)+"</TD>");
out.println("<TD>"+tabla.getString(5)+"</TD>");
out.println("<TD>"+tabla.getString(6)+"</TD>");


out.println("</TR>"); }; // fin while

out.println("</TABLE></CENTER></DIV></HTML>");

tabla.close(); } //fin try no usar ; al final de dos o mas catchs

catch(SQLException e) {};

try {canal.close();} catch(SQLException e) {};

};

// construyendo forma dinamica

out.println("<FORM ACTION=listPaciente.jsp METHOD=post>");
out.println("<p> Se buscara informacion del Paciente <p>");

out.println("CLAVE BUSCAR:<INPUT TYPE=TEXT NAME=CLAVEB><BR>");

out.println("<INPUT TYPE=SUBMIT NAME=OK VALUE=BUSCAR><BR>");

out.println("</FORM>");

%>


  </body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/MarcoMenu.html

```
<html>

<body text=#F8F8FF bgcolor=#0074BB link=#555555
  vlink=#5555ff alink=#55ff55>
<!--<body text="#000000" bgcolor="#e0e0e0" link="#555555" -->

<br>
<br>
<table COLS=1 WIDTH="100%" >
  <tr>
    <td >
      <a href="InicioSesion2.jsp" target="marcoderecha" style=" color:
#F8F8FF">Inicio de sesión</a>
    </td>
  </tr>
```

```
  <tr>
    <td>
    </td>
  </tr>

  <tr>
    <td>
    </td>
  </tr>

  <tr>
    <td><a href="Paciente.jsp" target="marcoderecha" style=" color:
#F8F8FF">Paciente</a>
    </td>
  </tr>

  <tr>
    <td><a href="CatalogImagenes.jsp" target="marcoderecha" style="
color: #F8F8FF">Catalogo de imagenes</a>
    </td>
  </tr>
<!--
   <tr>
    <td><a href="Pasaje3.jsp" target="marcoderecha" style=" color:
#F8F8FF">Pasaje 3</a>
    </td>
  </tr>
-->
  <tr>
    <td>
    </td>
  </tr>

  <tr>
    <td>
    </td>
  </tr>

  <tr>
    <td><a href="VerificarFinSes.jsp" target="marcoderecha" style=" color:
#F8F8FF">
      T□rmino de sesi□n</a>
    </td>
  </tr>
</table>

</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/MostrarImagen.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%--
The taglib directive below imports the JSTL library. If you uncomment it,
you must also add the JSTL library to the project. The Add Library... action
on Libraries node in Projects view can be used to add the JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
   <title>C□digo de ejemplo del cap□tulo 3</title>
</head>
<frameset cols="50%,50%">>


<%
String IdPaciente    = request.getParameter("IdPaciente");
String IdEstudio     = request.getParameter("IdEstudio");
String IdEquipo      = request.getParameter("IdEquipo");
String IdSerie       = request.getParameter("IdSerie");
String IdImagen      = request.getParameter("IdImagen");
%>
<%
    out.println ("<frame src=\"MuestraImagen.jsp?IdImagen=" +
                 IdImagen + "\"");
    out.println("frameborder=\"0\" noresize=\"noresize\"
name=\"marcomenuprincipal\">");
%>

<%
    out.println ("<frame src=\"MuestraInfo.jsp?" +
                 "IdPaciente=" + IdPaciente + "&" +
                 "IdEstudio="  + IdEstudio + "&" +
                 "IdEquipo="   + IdEquipo + "&" +
                 "IdSerie="    + IdSerie + "&" +
                 "IdImagen="   + IdImagen + " \"");
    out.println("  frame border=\"0\" name=\"marcoderecha\">");
%>


</frameset>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/MuestraImagen.jsp

```
<%@ page contentType="image/jpeg"
import="java.awt.*,
        java.awt.event.*,
        java.awt.image.*,
        com.sun.image.codec.jpeg.*,
        java.util.*,
        java.net.*,
        java.sql.*,
        javax.swing.*,
        java.io.*,
        javax.imageio.stream.*,
        javax.imageio.*"
%>


<%
 String IdImagen = request.getParameter("IdImagen");
%>


<%
// declarando y creando objetos globales
    Connection canal = null;
    ResultSet rs= null;
    Statement instruccion=null;
    String strcon =
"jdbc:mysql://loky.uam.mx/spacsm?user=admin&password=qwert";

    Image img = null;

// abriendo canal o enlace en su propio try-catch

  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    }
     catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion

    String q="select InstanceNumber, " +
             "lenght, PixelData from " +
             "Imagen where InstanceNumber = " +
            """ + IdImagen + """;


// mandando el sql a la base de datos

    try {
       rs = instruccion.executeQuery(q);

// mandando resultset a tabla html


       rs.next();


       int len=rs.getInt(2);
       byte [] b=new byte[len];
       InputStream in = rs.getBinaryStream(3);
       in.read(b);
       in.close();
       ImageIcon icon = new ImageIcon(b);
       img = icon.getImage();
       rs.close();
       } //fin try no usar ; al final de dos o mas catchs
       catch(SQLException e) {};
```

```
      try {canal.close();} catch(SQLException e) {};

// Crear la imagen
    int w = img.getWidth(null),
        h = img.getHeight(null);
    BufferedImage image = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);


// Obtener el contexto para dibujar

    Graphics2D imageGraphics = image.createGraphics();
    imageGraphics.drawImage(img, 0, 0, null);
    ServletOutputStream sos = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(sos);
    encoder.encode(image);
 %>
```

/root/Desarrollo/spacsm/SPACSM_1/web/MuestraInfo.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,java.net.*,java.sql.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>

        <title>MuestraInfo</title>
    </head>
    <body>

    <h1>ESTA ES LA INFORMACION ASOCIADA A LA IMAGEN</h1>
    <hr>
    <hr>

<%
/*
String IdPaciente   ="3";
String IdEstudio    ="3";
String IdEquipo     ="Philips Medical Systems";
String IdSerie     ="serie1";
String IdImagen     ="1";
*/
String IdPaciente   =request.getParameter("IdImagen");
String IdEstudio    =request.getParameter("IdEstudio");
String IdEquipo     =request.getParameter("IdEquipo");
String IdSerie     =request.getParameter("IdSerie");
String IdImagen     =request.getParameter("IdImagen");




// declarando y creando objetos globales
Connection canal = null;
ResultSet tabla= null;
Statement instruccion=null;
String strcon =
"jdbc:mysql://loky.uam.mx/spacsm?user=admin&password=qwert";


// abriendo canal o enlace en su propio try-catch

  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion

    out.println("<h1>Paciente</h1>");
    out.println("<hr><BR>");
//Paciente
    String query="SELECT * FROM Paciente WHERE
NumeroIdentificacionPaciente= " +
          """+ IdPaciente + """;


// mandando el sql a la base de datos

    try {
        ResultSet rs = instruccion.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();

        int columnCount = rsmd.getColumnCount();

        out.println("<TABLE Border=10 CellPadding="+ "2" +">");
        while(rs.next()) {


            for (int i=1; i<= columnCount; i++) {
            out.println("<TR>");
            String s= rsmd.getColumnLabel(i);
            out.println("<TD bgcolor=#F8F8FF>"+ s +"</TD>");
            out.println("<TD>"+rs.getString(i)+"</TD>");
            out.println("</TR>");
            }
        }; // fin while
        out.println("</TABLE>");
        rs.close();
        } //fin try no usar ; al final de dos o mas catchs

        catch(SQLException e) {};

        try {canal.close();} catch(SQLException e) {};
////////////////////////////////////////////////////////////////
//Estudio
  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion

    out.println("<h1>Estudio</h1>");
    out.println("<hr><BR>");

        query="SELECT * FROM Estudio WHERE
IdentificadorInstanciaEstudio=" +
          """+ IdEstudio + """;


// mandando el sql a la base de datos

    try {




        ResultSet rs = instruccion.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();

        int columnCount = rsmd.getColumnCount();

        out.println("<TABLE Border=10 CellPadding="+ "2" +">");
        while(rs.next()) {


            for (int i=1; i<= columnCount; i++) {
            out.println("<TR>");
            String s= rsmd.getColumnLabel(i);
            out.println("<TD bgcolor=#F8F8FF>"+ s +"</TD>");
            out.println("<TD>"+rs.getString(i)+"</TD>");
            out.println("</TR>");
            }
        }; // fin while
        out.println("</TABLE>");
        rs.close();
        } //fin try no usar ; al final de dos o mas catchs

        catch(SQLException e) {};

        try {canal.close();} catch(SQLException e) {};
////////////////////////////////////////////////////////////////
//Equipo
  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion
```

```
    out.println("<h1>Equipo</h1>");
    out.println("<hr><BR>");

       query="SELECT * FROM Equipo  WHERE  FabricanteEquipo=" +
          """+ IdEquipo+ """;


// mandando el sql a la base de datos

    try {



       ResultSet rs = instruccion.executeQuery(query);
       ResultSetMetaData rsmd = rs.getMetaData();

       int columnCount = rsmd.getColumnCount();
        out.println("<TABLE Border=10 CellPadding="+ "2" +">");
       while(rs.next()) {


          for (int i=1; i<= columnCount; i++) {
          out.println("<TR>");
          String s= rsmd.getColumnLabel(i);
          out.println("<TD bgcolor=#F8F8FF>"+ s +"</TD>");
          out.println("<TD>"+rs.getString(i)+"</TD>");
          out.println("</TR>");
          }
       }; // fin while
       out.println("</TABLE>");

       rs.close();
       } //fin try no usar ; al final de dos o mas catchs

       catch(SQLException e) {};

       try {canal.close();} catch(SQLException e) {};
////////////////////////////////////////////////////////////////////

//Serie
  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion

    out.println("<h1>Serie</h1>");
    out.println("<hr><BR>");

       query="SELECT * FROM Serie  WHERE IdentificadorSerie=" +
          """+ IdSerie + """;


// mandando el sql a la base de datos

    try {



       ResultSet rs = instruccion.executeQuery(query);
       ResultSetMetaData rsmd = rs.getMetaData();

       int columnCount = rsmd.getColumnCount();
        out.println("<TABLE Border=10 CellPadding="+ "2" +">");
       while(rs.next()) {


          for (int i=1; i<= columnCount; i++) {
          out.println("<TR>");
          String s= rsmd.getColumnLabel(i);
          out.println("<TD bgcolor=#F8F8FF>"+ s +"</TD>");
          out.println("<TD>"+rs.getString(i)+"</TD>");
          out.println("</TR>");
          }
       }; // fin while
       out.println("</TABLE>");

       rs.close();
       } //fin try no usar ; al final de dos o mas catchs

       catch(SQLException e) {};

       try {canal.close();} catch(SQLException e) {};
////////////////////////////////////////////////////////////////////
//IMAGEN
  try {

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    canal=DriverManager.getConnection(strcon);
    instruccion = canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};


// construyendo select con condicion

    out.println("<h1>Imagen</h1>");
    out.println("<hr><BR>");

       query="SELECT * FROM Imagen  WHERE InstanceNumber=" +
          """+ IdImagen + """;


// mandando el sql a la base de datos

    try {
       ResultSet rs = instruccion.executeQuery(query);
       ResultSetMetaData rsmd = rs.getMetaData();

       int columnCount = rsmd.getColumnCount();
        out.println("<TABLE Border=10 CellPadding="+ "2" +">");
       while(rs.next()) {


          for (int i=1; i<= columnCount; i++) {
           out.println("<TR>");
           String s= rsmd.getColumnLabel(i);
           if (!s.equals("PixelData")){
              out.println("<TD bgcolor=#F8F8FF>"+ s +"</TD>");
             out.println("<TD>"+rs.getString(i)+"</TD>");
             out.println("</TR>");
           }

          }
       }; // fin while
       out.println("</TABLE>");

       rs.close();
       } //fin try no usar ; al final de dos o mas catchs

       catch(SQLException e) {};

       try {canal.close();} catch(SQLException e) {};
////////////////////////////////////////////////////////////////////




%>


   </body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/NoSesionInic.jsp

```
<html>
<body>
<center>
<font size="+2">
<b>
<br>
<br>
<br>
¡¡¡NO HA INICIADO UNA SESIÓN!!!
<br>
<br>
</font>
</center>
<font size="+1">
<br>
<br>
Para iniciar sesión, haga clic en el vínculo etiquetado <b><i>Inicio de
sesión</i></b> del menú
de la izquierda.
```

```
</font>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/Paciente.jsp

```
<%@ page info="Favorito #1 de Paul" %>
<%@ page errorPage="/comun/Excepcion.jsp" %>
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  if (!administradorSesion.mantenerViva(credenciales,getServletInfo())) {
%>
    <jsp:forward page="SolicitudInSes.jsp" />
<%
  }
%>
    <jsp:forward page="listPaciente.jsp" />
```

/root/Desarrollo/spacsm/SPACSM_1/web/Pasaje3.jsp

```
<%@ page info="Favorito #1 de Paul" %>
<%@ page errorPage="/comun/Excepcion.jsp" %>
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  if (!administradorSesion.mantenerViva(credenciales,getServletInfo())) {
%>
    <jsp:forward page="SolicitudInSes.jsp" />
<%
  }
%>
<html>
<body>
<center>
<br>
<br>
Esta página se ha dispuesto para que la vea el usuario
<%= credenciales.getUsuario() %>
</center>
<br>
<br>
"Would you tell me, please, which way I ought to go from
here?"<P>
<P>
"That depends a good deal on where you want to get to," said the
Cat.<P>
<P>
"I don"t much care where--" said Alice.<P>
<P>
"Then it doesn"t matter which way you go," said the Cat.<P>
<P>
"--so long as I get <I>somewhere</I>," Alice added as an
explanation.<P>
<P>
"Oh, you"re sure to do that," said the Cat, "if you only walk
long enough."<P>
<p>
Alice"s Adventures In Wonderland -- Lewis Carroll
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/PrimeroFinSes.jsp

```
<html>
<body>
<center>
<font size="+2">
<b>
<br>
<br>
<br>
¡¡¡PRIMERO FINALICE LA SESIÓN!!!
</font>
</center>
<font size="+1">
<br>
<br>
Sólo puede iniciar sesión como un usuario a la vez.
<p>
Si desea iniciar sesión como otro usuario. primero finalice sesión.
Si llegó aquí por accidente, seleccione la opción que desee del menú de la
izquierda.
</font>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/principal.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title></title>
</head>
<frameset rows ="80,*">

  <frame src="cabecera.html"
   frameborder="0" noresize="noresize" name="cabecera">
  <frame src="cuerpo.html"
   frameborder="0" name="cuerpo">
</frameset>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/Salir.jsp

```
<%@ page errorPage="/comun/Excepcion.jsp" %>
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  administradorSesion.salir(credenciales);
  session.invalidate();
%>
<%@ include file="Intro.html" %>
```

/root/Desarrollo/spacsm/SPACSM_1/web/SesionDuplicada2.jsp

```
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<jsp:setProperty name="credenciales" property="usuario" value="" />
<jsp:setProperty name="credenciales" property="password" value="" />
<html>
<head>
<title>FALLA DE INICIO DE SESION</title>
</head>
<body bgcolor="#FFFFFF">
<center>
<br>
<br>
<h1>
¡El usuario
<%= request.getParameter("usuario") %>
ya está conectado!</h1>
<br>
<b>
Haga clic en el botón de abajo para iniciar sesión como otro usuario.
</b>
<form method=get action="InicioSesion2.jsp">
<input type=submit name="submit" value="OK">
</form>
</center>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/SolicitudInSes.jsp

```
<html>
<body>
<center>
<font size="+2">
<b>
<br>
<br>
<br>
¡¡¡NO SE HA REGISTRADO PARA INICIAR SESIÓN!!!
<br>
<br>
```

```
</font>
</center>
<font size="+1">
Si aún no registra el inicio de su sesión, debe hacerlo antes de que se le
permita accesar alguna de las páginas disponibles.
<p>
Si ya registró su inicio de sesión, probablemente está viendo este mensaje
porque no se ha detectado actividad de su explorador y se terminó su
sesión
por razones de seguridad.
<p>
Para registrar su inicio de sesión, haga clic en el vínculo etiquetado
<b><i>Inicio de sesión</i></b> en el menú de la izquierda.
</font>
</body>
</html>
```

/root/Desarrollo/spacsm/SPACSM_1/web/ValidarUsuario2.jsp

```
<jsp:useBean id="administradorSesion"
class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<jsp:setProperty name="credenciales" property="*"/>
<%! String paginaSig; %>
<%
  if (administradorSesion.yaRegistrado(credenciales)) {
      paginaSig = "SesionDuplicada2.jsp";
  }
  else {
       if (administradorSesion.estaSuspendido(credenciales))
              if (administradorSesion.inses(credenciales)) {
                   credenciales.setIdSesion(session.getId());
                   administradorSesion.iniciarRegistroSesion(credenciales);
                   paginaSig = "Bienvenido.jsp";
              }
              else {
              paginaSig = "FallaInSes2.jsp";
              }
       else
            paginaSig = "CuentaSuspendida.jsp";
  }
%>
<jsp:forward page="<%= paginaSig %>"/>
<%@ page errorPage="/comun/Excepcion.jsp" %>
```

Anexo D

Clases para visualizar la información por medio de una navegador WEB

/root/Desarrollo/spacsm/SPACSM_1/src/java/javaservlets/timer/TimerListener.java

```
/*
 * @(#)TimerListener
 *
 * Copyright (c) 1998 Karl Moss. All Rights Reserved.
 *
 * You may study, use, modify, and distribute this software for any
 * purpose provided that this copyright notice appears in all copies.
 *
 * This software is provided WITHOUT WARRANTY either expressed or
 * implied.
 *
 * @author  Karl Moss
 * @version 1.0
 * @date    11Mar98
 *
 */

package javaservlets.timer;

/**
  * <p>This interface is implemented by classes that wish to
  * receive timer events. The Timer class will invoke the
  * TimerEvent method for every time interval specified when
  * the Timer is started. This gives the implementing class
  * an opportunity to perform some type of time-dependent
  * checking.
  */

public interface TimerListener
{
  /**
    * <p>Called for each timer clock cycle
    */
  void TimerEvent(Object object);
}
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/com/instantjsp/AdministradorSesion.java

```
package com.instantjsp;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;

import javax.servlet.ServletContext;

import javaservlets.jdbc.ConnectionPool;

public class AdministradorSesion {

  private Hashtable sesionesActuales;
  private ConnectionPool connectionPool;

  private static final int MINUTOS_FIN = 5;
  private static final long ONE_MINUTE = 60000L;

  private static final String ARCHIVO_CFG_DEPOSITO =
    "javaservlets/jdbc/AdministradorSesionConnectionPool.cfg";

  private static final String SELECCIONAR_PASSWORD =
    "SELECT password FROM idpassword WHERE idusuario = ";

  private static final String SELECCIONAR_ESTADO_CUENTA =
    "SELECT COUNT(*) FROM idpassword WHERE status = 0 AND idusuario =
";

  private static final String ACTUALIZAR_ESTADO_CUENTA =
    "UPDATE idpassword SET status = 0 WHERE idusuario = ";

  private static final String SELECCIONAR_ESTADIS_SESION =
    "SELECT COUNT(*) FROM estadsesion WHERE idsesion = ";

  private static final String ACTUALIZAR_ESTADIS_SESION =
    "UPDATE estadsesion SET horafin = ";

  private static final String INSERTAR_ESTADISTICAS =
    "INSERT INTO estadpags" +
    "(idusuario, idsesion, fe, ho, descpag) VALUES (";

  private static final String HITS_MAS_CALIFICADOR =
    "hits = hits + 1 WHERE idsesion = ";

  private static final String RPAREN = ")";
  private static final String COMMA = ",";
  private static final String QUOTE = "'";

  private class PerfilInSes extends CredencialesUsuario{
    boolean estaActivo;
    int intentosInSes;
    boolean haIniciadoSes;
    Date loginTime;
    Date lastActivity;
    boolean estaViva;
    int minutosParaFin;

    public PerfilInSes(CredencialesUsuario credenciales) {
      setUsuario(credenciales.getUsuario());
      setPassword(credenciales.getPassword());
      estaActivo = true;
      haIniciadoSes = false;
      intentosInSes = 0;
      estaViva = false;
      minutosParaFin = MINUTOS_FIN;
    }
  }

  class MonitorInactividad implements Runnable {

    public void run() {
      while (true) {
        try {
          Thread.sleep(ONE_MINUTE);
          synchronized (sesionesActuales) {
            Enumeration keyEnumerator = sesionesActuales.keys();
            Vector expiredKeys = new Vector();
            while (keyEnumerator.hasMoreElements()) {
              String key = (String)keyEnumerator.nextElement();
              processElement(expiredKeys, key);
            }
            Enumeration expiredEnumerator = expiredKeys.elements();
            while (expiredEnumerator.hasMoreElements()) {
              sesionesActuales.remove(expiredEnumerator.nextElement());
            }
          }
        }
        catch (InterruptedException e) {
        }
      } // termina while
    }

    private void processElement(Vector expiredKeyTable, String key)       {
     PerfilInSes profile = (PerfilInSes)sesionesActuales.get(key);
     if (!profile.estaActivo) {
       //si (!profile.estaViva) {
       if (profile.minutosParaFin <= 0) {
         // marca esto para eliminarla
         expiredKeyTable.add(key);
       }
       else {
         // prepara para eliminarla en 5 minutos...
         //profile.estaViva = false;
         --profile.minutosParaFin;
         sesionesActuales.put(key,profile);
       }
     }
    }
  }

  public AdministradorSesion() throws Exception {
    connectionPool = new ConnectionPool();
    connectionPool.initialize(ARCHIVO_CFG_DEPOSITO);
    sesionesActuales = new Hashtable();
    MonitorInactividad monitor = new MonitorInactividad();
    Thread monitorThread = new Thread(monitor);
    monitorThread.start();
  }

  public boolean estaActivo(CredencialesUsuario credenciales) {
    boolean loggingIn = false;
    String usuario = credenciales.getUsuario();
    synchronized (sesionesActuales) {
      if (sesionesActuales.containsKey(usuario)) {
```

```
      PerfilInSes aPerfil =
        (PerfilInSes)sesionesActuales.get(usuario);
      loggingIn = aPerfil.estaActivo;
    }
  }
  return loggingIn;
}

public boolean yaRegistrado(CredencialesUsuario credenciales) {
  boolean registrado = false;
  String usuario = credenciales.getUsuario();
  synchronized (sesionesActuales) {
    if (sesionesActuales.containsKey(usuario)) {
      PerfilInSes aPerfil =
        (PerfilInSes)sesionesActuales.get(usuario);
      registrado = aPerfil.haIniciadoSes;
    }
  }
  return registrado;
}

public int getIntentosInSes(CredencialesUsuario credenciales) {
  int intentosInSes = 0;
  String usuario = credenciales.getUsuario();
  synchronized (sesionesActuales) {
    if (sesionesActuales.containsKey(usuario)) {
      PerfilInSes aPerfil =
        (PerfilInSes)sesionesActuales.get(usuario);
      intentosInSes = aPerfil.intentosInSes;
    }
  }
  return intentosInSes;
}

public void aumentarIntentosInSes(CredencialesUsuario credenciales) {
  String usuario = credenciales.getUsuario();
  synchronized (sesionesActuales) {
    if (sesionesActuales.containsKey(usuario)) {
      PerfilInSes aPerfil =
        (PerfilInSes)sesionesActuales.get(usuario);
      ++aPerfil.intentosInSes;
      sesionesActuales.put(usuario, aPerfil);
    }
  }
}

public boolean mantenerViva(CredencialesUsuario credenciales,
    String infoPagina) throws SQLException {
  if (!yaRegistrado(credenciales)) {
    return false;
  }
  actualizarEstadSesion(credenciales);
  String usuario = credenciales.getUsuario();
  synchronized (sesionesActuales) {
    if (sesionesActuales.containsKey(usuario)) {
      PerfilInSes aPerfil =
        (PerfilInSes)sesionesActuales.get(usuario);
      aPerfil.estaViva = true;
      aPerfil.minutosParaFin = MINUTOS_FIN;
      sesionesActuales.put(usuario, aPerfil);
    }
  }
  registrarEstad(credenciales, infoPagina);
  return true;
}

private int getCount(ResultSet rs) throws SQLException {
  rs.next();
  return rs.getInt(1);
}

public boolean estaSuspendido(CredencialesUsuario credenciales)
    throws SQLException {
  String usuario = credenciales.getUsuario();
  Connection conn = connectionPool.getConnection();
  Statement qs = conn.createStatement();
  ResultSet rs = qs.executeQuery(SELECCIONAR_ESTADO_CUENTA +
    QUOTE + usuario + QUOTE);

  boolean tf = (getCount(rs) < 1);
  System.out.println ("aqui enbtro en estasuspendido<" + tf + ">");
  System.out.flush();
  connectionPool.close(conn);
  return tf;
}

public void suspenderCuenta(CredencialesUsuario credenciales)
    throws SQLException {
  String usuario = credenciales.getUsuario();
  synchronized (sesionesActuales) {
    sesionesActuales.remove(usuario);
  }
  Connection conn = connectionPool.getConnection();
  Statement us = conn.createStatement();
  us.executeUpdate(ACTUALIZAR_ESTADO_CUENTA +
    QUOTE + usuario + QUOTE);
  connectionPool.close(conn);
}

public boolean inses(CredencialesUsuario credenciales)
    throws SQLException {
  if (yaRegistrado(credenciales)) {
    return false;
  }
  String usuario = credenciales.getUsuario();
  PerfilInSes profile;
  synchronized (sesionesActuales) {
    if (sesionesActuales.containsKey(usuario)) {
      profile = (PerfilInSes)sesionesActuales.get(usuario);
    }
    else {
      profile = new PerfilInSes(credenciales);
    }
  }
  Connection conn = connectionPool.getConnection();
  Statement qs = conn.createStatement();
  ResultSet rs =
    qs.executeQuery(SELECCIONAR_PASSWORD +
    QUOTE + usuario + QUOTE);
  while (rs.next()) {
    if (rs.getString(1).equals(credenciales.getPassword())) {
      profile.estaActivo = false;
      profile.haIniciadoSes = true;
      break;
    }
  }
  synchronized (sesionesActuales) {
    sesionesActuales.put(usuario, profile);
  }
  return profile.haIniciadoSes;
}

public void iniciarRegistroSesion(CredencialesUsuario credenciales)
    throws SQLException {
  String usuario = credenciales.getUsuario();
  String idSesion = credenciales.getIdSesion();
  Connection conn = connectionPool.getConnection();
  Statement qs = conn.createStatement();
  ResultSet rs = qs.executeQuery(SELECCIONAR_ESTADIS_SESION +
    QUOTE + idSesion + QUOTE + " AND idusuario = " +
    QUOTE + usuario + QUOTE);
  if (getCount(rs) == 0) {
    java.util.Date dt = new java.util.Date();
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
    String ft = sdf.format(dt);
    Statement us = conn.createStatement();
    us.executeUpdate("insert into estadsesion " +
      "(idsesion, idusuario, horainicio, horafin, hits) VALUES (" +
      QUOTE + idSesion + QUOTE + COMMA +
      QUOTE + usuario + QUOTE + COMMA +
      QUOTE + ft + QUOTE + COMMA +
      QUOTE + ft + QUOTE + COMMA +
      "0" + RPAREN);
  }
  connectionPool.close(conn);
}

public void registrarEstad(CredencialesUsuario credenciales, String
infoPagina)
    throws SQLException {
  String usuario = credenciales.getUsuario();
  String idSesion = credenciales.getIdSesion();
  java.util.Date dt = new java.util.Date();
  SimpleDateFormat sdfd = new SimpleDateFormat("yyyy-MM-dd");
  String fd = sdfd.format(dt);
  SimpleDateFormat sdft = new SimpleDateFormat("HH:mm:ss");
  String ft = sdft.format(dt);
  Connection conn = connectionPool.getConnection();
  Statement us = conn.createStatement();
  us.executeUpdate(INSERTAR_ESTADISTICAS +
    QUOTE +usuario + QUOTE + COMMA +
    QUOTE + idSesion + QUOTE + COMMA +
    QUOTE + fd + QUOTE + COMMA +
    QUOTE + ft + QUOTE + COMMA +
    QUOTE + infoPagina.replace("\"","`") + QUOTE + RPAREN);
  connectionPool.close(conn);
}

public void actualizarEstadSesion(CredencialesUsuario credenciales)
```

```
    throws SQLException {
   String usuario = credenciales.getUsuario();
   String idSesion = credenciales.getIdSesion();
   Connection conn = connectionPool.getConnection();
   Statement qs = conn.createStatement();
   ResultSet rs = qs.executeQuery(SELECCIONAR_ESTADIS_SESION +
     QUOTE + idSesion + QUOTE + " and idusuario = " +
     QUOTE + usuario + QUOTE);
   if (getCount(rs) > 0) {
     java.util.Date dt = new java.util.Date();
     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
     String ft = sdf.format(dt);
     Statement us = conn.createStatement();
     us.executeUpdate(ACTUALIZAR_ESTADIS_SESION +
       QUOTE + ft + QUOTE + COMMA +
       HITS_MAS_CALIFICADOR + QUOTE + idSesion + QUOTE);
   }
   connectionPool.close(conn);
 }

 public void salir(CredencialesUsuario credenciales) {
   synchronized (sesionesActuales) {
     sesionesActuales.remove(credenciales.getUsuario());
   }
 }
}
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/javaservlets/jdbc/AdministradorSesionConnectionPool.cfg

```
#ConnectionPool.cfg
JDBCDriver=com.mysql.jdbc.Driver
JDBCConnectionURL=jdbc:mysql://localhost/usuarios
ConnectionPoolSize=5
ConnectionPoolMax=100
ConnectionUseCount=10
ConnectionTimeout = 10
User=admin
Password=qwert
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/com/instantjsp/BeanAlmacenamiento.java

```
/*
 * BeanAlmacenamiento.java
 *
 * Created on March 6, 2006, 7:25 PM
 *
 * To change this template, choose Tools | Options and locate the template
under
 * the Source Creation and Management node. Right-click the template and
choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package com.instantjsp;


import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;



import javaservlets.jdbc.ConnectionPool;



/**
 *
 * @author root
 */
public class BeanAlmacenamiento {


  private ConnectionPool connectionPool;

  private static final String ARCHIVO_CFG_DEPOSITO =
    "javaservlets/jdbc/BeanAlmacenamientoConnectionPool.cfg";

  private static final String SELECCIONAR_PACIENTES =
    "SELECT * FROM Paciente ";



/*********
  private static final String SELECCIONAR_ESTADO_CUENTA =
    "SELECT COUNT(*) FROM idpassword WHERE status = 0 AND idusuario =
";

  private static final String ACTUALIZAR_ESTADO_CUENTA =
    "UPDATE idpassword SET status = 0 WHERE idusuario = ";

  private static final String SELECCIONAR_ESTADIS_SESION =
    "SELECT COUNT(*) FROM estadsesion WHERE idsesion = ";

  private static final String ACTUALIZAR_ESTADIS_SESION =
    "UPDATE estadsesion SET horafin = ";

  private static final String INSERTAR_ESTADISTICAS =
    "INSERT INTO estadpags" +
    "(idusuario, idsesion, fe, ho, descpag) VALUES (";

  private static final String HITS_MAS_CALIFICADOR =
    "hits = hits + 1 WHERE idsesion = ";
**/

  private static final String RPAREN = ")";
  private static final String COMMA = ",";
  private static final String QUOTE = "'";


    /** Creates a new instance of BeanAlmacenamiento */
    public BeanAlmacenamiento() throws Exception {
    connectionPool = new ConnectionPool();
    connectionPool.initialize(ARCHIVO_CFG_DEPOSITO);
    }

    /**
     *Regresa todos registros de paciente en un vector
     *
     */
    public Vector getPaciente() throws SQLException {
       Vector vect_out = new Vector();
       Vector fila = new Vector();
       Connection conn = connectionPool.getConnection();
       Statement us = conn.createStatement();
       ResultSet rs= us.executeQuery(SELECCIONAR_PACIENTES);

       while (rs.next()){
           fila.clear();
           fila.add(rs.getString(1));
           fila.add(rs.getString(2));
           fila.add(rs.getDate (3) );
           fila.add(rs.getString(4));
           fila.add(rs.getString(5));
           fila.add(rs.getString(6));
           vect_out.add(fila);
       }

       connectionPool.close(conn);

       return vect_out;

    }



}
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/javaservlets/jdbc/BeanAlmacenamientoConnectio

```
#ConnectionPool.cfg
JDBCDriver=com.mysql.jdbc.Driver
JDBCConnectionURL=jdbc:mysql://localhost/spacsm
ConnectionPoolSize=5
ConnectionPoolMax=100
ConnectionUseCount=10
ConnectionTimeout = 10
User=admin
Password=qwert
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/javaservlets/jdbc/ConnectionPool.java

```
/*
 * @(#)ConnectionPool
 *
 * Copyright (c) 1998 Karl Moss. All Rights Reserved.
 *
 * You may study, use, modify, and distribute this software for any
 * purpose provided that this copyright notice appears in all copies.
 *
 * This software is provided WITHOUT WARRANTY either expressed or
 * implied.
 *
 * @author  Karl Moss
 * @version 1.0
 * @date    11Mar98
 *
 */

package javaservlets.jdbc;

import java.sql.*;

/**
 * <p>This class serves as a JDBC connection repository. Since
 * creating database connections is one of the most time
 * intensive aspects of JDBC, we"ll create a pool of connections.
 * Each connection can be used and then replaced back into the
 * pool.
 *
 * <p>A properties file "ConnectionPool.cfg" will be used to
 * specify the types of JDBC connections to create, as well as
 * the minimum and maximum size of the pool. The format of
 * the configuration file is:
 *
 *   #(comment)
 *   JDBCDriver=<JDBC driver name>
 *   JDBCConnectionURL=<JDBC Connection URL>
 *   ConnectionPoolSize=<minimum size of the pool>
 *   ConnectionPoolMax=<maximum size of the pool, or -1 for none>
 *   ConnectionUseCount=<maximum usage count for one connection>
 *   ConnectionTimeout=<maximum idle lifetime (in minutes) of
 *       a connection>
 *   <other property for JDBC connection>=<value>
 *
 * <p>Any number of additional properties may be given (such
 * as username and password) as required by the JDBC driver.
 *
 */

public class ConnectionPool
  implements javaservlets.timer.TimerListener
{
  // JDBC Driver name
  String m_JDBCDriver;

  // JDBC Connection URL
  String m_JDBCConnectionURL;

  // Minimum size of the pool
  int m_ConnectionPoolSize;

  // Maximum size of the pool
  int m_ConnectionPoolMax;

  // Maximum number of uses for a single connection, or -1 for
  // none
  int m_ConnectionUseCount;

  // Maximum connection idle time (in minutes)
  int m_ConnectionTimeout;

  // Additional JDBC properties
  java.util.Properties m_JDBCProperties;

  // The Connection pool. This is a vector of ConnectionObject
  // objects
  java.util.Vector m_pool;

  // The maximum number of simultaneous connections as reported
  // by the JDBC driver
  int m_MaxConnections = -1;

  // Our Timer object
  javaservlets.timer.Timer m_timer;

  /**
   * <p>Initializes the ConnectionPool object using
   * "ConnectionPool.cfg" as the configuration file
   *
   * @return true if the ConnectionPool was initialized
   * properly
   */
  public boolean initialize() throws Exception
  {
    return initialize("javaservlets/jdbc/ConnectionPool.cfg");
  }

  /**
   * <p>Initializes the ConnectionPool object with the specified
   * configuration file
   *
   * @param config Configuration file name
   * @return true if the ConnectionPool was initialized
   * properly
   */
  public boolean initialize(String config) throws Exception
  {
    // Load the configuration parameters. Any leftover parameters
    // from the configuration file will be considered JDBC
    // connection attributes to be used to establish the
    // JDBC connections
    boolean rc = loadConfig(config);

    if (rc) {
      // Properties were loaded; attempt to create our pool
      // of connections
      createPool();

      // Start our timer so we can timeout connections. The
      // clock cycle will be 20 seconds
      m_timer = new javaservlets.timer.Timer(this, 20);
      m_timer.start();
    }
    return rc;
  }

  /**
   * <p>Destroys the pool and it"s contents. Closes any open
   * JDBC connections and frees all resources
   */
  public void destroy()
  {
    try {

      // Stop our timer thread
      if (m_timer != null) {
        m_timer.stop();
        m_timer = null;
      }

      // Clear our pool
      if (m_pool != null) {

        // Loop throught the pool and close each connection
        for (int i = 0; i < m_pool.size(); i++) {
          close((ConnectionObject) m_pool.elementAt(i));
        }
      }
      m_pool = null;
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }

  /**
   * <p>Gets an available JDBC Connection. Connections will be
   * created if necessary, up to the maximum number of connections
   * as specified in the configuration file.
   *
   * @return JDBC Connection, or null if the maximum
   * number of connections has been exceeded
   */
  public synchronized java.sql.Connection getConnection()
  {
    // If there is no pool it must have been destroyed
    if (m_pool == null) {
      return null;
    }

    java.sql.Connection con = null;
    ConnectionObject connectionObject = null;
    int poolSize = m_pool.size();

    // Get the next available connection
    for (int i = 0; i < poolSize; i++) {

      // Get the ConnectionObject from the pool
      ConnectionObject co = (ConnectionObject)
        m_pool.elementAt(i);
```

```
      // If this is a valid connection and it is not in use,
      // grab it
      if (co.isAvailable()) {
        connectionObject = co;
        break;
      }
    }

    // No more available connections. If we aren"t at the
    // maximum number of connections, create a new entry
    // in the pool
    if (connectionObject == null) {
      if ((m_ConnectionPoolMax < 0) ||
          ((m_ConnectionPoolMax > 0) &&
           (poolSize < m_ConnectionPoolMax))) {

        // Add a new connection.
        int i = addConnection();

        // If a new connection was created, use it
        if (i >= 0) {
          connectionObject = (ConnectionObject)
            m_pool.elementAt(i);
        }
      }
      else {
        trace("Maximum number of connections exceeded");
      }
    }

    // If we have a connection, set the last time accessed,
    // the use count, and the in use flag
    if (connectionObject != null) {
      connectionObject.inUse = true;
      connectionObject.useCount++;
      touch(connectionObject);
      con = connectionObject.con;
    }

    return con;
  }

/**
  * <p>Places the connection back into the connection pool,
  * or closes the connection if the maximum use count has
  * been reached
  *
  * @param Connection object to close
  */
public synchronized void close(java.sql.Connection con)
  {
    // Find the connection in the pool
    int index = find(con);

    if (index != -1) {
      ConnectionObject co = (ConnectionObject)
        m_pool.elementAt(index);

      // If the use count exceeds the max, remove it from
      // the pool.
      if ((m_ConnectionUseCount > 0) &&
          (co.useCount >= m_ConnectionUseCount)) {
        trace("Connection use count exceeded");
        removeFromPool(index);
      }
      else {
        // Clear the use count and reset the time last used
        touch(co);
        co.inUse = false;
      }
    }
  }

/**
  * <p>Prints the contents of the connection pool to the
  * standard output device
  */
public void printPool()
  {
    System.out.println("--ConnectionPool--");
    if (m_pool != null) {
      for (int i = 0; i < m_pool.size(); i++) {
        ConnectionObject co = (ConnectionObject)
          m_pool.elementAt(i);
        System.out.println("" + i + "=" + co);
      }
    }
  }

/**
  * <p>Removes the ConnectionObject from the pool at the
  * given index
  *
  * @param index Index into the pool vector
  */
private synchronized void removeFromPool(int index)
  {
    // Make sure the pool and index are valid
    if (m_pool != null) {

      if (index < m_pool.size()) {

        // Get the ConnectionObject and close the connection
        ConnectionObject co = (ConnectionObject)
          m_pool.elementAt(index);
        close(co);

        // Remove the element from the pool
        m_pool.removeElementAt(index);
      }
    }
  }

/**
  * <p>Closes the connection in the given ConnectionObject
  *
  * @param connectObject ConnectionObject
  */
private void close(ConnectionObject connectionObject)
  {
    if (connectionObject != null) {
      if (connectionObject.con != null) {
        try {

          // Close the connection
          connectionObject.con.close();
        }
        catch (Exception ex) {
          // Ignore any exceptions during close
        }

        // Clear the connection object reference
        connectionObject.con = null;
      }
    }
  }

/**
  * <p>Loads the given configuration file.  All global
  * properties (such as JDBCDriver) will be
  * read and removed from the properties list. Any leftover
  * properties will be returned. Returns null if the
  * properties could not be loaded
  *
  * @param name Configuration file name
  * @return true if the configuration file was loaded
  */
private boolean loadConfig(String name)
  throws Exception
  {
    boolean rc = false;

    // Get our class loader
    ClassLoader cl = getClass().getClassLoader();

    // Attempt to open an input stream to the configuration file.
    // The configuration file is considered to be a system
    // resource.
    java.io.InputStream in;

    if (cl != null) {
      in = cl.getResourceAsStream(name);
    }
    else {
      in = ClassLoader.getSystemResourceAsStream(name);
    }

    // If the input stream is null, then the configuration file
    // was not found
    if (in == null) {
      throw new Exception("ConnectionPool configuration file "" +
                          name + "" not found");
    }
    else {
      try {
        m_JDBCProperties = new java.util.Properties();

        // Load the configuration file into the properties table
        m_JDBCProperties.load(in);
```

```
      // Got the properties. Pull out the properties that we
      // are interested in
      m_JDBCDriver = consume(m_JDBCProperties, "JDBCDriver");
      m_JDBCConnectionURL = consume(m_JDBCProperties,
                                    "JDBCConnectionURL");
      m_ConnectionPoolSize = consumeInt(m_JDBCProperties,
                                        "ConnectionPoolSize");
      m_ConnectionPoolMax = consumeInt(m_JDBCProperties,
                                       "ConnectionPoolMax");
      m_ConnectionTimeout = consumeInt(m_JDBCProperties,
                                       "ConnectionTimeout");
      m_ConnectionUseCount = consumeInt(m_JDBCProperties,
                                        "ConnectionUseCount");
      rc = true;
    }
    finally {
      // Always close the input stream
      if (in != null) {
        try {
          in.close();
        }
        catch (Exception ex) {
        }
      }
    }
  }
  return rc;
}

/**
 * <p>Consumes the given property and returns the value.
 *
 * @param properties Properties table
 * @param key Key of the property to retrieve and remove from
 * the properties table
 * @return Value of the property, or null if not found
 */
private String consume(java.util.Properties p, String key)
 {
  String s = null;

  if ((p != null) &&
      (key != null)) {

    // Get the value of the key
    s = p.getProperty(key);

    // If found, remove it from the properties table
    if (s != null) {
      p.remove(key);
    }
  }
  return s;
}

/**
 * <p>Consumes the given property and returns the integer
 * value.
 *
 * @param properties Properties table
 * @param key Key of the property to retrieve and remove from
 * the properties table
 * @return Value of the property, or -1 if not found
 */
private int consumeInt(java.util.Properties p, String key)
 {
  int n = -1;

  // Get the String value
  String value = consume(p, key);

  // Got a value; convert to an integer
  if (value != null) {
    try {
      n = Integer.parseInt(value);
    }
    catch (Exception ex) {
    }
  }
  return n;
}

/**
 * <p>Creates the initial connection pool. A timer thread
 * is also created so that connection timeouts can be
 * handled.
 *
 * @return true if the pool was created
 */
private void createPool() throws Exception
{
  // Sanity check our properties
  if (m_JDBCDriver == null) {
    throw new Exception("JDBCDriver property not found");
  }
  if (m_JDBCConnectionURL == null) {
    throw new Exception("JDBCConnectionURL property not found");
  }
  if (m_ConnectionPoolSize < 0) {
    throw new Exception("ConnectionPoolSize property not found");
  }
  if (m_ConnectionPoolSize == 0) {
    throw new Exception("ConnectionPoolSize invalid");
  }
  if (m_ConnectionPoolMax < m_ConnectionPoolSize) {
    trace("WARNING - ConnectionPoolMax is invalid and will " +
          "be ignored");
    m_ConnectionPoolMax = -1;
  }
  if (m_ConnectionTimeout < 0) {
    // Set the default to 30 minutes
    m_ConnectionTimeout = 30;
  }

  // Dump the parameters we are going to use for the pool.
  // We don"t know what type of servlet environment we will
  // be running in - this may go to the console or it
  // may be redirected to a log file
  trace("JDBCDriver = " + m_JDBCDriver);
  trace("JDBCConnectionURL = " + m_JDBCConnectionURL);
  trace("ConnectionPoolSize = " + m_ConnectionPoolSize);
  trace("ConnectionPoolMax = " + m_ConnectionPoolMax);
  trace("ConnectionUseCount = " + m_ConnectionUseCount);
  trace("ConnectionTimeout = " + m_ConnectionTimeout +
        " seconds");

  // Also dump any additional JDBC properties
  java.util.Enumeration enum = m_JDBCProperties.keys();
  while (enum.hasMoreElements()) {
    String key = (String) enum.nextElement();
    String value = m_JDBCProperties.getProperty(key);
    trace("(JDBC Property) " + key + " = " + value);
  }

  // Attempt to create a new instance of the specified
  // JDBC driver. Well behaved drivers will register
  // themselves with the JDBC DriverManager when they
  // are instantiated
  trace("Registering " + m_JDBCDriver);
  java.sql.Driver d = (java.sql.Driver)
    Class.forName(m_JDBCDriver).newInstance();

  // Create the vector for the pool
  m_pool = new java.util.Vector();

  // Bring the pool to the minimum size
  fillPool(m_ConnectionPoolSize);
}

/**
 * <p>Adds a new connection to the pool
 *
 * @return Index of the new pool entry, or -1 if an
 * error has occurred
 */
private int addConnection()
 {
  int index = -1;

  try {
    // Calculate the new size of the pool
    int size = m_pool.size() + 1;

    // Create a new entry
    fillPool(size);

    // Set the index pointer to the new connection if one
    // was created
    if (size == m_pool.size()) {
      index = size - 1;
    }
  }
  catch (Exception ex) {
    ex.printStackTrace();
  }
  return index;
}

/**
 * <p>Brings the pool to the given size
```

```
  */
private synchronized void fillPool(int size) throws Exception
  {
    boolean useProperties = true;
    String userID = null;
    String password = null;

    // If the only properties present are the user id and
    // password, get the connection using them instead of
    // the properties object
    if (m_JDBCProperties != null) {

      // Make sure there are only 2 properties, and they are
      // the user id and password
      if (m_JDBCProperties.size() == 2) {
        userID =
          getPropertyIgnoreCase(m_JDBCProperties, "user");
        password =
          getPropertyIgnoreCase(m_JDBCProperties, "password");

        // If all we"ve got is a user id and password then
        // don"t use the properties
        if ((userID != null) && (password != null)) {
          useProperties = false;
        }
      }

    }

    // Loop while we need to create more connections
    while (m_pool.size() < size) {

      ConnectionObject co = new ConnectionObject();

      // Create the connection
      if (useProperties) {
        co.con = DriverManager.getConnection(m_JDBCConnectionURL,
                                             m_JDBCProperties);
      }
      else {
        co.con = DriverManager.getConnection(m_JDBCConnectionURL,
                                             userID, password);
      }

      // Do some sanity checking on the first connection in
      // the pool
      if (m_pool.size() == 0) {

        // Get the maximum number of simultaneous connections
        // as reported by the JDBC driver
        java.sql.DatabaseMetaData md = co.con.getMetaData();
        m_MaxConnections = md.getMaxConnections();
      }

      // Give a warning if the size of the pool will exceed
      // the maximum number of connections allowed by the
      // JDBC driver
      if ((m_MaxConnections > 0) &&
          (size > m_MaxConnections)) {
        trace("WARNING: Size of pool will exceed safe maximum of " +
              m_MaxConnections);
      }

      // Clear the in use flag
      co.inUse = false;

      // Set the last access time
      touch(co);

      m_pool.addElement(co);
    }
  }

/**
 * Gets a the named propery, ignoring case. Returns null if
 * not found
 * @param p The property set
 * @param name The property name
 * @return The value of the propery, or null if not found
 */
private String getPropertyIgnoreCase(java.util.Properties p,
                                     String name)
  {
    if ((p == null) || (name == null)) return null;

    String value = null;

    // Get an enumeration of the property names
    java.util.Enumeration enum = p.propertyNames();

    // Loop through the enum, looking for the given property name
    while (enum.hasMoreElements()) {
      String pName = (String) enum.nextElement();
      if (pName.equalsIgnoreCase(name)) {
        value = p.getProperty(pName);
        break;
      }
    }

    return value;
  }

/**
 * <p>Find the given connection in the pool
 *
 * @return Index into the pool, or -1 if not found
 */
private int find(java.sql.Connection con)
  {
    int index = -1;

    // Find the matching Connection in the pool
    if ((con != null) &&
        (m_pool != null)) {
      for (int i = 0; i < m_pool.size(); i++) {
        ConnectionObject co = (ConnectionObject)
          m_pool.elementAt(i);
        if (co.con == con) {
          index = i;
          break;
        }
      }
    }
    return index;
  }

/**
 * <p>Called by the timer each time a clock cycle expires.
 * This gives us the opportunity to timeout connections
 */
public synchronized void TimerEvent(Object object)
  {
    // No pool means no work
    if (m_pool == null) {
      return;
    }

    // Get the current time in milliseconds
    long now = System.currentTimeMillis();

    // Check for any expired connections and remove them
    for (int i = m_pool.size() - 1; i >= 0; i--) {
      ConnectionObject co = (ConnectionObject)
        m_pool.elementAt(i);

      // If the connection is not in use and it has not been
      // used recently, remove it
      if (!co.inUse) {
        if ((m_ConnectionTimeout > 0) &&
            (co.lastAccess +
             (m_ConnectionTimeout * 1000) < now)) {
          removeFromPool(i);
        }
      }
    }

    // Remove any connections that are no longer open
    for (int i = m_pool.size() - 1; i >= 0; i--) {
      ConnectionObject co = (ConnectionObject)
        m_pool.elementAt(i);
      try {
        // If the connection is closed, remove it from the pool
        if (co.con.isClosed()) {
          trace("Connection closed unexpectedly");
          removeFromPool(i);
        }
      }
      catch (Exception ex) {
      }
    }

    // Now ensure that the pool is still at it"s minimum size
    try {
      if (m_pool != null) {
        if (m_pool.size() < m_ConnectionPoolSize) {
          fillPool(m_ConnectionPoolSize);
        }
      }
    }
    catch (Exception ex) {
```

```
        ex.printStackTrace();
      }
    }

  /**
    * <p>Sets the last access time for the given ConnectionObject
    */
  private void touch(ConnectionObject co)
    {
      if (co != null) {
        co.lastAccess = System.currentTimeMillis();
      }
    }

  /**
    * <p>Trace the given string
    */
  private void trace(String s)
    {
      System.out.println("ConnectionPool: " + s);
    }
}

// This package-private class is used to represent a single
// connection object
class ConnectionObject
{
  // The JDBC Connection
  public java.sql.Connection con;

  // true if this connection is currently in use
  public boolean inUse;

  // The last time (in milliseconds) that this connection was used
  public long lastAccess;

  // The number of times this connection has been used
  public int useCount;

  /**
    * <p>Determine if the connection is available
    *
    * @return true if the connection can be used
    */
  public boolean isAvailable()
    {
      boolean available = false;

      try {

        // To be available, the connection cannot be in use
        // and must be open
        if (con != null) {
          if (!inUse &&
              !con.isClosed()) {
            available = true;
          }
        }
      }
      catch (Exception ex) {
      }

      return available;
    }

  /**
    * <p>Convert the object contents to a String
    */
  public String toString()
    {
      return "Connection=" + con + ",inUse=" + inUse +
        ",lastAccess=" + lastAccess + ",useCount=" + useCount;
    }
}
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/com/instantjsp/CredencialesUsuario.java

```
package com.instantjsp;

public class CredencialesUsuario {
  private String  usuario;
  private String  password;
  private String idSesion;

  public CredencialesUsuario( ) {
    usuario = "";
    password = "";
    idSesion = "";
  }

  public String getUsuario( ) {
    return usuario;
  }

  public void setUsuario( String usuario ) {
    this.usuario = usuario;
  }

  public String getPassword( ) {
    return password;
  }

  public void setPassword( String password ) {
    this.password = password;
  }

  public String getIdSesion() {
    return idSesion;
  }

  public void setIdSesion(String idSesion) {
    this.idSesion = idSesion;
  }
}
```

/root/Desarrollo/spacsm/SPACSM_1/src/java/javaservlets/timer/Timer.java

```
/*
 * @(#)Timer
 *
 * Copyright (c) 1998 Karl Moss. All Rights Reserved.
 *
 * You may study, use, modify, and distribute this software for any
 * purpose provided that this copyright notice appears in all copies.
 *
 * This software is provided WITHOUT WARRANTY either expressed or
 * implied.
 *
 * @author  Karl Moss
 * @version 1.0
 * @date    11Mar98
 *
 */

package javaservlets.timer;

/**
  * <p>This class implements a simple timer. Every time the
  * timer clock cycle that is specified expires the TimerEvent
  * method on the given TimerListener object will be invoked.
  * This gives the object a chance to perform some type of
  * timeout checking.
  */

public class Timer extends Thread
{
  // TimerListener to receive TimerEvent notifications
  TimerListener m_timerListener;

  // Number of seconds in each timer cycle
  int m_cycle;

  // Object to be supplied with the TimerEvent notification
  Object m_object;

  /**
    * <p>Constructs a new Timer object
    *
    * @param timerListener Object that will receive TimerEvent
    * notifications
    * @param cycle Number of seconds in each timer cycle
    */
  public Timer(TimerListener timerListener, int cycle)
    {
      m_timerListener = timerListener;
      m_cycle = cycle;
      m_object = null;

    }
  /**
    * <p>Constructs a new Timer object
    *
    * @param timerListener Object that will receive TimerEvent
    * notifications
    * @param cycle Number of seconds in each timer cycle
    * @param object Object to be supplied with the TimerEvent
```

```
    * notification
    */
  public Timer(TimerListener timerListener, int cycle,
               Object object)
    {
      m_timerListener = timerListener;
      m_cycle = cycle;
      m_object = object;
    }

  /**
    * <p>Runs the timer. The timer will run until stopped and
    * fire a TimerEvent notification every clock cycle
    */
  public void run()
    {
      // Loop until stopped
      while (true) {
        try {

          // Sleep for the clock cycle
          sleep(m_cycle * 1000);
        }
        catch (InterruptedException ex) {
        }

        // Fire a TimerEvent
        if (m_timerListener != null) {
          m_timerListener.TimerEvent(m_object);
        }
      }
    }
}
```