Universidad Politécnica de Valencia

Departamento de Sistemas Informáticos y Computación

Diseño y Desarrollo de una Arquitectura Software Genérica Orientada a Servicios para la Construcción de un Middleware Grid Orientado a la Gestión y Proceso Seguro de Información en formato DICOM sobre un Marco Ontológico.

Tesis Doctoral

Requisito para la Obtención del Grado de Doctor en Informática por la Universidad Politécnica de Valencia

Valencia, 15 de Enero de 2007

Autor:
J. Damian Segrelles Quilis

Directores:
Dr. D. Ignacio Blanquer Espert
Dr. D. Vicente Hernández García

Esta tesis se la dedico a mi mujer Mª Carmen, por su paciencia, comprensión y los ánimos que me ha dado de forma constante a lo largo de los años que ha durado el desarrollo de este trabajo, y porque gracias a ella he visto que hay vida mas allá del Grid.

Agradecimientos

Agradecer a mis padres y hermanos el apoyo incondicional recibido a lo largo de toda una vida, hasta llegar aquí.

Destacar por encima de todo el esfuerzo de Nacho Blanquer, al cual considero el abuelo biológico de esta tesis y de todos mis trabajos realizados en el grupo. Sin él, esta tesis no hubiera existido.

Destacar el esfuerzo de Erik Torres en el área concreta de esta tesis y por demostrarme que con mucho empeño y trabajo se puede conseguir cualquier objetivo.

A Carlos, Migue, Gabi, German y Eli, gracias a ellos estos años han sido inolvidables.

A los directores de tesis, tanto a Vicente Hernández como a Nacho Blanquer, por los consejos, apoyo, motivación y buen hacer.

A todos los compañeros del GRyCAP, tanto a los actuales como los que ya no están, por hacer que el ir a trabajar sea un placer.

También agradecer a AENA y autoridades aeroportuarias de todo el mundo, por los retrasos constantes en los vuelos, lo cual ha propiciado ganar tiempo en la corrección de esta tesis.

Diseño y Desarrollo de una Arquitectura Software Genérica Orientada a Servicios para la Construcción de un Middleware Grid Orientado a la Gestión y Proceso Seguro de Información en formato DICOM sobre un Marco Ontológico

Una de las áreas que más se ha beneficiado del soporte digital es la imagen médica, reforzada por la aparición del estándar Digital Imaging and Communication in Medicine (DICOM), evolucionado a lo largo de los años para soportar no sólo imágenes médicas sino otros tipos de información médica como videos, señales e incluso informes estructurados (DICOM-SR). Con la aparición de DICOM, se ha conseguido la integración de dispositivos de adquisición, visualización, impresión y almacenamiento de imágenes médicas de diferentes fabricantes, al ser este un estándar utilizado por todas las compañías proveedoras.

En la actualidad, los sistemas que trabajan con imagen médica digital, como PACS, RIS y HIS permiten integrar los datos a nivel departamental y hospitalario, existiendo soluciones comerciales muy efectivas. En estos sistemas, la seguridad de usuarios y datos se gestiona en un único dominio administrativo restringido. A consecuencia de su uso en producción en la práctica clínica, existe en la actualidad una gran cantidad de información en formato DICOM cuya utilización se restringe generalmente al tratamiento de los pacientes individuales. Sin embargo, la investigación médica requiere consolidar información multicéntrica para la extracción de patrones y la validación de técnicas y diagnósticos, realizándose esta actividad de forma manual y sin herramientas especiales.

En la presente tesis se plantea como objetivo general la definición de una Arquitectura Orientada a Servicios (SOA) y la implementación de un Middleware Grid basado en esta arquitectura, cuya principal función será la gestión, integración y proceso de información en formato DICOM almacenada de forma distribuida en diferentes dominios administrativos, de forma segura y estructurada semánticamente mediante la definición de ontologías médicas basadas en el informe estructurado y los estudios DICOM. Este middleware, proporciona a los desarrolladores un interfaz de alto nivel orientado a objetos que permite aumentar la productividad en el desarrollo de aplicaciones en diferentes ámbitos médicos.

Las aportaciones más destacables de esta tesis son las siguientes:

- Diseño de una Arquitectura Grid de propósito general para virtualizar el almacenamiento distribuido y proceso de datos en formato DICOM, basada en Web Services Resorce Framework (WSRF), frente a otras arquitecturas existentes que no utilizan estándares.
- Desarrollo de un modelo para la indexación y estructuración de datos DICOM basado en ontologías médicas obtenidas a partir de informes radiológicos, frente a los modelos convencionales basados en nombres identificadores y metadata básica.
- Diseño e implementación de un sistema de autorización que estructura los permisos de los miembros de las organizaciones virtuales a partir de ontologías médicas.
- Implementación de una plataforma y de los objetos de alto nivel necesarios, junto con diversas aplicaciones para la asistencia en la investigación en Diagnóstico por Imagen.

Disseny i Desenvolupament d'una Arquitectura Software Genèrica Orientada a Serveis per a la Construcció d'un Middleware Grid Orientat a la Gestió i Procés Segur d'informació en format DICOM sobre un Marc Ontològic

Una de les àrees que més s'ha beneficiat del suport digital és la imatge mèdica, reforçada per l'aparició de l'estàndard Digital Imaging and Comunication in Medicine (DICOM), que ha evolucionat al llarg dels anys fins a funcionar no sols amb imatges, sinó també amb altres tipus d'informació mèdica, com vídeos, senyals i fins i tot informes estructurats (DICOM-SR). Amb l'aparició de DICOM, s'ha aconseguit la integració de dispositius d'adquisició, visualització, impressió i emmagatzemament d'imatges mèdiques de diferents fabricants, ja que es tracta d'un estàndard utilitzat per totes les empreses proveïdores.

En l'actualitat, els sistemes que treballen amb imatge mèdica digital, com PACS, RIS i HIS, permeten integrar les dades en l'àmbit departamental i hospitalari, i hi ha solucions comercials molt efectives. En aquests sistemes, la seguretat d'usuaris i dades es gestiona en un únic domini administratiu restringit. A conseqüència del seu ús en producció en la pràctica clínica, en l'actualitat hi ha una gran quantitat d'informació en format DICOM la utilització de la qual es restringeix generalment al tractament dels pacients individuals. Per contra, la investigació mèdica requereix consolidar informació multicèntrica per a l'extracció de patrons i la validació de tècniques i diagnòstics, una activitat que es realitza de forma manual i sense eines especials.

En aquesta tesi es planteja com a objectiu general la definició d'una arquitectura orientada a serveis (SOA) i la implementació d'un Middleware Grid basat en aquesta arquitectura, que té com a principal funció la gestió, la integració i el processament d'informació en format DICOM emmagatzemada de forma distribuïda en diferents dominis administratius, de forma segura i estructurada semànticament mitjançant la definició d'ontologies mèdiques basades en l'informe estructurat i els estudis DICOM. Aquest programa intermediari (*middleware*) proporciona als desenvolupadors una interfície d'alt nivell orientada a objectes que permet augmentar la productivitat en el desenvolupament d'aplicacions en diferents àmbits mèdics.

Les aportacions més destacables en aquesta tesi són les següents:

- Disseny d'una arquitectura Grid de propòsit general per virtualitzar l'emmagatzemament distribuït i el processament de dades en format DICOM, basada en Web Services Resource Framework (WSRF), davant d'altres arquitectures existents que no utilitzen estàndards.
- Desenvolupament d'un model per a la indexació i l'estructuració de dades DICOM basades en ontologies mèdiques obtingudes a partir d'informes radiològics, davant models convencionals basats en noms identificadors i metadada bàsica.
- Disseny i implementació d'un sistema d'autorització que estructura els permisos dels membres de les organitzacions virtuals a partir d'ontologies mèdiques.
- Implementació d'una plataforma i dels objectes d'alt nivell necessaris, amb diverses aplicacions per a l'assistència en la investigació en diagnòstic per imatge.

Design and Development of a Service-Oriented Generic Software Architecture to Build up a Grid Middleware Oriented to Secure Management and Processing of DICOM Objects on a Ontolologic Framework

One of the areas that have taken profit most of the digital management of clinical data is medical imaging. The appearance of the Digital Imaging and Communication in Medicine standard (DICOM), has envolved throughout the years to support not only medical images but also other types of medical information like videos, signals and even structured reports (DICOM-SR). DICOM has enabled the acquisition, visualization, printing and storage devices for medical images of different manufacturers.

Today, systems working with medical digital image, such as PACS, RIS and HIS, integrate effectively the information at department and hospital level. In these systems, the security of users and information is managed only in a single administrative restricted domain. Moreover, a result of its use in production in the clinical practice is a huge quantity of information available in DICOM format, but it is used only for treatment of individual patients. Nevertheless, the medical research needs to consolidate multicentral information for the extraction of patterns and the validation of procedures and diagnostics, performing currently this activity manually and without special tools.

In this thesis, the main objective is the definition of a Service Oriented Architecture (SOA) and the implementation of a Grid Middleware based on this architecture, whose principal function will be the management, integration and process of DICOM information stored in a secure and distributed way in different administrative domains, and constructed semantically by means of the definition of medical ontologies based on the structured reports and DICOM studies. This middleware, provides the developers with a high-level object-oriented interface that increases the productivity in the development of applications in different medical areas.

The most important contributions of this thesis are the following:

- Design of a general Grid Architecture Grid to virtualize a distributed storage for the processing of DICOM information, based on Web Services Resource Framework (WSRF), extending the possibilities of other architectures that do not use standards.
- Development of a model for the indexation and structuring of DICOM information based on medical ontologies obtained from radiological reports, as an advance to the conventional models based on clinical keys and basic metadata.
- Design and implementation of an authorization system that organizes the permissions of the virtual organizations users using medical ontologies.
- Implementation of a platform and of the high-level objects required, along with diverse applications for the assistance in the research in image diagnosis.

Índice General

Introducción

Hoy en día, la gran mayoría de los centros médicos utilizan el formato digital para el almacenamiento y procesado de la mayor parte de la información con la que trabajan (datos demográficos, imágenes, señales, videos, informes etc.). Las ventajas que aportan los formatos digitales respecto a los tradicionales son numerosas. Una de las principales ventajas del soporte digital es la mejora de la capacidad de almacenamiento de datos, al ocupar menos espacio físico que el formato analógico convencional (como placas de radiografía, informes de diagnóstico en formato papel, cintas de video de ecografías, etc.). Por otra parte, los datos en formato digital pueden procesarse mediante computadores, complejos algoritmos y programas especializados, según el área o información que se quiera tratar. Otra de las ventajas a destacar, viene de las facilidades que nos proporcionan las tecnologías actuales de comunicación para la transferencia de información digitalizada a través de redes informáticas (LAN, WAN, Internet) y que permiten tener un acceso más fácil y organizado de los contenidos, facilitando la colaboración e intercambio de información entre diferentes entidades u organizaciones de manera segura.

La imagen médica es uno de los campos tradicionales en los que el formato digital cobra una especial importancia, la continua evolución y aparición de nuevas herramientas y algoritmos aportan nueva información y conocimiento de gran utilidad para la toma de decisiones en la realización de diagnósticos y el seguimiento de terapias en los pacientes. Los inicios de la utilización de imágenes digitalizadas fueron difíciles, principalmente por la falta de estandarización, siendo habitual que las empresas proveedoras de dispositivos de captación de imágenes utilizaran formatos digitales propietarios, impidiendo la comunicación de dispositivos de diferentes proveedores. Esto tenía como consecuencia la dificultad de ampliación e integración de nuevos sistemas, tanto a nivel departamental como a nivel de centro. A nivel departamental, los Sistemas de Información para Imagen Médica [1] (PACS) que se encargaban de la gestión de las imágenes tenían dificultades al insertar nuevos dispositivos de diferentes empresas (impresoras, workstations, dispositivos de captación de imágenes, etc…), dado que se hacía muy difícil su integración en el flujo de información del propio sistema, al no existir un estándar para almacenar y transmitir las imágenes. A nivel de centro, la carencia de estándares en los Sistemas de Información Radiológica [2] (RIS) o Sistema de Información Hospitalaria [3] (HIS), implicaba en muchos casos duplicar datos, al no poder comunicarse y compartir información entre los diferentes departamentos. Por tanto, en los inicios del uso de la imagen médica digital, era necesario adjudicar la gestión de todos los dispositivos y sistemas de información a un mismo proveedor para garantizar la integración de los mismos.

En la actualidad, este tipo de problemas se ve notablemente reducido al existir soluciones tecnológicas que los resuelven. Para facilitar la integración de diferentes dispositivos y sistemas que tratan imágenes médicas se utiliza el estándar Digital Imaging and Comunication in Medicine [4] (DICOM). DICOM, nacido en 1985, no sólo define el formato digital con el que se guardan las imágenes como objetos DICOM, sino también los protocolos de comunicación a utilizar y la información a intercambiar entre los diferentes dispositivos implicados. Actualmente, todas las empresas proveedoras de dispositivos relacionados con imagen médica soportan este estándar, que es utilizado por la mayoría de centros médicos en todo el mundo.

El estándar DICOM fue concebido para una serie de objetivos concretos, que por su uso generalizado se han ido ampliando a lo largo de los años. Si repasamos la historia del estándar, en sus inicios fue pensado para el intercambio y almacén únicamente de imágenes médicas e información relacionada con las áreas de radiología y cardiología. Actualmente, DICOM se esta ampliando a otros campos de la medicina. Esto ha obligado a realizar diversas ampliaciones en el estándar que permitan cubrir las nuevas necesidades que estas áreas demandan. Entre las ampliaciones más notorias, cabe destacar la incursión en el estándar para el tratamiento de señales y videos codificados en formato digital, y la de informes estructurados DICOM Structured Reporting [5] (DICOM-SR). Esta última ampliación, surge de la necesidad de poder codificar aquella información relativa al análisis de la información contenida en los objetos DICOM y que en el estándar no estaban inicialmente contemplados. Por todo ello, los centros médicos almacenan en la actualidad una gran cantidad de información en formato DICOM, que podría ser de muchísima utilidad para la realización de análisis y estudios epidemiológicos que permitan apoyar la toma de decisiones, la personalización de la terapia o su seguimiento. Disponer de una herramienta que permita compartir esta información distribuida de una manera segura, homogénea y colaborativa podría conducir a grandes beneficios.

El manejo estándar de toda la diversidad de información médica digitalizada, generada por distintos dispositivos, queda solucionado de una forma adecuada al utilizar el estándar DICOM. La manera en que se trata la información para su lectura, escritura y comunicación es independiente de sus características y contenido (imágenes 3D de Tomografía Axial Computerizada y Resonancia Magnética, imágenes 2D de Rayos X, películas de Cardiogramas, Ecografías, Informes Estructurados, etc.). Junto con DICOM, existen también otros estándares como el High Level Seven [8] (HL7) y normativas para la aplicación de estos, como las definidas por Integrating the Healthcare Enterprise [9] (IHE) orientadas a la integración de los diferentes sistemas existentes en un centro médico en general.

Por tanto, tal y como se ha comentado anteriormente, existen tecnologías que permiten integrar la información digitalizada mediante los protocolos definidos en DICOM y otros estándares como HL7 que pueden incluso aportar información adicional a la contenida en los propios objetos DICOM almacenados, al poder interactuar con los diferentes sistemas de los centros (PACS, RIS, HIS). Sin embargo, aparecen nuevos problemas al compartir esta información cuando se transciende el ámbito de centro o departamento, referentes entre otros a la confiabilidad de la información que reside en los sistemas implantados, tanto en términos de autenticidad (detectar modificaciones no autorizadas), como de confidencialidad (prevenir la exposición no autorizada de datos). La información DICOM, al estar digitalizada, reside únicamente en los computadores y por tanto la confiabilidad de los usuarios con respecto a ésta no es la misma que cuando se almacena de manera convencional. Por ejemplo, la percepción de la inmutabilidad de la información cuando se ven radiografías impresas o informes sobre papel es muy diferente frente a la información digitalizada. Por otra parte, la autenticidad y confidencialidad de la información no sólo se reduce a un problema ético o técnico, ya que los sistemas de información de salud se desarrollan en un marco legislativo muy estricto que obliga a garantizar la protección de los datos personales contra el procesamiento, y define las condiciones y reglas bajo las cuales el procesamiento es permitido. Existen muchas regulaciones a nivel europeo [10][11] y legislaciones adicionales implementadas en estados miembros [12].

Dentro de un sistema hospitalario, los sistemas de información actuales tienen soluciones para garantizar la autenticidad y privacidad de los datos, pues se desarrollan dentro de un mismo dominio administrativo y por tanto los datos están protegidos mediante los roles de usuario y permisos que se administran en el sistema, impidiendo o dando los permisos necesarios a los usuarios para realizar sus operaciones.

Por tanto, todos estos estándares y sistemas están bien definidos, son seguros y están implantándose en la actualidad con un buen grado de satisfacción por parte de los usuarios. Sin embargo, estos protocolos no satisfacen las necesidades que presenta la integración inter-hospitalaria o corporativa. Estas necesidades son susceptibles de abrir nuevas áreas de investigación en un ámbito colaborativo, para compartir recursos de almacenamiento y proceso de información en formato DICOM de manera segura en términos de autenticidad y confidencialidad de datos, mediante la definición y colaboración de diferentes organizaciones virtuales. Con la integración de toda la información a nivel inter-hospitalario o corporativo, la gestión de los datos debe cambiar, ya que la cantidad de información es mucho más elevada, con lo que la organización semántica de ésta cobra especial importancia para optimizar su manejo.

Hoy en día, las tecnologías Grid son utilizadas en otros ámbitos científicos para solucionar este tipo de problemas. El termino Grid se define, entre otras acepciones, como el conjunto de técnicas que permite compartir recursos distribuidos geográficamente de forma coordinada, entre las denominadas “organizaciones virtuales”, caracterizadas por ser dinámicas y multi institucionales, teniendo en cuenta que no sólo se comparten archivos o datos, sino también capacidad de proceso [39]. Por tanto, este trabajo plantea ofrecer una solución a las necesidades comentadas anteriormente, mediante la aplicación de las tecnologías Grid, pues estas tecnologías ofrecen soluciones efectivas para la creación de entornos de colaboración seguros y distribuidos.

Motivación

En la introducción se ha expuesto un breve resumen sobre la importancia de la información médica digitalizada y su uso en la actualidad, además de describir su evolución y destacar la importancia del estándar DICOM. Tal y como se ha descrito al final del apartado anterior, los protocolos y estándares actuales cubren satisfactoriamente las necesidades que plantean en la actualidad los centros médicos a nivel interno, aunque no resuelven los problemas que plantea compartir y procesar información a nivel inter-hospitalario o entre diferentes organizaciones reales. Toda la información existente en los diferentes almacenes digitales, puede tener un gran valor si se trata de forma adecuada en la investigación médica y clínica. Éste es el punto de partida y motivación de la presente tesis, de la cual derivarán diferentes necesidades y objetivos, propias de la naturaleza del problema que se plantea. Estas nuevas necesidades a las que responde esta tesis, están relacionadas con la seguridad, privacidad, organización semántica de los datos, gestión de recursos distribuidos, proceso de altas prestaciones y creación de aplicaciones, todo ello sobre datos digitalizados y almacenados en objetos DICOM.

Para abordar todas estas nuevas necesidades, en esta tesis se propone como primer objetivo definir una Arquitectura Orientada a Servicios (SOA) que permita crear aquellos servicios necesarios para compartir y procesar recursos que manejen objetos DICOM de manera segura y utilizando diferentes vistas semánticas, definidas sobre la información contenida en estos objetos. Desde un punto de vista técnico, la arquitectura SOA consiste en permitir la reutilización de la lógica y los datos existentes mediante interfaces estándar flexibles (por ejemplo, servicios Web o servicios Grid). Las organizaciones pueden crear nuevas aplicaciones combinando servicios y componentes de alto nivel reutilizables. Dichos componentes serán orientados a objetos y proporcionarán, a las aplicaciones que trabajan sobre información DICOM, los servicios que requieran y que estén desplegados en la infraestructura de la arquitectura. Existen en la actualidad diferentes definiciones de arquitecturas y nuevas aplicaciones, pero pocas plantean incluir la definición de componentes de alto nivel que faciliten la programación de aplicaciones.

La complejidad de la definición de una arquitectura de estas características es alta, especialmente si se pretende generalizarla para que sea adecuada para el mayor número de casos de uso correspondientes a diferentes áreas médicas o grupos de investigación posible. Homogeneizar el acceso a la información implica, primero, cubrir un amplio abanico de modalidades que, aunque soportadas por DICOM, requieren tratamientos diferentes. Además, tanto el soporte de almacenamiento de la información digital (PACS, RIS/HIS, Discos Duros etc.) como los gestores de las mismas (bases de datos relacionales, gestores de discos, servidores DICOM, etc.) depende de los centros y se ubican en dominios administrativos que también pueden ser diferentes. Por tanto, todos estos puntos (diferentes ubicaciones, gestores, modalidades y dominios administrativos) hacen difícil compartir y procesar la información DICOM de una forma transparente para los usuarios, dentro de un entorno colaborativo.

Otro punto a tener en cuenta, en la complejidad de definir la arquitectura, es la gran dimensión de los repositorios de información clínica, que implica una mayor dificultad a la hora de organizar el contenido. Cada grupo de investigación o área medica, tiene diferentes intereses y

requerimientos, necesitando acceder a aquella información que les sea útil. El exceso de información puede dificultar notablemente la usabilidad de los sistemas. Por ello también surge la necesidad de crear un marco semántico u ontología sobre las mismas bases de información en función del área médica o grupo de investigación con el que se trabaje, de forma que permita organizar la información. Dentro de una misma área médica o grupo de investigación estaría bien definir diferentes subconjuntos de información en función del experimento que se quiera realizar (por ejemplo, sobre una ontología apropiada para la investigación en oncología del Sistema Nervioso Central se podría realizar un estudio sobre neuroblastoma, de forma que haga accesibles únicamente la información relacionada con el neuroblastoma). En este caso también se trabaja sobre las mismas bases de datos pero se accede a diferentes niveles de información.

De la arquitectura a definir subyace la necesidad de contemplar el problema de la privacidad de los datos. El cumplimiento de la legislación actual, y la previsión de posibles cambios razonables, son unos de los principales problemas del tratamiento distribuido de la información médica, ya que la información se guarda en entornos administrativos diferentes. La información de los pacientes sólo debería poder ser leída por usuarios debidamente autorizados, impidiendo el acceso, incluso a los administradores de los recursos en donde se almacena la información, al ser esta confidencial. Para solucionar este problema, en la actualidad se utilizan técnicas de cifrado de forma que la información sólo puede ser descifrada por los servicios y usuarios autorizados para ello. Por tanto, este trabajo incorpora también un modelo de cifrado que se integra en los diferentes servicios definidos en la arquitectura.

Hoy en día, existe una gran diversidad de proyectos que pretenden compartir imágenes médicas de forma segura, aunque estos generalmente se centran en el desarrollo de soluciones particulares para problemas en determinadas áreas, sin abstraer aquellos aspectos que podrían generalizarse para cualquier tipo de aplicación que requiera de la gestión de un conjunto de datos médicos digitales, tanto para sus consultas, como para su almacenamiento o proceso. La arquitectura que se define en esta tesis pretende abarcar las necesidades de una manera genérica, sin centrarse en un tipo de aplicación o problema de un área determinada, además de permitir la definición de nuevos aspectos que sean particulares para cada caso.

Otra de las motivaciones, que se tienen en cuenta en la definición de la arquitectura, es la creación de componentes de alto nivel que actúen sobre los servicios que ofrece la arquitectura, con el objetivo de reducir el tiempo de desarrollo de nuevas aplicaciones, evitando que estas tengan que interactuar directamente con los servicios ofrecidos. De esta manera, se conseguirá una mayor productividad en el desarrollo, una mayor reutilización de componentes software de alto nivel y una mayor interoperabilidad.

Por tanto, y como resumen de lo que seria el objetivo general de esta tesis, se aborda la investigación referente a la especificación de una arquitectura SOA que permita definir e implementar los servicios necesarios para compartir y procesar recursos que manejan información DICOM de una manera segura y organizados mediante ontologías, así como la definición de componentes de alto nivel para el manejo de estos servicios, además de su implementación y despliegue en un entorno de producción real.

CAPÍTULO I. Antecedentes y Estado Actual

Tal y como se ha descrito en el apartado anterior, el objetivo del trabajo desarrollado en esta tesis es definir una arquitectura SOA que permita crear los servicios necesarios para compartir y procesar recursos que manejan objetos DICOM de manera segura, utilizando diferentes ontologías definidas sobre los mismos, implementando dicho entorno y llevándolo a producción. El objetivo de esta arquitectura es cubrir todas las necesidades expuestas en el apartado de motivación y solucionar los problemas que esto conlleva. El carácter interdisciplinar de esta tesis hace necesario el uso de tecnologías de ámbitos diferentes, tanto en la especificación de la arquitectura como en su implementación. Las principales áreas involucradas son las relativas a la informática médica y a los sistemas distribuidos y de altas prestaciones.

Por este motivo, se ha dividido este capítulo en siete secciones, según el área concreta con la que se relaciona, en los que se pretende dar una visión de aquellas tecnologías y conceptos en los que se basa la presente arquitectura, además de una justificación de las tecnologías que se van a utilizar como base para abordar el objetivo general.

En primer lugar se describen los diferentes Metalenguajes, al menos en los puntos mas importantes relacionados con las tareas a desarrollar en la tesis, que básicamente son: estructuras de datos para el intercambio de información entre los servicios a implementar en la arquitectura; definición de ontologías sobre bases de datos distribuidas de objetos DICOM y definición de plantillas (templates) de DICOM-SR, que actualmente se definen mediante ficheros de texto y no mediante lenguajes formales interpretables por los ordenadores. Además, en este bloque se expondrá una discusión de la conveniencia de uno u otro metalenguaje en función de la tarea para la que se pretende utilizar.

El segundo apartado es el relativo al estándar DICOM, posiblemente el concepto más importante junto con el concepto de Grid en esta tesis. En este apartado se da una explicación del estándar en general, así como la evolución de este a lo largo de los años, incidiendo en aquellos puntos en los que el estándar toma una especial relevancia, como la estructura de la información de los objetos DICOM y la parte referente a los informes estructurados (DICOM-SR).

En el tercer apartado se describen los sistemas de información actuales, que utilizan o interactúan con el estándar DICOM (PACS, RIS y HIS), mostrando en qué grado y a qué nivel (departamental u hospitalario) cubren las necesidades los sistemas actuales. Además también se dará un breve resumen de aquellos protocolos y estándares que conviven con el estándar

DICOM (IHE y HL7) en los entornos médicos, para así entender mejor el funcionamiento de los sistemas hospitalarios.

En el cuarto apartado se discuten las tecnologías Grid, describiendo las diferentes definiciones del concepto de Grid así como su evolución a lo largo de los años. Se expone también un estudio de las tecnologías Grid emergentes en la actualidad, así como una definición de lo que debe cumplir una arquitectura Grid, centrándose principalmente en Open Grid Service Architecture [36] (OGSA), pues será la especificación que se empleará en la tesis.

En el quinto y sexto apartado, se profundizará en aquellos puntos clave para la definición de una arquitectura Grid y que inciden directamente en las necesidades planteadas, además de una justificación de cuales serán las bases tecnológicas que se van a utilizar en esta tesis para abordarlos. Estos puntos clave son los sistemas de monitorización e información en un Grid y la seguridad en sus tres aspectos más importantes (autenticación y autorización de usuarios, sistemas de autorización en el Grid y modelos de cifrado de datos).

Por último, en el séptimo apartado se analizarán las soluciones propuestas en diferentes proyectos actuales, relacionados con los campos en los que se desarrolla la presente tesis, para así explicar las aportaciones fundamentales que ofrece el middleware Grid propuesto respecto a estos proyectos.

1.1. Metalenguajes

Un metalenguaje se define como un lenguaje natural o formal que se usa para explicar o hablar del lenguaje mismo o de una lengua, es decir, un lenguaje para la definición de otros lenguajes.

A pesar de la gran variedad de metalenguajes existentes en la actualidad, en este documento se destacan los más importantes por su relación con los puntos que se plantearán a continuación. Estos metalenguajes son: a) Standard Generalized Markup Language [13] (SGML); b) Extensible Markup Language [14] (XML); c) Resource Description Framework [15] (RDF); d) Web Ontology Language [16] (OWL). Todos estos metalenguajes tienen ventajas e inconvenientes, en función de para qué se quieran aplicar.

Los metalenguajes son una herramienta comúnmente utilizada para la estructuración de contenido en formato texto y su interpretación. Para la definición de la arquitectura que se propone, surgen una serie de puntos que necesitan de metalenguajes:

- Definición de estructuras de datos para el intercambio de información entre los servicios a definir en la arquitectura.
- Definición de ontologías sobre bases de datos distribuidas de objetos DICOM para la organización del contenido.
- Definición de plantillas para la creación de informes estructurados incluidos en objetos DICOM-SR.

Por tanto, en este apartado describiremos las diferentes alternativas y justificaremos la elección del metalenguaje a utilizar en cada caso. Para ello, primero se incluye una breve descripción de los metalenguajes citados (SGML, XML, RDF y OWL) para después enmarcarlos en los puntos de la tesis asociados.

1.1.1. Metalenguajes Estándar Actuales

1.1.1.1. El Standard Generalized Markup Language (SGML)

El Standard Generalized Markup Language [13] (SGML) es en un sistema semántico para la organización y etiquetado de documentos. La Organización Internacional de Estándares [17] (ISO) normalizó este lenguaje en 1986 aunque su desarrollo se realizó entre los años 60 y 70. La industria de la publicación de documentos constituye uno de los principales usuarios del lenguaje SGML. Empleando este lenguaje, se crean y mantienen documentos que luego son llevados a otros formatos finales como HTML, Postscript, o RTF. SGML fue diseñado para permitir el intercambio de información entre distintas plataformas, soportes físicos, lógicos y diferentes sistemas de almacenamiento y presentación (bases de datos, edición electrónica, etc.), con independencia de su grado de complejidad.

En términos generales, el SGML utiliza etiquetas para dar formato a los documentos, es decir, SGML permite especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial. Los documentos que se especifican mediante los lenguajes definidos por SGML separan la parte referente a la estructura de documento de la presentación y

la semántica del contenido del mismo, lo cual supone un gran beneficio a la hora de manipular la información contenida en los documentos.

Un ejemplo claro de lenguaje generado a partir de SGML es el HyperText Markup Language [18] (HTML):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <TITLE>Mi primer documento HTML</TITLE>
  </HEAD>
  <BODY>
    <P>¡Hola mundo!</P>
  </BODY>
</HTML>
```

Un documento HTML se divide en una sección de cabecera (entre <HEAD> y </HEAD>) y un cuerpo (entre <BODY> y </BODY>). El título del documento aparece en la cabecera (junto con otras informaciones sobre el documento), y el contenido del documento aparece en el cuerpo. El cuerpo de este ejemplo contiene únicamente un párrafo, codificado o marcado como <P>.

Cada lenguaje de formato de documentos definido con SGML se denomina aplicación SGML. Una aplicación SGML se caracteriza generalmente por:

- La Declaración SGML. Especifica qué caracteres y delimitadores pueden aparecer en la aplicación.
- El Document Type Definition [31] (DTD). El DTD define la sintaxis de las estructuras de formato. Los DTD describen la estructura de los nuevos documentos.
- La Especificación Semántica. Una especificación que describe la semántica que se debe conferir al código de formato. Esta especificación también impone restricciones de sintaxis que no pueden expresarse dentro del DTD.
- Los Documentos. Elementos que contienen datos (contenido) y código (etiquetas). Cada documento contiene una referencia al DTD que debe usarse para interpretarlo.

Por el contrario, aunque SGML es un metalenguaje muy potente y general a la hora de definir un lenguaje basado en etiquetas, también tiene como uno de los principales problemas el que su definición es demasiado extensa y su uso puede generar lenguajes incompletos en su definición como sucede con el HTML.

1.1.1.2. Extensible Markup Language (XML)

Como resultado de la complejidad del SGML, el Extensible Markup Language [14] (XML) nació como un nuevo lenguaje más restrictivo y basado en SGML. El XML fue publicado como una recomendación por el World Wide Web Consortium [19] (W3C) en 1998.

Dependiendo de los requerimientos de uso, las diferencias entre XML y SGML pueden ser sutiles o inmensas. SGML es más adaptable que XML, es decir, más flexible y potente. Por el contrario, el SGML es mucho más costoso de implementar. Las diferencias entre XML y SGML vienen recogidas por W3C en [20].

XML es un lenguaje que proporciona un conjunto de reglas para la creación de etiquetas que dotan de estructura a los datos y documentos, por ejemplo en la Web. Estas reglas definen

claramente la sintaxis de los documentos a crear. XML es una sintaxis base para documentos potentes y flexibles, pero no impone restricciones semánticas al significado de esos documentos, siendo esta, posiblemente, una de las diferencias mas grandes con respecto al SGML.

De XML cabe destacar las siguientes características:

- Es un lenguaje independiente de la plataforma sobre la que se trabaje. Es UNICODE [21], es decir, asigna un identificador único a cada carácter, lo que hace que pueda ser utilizado en múltiples lenguajes.
- Es independiente de la información con respecto a la representación.
- Los cambios en el documento no suponen un problema para su interpretación, ya que siempre hay que leer el DTD asociado para interpretarlo.
- XML sigue un estándar recogido por W3C.
- Los datos sólo dependen de los datos en sí y no de su formato. Existen tecnologías asociadas para validación y definición de formatos y estructura de datos (Esquemas XML).

XML, al igual que el SGML, define la sintaxis para la creación de nuevos lenguajes basado en etiquetas. XML ha provocado la creación de nuevas tecnologías de apoyo a este lenguaje, debido a su sencillez, claridad en su especificación y potencia de uso. Todas estas tecnologías quedan integradas dentro de W3C, lo cual supone una de las principales ventajas con respecto al SGML. Las tecnologías más conocidas y utilizadas relacionadas con XML son las siguientes:

- eXtensible StyleSheet Language [22] (XSL). Define el estándar para las hojas de estilo de XML. Es la ampliación y modificación de las Cascading Style Sheets [23] (CSS). Esta tecnología está enfocada a dar presentación al contenido de los conceptos.
- XML Schemas [24]. Ayuda a los desarrolladores a definir estructuras precisas basadas en XML. Esta tecnología está enfocada a dar estructura a los documentos y tipos de datos de la información contenida, permitiendo validar los documentos una vez creados.
- eXtensible StyleSheet Language Transformations [25] (XSLT). Es un lenguaje de transformación que se usa para ordenar, añadir y eliminar etiquetas y atributos.
- XML Linking Language [26] (XLink). Define la forma estándar de añadir enlaces dentro de un documento XML.
- XML Pointer Language [27] (XPointer). Define cómo poder hacer referencia a partes dentro del documento XML. Es simular al concepto de URL, pero haciendo referencia a partes dentro del documento XML.

Todas estas tecnologías estandarizadas hacen del lenguaje XML una herramienta muy potente, lo que unido a su aceptación por los comités de estandarización hace que en la actualidad lo soporten multitud de herramientas, por ejemplo el XML Spy [28], destacándose por ello frente al SGML.

1.1.1.3. Resource Description Framework (RDF)

El Resource Description Framework [15] (RDF), desarrollado por el W3C, es una infraestructura que permite la codificación, intercambio y reutilización de estructuras de documentos (metadatos). De alguna manera, RDF es similar a SGML y XML a la hora de

definir la estructura de documentos. RDF utiliza XML como sintaxis para el intercambio y proceso de documentos, lo cual permite una integración plena con los documentos generados mediante XML. Para explotar las características de XML, RDF define estructuras que permiten definir expresiones no ambiguas que posibilitan la codificación consistente, el intercambio y proceso de metadatos estándar.

El RDF es una infraestructura para la descripción de recursos y constituye un estándar que permite descripciones sencillas. XML es a la sintaxis lo que RDF a la semántica, es decir, un conjunto claro de reglas para proporcionar información descriptiva sencilla. El esquema RDF proporciona un modo de combinar esas descripciones en un vocabulario único.

En la mayoría de los casos, cuando se intercambia información a través de la Web, se utiliza en la actualidad XML como sintaxis de intercambio para los documentos. Las especificaciones RDF se utilizan para proporcionar una infraestructura para el intercambio de conocimiento o de las reglas semánticas definidas para el documento a intercambiar. Como se ha comentado en el apartado anterior, XML no permite la inclusión de información semántica en los documentos, con lo que el uso de RDF puede resultar muy útil en los casos en los que se requiera de este tipo de información, también codificada de acuerdo con la sintaxis del XML. Por tanto, se puede decir que RDF es un avance significativo en el potencial de la Web, ya que permite el intercambio de información semántica.

Las especificaciones RDF se han convertido en una base práctica y matemáticamente precisa sobre la que se puede construir otros lenguajes, tales como Ontology Web Lenguaje [16] (OWL) y el resto de la Web Semántica, permitiendo un intercambio semántico de información mucho mas potente.

1.1.1.4. Ontology Web Language (OWL)

El Ontology Web Language [16] (OWL) esta diseñado para el uso de aplicaciones que necesitan procesar el contenido de información en lugar de presentar dicha información. OWL facilita mejor la interpretabilidad del contenido Web que el lenguaje XML y RDF, al proporcionar un vocabulario adicional junto con la información semántica.

Lo que no abarca XML ni RDF es la posibilidad de desarrollar vocabularios para un dominio específico, lo que viene a constituir una ontología completa en todos los sentidos. Una ontología define los términos utilizados para describir y representar un área de conocimiento. Las ontologías son utilizadas por personas, bases de datos, y aplicaciones que necesitan compartir información específica sobre un determinado asunto (o dominio), como las neurociencias, laboratorio, genética etc. Las ontologías incluyen definiciones utilizables por máquinas sobre conceptos básicos del dominio y de las relaciones existentes entre los mismos. Codifican conocimiento en un dominio y también conocimiento que se expande a través de varios dominios. De este modo, hacen que este conocimiento sea reutilizable.

OWL, proporciona un lenguaje para la definición de ontologías estructuradas, basadas en la Web, que ofrece una integración e interoperabilidad de datos más rica entre comunidades descriptivas. Los lenguajes anteriores se utilizaron para desarrollar herramientas y ontologías para comunidades de usuarios específicas (particularmente en las ciencias y en aplicaciones de comercio electrónico de compañías específicas), pero no fueron definidos para ser compatibles con la arquitectura de la World Wide Web en general, y de la Web Semántica en particular.

OWL se construye sobre RDF y añade más vocabulario para la descripción de clases y propiedades que se requieren para la definición de la información semántica. Entre las características de OWL respecto a XML/RDF destacan: a) Relaciones entre clases (p.e. inconexión); b) cardinalidad (p.e. "exactamente uno"); c) mayor riqueza de tipos en las propiedades; d) Características de propiedades (p.e. simetría); y e) Clases enumeradas.

OWL supone un gran paso adelante en la representación y organización de conocimiento en la World Wide Web.

1.1.2. Aplicación de los Metalenguajes

Como se ha podido observar, todos los metalenguajes descritos (SGML, XML, RDF y OWL) poseen muchas características comunes y también muchas características que los diferencian de manera significativa. Por tanto, para las diferentes tareas a realizar en esta tesis que requieren del uso de Metalenguajes, se utilizará en cada caso el más adecuado. Por ello en los apartados siguientes se realiza una breve discusión sobre qué aporta cada metalenguaje en función de la tarea para la que se quiere utilizar. Recordemos que básicamente eran tres las partes para las que los metalenguajes se requerían: a) estructuras de datos para el intercambio de datos entre los servicios a implementar en la arquitectura; b) la definición de ontologías sobre bases de datos distribuidos de objetos DICOM; c) y plantillas (templates) de DICOM-SR, que actualmente se definen mediante ficheros de texto y no mediante lenguajes formales interpretables por los ordenadores.

1.1.2.1. Definición de Estructuras de Datos

La definición de una arquitectura que ofrece diferentes servicios con los que se intercambia información, y en la que cada servicio corresponde con un recurso a compartir, requiere definir esta información mediante estructuras de datos bien formadas, porque los datos que intercambia pueden estar guardados en recursos de diferente naturaleza aunque se deban de tratar e interpretar de la misma manera (p.e. si hablamos de almacenes de objetos DICOM, éstos pueden ser Bases de Datos Relacionales, directorios de fichero, sistemas PACS, etc.). Es fundamental que estas estructuras de datos sean independientes de la plataforma donde se procesan o guardan, pues los datos quedan enmarcados dentro de los servicios definidos en la arquitectura, con lo que pueden ser de naturaleza diversa aunque representen una misma estructura o funcionalidad. Por ello es importante tener las herramientas adecuadas que permitan validar estas estructuras de una forma eficaz y robusta.

Repasando los metalenguajes mencionados en los apartados anteriores, los que más se pueden encuadrar dentro de este ámbito son SGML y XML. Tal y como se ha comentado, estos lenguajes pueden ser muy diferentes o muy iguales en función de la aplicación que se les quiera dar. En este caso concreto se podría decir que para la definición de las estructuras de los datos que tratan, estos lenguajes pueden resultar prácticamente idénticos, de manera que son capaces de suplir las necesidades de definir estructuras de datos de manera efectiva. Ambos lenguajes fueron diseñados para intercambiar información entre diferentes plataformas, y además permiten definir estructuras de datos separando la presentación, la estructura y el contenido. En el caso de ésta tesis, interesa la estructura de los datos, su validación y su contenido para poder transmitirlos. Cabe decir que SGML tiene mayor flexibilidad que XML, pero puede resultar

muy costoso por la extensa y compleja definición de este lenguaje. Por el contrario XML es un lenguaje más reducido, fácil de usar y tiene una mayor implementación en el ámbito Web. Además tiene la tecnología específica desarrollada de apoyo de esquemas XML para la definición y validación de estructuras de datos, que permiten validar los documentos de forma sencilla y en la que, actualmente, existe una gran cantidad de herramientas con las que se pueden trabajar para realizar desarrollos.

Por lo tanto, en la parte referente a la definición de estructuras de datos para el intercambio de información con los servicios de la arquitectura, se utilizarán esquemas XML y los datos serán transmitidos como documentos XML que deberán ser validados mediante estos esquemas.

1.1.2.2. Definición de Ontologías sobre Objetos DICOM

El segundo aspecto referente al uso de metalenguajes es el relativo a la anotación semántica de los datos y su organización. La organización semántica se plantea como una necesidad debido a la gran dimensión de los repositorios de información clínica que existen en la actualidad, lo cual provoca una mayor dificultad a la hora de organizar el contenido por parte de los grupos de investigación o personal de diferentes áreas medicas. Por ello se requiere estructurar y definir las mismas bases de información para poder clasificarlos, permitiendo compartir, en función del área médica o grupo de investigación con el que se quiera trabajar, aquellos objetos DICOM existentes en diferentes almacenes físicos, gestionándolos como un único almacén virtual, que satisfaga los requerimientos de una especificación previamente definida sobre la información clínica y médica que almacenan.

En este ámbito, nace el concepto de ontología. Aunque el concepto de ontología ha estado presente desde hace mucho tiempo en la filosofía, recientemente se utiliza en informática para definir vocabularios que las máquinas puedan entender, y que sean especificados con tanta precisión como para permitir diferenciar términos y referenciarlos de una manera precisa.

Una definición más exacta en términos generales del concepto de ontología es la que propone W3C y es la siguiente: Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento). Las ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son interpretables por los ordenadores. Codifican el conocimiento de un dominio y también el conocimiento que extienden los dominios. En este sentido, hacen el conocimiento reutilizable.

Podríamos decir que el estándar DICOM define una ontología respecto a la información contenida respecto al objeto que contienen, pues claramente define la codificación de términos junto con su significado. Igualmente ocurre con las plantillas definidas en estándar DICOM para la creación de informes estructurados, pues también definen claramente el significado de los campos además de proporcionar codificaciones de los conceptos a expresar. Por otra parte, los propios DICOM-SR, utilizan otras ontologías y terminologías (SNOMED, LOINC, CIE9). En todos estos casos, las ontologías vienen predefinidas por los propios estándares a utilizar.

Además de las ontologías descritas, también se debe definir en éste trabajo una ontología para la compartición de información de los diferentes sistemas distribuidos y en la que se deben incluir tres niveles. Como primer nivel se definirán aquellos campos susceptibles de ser

utilizados, que queramos referenciar en los objetos DICOM que corresponden a un área determinada (neurología, oncología abdominal, etc.). Como segundo nivel, la ontología debe definir aquellos campos a indexar a la hora de crear los almacenes virtuales de un área determinada (modalidad, sexo, patología en el informe estructurado, etc.). Finalmente, como tercer nivel se debe definir aquellos campos por los cuales se puede filtrar en función del experimento que se pretenda realizar (grupos de edades, tiempo de tratamiento, etc.).

Por tanto, realmente lo que se necesita es un lenguaje que sea capaz de definir ontologías que únicamente referencien los campos correspondientes a los objetos DICOM en los diferentes niveles ontológicos sin considerar relaciones entre los campos.

Por lo tanto, todos los lenguajes mencionados anteriormente son suficientes para definir las ontologías. SGML es quizás la herramienta más genérica y potente, pero también es la que comporta una dificultad de uso mayor. RDF se definió para complementar el XML e introducir información semántica sencilla, con lo que no nos aporta nada en este caso concreto, aunque es una herramienta a tener en cuenta si en un futuro se definieran ontologías más complejas en las que se introdujera información semántica más compleja sobre los objetos DICOM. OWL es el lenguaje por excelencia de la Web para definir Ontologías, mucho más potente que XML y RDF, quizá sea el lenguaje ideal para la definición de ontologías completas y complejas, pero tal y como hemos visto necesitamos definir ontologías sencillas, con lo que OWL resultaría ser excesivo, aunque también es una herramienta a tomar en cuenta para posibles ampliaciones futuras en las definiciones de las ontologías mucho mas complejas y con una semántica mas fuerte. Por tanto, se propone de nuevo XML como metalenguaje para la definición de ontologías sobre los objetos DICOM. XML permite definir documentos con las referencias de los campos DICOM relevantes para una ontología, pudiendo ser validados mediante esquemas XML.

Por tanto, en el caso de definición de ontologías utilizamos documentos XML que son validados contra el esquema XML que define su estructura.

1.1.2.3. Definición de plantillas para DICOM-SR

Los componentes del Middleware Grid a desarrollar permitirán crear objetos DICOM que representen documentos estructurados según el estándar (DICOM-SR). Los DICOM-SR definen una serie de plantillas que contienen los campos y códigos adecuados para cada informe en cada patología, modalidad o protocolo. En la actualidad no se utilizan plantillas estructuradas, sino directamente la información se codifica en texto plano, lo que dificulta su proceso por computador. Por tanto, no existe ningún mecanismo mediante el cual pueda validarse de una forma eficiente y automática la corrección de un DICOM-SR con respecto a su plantilla de referencia. Existen grupos de trabajo que están migrando los objetos DICOM a documentos XML de una manera formalizada, debido a la potencia que ofrece XML y las numerosas herramientas y tecnologías asociadas que existen. Con respecto a la validación de las plantillas de los DICOM-SR, también se está trabajando en la línea de formalizar estos esquemas mediante la tecnología de esquemas XML [29], en estos trabajos se presentan las limitaciones que suponen los DTD [31] para este tipo de formalización, proponiendo soluciones a estas mediante los esquemas XML.

Por ello ésta tesis apuesta por el uso de XML como el lenguaje para representar las plantillas de DICOM-SR y los esquemas XML para su validación.

1.2. Estándar DICOM

El formato Digital Imaging and Communications in Medicine [4] (DICOM) es el estándar de facto para la codificación de la información radiológica, que constituye el tipo de información objetivo de ésta tesis. En éste apartado se describen aquellos puntos del estándar que tienen mayor relevancia en los desarrollos e implementaciones realizadas para abordar la definición e implementación de la arquitectura genérica SOA de éste trabajo.

Primero se mostrará una breve introducción de los orígenes del estándar y un breve repaso de las diferentes versiones que se han ido sucediendo a lo largo del tiempo, para después mostrar aquellas ampliaciones que se consideran más importantes. A continuación se describe las características más importantes de DICOM en la actualidad, así como la estructura de la información incluida en los objetos DICOM, para así poder entender como la información DICOM puede ser mostrada, insertada, modificada, consultada y referenciada. Finalmente se incidirá en la parte referente a los informes estructurados.

1.2.1. Evolución del Estándar

El estándar DICOM, además de constituir un formato para la codificación de la imagen médica, es un protocolo que ha sido adoptado mundialmente para la comunicación entre entidades que generan o manejan imágenes médicas digitales.

Con la introducción en los años 70 de las imágenes tomográficas, como la Tomografía Axial Computerizada (TAC) o la Resonancia Magnética (RM), y el incremento del uso de ordenadores en aplicaciones clínicas, el American College of Radiology (ACR) y la National Electrical Manufactures Association (NEMA) recogieron las necesidades que permiten definir un método estándar para la transferencia de imágenes y la información asociada entre dispositivos de diferentes fabricantes que utilizaban diferentes formatos de imagen digital. Finalmente, ACR y NEMA formaron un comité mixto en 1983 para desarrollar el estándar que cumpliera con los siguientes requisitos:

- Impulsar la comunicación de la información referente a la imagen digital entre dispositivos, independientemente de las empresas proveedoras de dispositivos captadores de imágenes.
- Facilitar el desarrollo y expansión de sistemas PACS y que puedan también ser un interfaz con otros sistemas de información hospitalaria (HIS/RIS).
- Permitir la creación de bases de datos de información diagnóstica que puedan ser interrogadas con una gran variedad de dispositivos distribuidos.

La publicación del estándar ACR-NEMA No. 300-1985 constituyó la versión 1.0. A este estándar le siguieron dos revisiones: La No. 1, fechada en Octubre de 1986 y la No. 2, fechada en Junio de 1988. El estándar ACR-NEMA No. 300-1988, la Versión 2.0, abarcaba la versión 1.0 además de incluir algunas revisiones y nuevo material para dar soporte a dispositivos de visualización, introducir un nuevo esquema jerárquico para identificar cada imagen y poder agregar nuevos elementos para permitir una mayor descripción de las imágenes. Todos estos estándares especificaban además, un interface hardware, un mínimo conjunto de comandos para interactuar con los dispositivos y un conjunto consistente de formatos de datos.

Posteriormente ACR y NEMA desarrollaron DICOM 3.0 (versión de 1996) y DICOM 3.0 (versión de 1999) absorbiendo a todos sus predecesores ACR-NEMA 2.0 (1988) y ACR-NEMA 1.0 (1985). Curiosamente DICOM 3.0, a pesar de ser revisado y ampliado en sucesivas ocasiones desde 1996, sigue siendo la versión 3.0.

En la actualidad, DICOM es el estándar utilizado por la mayoría de centros médicos, en las áreas de radiología y cardiología para el manejo de imágenes médicas digitales, aunque también se está introduciendo en nuevas áreas, lo cual ha provocado sucesivas ampliaciones. Actualmente DICOM soporta prácticamente cualquier modalidad de imagen médica digital, como TACs, RMs, Tomografía por Emisión de Positrones (PET), Medicina Nuclear, Ultrasonidos, Rayos X, Radiografías Digitalizadas, Videos Digitalizados, información HIS/RIS etc…

Una de las últimas actualizaciones del estándar DICOM es la incorporación en el año 2000 de documentos estructurados, DICOM Structured Reporting [5] (DICOM-SR), que merece un capítulo aparte debido a su importancia dentro de esta tesis.

Cabe destacar que en esta tesis solo se tendrá en cuenta la parte estática del estándar DICOM, es decir, aquella en la que tengamos un objeto DICOM autocontenido (una imagen digitalizada junto con la información asociada a ésta, o un informe estructurado). La parte de protocolos de comunicación, la forma de transferencia o los mensajes entre los diferentes dispositivos generadores de información en formato DICOM no son objetivos de esta tesis.

1.2.2. Características del Estándar DICOM

Tal y como se encuentra en la actualidad el estándar DICOM, las principales características destacables para este trabajo son:

- Intercambiabilidad de objetos en redes de comunicación y en medios de almacenamiento a través de protocolos y servicios, manteniendo, sin embargo, independencia de la red y del almacenamiento físico. Todo esto a través de comandos definidos por una sintaxis y una semántica muy estricta, a los que se les asocian datos.
- Como se maneja el flujo de datos correspondiente a las transferencias de los diferentes objetos DICOM. La especificación exige para cada objeto DICOM una cabecera “File Meta Information Header”, la cual es guardada al comienzo de iniciar el flujo de datos. Esta cabecera contiene información que permite al que recibe reconocer que se trata de un fichero DICOM, y define como está codificada mediante la sintaxis de transferencia (“Transfer Syntax”) y como ésta puede ser interpretada, además de una serie de parámetros para detectar los diferentes campos de datos que se encuentran en la misma. También se utilizan las estructuras de “Data Elements” para guardar este tipo de información.
- Incluye información explícita de objetos a través de estructuras de datos, que facilitan su manipulación como entidades autocontenidas. Los objetos no son únicamente imágenes digitales y gráficas, sino también estudios o informes, etc.
- Define una identidad única de objetos, como instancias con operaciones permitidas definidas a través de clases.

- Proporcionan interoperabilidad entre servicios y aplicaciones a través de una configuración definida por el estándar, manteniendo una comunicación eficiente entre el usuario y el proveedor de los servicios.
- Ofrece una representación de aspectos del mundo real, utilizando objetos compuestos que describen un contexto completo, y objetos normalizados como entidades del mundo real.

Todas estas características han hecho que DICOM sea el estándar utilizado por la gran mayoría de centros médicos en el mundo.

1.2.3. Estructura del Estándar DICOM

Con respecto a la parte dinámica del estándar, es decir, la interacción de DICOM con los diferentes dispositivos, cabe decir que las aplicaciones que soporta DICOM se componen por un conjunto de funcionalidades llamadas Clases de Servicios ("Services Classes") definidas en el estándar. Las Clases de Servicios son clases que contienen funcionalidades específicas para determinadas aplicaciones. Por ejemplo, una aplicación envía o recibe imágenes a través de Clases de Servicios de almacenamiento, mediante la funcionalidad de enviar y recibir objetos DICOM.

Por otra parte, respecto a la información médica codificada DICOM, cabe destacar que la estructura de la información contenida en DICOM permite incorporar la compresión en diferentes formatos aceptados por la comunidad de usuarios (como JPEG y JPEG2000), siendo consideradas parte del estándar. Normalmente, las imágenes al ser generadas no se comprimen, aunque pueden ser comprimidas tras su proceso en los sistemas de almacenamiento, manteniendo la compatibilidad con el formato DICOM, al existir entre sus opciones diversos algoritmos de compresión aceptados. En la compresión de datos médicos debe garantizarse la reversibilidad de proceso, es decir, que la compresión no produzca pérdida de información, tal y como habitualmente ocurre en el formato JPEG. La perdida de información implica la pérdida de la validez diagnóstica.

La estructura de los objetos DICOM, además de representar los datos médicos, contiene también información referente a sus características, como los parámetros relacionados con las características de la imagen, datos demográficos del paciente, datos relativos al equipo y el protocolo de exploración. Toda esta información se almacena en la cabecera DICOM de la imagen y se organiza en diferentes unidades de datos llamadas "Data Elements" que viene determinada por el Information Object Definition [6] (IOD) que se este aplicando al objeto DICOM. Un objeto DICOM siempre se corresponde con un IOD determinado, que también es conocido como un "Data Set". Un IOD no es más que un modelo de datos predefinido por el estándar, que constituye una abstracción de un tipo de información en el mundo real (como una imagen o un estudio). Este modelo de datos define la naturaleza y los atributos relevantes del objeto del mundo real representado. La unidad básica de un "Data Set" es un "Data Element". Por lo tanto, un "Data Set" no es más que una colección de "Data Elements", que describen atributos de la imagen, paciente, estudio o aquel dato que pueda ser relevante al objeto que representa. Un objeto DICOM puede también tener varios "Data Elments" anidados ("Data Elements" dentro de otros "Data Elements"). Los diferentes Data Elements que pueden existir en un documento DICOM quedan claramente especificados en el estándar DICOM [7].

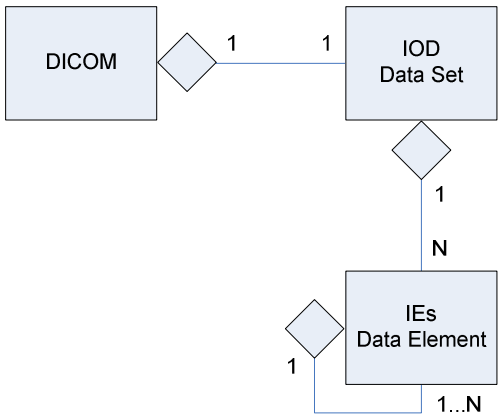

figura 1. Estructura de la Información de un Objeto DICOM.

1.2.4. DICOM Structured Reporting (DICOM-SR)

Tal y como se ha expuesto anteriormente, DICOM es el estándar adoptado y utilizado tradicionalmente por la mayoría de centros médicos en las áreas de radiología y cardiología. Debido a la utilización del estándar en nuevas áreas de la medicina, DICOM ha sufrido diversas ampliaciones para cubrir las necesidades generadas por estas nuevas áreas. Una de las últimas incorporaciones en el estándar, ha sido la inclusión de informes estructurados mediante DICOM-SR. Este tipo de datos cobra un especial interés en ésta tesis, motivado por la necesidad de agregar información semántica a las imágenes médicas.

Debido a la gran cantidad de áreas que pueden utilizar DICOM y los diferentes tipos de datos que se manejan, la especificación de informes estructurados genérica, independientes del área en la que se esté tratando resulta altamente complejo. La información que se debe plasmar en un informe estructurado requiere de una estructura que describa de forma exhaustiva la casuística del informe radiológico. En la actualidad es común la utilización de texto plano para la elaboración de los informes, por lo que el aprovechamiento de la información contenida para la realización de posteriores estudios es escasa, puesto que no se sigue una estructura homogénea y por tanto, operaciones automatizadas por computador, como la búsquedas por contenido o extracción de valores concretos, resultan poco eficientes.

La decisión de cómo estructurar la información semántica es muy importante. No es la misma conceptualización de un informe estructurado la de un radiólogo, que se encarga de detectar y valorar hallazgos en una imagen, que la de un traumatólogo, valorando la necesidad de una intervención quirúrgica. No sólo la estructuración es diferente y compleja, sino también la unificación de los conceptos que definen la información.

La codificación de conceptos puede ayudar a la mejora en la estructura de documentos. Existen en la actualidad muchas codificaciones diferentes (diagnósticos, tratamientos, nomenclaturas, etc.) que responden a diferentes necesidades. Entre las más utilizadas destaca la Clasificación Internacional de Enfermedades o International Classification of Diseases [32] (ICD9); la Codificación Internacional de Enfermedades (CIE9), que es una codificación para determinar diagnósticos y métodos; la Systematized Nomenclature of Medicine [33] (SNOMED), que es una nomenclatura sistematizada de medicina; o la Logical Observation Identifiers Names and Codes [34] (LOINC), que es una base de datos cuyo propósito es facilitar el intercambio y la elaboración de un fondo común de resultados, tales como, por ejemplo,

hemoglobina en sangre o constantes vitales, para la atención clínica, el tratamiento de los resultados y la investigación.

Las ventajas que nos aportan las codificaciones, tanto a la hora de generalizar o estandarizar conceptos como a la hora de explotar esta información, son evidentes. Las codificaciones nos permiten evitar diferentes interpretaciones y ambigüedades, puesto que cada código referente a un concepto tiene un significado concreto para una codificación determinada.

Por tanto la codificación es imprescindible para su posterior explotación, evitando así interpretaciones ambiguas que pueden llevar a confusiones. La gran diversidad de codificaciones existentes es otro valor añadido a la complejidad que supone unificar los conceptos de una manera común, a la hora de generar la información de un objeto DICOM.

El estándar DICOM recoge una codificación para generar informes estructurados en el área médica. El DICOM Structured Reporting [5] (DICOM-SR) permite integrar las codificaciones más importantes en el mundo de la imagen médica existentes, e incluso incluir codificaciones particulares. También señala las pautas a seguir a la hora de elaborar un documento estructurado para imágenes DICOM. Además permite una codificación estándar de la información relativa al informe en formato DICOM (conjuntos de "Data Elements" que correspondan con un IOD determinado). DICOM-SR permite estructurar información semántica y no da un interfaz de presentación a la información que contiene. La representación de un DICOM-SR vendrá definido por los IODs, (concretamente en DICOM-SR sólo se especifican tres tipos de IODs para definir un documento estructurado: Basic Text, Enhanced SR y Comprehensive SR). Los objetos DICOM-SR incluyen la estructura en árbol del documento estructurado, al que representan dentro de la información codificada mediante los "Data Elements". Ésta estructura en árbol se la conoce como el "SR Document Content" y es donde realmente vendrá definido el documento estructurado. El "Document Content" tendrá una estructura en árbol en la que cada nodo guardará una relación con su nodo padre, además de ser de un tipo determinado y definido en el estándar DICOM-SR, tal y como se muestra en la figura 2. En esta figura se representa el hallazgo de un tumor maligno de forma redondeada de un tamaño de 1,5 cm.

Por otra parte el "SR Document Content" puede tener diferente estructura según para qué se utilice el DICOM-SR. Por ello existen plantillas o "templates" para los DICOM-SR, que definen las estructuras y las restricciones que se requiera (como las codificaciones a emplear o la relación entre los nodos). En DICOM-SR también se pueden validar estructuras determinadas por medio de estas plantillas al igual que se hace con los esquemas XML. Las plantillas vienen definidas en el estándar DICOM. Muchas de estas plantillas se encuentran en proceso de especificación y todavía a día de hoy no están completadas. En la actualidad se están realizando trabajos para migrar los DICOM-SR a documentos XML [30] y también para generar esquemas XML a partir de las plantillas de los DICOM-SRs [29].

Por tanto, la utilización de DICOM-SR para la generación de informes estructurados es una buena herramienta, pues se integra perfectamente en el mundo DICOM y además permite generar, tratar y validar estos informes de una manera sencilla utilizando las tecnologías actuales basadas en XML. Es posible codificar los documentos estructurados DICOM-SR en documentos XML, de manera que el tratamiento de estos documentos no será diferente al tratamiento de cualquier documento XML.

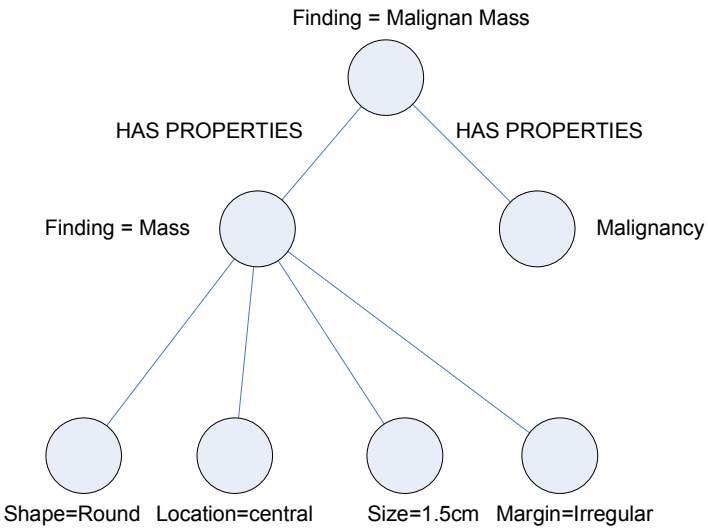

figura 2. Estructura General de un Árbol de Contenido de un Objeto DICOM-SR.

Por otra parte, en el estándar Health Level Seven [8] (HL7) también se incorpora en su estándar de forma nativa, la gestión de documentos estructurados, llamados Clinical Data Architecture [35] (CDA). Este formato fue incorporado a HL7 en 1997 para estructurar el contenido semántico de los documentos clínicos. Todo el contenido de un CDA se define mediante información codificada en el estándar XML. Sin embargo, DICOM-SR es una aproximación complementaria para documentos estructurados porque HL7 no propone en la actualidad una estructura rigurosa con CDAs, es decir, no incorpora plantillas a la hora de definir un informe estructurado. Por este motivo el middleware planteado en el presente trabajo utilizará DICOM-SR en la gestión de documentos estructurados.

1.3. Sistemas de Información Médica y Protocolos.

Esta sección ofrece una breve descripción de los sistemas de información y protocolos, que en la actualidad están implantados en la mayoría de los hospitales, que utilizan el formato digital para el almacenamiento de los datos. Esta sección pretende dar una visión general, mostrando sus carencias con respecto a las necesidades planteadas en la ésta tesis, y más concretamente en la parte referente a gestión compartida de información en formato DICOM.

1.3.1. Sistemas de Información para Imagen Médica (PACS)

En los hospitales y centros médicos actuales, la adquisición de imágenes médicas digitales es una práctica común y muy importante en el proceso clínico. A la hora de tratar estas imágenes cabe tener en cuenta una serie de factores, como por ejemplo, las diferentes modalidades de imágenes médicas que se manejan, las diferentes áreas del hospital que requieren consultar dichas imágenes, el intercambio de información con otros centros, etc. Para la realización de estas tareas se utilizan sistemas de manejo de imágenes, conocidos como Picture Archiving and Communications Systems [1] (PACS). Los PACS utilizan generalmente los estándares DICOM y Health Level 7 [8] (HL7) para sus comunicaciones, almacenando la información generalmente en otros formatos, aunque siguiendo las especificaciones marcadas en Integrating the Healthcare Enterprise [9] (IHE) que serán brevemente explicados en el apartado 1.3.5.

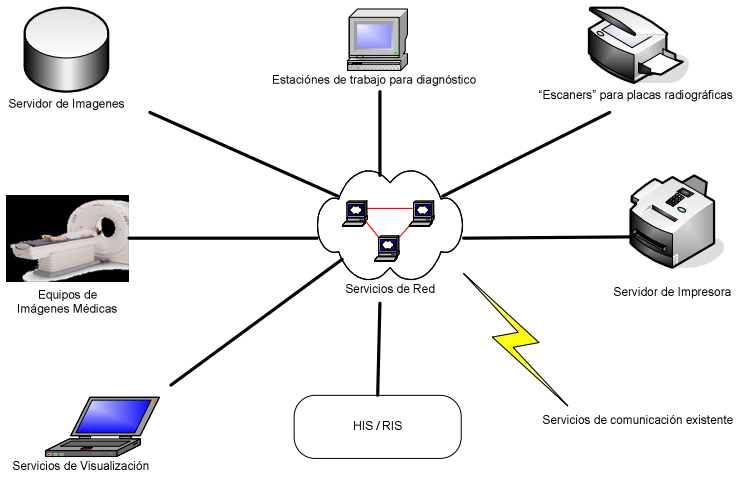

figura 3. Estructura General de un sistema PACS.

Básicamente los servicios que debe de ofrecer un PACS son los siguientes:

- Adquisición de imágenes. Los diferentes dispositivos de captación de imágenes deben estar integrados, registrados y debidamente configurados.
- Almacenamiento de información. Se incluye un sistema para almacenar las imágenes de forma eficiente, generalmente estructurado en varios niveles de almacenamiento correspondiendo a la probabilidad de consulta (como almacenamiento local, en línea o en cinta).
- Distribución de imágenes. Las imágenes deben poder ser distribuidas entre aquellas partes implicadas en su gestión.

- Visualización de imágenes (consulta, interpretación o diagnóstico). Las imágenes deben poder ser consultadas y mostradas para poder realizar las valoraciones pertinentes.
- Registro de resultados. Debe existir un registro de los resultados de las diferentes valoraciones que se vayan produciendo a lo largo de la vida útil de las imágenes.
- Interfaz con otros sistemas. El PACS debe ofrecer un sistema para poder interactuar con otros sistemas de información que conviven junto al PACS en los centros hospitalarios.
- Seguridad del sistema. La seguridad del sistema debe garantizar la autenticidad de los usuarios y su autorización para el acceso y manejo de los datos.

El Estándar DICOM ofrece servicios para poder entrelazar los diferentes sistemas de información en un hospital, como los PACS, Sistemas de Información de Radiología (RIS) y Sistemas de Información Hospitalaria (HIS).

Los sistemas PACS ofrecen una solución adecuada a la gestión del flujo de las imágenes médicas y su almacenamiento a nivel hospitalario, pero no son una buena solución a la hora de compartir datos entre diferentes hospitales en un marco colaborativo. Los sistemas PACS están diseñados para funcionar dentro de un mismo centro médico, sin tener en cuenta los problemas adicionales que plantea la colaboración entre centros médicos diferentes (seguridad de los datos, cifrado, transmisión eficiente, roles de los usuarios, etc.). Por ello se ha definido en la arquitectura propuesta en ésta tesis unos servicios (Servicios Grid) que se encargarán de interactuar con los sistemas PACS correspondientes para manejar de manera segura la información DICOM que se quiera compartir.

1.3.2. Sistemas de Información Radiológica (RIS)

Los sistemas Radiology Information System [2] (RIS) son sistemas de información radiológica orientados a la gestión del flujo de las imágenes e informes radiológicos, y responden a las necesidades de los distintos grupos de usuarios implicados (radiólogos, técnicos, administrativos). Casi todos los productos existentes están basados en una arquitectura cliente-servidor. Sin embargo, algunos fabricantes empiezan a proponer soluciones basadas en tecnología Web, que presentan importantes ventajas. Los sistemas RIS almacenan datos administrativos del paciente y el texto del informe radiológico, además de interactuar con los sistemas PACS para el acceso a las imágenes.

A continuación se detallan los puntos principales a tener en cuenta a la hora de evaluar un RIS:

- Herramientas para generación de informes. El proceso de informado, empleando un ordenador en el que se disponga automáticamente de los datos del paciente y de sus imágenes, supone de por sí una importante mejora frente al dictado con cinta y la visualización en el negatoscopio. Los sistemas actuales de dictado con reconocimiento de voz permiten incluso eliminar el proceso de trascripción del informe por parte de un administrativo, y en la actualidad se encuentran plenamente desarrollados. La seguridad de los datos mediante cifrado, firma electrónica o autologout en las estaciones de trabajo es también prioritaria, dado que se trabaja con datos sensibles. Ha de asegurarse el cumplimiento de la legislación vigente, así como de las normativas existentes en el hospital.

- Deben permitir una transición suave alterando de forma progresiva el modo de trabajo. Generalmente contemplan la posibilidad de incluir imágenes analógicas del histórico de los pacientes en el sistema mediante una estación adecuada y un escáner especial.
- Debe permitir el uso concurrente por varios usuarios. El número de licencias concurrentes del RIS ofertado debe evaluarse también, pues cada modalidad dispondrá de una estación RIS para el seguimiento y finalización de estudios, a las que se deben sumar las estaciones de diagnóstico, los puestos administrativos y otros en los que se desee disponer de los datos del RIS con o sin estación PACS, como por ejemplo la sala de sesiones clínicas.
- Accesibilidad. Aunque actualmente hay pocos fabricantes que lo ofrezcan, cada vez es más interesante la posibilidad de que todo el RIS o cierta parte de él (petición de pruebas, citación, consulta de estadísticas...) tenga un módulo accesible vía Web. Esto permitirá evitar la instalación de clientes en puestos remotos del hospital, y es la tendencia actual en muchas aplicaciones del ámbito hospitalario.
- Integración con los sistemas PACS. Este aspecto es de suma importancia. Algunos proveedores distribuyen RIS y PACS de distintos desarrolladores, y deben asegurar la correcta integración, a nivel de estación de trabajo conjunta, control de consistencia de las bases de datos, o si es posible, incluso base de datos única.

Se puede decir que el RIS es un paso más para integrar el sistema de información radiológica en el Sistema de Información Hospitalario (HIS). Cabe decir que este tipo de sistemas, aunque constituyen un avance a la hora de compartir información, están diseñados para la gestión de un centro concreto. Una de las deficiencias mas importantes podría ser la carencia de definición de roles colaborativos. Por tanto, en la arquitectura a definir en ésta tesis, se debe recoger también la necesidad de definir servicios que interactúen con este tipo de sistemas si se requiere información que no venga definida en los sistemas PACS, como informes de estudios DICOM o datos adicionales del paciente que no estén incluidas en la metainformación DICOM.

1.3.3. Sistema de información Hospitalaria (HIS)

Los Hospital Information System [3] (HIS) se encargan de gestionar todo lo referente a la información de los pacientes y a los distintos departamentos hospitalarios, entrelazando y unificando información común entre los distintos departamentos. El resto de los sistemas de información, como por ejemplo el RIS, están altamente implicados con el HIS, pues estos sistemas obtienen parte de la información de pacientes a través del HIS.

En el caso de este trabajo, se busca una interacción directa con aquellos elementos relacionados con los datos DICOM de un centro médico. Los sistemas HIS sólo se considerarán a través de su interacción con los RIS y PACS, ya que la información que manejan no interaccionan con los servicios a definir en la arquitectura.

Al igual que ocurre con los sistemas PACS y RIS, los HIS constituyen un avance a la hora de compartir información, aunque estos sistemas son diseñados para la gestión de los datos de interés de un centro. Una de las deficiencias más importantes podría ser la carencia de definición de roles colaborativos, pues se definen roles locales a través de grupos de usuarios y dominios administrativos.

Por tanto, la arquitectura a definir en esta memoria, también debe recoger la necesidad de definir servicios que interactúen con los sistemas HIS si se requiere información que no venga definida en los sistemas PACS y RIS.

1.3.4. Health Level 7 (HL7)

El Health Level 7 [8] (HL7) es la única organización acreditada ANSI para la estandarización de procesos clínicos. HL7 provee de estándares para el intercambio, gestión e integración de datos que soportan la asistencia médica al paciente, y la evaluación de los datos de salud en los diferentes departamentos que componen los centros médicos. HL7 empezó en 1983 en los Estados Unidos y está presente en Europa desde 1993.

La idea principal de HL7 es la integración jerárquica de los diferentes sistemas de información existentes en los diferentes departamentos hospitalarios, sin obligar a su redefinición. Cada departamento tiene sus propios problemas y casos particulares, aunque todos ellos contienen elementos comunes (p.e. datos demográficos de los pacientes) y sufren de necesidades similares. HL7 pretende unificar conceptos y dar estándares para poder interactuar entre éstos de forma sencilla. HL7 también pretende dar un vocabulario controlado de los elementos que se transfieren, para unificar conceptos entre los diferentes sistemas existentes y así evitar problemas de interpretación de conceptos.

HL7, corresponde con la capa de nivel siete dentro del modelo de la arquitectura de comunicación definida por Open Systems Interconnection Reference Model [52] (IS-OSI). Este estándar se aplica en la actualidad en el 90% de los hospitales norteamericanos y se usa en todos los departamentos, excepto aquellos que se encargan de la gestión de imagen. Actualmente se está organizando en España. Este estándar incidirá en el desarrollo de ésta tesis si compartimos información más allá de la compartida por el objeto DICOM autocontenido, es decir, que esté en los sistemas de información RIS.

1.3.5. Integrating the Healthcare Enterprise (IHE)

Los estándares DICOM y HL7 muchas veces se intersectan y son aplicables en los mismos casos de usos y sistemas de información. Para consensuar el uso de los diferentes estándares se creó Integrating the Healthcare Enterprise [9] (IHE).

IHE fué promovido por la Radiological Society of North America (RSNA) y la Healthcare Information Systems and Management Society (HIMSS). Su principal objetivo es facilitar que los departamentos estén totalmente integrados a través del uso de estándares HL7 y DICOM, quedando claramente especificado dónde se debe de aplicar un estándar u otro a la hora de hacer fluir la información. IHE define escenarios clínicos y autores que especifican los estándares que se ven implicados para cada caso y en qué momentos se deben utilizar.

Las recomendaciones IHE son relevantes para el desarrollo de los servicios de la arquitectura que se plantea en ésta tesis, pues el sistema debe ser consecuente con qué estándar se debe utilizar para recuperar adecuadamente la información no presente en los objetos DICOM, es decir, los protocolos a utilizar a la hora de interrogar los sistemas RIS o HIS si se da el caso.

1.4. Tecnologías Grid

Las tecnologías Grid suponen la base tecnológica y conceptual de la arquitectura software que se propone en este trabajo. En este apartado se describen con mayor nivel de detalle los objetivos y capacidades de las tecnologías Grid, prestando especial atención a la Open Grid Service Architecture [36] (OGSA) y las implementaciones de los servicios de esta arquitectura mediante la infraestructura de Web Services Resource Framework [37] (WSRF).

En este apartado se propone una definición del Grid, además de exponer un estudio de las diversas infraestructuras implementadas sobre OGSA existentes en la actualidad, aportando una comparativa entre ellas y justificando la elección de WSRF para la implementación de los servicios de la arquitectura que se desarrollarán en esta tesis.

1.4.1. El concepto de Grid

La tecnología Grid es una de las tecnologías principales utilizadas en la presente tesis. Por ello, en este apartado se va a dar una definición de lo que es un Grid así como aquellos puntos que influyen directamente en los desarrollos a realizar dentro del marco de ésta tesis.

Una definición del término Grid sería el conjunto de protocolos, servicios y aplicaciones que permiten compartir recursos distribuidos geográficamente de forma coordinada, entre diferentes usuarios estructurados en forma de "organizaciones virtuales", caracterizadas por ser dinámicas y multi-institucionales, teniendo en cuenta una estrategia conjunta para la solución de problemas y el hecho de que no se comparten simplemente recursos como archivos o datos, sino también recursos de proceso y almacenamiento [39].

Con respecto a las nuevas necesidades surgidas y planteadas en ésta tesis, referentes a la necesidad de tener acceso a información DICOM independientemente de su ubicación, así como de recursos capaces de realizar operaciones de proceso, esta primera definición del Grid pone de manifiesto que el Grid sea una prometedora tecnología para solucionar los problemas que pueden plantear estas nuevas necesidades.

Siguiendo con la definición de los que son las tecnologías Grid y para entender mejor sus fundamentos, es importante comprender el concepto de Organización Virtual (OV) enunciado por Ian Foster, Carl Kesselman y Steven Tuecke [39]. Las OVs son organizaciones que emergen a partir de varios participantes "del mundo real" que colaboran en un entorno Grid para lograr un objetivo común. Un interesante símil con respecto a esta definición, son las uniones temporales de empresas, que comparten recursos de forma temporal para la consecución de un fin, el cual no serían capaces de abordar de forma individual. Es interesante destacar que las entidades reales participantes pueden tener diversos grados de relación entre ellos, o incluso puede que no tengan ninguna relación o garantía de confianza. Pese a esto, el objetivo de las OVs es colaborar para efectuar las tareas que les fueron asignadas. Esta paradójica situación junto con algunas otras que se mencionarán más adelante, son las que provocan aquellos problemas particulares que deben resolverse al implementar una estrategia Grid.

Volviendo a los trabajos a realizar en ésta tesis, y desde la perspectiva Grid, se podría decir que para compartir los recursos de almacenamiento y proceso de objetos DICOM ubicados en diferentes hospitales o centros médicos ("mundo real"), y cuyo objetivo es compartir recursos

para conseguir un resultado, generalmente científico, inabordable de forma individual, éstos se deberían organizar en una OV que determinara los acuerdos entre las diferentes entidades para definir su colaboración.

Por otra parte, para que una arquitectura Grid sea posible, debe haber una capa de software que se sitúe lógicamente entre las aplicaciones que hagan uso de su potencial, y los recursos a compartir. Esta capa es lo que se conoce como el middleware Grid, que abstrae de los aspectos de más bajo nivel, proporcionando un acceso transparente a las aplicaciones que requieran un uso intensivo de recursos. Un ejemplo de middleware Grid es el Globus Toolkit [40], desarrollado de forma comunitaria y en licencia de Open Source, y que hoy en día es una referencia de las tecnologías Grid en todo el mundo.

Estas herramientas, basadas e implementadas sobre arquitecturas Grid, deben proporcionar la siguiente funcionalidad básica:

- Gestión compartida de los recursos. Los recursos deben verse como algo compartido por todas las entidades participantes, de manera que su gestión debe ser manejada por los administradores de la OV.
- Entorno de Seguridad a nivel de usuarios, recursos y procesos. Deben de garantizarse las identidades de los usuarios y de los recursos, además de asegurar que las comunicaciones se realizan a través de canales seguros.
- Gestión eficiente de los recursos. El acceso a estos recursos debe ser controlado de manera eficiente.
- Transmisión rápida y fiable de datos. Se definen e implementan protocolos y redes adecuados a las necesidades.
- Estándares abiertos. Con el objetivo de permitir una mejor integración de nuevos componentes.

1.4.2. Arquitecturas Grid

En esta tesis se pretende definir una arquitectura basada en tecnologías Grid que cubra las funcionalidades fundamentales descritas en el apartado anterior. Para ello, este apartado plantea aquellos puntos básicos que debe considerar una arquitectura Grid, dentro de los objetivos de la tesis.

Los diferentes protocolos, servicios y aplicaciones de una arquitectura basada en tecnologías Grid se estructuran en diferentes capas como las que se describen a continuación:

- Infraestructura: Corresponde a los recursos que se van a desplegar para su uso por los usuarios de cada OV. Los recursos pueden ser computadores, clusters, supercomputadores, bases de datos, sistemas de almacenamiento en red, etc.
- Sistemas de Monitorización e Información: En un entorno Grid se requiere de un sistema que monitorice y mantenga información actualizada sobre los recursos disponibles en la infraestructura.
- Conectividad: Se refiere tanto a los medios existentes para comunicarse con los recursos (por ejemplo Internet), como a los protocolos utilizados.

- Seguridad: El sistema debe garantizar el acceso a los recursos de una manera fiable, asegurando la privacidad de los datos, la autenticación de los usuarios y la gestión de recursos a nivel de una OV.
- Aplicaciones: Aquéllas que hacen uso de las infraestructuras Grid mediante un Middleware Grid.

1.4.3. Open Grid Services Architecture (OGSA)

Como se ha visto al final del apartado anterior, el primero de los puntos clave a la hora de definir una arquitectura Grid es la infraestructura a emplear, es decir, los recursos a desplegar y de qué manera estos recursos pueden ser definidos y como se ha de interactuar con ellos. Por ello, en este apartado se incluye una visión de las tecnologías actuales que se utilizan para la definición de los recursos a desplegar. Entre las tecnologías más utilizadas en la actualidad destacan los Servicios Web (WS), los servicios Grid (OGSI o WSRF) y CORBA, entre otros.

En cuanto a los requerimientos de una arquitectura Grid, establecer una serie de requerimientos de hardware y otros de software a la hora de crear la infraestructura. Para entender las necesidades desde el punto de vista de una aplicación, podemos preguntarnos, por ejemplo, para la ejecución de una tarea desde un computador en el Grid, ¿Cómo se identificarán los recursos de la arquitectura a utilizar? ¿Cómo se describen las tareas? ¿Cómo se envían a las tareas los parámetros o archivos que éstas requiere para su ejecución? En las primeras versiones de los middleware Grid, la mayor parte de estas acciones se realizaban de forma explícita por los desarrolladores y usuarios. Sin embargo, en la actualidad los esfuerzos se dirigen hacia la estandarización y la automatización de estos procesos de forma que exista una descripción común de las funcionalidades básicas, que puedan luego extenderse a los campos específicos en los que se quiera trabajar. Esta tarea fue asumida por el Open Grid Forum [41] (OGF), que desarrolló una especificación para una arquitectura Grid orientada a servicios llamada Open Grid Services Architecture [36] (OGSA). OGSA soporta la creación, mantenimiento y agrupación de servicios integrados dentro de las OVs. Pese a esto, el estándar OGSA no se compromete con la forma en que se implementarán los servicios que sustenta, especificando únicamente las interfaces y no el comportamiento. Sobre la arquitectura OGSA se definieron entonces especificaciones de servicios que cada vez más son aceptadas en la comunidad informática. En este marco, se identificaron los Servicios Web como la base de los nuevos Servicios Grid. Dado que OGSA define una arquitectura estándar y abierta para aplicaciones basadas en tecnologías Grid, el objetivo principal del OGF está siendo estandarizar los servicios que se necesitan en el desarrollo de una aplicación Grid (seguridad, servicios para gestionar recursos, servicios para el manejo de trabajos, etc.), desarrollando y especificando un conjunto de interfaces estándar genéricos para toda aplicación Grid que pretenda beneficiarse de los servicios que la arquitectura pueda ofrecer.

Para el desarrollo de los requisitos comunes de una arquitectura Grid, se podría haber tomado como base, los middlewares anteriores al Grid relacionados con la computación distribuida (como CORBA, RMI, o incluso RPC), ya que son estándares altamente aceptados por la comunidad informática. Algunas de las razones más importantes por las que se eligió la tecnología de Servicios Web se exponen a continuación.

Los Servicios Web permiten crear aplicaciones cliente/servidor al igual que otros midlewares como CORBA o RMI. En este caso, una aplicación basada en Servicios Web consta de la parte cliente que ejecuta los servicios ofrecidos por los servidores, permitiendo el descubrimiento del servicio, de modo que el cliente busca el servicio que necesita y una vez lo encuentra lo ejecuta. Los Servicios Web utilizan Web Services Description Language [42] (WSDL) para permitir a los clientes poder descubrir e interactuar con los servicios. De nuevo, en CORBA existe un lenguaje análogo que es el Interface Definition Language (IDL). Sin embargo, la principal ventaja de WSDL es que es estándar, por lo que descubrir los servicios resulta más fácil, y por tanto las herramientas para poder tratar e interpretar este tipo de información son mucho más extensas.

Otra de las ventajas de los Servicios Web y que otras tecnologías como CORBA no ofrece, es que los Servicios Web utilizan para la comunicación el protocolo de alto nivel Simple Object Access Protocol [70] (SOAP), que puede funcionar sobre otros protocolos ya existentes como por ejemplo http, ftp, https y ftps. Al ser estos protocolos ampliamente utilizados y soportados, este aspecto aporta una gran ventaja respecto a otros middlewares como CORBA, evitando numerosos problemas organizativos asociados (como cortafuegos, proxies de Internet, …).

También existen puntos negativos en los Servicios Web con respecto a las necesidades de OGSA. Uno de los aspectos más notorios es la carencia de la persistencia del estado de un servicio, aunque también existen otras como la carencia de notificaciones de eventos, manejo del ciclo de vida de un servicio, transacciones etc…Si bien es posible implementar de forma particular soluciones a estos problemas, se hace necesaria la extensión de un estándar. Estas carencias son recogidas por OGSA y se corrigen en las especificaciones de Web Services Resource Framework [37] (WSRF). También existen otras especificaciones que cumplen con OGSA, como Open Grid Service Infraestructure [38] (OGSI), aunque esta última está en desuso y actualmente no es mantenida.

A continuación se introducen brevemente OGSI y WSRF, planteando las ventajas e inconvenientes de cada una de las especificaciones a la hora de utilizar una u otra infraestructura, para la creación de los recursos a desplegar dentro de una arquitectura Grid.

1.4.3.1. Open Grid Service Infraestructure (OGSI)

Open Grid Service Infraestructure [38] (OGSI) es una especificación de OGSA desarrollada por los miembros del Global Grid Forum (GGF). OGSI fue creado con la esperanza de que acabara convergiendo a Servicios Web, lo que no ha ocurrido principalmente por diversos motivos:

- La especificación OGSI es extremadamente densa y compleja lo cual dificulta los desarrollos.
- OGSI no interactúa bien con herramientas que trabajan con Servicios Web, se requiere de herramientas específicas como Globus 3.X o OGSI.NET.
- Su intensa orientación a objetos, a pesar de las ventajas que este paradigma proporciona, implica ciertas incompatibilidades con los Servicios Web que no siempre se ciñen a este modelo.

Por tanto, para garantizar la convergencia a Servicios Web, se substituyó esta especificación por el Web Services Resource Framework (WSRF).

1.4.3.2. Web Services Resource Framework (WSRF)

El Web Services Resource Framework [37] (WSRF) es una especificación desarrollada por la "Organization for the Advancement of Structured Information Standards" [43] (OASIS). Básicamente especifica como proporcionar estado a los Servicios Web y poder agregar las características definidas en OGSA a los Servicios Web. Se puede decir que WSRF es una extensión de los Servicios Web para ser aceptado por OGSA.

WSRF garantiza la convergencia a Servicios Web, pues estos servicios son utilizados como plataforma de comunicación de los Servicios Grid, por el contrario OGSI no utilizaba los Servicios Web e interactuaba directamente con los Servicios Grid.

1.4.3.3. Diferencias entre OGSI y WSRF

Las principales ventajas que nos aporta WSRF frente a OGSI son las siguientes:

- Las dos especificaciones abarcan los mismos requisitos determinados por OGSA, pero la especificación WSRF es mucho menos densa que la de OGSI, conteniendo sólo 5 documentos que se corresponden estrictamente con WSRF.
- OGSI no se integra bien con las herramientas actuales de los Servicios Web, mientras que WSRF sí interactúa con estas herramientas correctamente al estar basado puramente en Servicios Web.
- OGSI esta muy orientado a objetos, lo cual provoca ciertas incompatibilidades con los Servicios Web. Por el contrario, WSRF separa claramente la parte de Servicios Web con la parte de gestión de recursos (como creación de estancias y estado).

En el desarrollo e implementación de los servicios en la arquitectura definida en ésta tesis, se ha utilizado los WSRF, particularmente el toolkit de Globus 4, dado que es ésta la infraestructura más completa e integrada en el mundo Web, además de tener unas perspectivas de futuro mucho mejores que OGSI, actualmente no mantenida.

1.5. Sistemas de Monitorización e Información

Tal y como se ha comentado en el apartado 1.4.2, otro aspecto importante, además de la infraestructura a utilizar para el despliegue de una arquitectura Grid, son los Sistemas de Monitorización e Información de los recursos.

Los Sistemas de Monitorización e Información tienen como principal función el mantener información actualizada referente a todos los recursos o servicios desplegados en la capa de infraestructura. En el caso de ésta memoria, el aspecto más relevante es la búsqueda de recursos, cuyo objetivo es informar sobre dónde están ubicados los recursos para su posterior uso.

Todo sistema de información tiene unos componentes y características comunes, aunque esto pueden estar implementados de manera diferente. Entre ellos destacamos los siguientes:

- Nombre de Servicio: Identifica los recursos o servicios desplegados en la arquitectura de manera inequívoca.
- Acceso Ubicuo: El sistema de búsqueda debe ser accesible desde cualquier punto del sistema.
- Transparencia: El funcionamiento debe ser totalmente transparente al usuario de manera que se oculte la complejidad interna del sistema. El cliente debe acceder al sistema, y de forma simple obtener la localización del servicio buscado.
- Escalabilidad: El sistema de información debe ser capaz de mantener información de nuevos recursos que se desplieguen en la capa Infraestructura de una manera fácil y eficiente. Debe permitir, de forma sencilla, la adición de nuevos componentes al sistema, tanto de nuevos servicios básicos, como de servicios indexadores del propio sistema de información sin que esto represente una pérdida significativa en la eficiencia.

Por otra parte, el sistema de información debe tener en cuenta algunos aspectos a la hora de implementar los recursos implicados en el sistema Monitorización e Información. Estos recursos, acordes con las especificaciones OGSA, se implementan en ésta tesis bajo la infraestructura WSRF, como se ha comentado y justificado en el apartado 1.4.3.

Por otra parte y aunque no sean requisitos imprescindibles, también existen una serie de características que estos sistemas se recomienda que cumplan, entre las que destacan las siguientes:

- El sistema de Monitorización e Información debería ser completamente distribuido, es decir, desde cualquier recurso implicado en el sistema de información debería ser posible interrogar el sistema completo, sin recurrir a un sistema centralizado en un único recurso.
- Fácil Accesibilidad. Para ello el sistema debe estar diseñado para permitir la existencia de diferentes puntos de acceso, de manera que siempre exista uno cercano y conocido por los clientes del servicio.
- Tolerante a fallos. Es muy importante que el sistema proporcione un gran nivel de tolerancia a fallos, haciendo que el fallo de un componente del sistema únicamente afecte de forma local y no provoque un fallo del sistema completo. Además, el sistema debe ser capaz de detectar esos fallos a la hora de realizar las búsquedas, para distinguir cuando un servicio no se encuentra porque no existe o porque temporalmente no se encuentra disponible.

A continuación se describen diferentes sistemas de Monitorización e Información existentes en los sistemas Grid actuales, destacando aquellas características que son importantes respecto al objetivo general planteado en la tesis, así como una justificación de cual será el sistema empleado en la arquitectura propuesta.

1.5.1. Sistemas de información actuales

1.5.1.1. Grid Monitoring Architecture (GMA)

El primer Sistema de Monitorización e Información que se presenta es el Grid Monitoring Architecture [44] (GMA) propuesto por el GGF. Tal y como se muestra en la figura 4, se compone básicamente de tres componentes: Consumers, Producers y un servicio directorio, denominado Registro. Algunos middlewares, como por ejemplo gLite [46], utilizan este modelo de sistema de información.

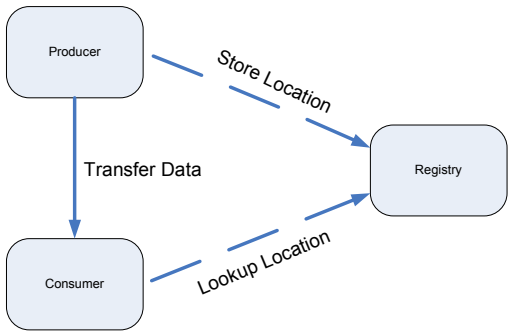

figura 4. Esquema General de un sistema GMA.

En el GMA, los "producers" se inscriben en el registro y describen el tipo y la estructura de la información que pretenden publicar, para hacerla disponible en todo el Grid. Los "consumers" pueden preguntar al registro para encontrar el tipo de información que requieren y localizar así a aquellos "producers" que disponen el tipo de información buscado. Una vez se localizan éstos, el "consumer" contacta directamente para obtener los datos requeridos.

Mediante la especificación del protocolo "Consumer/Producer" y los interfaces con el registro, se pueden generar servicios para la realización de operaciones de publicación y consulta de información. La comunicación con el registro se muestra en la figura anterior con una línea discontinua, mientras que el flujo principal de datos viene reflejado mediante una línea continua.

La definición actual del GMA también incluye en el registro a los "consumers", permitiendo que un "producer" pueda encontrar un "consumer". De esta forma, el registro puede notificar a los "consumers" sobre los cambios producidos en el conjunto de "producers" con los que están relacionados. La arquitectura GMA fue ideada para la monitorización pero también se puede utilizar para sistemas de búsqueda de recursos e información.

GMA no se restringe a ningún protocolo ni al modelo de datos que subyace en la implementación, de manera que ésta queda abierta para adoptar el modelo más adecuado para la complejidad de las consultas sobre los datos que se requiera implementar.

Una interesante implementación del modelo GMA es el R-GMA. R-GMA es una implementación del GMA basada en un modelo relacional, desarrollado en el marco del

proyecto European DataGrid (EDG). R-GMA crea la virtualización de una base de datos relacional por OV, distribuida entre los diferentes recursos. Sin embargo es importante apreciar que este sistema es una forma de utilizar el modelo relacional en un entorno Grid, en el sentido de que los "producers" registran la información mediante sentencias del lenguaje SQL del estilo CREATE TABLE y la publican mediante sentencias del tipo INSERT. Por otra parte, los "consumers" utilizan sentencias SQL del tipo SELECT para recuperar la información requerida. El R-GMA se ha desarrollado utilizando la tecnología de servlets y está siendo migrada a Servicios Web para adecuarse a las especificaciones OGSA.

1.5.1.2. Sistema de Monitorización y Descubrimiento Versión 2 (MDS2)

El Servicio de Monitorización y Descubrimiento (MDS2) está diseñado para proporcionar un mecanismo estándar para publicar y descubrir información sobre los recursos desplegados por una infraestructura Grid, adaptándose a los cambios y estado del sistema. Algunos middlewares actuales como el Toolkit de Globus 2 (GT2) y el LCG [47] utilizan este sistema. MDS2 proporciona un interface uniforme y flexible para los datos recogidos mediante los proveedores de información, situados en los propios recursos desplegados.

La estructura del MDS2 está descentralizada de manera que permite una buena escalabilidad, pudiendo manejar tanto datos estáticos como dinámicos. Además, MDS2 se integra perfectamente con la infraestructura de seguridad Grid Security Infraestructure [45] (GSI), permitiendo utilizar las credenciales GSI con características de autorización que provee el propio MDS2 a la hora de autorizar los accesos.

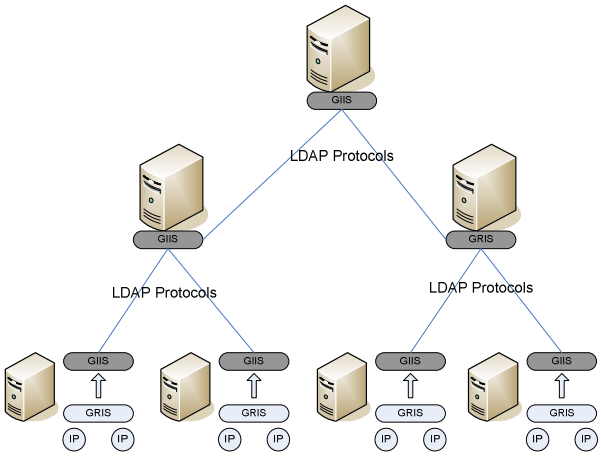

figura 5. Esquema General de un sistema MDS2.

MDS2 reduce el número de mecanismos requeridos para acceder al sistema de información. Los sistemas locales pueden usar una amplia variedad de mecanismos para la generación y publicación de información, pero los usuarios y servicios sólo necesitan un único mecanismo MDS2 para tener acceso a toda la información. MDS2 utiliza el protocolo LDAP como una interfaz de acceso a la información mediante consultas en el lenguaje LDIF. Al integrar LDAP como interfaz de acceso, cualquier navegador LDAP puede ser utilizado para la consulta tanto de los GIIS con los GRIS de un MDS2.

Por otra parte, la estructura jerárquica de MDS2 esta compuesto por cuatro clases de componentes principalmente. El Grid Index Structure Service (GIIS) provee un directorio de los

datos que se ubican en diferentes niveles de la jerarquía. El Grid Resource Information Service (GRIS) se sitúa en los recursos en sí, y actúa como una entrada de información del recurso al sistema de información a la hora de publicar datos. Los Information Providers (IPs) son el interface de cualquier tipo de colección de datos a publicar que se encargan de trasladar estos al GRIS para que posteriormente se publiquen en el GIIS de nivel superior. El GIIS a su vez puede publicar sus datos a un GIIS de nivel superior. Los datos que proporcionan los IPs de los recursos suelen estar relacionados con las características del recurso, como la configuración de la CPU, espacio en disco, memoria RAM, etc. El esquema del GIIS tiene como inconveniente que su escalabilidad es reducida. Como respuesta a este problema se desarrolla el Berkely Database Information Index (BDII). El BDII lo forman dos servidores LDAP, un servidor para accesos de lectura y otro para accesos de escritura. Este componente se encarga de recoger información cada dos minutos de los GIIS que tiene registrados en su fichero de configuración. Algunas OVs tienen su propio BDII en el cual recogen información de los GIIS que pertenecen a las OVs correspondientes. La principal función del BDII es la de ser una caché de toda la información albergada en un conjunto de GIIS.

1.5.1.3. Sistema de Monitorización y Descubrimiento Versión 4 (MDS4)

Al igual que ocurre con el MDS2, el Servicio de Monitorización y Descubrimiento MDS4 está diseñado para proporcionar un mecanismo estándar para publicar y descubrir información sobre los recursos del Grid, adaptándose a los cambios y estado del sistema. MDS4 es incorporado por middlewares actuales como el Toolkit de Globus 4 [48] y a diferencia del MDS2, está basado en WSRF y por tanto acorde con las especificaciones marcadas por OGSA.

MDS4 implementa un interfaz estándar mediante Servicios Web para la monitorización de los recursos y publicación de información en el Grid. La estructura es básicamente la misma que en el MDS2 y también tiene una estructura jerárquica. La información que los recursos quieren publicar en el MDS4 es enviada a éste mediante esquemas XML. Para la consulta de los datos publicados del MDS se utilizan consultas mediante interfaces definidas por Servicios Web y el lenguaje de consultas de ficheros XML XPath.

MDS4 se fundamenta sobre protocolos de consulta e interfaces definidos por WSRF. Partiendo de esta base, se han implementado un conjunto de Information Providers (IPs) utilizados para recolectar información desde los propios recursos.

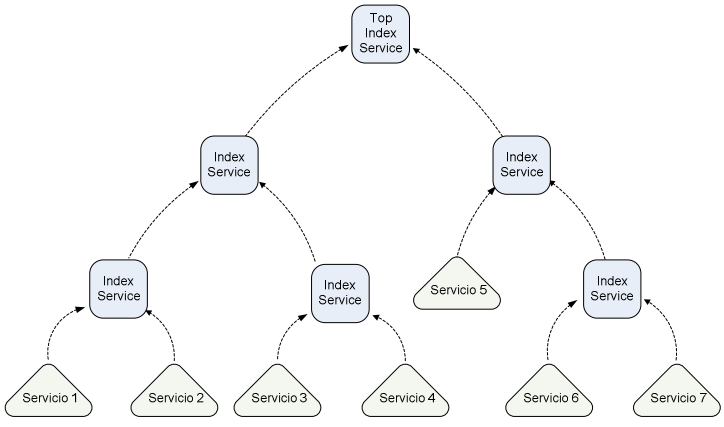

figura 6. Estructura en árbol de los IDXSRV de GT4.

Los componentes que componen el MDS4 son los Index Services (IDXSRV), los cuales recogen y publican información sobre las fuentes o recursos que se conectan a éste. Por otro lado, también existen los Trigger Services, que recogen de forma automática la información definida cuando se cumplen ciertas condiciones.

En una configuración estándar, cada Organización Virtual dispone de un IDXSRV, de manera que en ese IDXSRV reside toda la información. Para descubrir la información publicada en un MDS4, existe el interface Web llamado WebMDS.

1.5.1.4. Distributed Discovery Architecture for Grid Environments (DIDA)

Este sistema de Información y Monitorización ha sido desarrollado en el Grupo de gRid y Computación de Altas Prestaciones (GRyCAP) de la Universidad Politécnica de Valencia (UPV) en el marco del proyecto gCitizen [49]. El sistema Distributed Discovery Architecture for Grid Environments [50] (DIDA) es un sistema para complementar el sistema de búsqueda MDS4, que utiliza la Federated Advanced Directory Architecture [51] (FADA).

FADA es un sistema de directorio dinámico compuesto por un conjunto de servidores, que conforman un servidor "virtual". Los nodos colaboran entre sí para que, independientemente del nodo que se acceda, la visión del sistema sea completa. FADA proporciona un registro de servicios totalmente distribuido, y tolerante a fallos.

La arquitectura FADA está compuesta por una serie nodos que se van añadiendo al sistema, notificando su presencia a los nodos ya existentes que forman la infraestructura. La organización de los nodos en FADA conforma una estructura llamada "Scale-Free Network". En esta red, la caída de un nodo cualquiera, no afecta gravemente al sistema en conjunto, incrementándose la robustez y la capacidad de autoorganización del sistema.

Los servicios se van registrando en cualquiera de los nodos, pasando a formar parte del sistema. Para encontrarlos, basta con realizar una petición de búsqueda a cualquiera de los nodos FADA, quienes envían dicha búsqueda a todos los nodos que forman la infraestructura (a modo de broadcast).

Con estas características, FADA puede complementar a MDS4 con su sistema de búsqueda distribuida. En concreto, el sistema de búsqueda que se propone para gCitizen, consiste en el mantenimiento de una red de servicios IDXSRV de GT4, indexados mediante FADA, pero modificando la topología de ésta [50] de manera que para la conexión de los nodos FADA siempre existan 2 caminos disjuntos entre dos nodos cualesquiera del sistema. A partir de esta construcción, se dispondrá de acceso a cualquiera de estos indexadores mediante FADA, para posteriormente interrogarlos acerca de los servicios que indexan, utilizando los protocolos propios de GT4.

El esquema de las búsquedas se muestra en la figura 7. Este esquema propone que cada nodo del sistema MDS4, además de los servicios propios desplegados en él, tenga su propio IDXSRV, donde se registrarán todos los servicios locales, y un servicio de búsqueda al que los clientes se podrán conectar para realizar las peticiones de búsqueda a todo el sistema. Excepcionalmente, ciertas entidades, sobre todo las de pequeño tamaño, no tendrán su propio IDXSRV, sino que delegarán a otra entidad de mayor tamaño, donde se registrarán sus servicios. Además, los IDXSRV pueden tener una estructura jerárquica, de manera que existan

IDXSRV que agrupen los pertenecientes a diferentes nodos del sistema, para facilitar las búsquedas. Para proporcionar un acceso ubicuo, fácil y transparente al sistema de búsqueda diseñado se despliegan, para cada una de las entidades del sistema, un servicio de búsqueda que será el punto de entrada al sistema, y por tanto, debe ser conocido por todos los usuarios de dicha entidad.

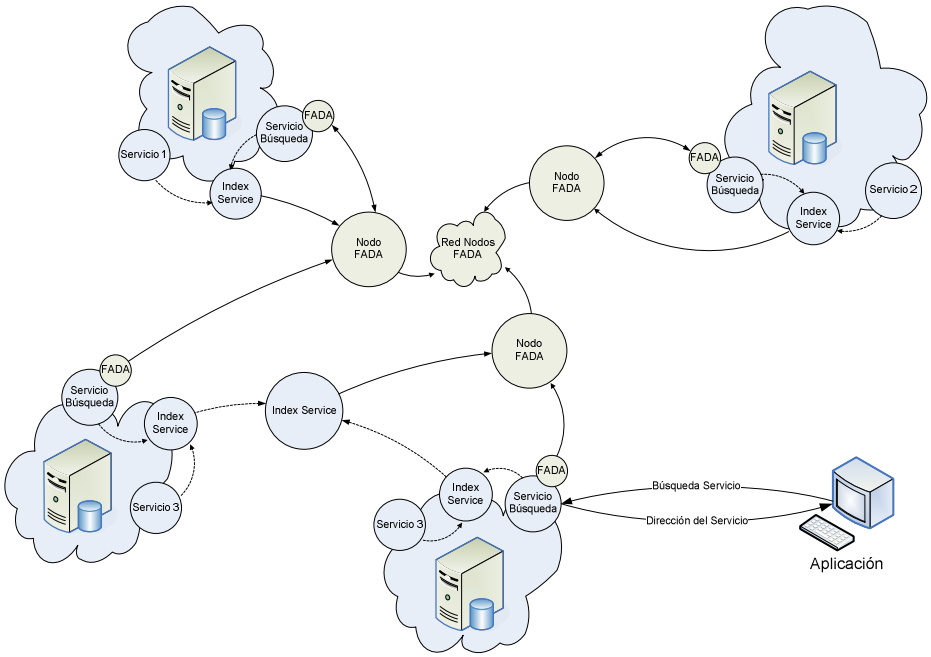

figura 7. Sistema de Búsqueda FADA – GT4.

Los servicios de búsqueda deben ser implementados como servicios estándar Globus 4. Cada uno de los servicios conoce algún nodo de la red FADA, con el que interactuará para descubrir todos los Index Services del sistema.

El funcionamiento del servicio de búsqueda será el siguiente:

- Una aplicación accede al servicio de búsqueda indicándole la consulta XPath que quiere realizar al sistema.
- El servicio de búsqueda conectará con algún nodo FADA conocido, para obtener la lista de todos los Index Services, disponibles en el sistema.
- Este nodo FADA conectará, usando el protocolo interno de FADA, con el resto de nodos de la red FADA, devolviendo finalmente la lista de los IS que componen el sistema.
- El servicio de búsqueda realiza la pregunta XPath a cada uno de los IS, de la lista obtenida.
- Finalmente todos los resultados de las preguntas XPath realizadas a los IS, son devueltos a la aplicación cliente.

1.5.2. Aplicación de los Sistemas de Información

Todos los sistemas de Monitorización e Información descritos anteriormente, pueden ser utilizados para el desarrollo de la arquitectura que se propone como objetivo general de ésta tesis. Tomando como base las características que debe tener un Sistemas de Monitorización e Información, planteados en la introducción la sección 1.5, todos estos sistemas tienen ventajas e inconvenientes, que hay que tener en cuenta a la hora de valorar el mejor a aplicar e integrarlo

en la arquitectura. Las características más destacables de estos sistemas se muestran en la siguiente tabla a modo de resumen:

	Carácter Distribuido Datos	OGSA	Escalabilidad	Tolerante a Fallos	Acceso Toda Información	Lenguaje de Interrogación	Interfaz
RGMA	Bueno	Sí	Buena	Bajo (Punto único de Fallo)	Punto Único	SQL	WS
MDS2	Bueno	No	Media	Medio (Estructura en Árbol)	Punto Único	LDIF	LDAP
MDS4	Bueno	Sí	Media	Medio (Estructura en Árbol)	Punto Único	XPATH	WSRF
DiDA	Bueno	Sí	Regular	Alto (Estructura Grafo)	Múltiples puntos	XPATH	WSRF

tabla 1. Tabla resumen de las características más significativas de los sistemas de Monitorización e Información explicados en este apartado.

Como se puede observar en la tabla, todos los sistemas presentan un carácter distribuido de los datos, al albergar la información en diferentes puntos. Esta característica es muy importante, sobre todo porque repercute en la escalabilidad del sistema.

Centrándonos en las ventajas e inconvenientes que el sistema GMA con su variante R-GMA ofrece, cabría destacar que la utilización de SQL es un punto a favor muy importante, al ser este un lenguaje de consulta muy potente, además de las ventajas que nos aporta el modelo relacional a la hora de organizar los datos. El carácter distribuido de los datos y el uso del modelo relacional proporcionan una buena escalabilidad al sistema, de manera que la información reside en cada recurso implicado y que se proporciona cuando es requerida por los "consumers". Cabe decir que su adecuación a OGSA e interfaz mediante web services coincide con la tendencia que se está siguiendo en las arquitecturas Grid actuales. A pesar de todas estas ventajas, no se considera el sistema adecuado para la arquitectura que se propone como objetivo de esta tesis, pues en los sistemas R-GMA existen puntos únicos de fallo. Si el registro deja de funcionar, todo el sistema queda inoperativo con lo que la tolerancia a fallos es baja. De forma similar, al existir un único punto de conexión para las consultas, las consultas de todos los clientes podrían saturar el sistema.

Con respecto al sistema MDS2, la robustez de este sistema es mayor que la del GMA, al poder interrogarse cualquier GIIS o GRIS del sistema, lo cual permite que, incluso aunque un GIIS deje de funcionar, esto sólo afecte a la parte del sistema que cuelga de éste, con lo que la tolerancia a fallos mejora de manera notable con respecto al GMA. El carácter distribuido de la información también hace escalable el sistema pero hasta cierto punto, pues los protocolos que se utilizan y el sistema de almacenar la información no son muy eficientes a la hora de tratar grandes volúmenes de datos. A pesar de ello, tampoco se considera un sistema adecuado para la arquitectura que se plantea debido a los problemas que presenta. Uno de los principales problemas es debido a su estructura en árbol, que obliga interrogar al servidor LDAP raíz para tener acceso a la información de todo el Grid, con lo cual, al igual que ocurría en el GMA con el registro, solo hay un único punto desde donde interrogar al sistema completo. También es importante indicar que el lenguaje a emplear para las consultas es el LDIF, más complejo y

menos versátil que el SQL. Finalmente, indicar que el MDS2 no está integrado con las especificaciones OGSA, ya que no está diseñado como un sistema orientado a servicios. Además, utiliza como único protocolo de comunicación el protocolo LDAP, que bajo entornos Web podría resultar problemático.

El sistema MDS4 también presenta una robustez mayor que la del GMA, al igual que ocurría con MDS2, al poderse interrogar cualquier IDXSRV del sistema, lo cual permite que incluso aunque un IDXSRV deje de funcionar, esto sólo afecte a la parte del sistema que cuelga de éste. El carácter distribuido de la información permite escalar el sistema hasta cierto punto, ya que el almacenamiento de la información es mediante ficheros XML, con lo que a la hora de gestionar grandes volúmenes de datos puede resultar ineficiente. Una de las ventajas que aporta respecto a MDS2 es su integración con OGSA, ya que existen middlewares como Globus 4 que implementan MDS4 y cumplen las especificaciones OGSA mediante interfaces de WSRF. Además el lenguaje de interrogación es el XPATH, más sencillo que LDIF aunque un poco más complejo que el SQL, con lo que consultas de estructuras complejas pueden resultar problemáticas. Respecto a los problemas que presenta MDS4, destacar que al igual que en MDS2, el acceso a todo el sistema de información solo se puede hacer desde un único punto. A pesar de estas desventajas, se ha considerado este sistema como el sistema a emplear en la arquitectura a desarrollar, sobre el que se realizarán modificaciones para que sea más escalable y para que las consultas sobre estructuras complejas sean más sencillas e intuitivas.

Con respecto al sistema DIDA, la principal ventaja es la tolerancia a fallos, pues desde cualquier IDXSRV se puede consultar toda la información del sistema, por lo que la caída de un nodo no implica que no se pueda acceder a una parte de la información, como ocurría en la estructura de árbol del MDS4. En DIDA los datos se organizan mediante un grafo en el que todos los nodos podrán ser accedidos desde cualquier otro nodo mediante dos caminos. Otra ventaja importante es que al ser un sistema basado sobre MDS4 también mantiene las especificaciones OGSA mediante interfaces WSRF. El lenguaje de consultas es XPATH al estar basado sobre MDS4. A pesar de estas ventajas respecto a MDS4, no se ha contemplado como el sistema adecuado para aplicar en la arquitectura, pues el sistema a aplicar debe de ser una sistema escalable y una de las principales desventajas es la baja escalabilidad, pues una simple consulta implica la interrogación de todos los nodos del sistema, con lo que para sistemas de muchos nodos puede resultar ineficiente, además de presentar los mismos problemas que presenta MDS4 a la hora de gestionar grandes volúmenes de datos.

1.6. Infraestructura de Seguridad en el Grid

Otro punto clave en la definición de las arquitecturas Grid es la infraestructura de seguridad. La seguridad es uno de los puntos más importantes, ya que el objetivo de las infraestructuras Grid y de la plataforma propuesta en éste trabajo, es compartir información y recursos entre diferentes entidades relacionadas mediante OVs, con diferentes dominios administrativos. A esto se suma el uso de una red como Internet, que por su propia naturaleza requiere de las máximas precauciones. En este ámbito se tratarán los problemas de autenticación de usuarios para garantizar tanto que los interlocutores en las comunicaciones son realmente quienes dicen ser, como que se proteja la privacidad de los datos que se transfieren y almacenan, pues al ser Internet una red pública, se debe garantizar que la transmisión se realice mediante canales seguros. Por tanto, en éste apartado se describen los mecanismos actuales que aseguran la privacidad de datos y autenticación de los usuarios en las comunicaciones, mediante tecnologías que pueden ser aplicables sobre los protocolos más comunes que se utilizan en Internet (http, ftp, etc.). Estos protocolos se basan en técnicas criptográficas y por tanto también se introducen conceptos básicos de criptografía.

Además de la autenticación de usuarios y privacidad de los datos en las comunicaciones, en las tecnologías Grid aparecen nuevas necesidades adicionales de seguridad, que también se encuentran presentes en esta tesis. Al coexistir en el Grid recursos diferentes, con sus propios sistemas de administración y políticas de seguridad de acceso a la información o recursos, se debe fijar de alguna manera como se comparten con el resto de miembros de la OV y que no pertenecen a su propio dominio administrativo. Los servicios de autorización Grid son los encargados de especificar las relaciones entre las políticas de seguridad locales de cada recurso con las políticas de seguridad que se gestionan a nivel de OV. Estos dos entornos son diferentes pero se relacionan estrechamente. Por tanto, en este apartado también se expondrán diferentes sistemas de autorización actuales para entornos Grid y se dará una breve explicación de aquellas ventajas e inconvenientes que nos aportan a la hora de realizar la definición de la arquitectura. Por otra parte se justificará el sistema que se empleará para los desarrollos de los servicios o recursos de la arquitectura a la hora de definir las políticas de seguridad.

También, cabe resaltar que, en los entornos médicos, otro aspecto clave en el ámbito de la seguridad es el garantizar la privacidad e integridad de la información que se guarda en dominios administrativos en los que el propietario de la información no es el mismo que el administrador del dominio local, garantizando que sólo sea accesible por los usuarios Grid que estén autorizados para ello y que el administrador local del dominio no sea capaz de acceder a los datos si no está autorizado. Para solventar este tipo de problema, se utilizan técnicas de cifrado de datos. Por tanto, se dará una breve explicación de modelos de cifrado de datos que se pueden aplicar en entornos Grid para solucionar este tipo de problemas.

En resumen, en las siguientes secciones se dará un repaso a conceptos básicos sobre la criptografía de datos, ya que todos los puntos referentes a la seguridad se basan en conceptos relacionados con ésta. A continuación se comentarán aquellas tecnologías actuales que se utilizarán para garantizar la privacidad y autenticación de datos y usuarios en las comunicaciones cuando se utiliza Internet, los sistemas de autorización actuales para entornos

Grid y las técnicas de cifrado de datos que permitirán mantener su privacidad e integridad aunque estén almacenados en dominios administrativos diferentes.

1.6.1. Conceptos Generales Sobre la Seguridad en el Grid

Tal y como se ha comentado en el apartado anterior, los tres bloques relacionados con la seguridad en ésta tesis (autenticación y privacidad de datos en las comunicaciones, sistemas de autorización Grid y modelos de cifrado de datos) se sustentan bajo conceptos relacionados con la criptografía de datos y que por tanto, su conocimiento es necesario para una correcta comprensión de estos tres bloques. En esta sección, se presentarán aquellos conceptos generales relacionados con la criptografía y que son necesarios para el entendimiento a secciones posteriores

1.6.1.1. Criptografía

Como primer concepto general, definiremos criptografía como el estudio de los mecanismos para convertir la información de su forma normal y comprensible a un formato incomprensible, haciéndola ilegible sin el conocimiento de una clave.

En la terminología básica de la criptografía, cabe resaltar algunos conceptos necesarios para la compresión de todo proceso relacionado con la criptografía. Estos conceptos son:

- Texto plano. La información original que será protegida.
- Cifrado. Es el proceso de convertir el texto plano en una forma ilegible.
- Descifrado. El proceso reverso al cifrado, que consiste en recuperar el texto plano de un cifrado.
- "Cipher". Algoritmo para el cifrado y descifrado. La operación exacta de un "cipher" es controlada normalmente por una clave, que no es más que un fragmento de información secreta que caracteriza la forma en que se produce el cifrado. Los protocolos especifican los detalles de cómo los "ciphers" y otras primitivas criptográficas son utilizadas para las tareas de cifrado y descifrado.
- Sistema criptográfico. Un conjunto de protocolos, "ciphers", manejadores de claves y acciones prescritas por el usuario, implementados en conjunto como un sistema.

Tal y como se ha descrito, las operaciones de los "ciphers" son controladas por una clave. Se pueden definir dos tipos fundamentales de algoritmos criptográficos en función del tipo de claves a utilizar por los ciphers: la criptografía de clave simétrica y la criptografía de clave asimétrica o firma digital.

1.6.1.2. Criptografía de Clave Simétrica

Los "ciphers" de clave simétrica utilizan la misma clave para el cifrado y descifrado, o de manera más precisa, la clave utilizada para el descifrado puede ser obtenida con facilidad a partir de la clave utilizada para el cifrado y a la inversa. Este tipo de algoritmos son referidos con otros términos como son criptografía de "clave privada", "una clave" o "clave única".

Los "ciphers" de clave simétrica pueden ser clasificados de forma general en "ciphers" de bloques y "ciphers" de flujo. Los "ciphers" de flujo operan sobre un bit en cada momento, en contraste con los de bloques, que operan sobre un grupo de bits de cierta longitud (un bloque)

de una sola vez. Dependiendo del modo de operación, los “ciphers” de bloques pueden ser implementados como “ciphers” de flujo realimentados o Cipher FeedBack (CFB), en la que cada byte debe ser cifrado. De igual forma, los “ciphers” de flujo pueden trabajar sobre bloques individuales de texto plano. De esta forma, existe alguna dualidad entre los dos tipos. Los “ciphers” de bloques Delivered Ex Ship (DES), International Data Encryption Algorithm (IDEA) y Avanced Encrypted Standard (AES), y el cipher de flujo Ron's Code 4 (RC4) están entre los más conocidos de clave simétrica [117].

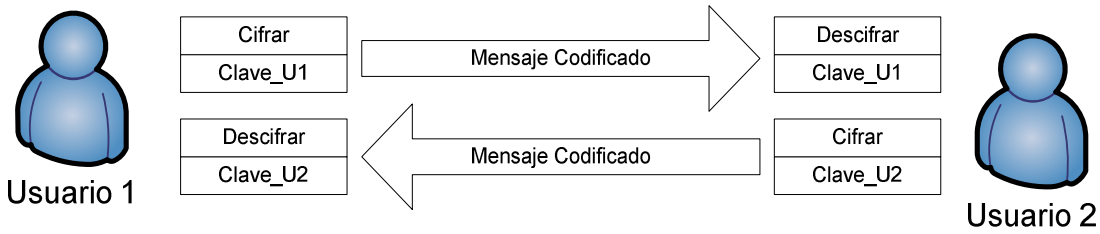

figura 8. Esquema de comunicación mediante Criptografía de Clave Simétrica.

1.6.1.3. Criptografía de Clave Asimétrica

Los “ciphers” de clave asimétrica utilizan un par de claves para el cifrado y descifrado. Las dos claves están relacionadas matemáticamente, aunque generalmente el proceso para obtener una de la otra es un problema computacionalmente intratable. Un mensaje cifrado por el algoritmo utilizando una clave puede ser descifrado por el mismo algoritmo utilizando la clave asociada correspondiente. En cierto sentido, una clave cierra una cerradura (cifrado), mientras que se necesita otra clave diferente para abrirla (descifrado).

No todos los algoritmos de clave asimétrica funcionan exactamente de la misma forma. Los más comunes tienen la propiedad de que las dos partes involucradas tienen dos llaves, ninguna de las cuales (hasta donde se conoce) es deducible a partir de la otra. Esta aproximación se conoce como criptografía de clave pública, debido a que una de las claves del par puede ser publicada sin afectar la seguridad del mensaje. Este esquema de dos claves enlazadas es utilizado en la actualidad por muchos algoritmos para aplicaciones sobre Internet, como por ejemplo el algoritmo Secure Sockets Layer [53] (SSL).

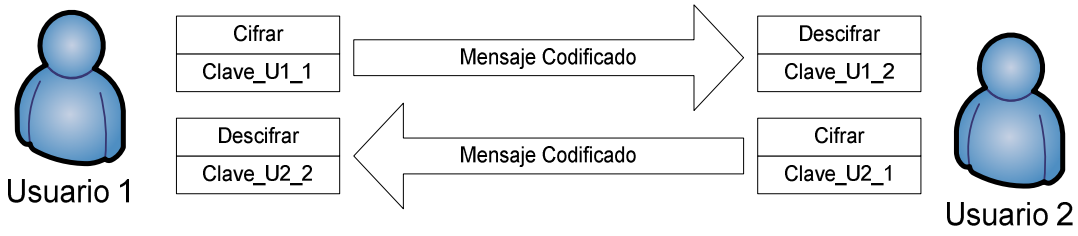

figura 9. Esquema de comunicación mediante Criptografía de Clave Asimétrica.

No existe un algoritmo de clave asimétrica que haya sido demostrado que es seguro ante un ataque matemático. No se ha podido demostrar que sea imposible la existencia de cierta relación entre el par de claves o una debilidad en la operación de los algoritmos que pueda ser utilizada para descifrar el mensaje sin la llave privada. La seguridad de los algoritmos de clave asimétrica se basa en la estimación de la dificultad en resolver el problema matemático subyacente. Estas

estimaciones han ido cambiando tanto con la disminución del coste del poder computacional, como con los nuevos descubrimientos matemáticos. De esta forma, el uso de algoritmos de clave asimétrica no garantiza la seguridad a muy largo plazo. Esta es un área de investigación activa para descubrir y proteger los sistemas de nuevos e inesperados ataques.

1.6.1.4. Firmas Digitales

La firma digital es un método para la autenticación de la información digital análogo a la firma física ordinaria en papel, pero implementado utilizando técnicas criptográficas. Los esquemas de firma digital utilizan la criptografía de clave pública basadas en criptografía de clave asimétrica. La firma digital nos dará la confianza de que la información no ha sido manipulada y sigue siendo la original.

Por razones de eficiencia, es usual que se aplique una función criptográfica de partición (hash) al mensaje antes de firmarlo, es decir, aplicar una función que reduzca el tamaño del mensaje. Esto hace que la firma sea más corta y de esta manera se ahorra tiempo porque los algoritmos de partición son usualmente más rápidos que los de firma. La firma no protege la confidencialidad del mensaje.

El cifrado de un mensaje con una clave privada, cuando se utiliza criptografía asimétrica, crea una firma digital de la persona o entidad que posee la clave privada. Cualquiera que posea la clave pública correspondiente puede descifrar la firma para obtener el mensaje y de esta forma verificar si en efecto corresponde al mensaje original.

Un esquema general de firma digital consiste en tres algoritmos: un algoritmo de generación de claves, un algoritmo de firma y un algoritmo de verificación.

Algunos de los algoritmos de firma digital más conocidos son: RSA, Digital Signature Algorithm (DSA), Elliptic Curve Digital Signature Algorithm (ECDSA), ElGamal [118].

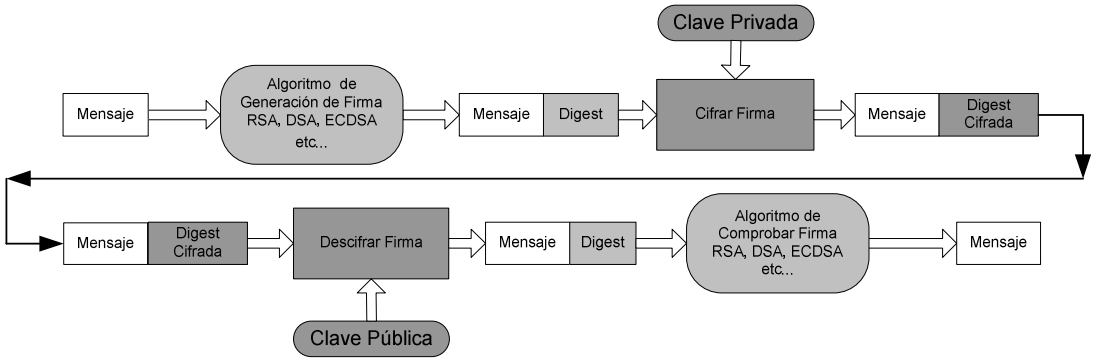

figura 10. Esquema para la Firma Digital de un mensaje.

1.6.1.5. Código de autenticación de mensajes (MAC)

El código de autenticación de mensajes o Message Authentication Code (MAC) se obtiene al aplicar una clave secreta al "digest" de un mensaje (representación del texto en forma de una cadena de dígitos, creado con una formula de "hashing" de una sola dirección). Para hacer esto se combina el mensaje M con una clave privada K y se les aplica un algoritmo "hash" h(M,K), de manera que la función, dada la clave K y el mensaje M, retorna un resultado X. El resultado X del proceso se envía y al llegar a su destino se comprueba la integridad de la clave privada K.

En caso positivo se demuestra que el origen del mensaje es el que tiene la parte privada de la clave K.

Al cifrar un "digest" de un mensaje con una clave privada se genera una firma digital y por tanto los MAC sirven para autenticar el origen de los mensajes así como también su integridad, como cualquier firma digital. Este método evita los ataques del tipo Man-In-The-Middle (MITM) porque un interceptor malicioso necesita conocer la clave secreta para sustituir el "digest" de un mensaje modificado. Lógicamente, para que este método funcione, se necesita compartir la clave de descifrado entre el emisor y el receptor del mensaje.

1.6.2. Autenticación de Usuarios y Privacidad de Datos

Una vez revisados los conceptos generales sobre los que se van a apoyar los siguientes apartados, el primer aspecto a tratar dentro de la seguridad Grid es que en todas las comunicaciones se debe garantizar la autenticación de los usuarios (garantía de que los usuarios que se conecten a un servicio sean realmente quienes dicen ser), para así posteriormente aplicar las políticas de seguridad pertinentes que se definan en el dominio local. Por otra parte, también se debe garantizar la integridad de los datos que se intercambien en la comunicación mediante canales seguros. Este problema está bien resuelto en la actualidad mediante protocolos como el Secure Sockets Layers [53] (SSL) y el uso de certificados digitales que cumplen el estándar X509. A continuación se da una breve explicación sobre el algoritmo SSL y el estándar X509, precedido de un breve resumen de lo que es una infraestructura de clave pública, base que se utilizará en la arquitectura propuesta en esta tesis.

1.6.2.1. Infraestructuras de Clave Pública (PKI)

En criptografía, una infraestructura de clave pública [54] (PKI) es una combinación, por un lado de hardware y software, y por otro de políticas y procedimientos que permiten asegurar la identidad de los participantes en un intercambio de datos. PKI utiliza criptografía asimétrica.

El termino PKI se utiliza para referirse tanto a la autoridad de certificación y al resto de componentes, como para referirse, de manera más amplia y a veces confusa, al uso de algoritmos de clave pública en comunicaciones electrónicas. Este último significado es incorrecto, ya que no se requieren métodos específicos de PKI para usar algoritmos de clave pública.

Una PKI permite a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes. En general, una PKI consiste en un software para los clientes, un software de servidor (por ejemplo una CA o autoridad de certificación), hardware (por ejemplo, tarjetas inteligentes) y unos procedimientos operacionales. Un usuario puede firmar digitalmente mensajes usando su clave privada, y otro usuario puede validar dicha firma (usando la clave pública del usuario contenida en el certificado que ha sido emitido y firmado por una autoridad de certificación de la PKI). Esto permite a dos (o más) entidades establecer una comunicación que garantiza la confidencialidad y la integridad del mensaje y la autenticación de los usuarios sin tener que intercambiar previamente ninguna información secreta.

Los sistemas basados en infraestructuras PKI se basan en las cadenas de certificado para establecer la identidad de las partes. Un certificado puede ser emitido por una autoridad

certificadora, cuya legitimidad fue establecida de antemano para esos propósitos a través de un certificado emitido, a la vez, por una autoridad certificadora de más alto nivel. El resultado de esta práctica produce una jerarquía de certificados compuesta a menudo por varias computadoras, una o más organizaciones, y diversos paquetes de software de varias fuentes que operan entre sí. Por esta razón, la utilización de estándares es imprescindible para el funcionamiento de una PKI y la utilización de estándares públicos en infraestructuras PKI destinados a operar en producción es más crítica aún. Una parte considerable de esta estandarización en esta área es llevada a cabo por el grupo de trabajo IETF PKIX [55].

Los entornos PKI empresariales, a menudo están ligados a un esquema de directorio empresarial en el cual la clave pública de cada usuario se almacena, embebida en un certificado, junto con otros detalles personales como pueden ser el número de teléfono, la dirección de e-mail, el departamento, etc. Actualmente, la tecnología más utilizada para directorios es el LDAP y de hecho, el formato más común de certificados es el X509, que tiene su raíz en el predecesor de LDAP, el esquema de directorio X500.

1.6.2.2. Estándar X.509

En criptografía, X509 es un estándar para la infraestructura de clave pública PKI. X509 especifica, entre otras cosas, los formatos estándares para los certificados de clave pública y un algoritmo de validación del camino de certificación.

La versión 1 de X509 se emitió en 1988, en asociación con el estándar X500, asumiendo un sistema estrictamente jerárquico de autoridades certificadoras (CAs) para la emisión de certificados. Este sistema contrasta con otros modelos de confianza, como Pretty Good Privacy (PGP), en los que cualquiera, sin ser una autoridad certificadora especial, puede firmar los certificados de otros.

La versión 2 de X509 introdujo el concepto de identificadores únicos de sujeto y emisor, y con ello la posibilidad de reutilizar los nombres del sujeto y/o el emisor a través del tiempo. Debido a que la mayor parte de los autores recomienda que no se reutilicen los nombres y que los certificados no deben hacer uso de identificadores únicos, la utilización de la versión 2 no ha sido muy extendida.

Finalmente, la versión 3 de X509, emitida en 1996, es la más reciente y da soporte a las extensiones, por las que cualquiera puede definir información adicional a incluir en el certificado. Entre las extensiones que más se han popularizado, están las que limitan el uso de las claves a un propósito particular y los nombres alternativos, que permiten asociar otras formas de identidad con la clave pública. Las extensiones pueden ser catalogadas como críticas para indicar que la extensión debe ser chequeada y de uso obligatorio. Por ejemplo, si un certificado contiene una extensión crítica que restringe el uso de la clave a la firma de certificados, este certificado será rechazado si se presenta al iniciar una comunicación SSL porque la extensión indica que la clave no puede ser usada para este propósito.

El estándar X509 define qué información puede ir en un certificado y describe el formato de datos en el que debe estar escrito el certificado. La estructura de cualquier certificado digital X509 v3 es la siguiente:

- Sección del Certificado
    - Versión. Indica la versión del estándar X509 que se aplica a este certificado, lo cual afecta a la información que puede ser especificada en el mismo. En este caso es la versión 3.
    - Número de serie. La entidad que creó el certificado es responsable de asignarle un número de serie que lo distinga de cualquier otro certificado emitido por la misma entidad. Esta información se utiliza, por ejemplo, para revocar un certificado, colocando su número de serie en una Lista de Revocación de Certificados o "Certificate Revocation List" (CRL).
    - Identificador del algoritmo de firma. Identifica el algoritmo utilizado por la CA para firmar el certificado.
    - Nombre de emisor. Contiene el nombre X500 de la entidad que firmó el certificado, que normalmente es una CA. El uso de este certificado implica confiar en la entidad que firmó el certificado. En algunos casos el emisor firma su propio certificado, como por ejemplo, las CA de alto nivel.
    - Periodo de validez. Cada certificado es válido solamente durante un período limitado de tiempo. Este período es descrito por una fecha de inicio y una de terminación y puede ser tan corto como unos segundos y tan largo como se deseé. El período de validez se selecciona considerando factores como la fortaleza de la clave privada utilizada para firmar el certificado. El periodo de validez es el lapso de tiempo que las entidades pueden garantizar la seguridad de un certificado si la clave privada asociada no se ve comprometida.
    - Nombre del sujeto. Contiene el nombre de la entidad cuya clave pública es identificada por el certificado. Este nombre utiliza el estándar X500, por lo que se espera que sea único en Internet. Generalmente se utiliza una estructura llamada Nombre Distintivo de la entidad "Distinguished Name" (DN), como por ejemplo: CN=J. Damià Segrelles, OU=Departamento de Sistemas Informáticos y Computación (DSIC), O=UPV, C=ES. Este nombre en concreto se refiere al nombre común del sujeto ("Common Name" o CN), a la unidad organizacional ("Organizational Unit" u OU), a la organización ("Organization" u O), y al país ("Country" o C).
    - Información de la clave pública del sujeto. Este apartado contiene la clave pública de la entidad nombrada en este certificado, junto al identificador del algoritmo que especifica el sistema criptográfico utilizado en la generación de la clave y los parámetros asociados a la misma, como por ejemplo el algoritmo de clave pública y la clave pública del sujeto.
    - Extensiones (Opcional). Información adicional sobre el certificado.
- Algoritmo de firma del certificado. Esta sección contiene el identificador del algoritmo utilizado para firmar este certificado.
- Firma del certificado. Comúnmente es una firma llevada a cabo por una CA.

X.509 incluye también estándares para la implementación de Listas de Revocación de Certificados, un aspecto que es descuidado a menudo en los sistemas PKI. La forma recomendada de verificar la validez de un certificado es a través de un protocolo conocido como Protocolo en Línea de Estado de Certificado ("Online Certificate Status Protocol" u OCSP).

Existen múltiples formatos para el almacenamiento e intercambio de certificados y credenciales, y de su conocimiento y utilización depende mucha de la funcionalidad y de la usabilidad de las aplicaciones. Entre los formatos más comunes para certificados X.509 se destacan:

- PEM. Define una forma simple de almacenar y transferir certificados o claves privadas codificados con el estándar Base64, el cuerpo del certificado está encerrado entre las cabeceras "-----BEGIN CERTIFICATE-----" y "-----END CERTIFICATE-----". El formato Base64 está definido en el estándar de Internet RFC 1421 con el objetivo de proporcionar una alternativa a los estándares codificados de forma binaria. Es un estándar de codificación que utiliza un formato imprimible que facilita exportar los certificados a otras aplicaciones por email o a través de cualquier otro mecanismo en texto plano. Se presenta en archivos con extensiones .pem.
- PKCS #12. Es un estándar que define un formato utilizado para intercambiar objetos públicos y privados en un solo archivo. Evolucionó del estándar Personal inFormation eXchange (PFX). Se presenta en archivos con extensiones .pfx, .p12 que pueden contener certificados públicos y/o claves privadas protegidas con una palabra clave.

1.6.2.3. Secure Sockets Layers (SSL)

Secure Sockets Layer [53] (SSL) es un protocolo que permite establecer comunicaciones seguras entre aplicaciones clientes y servidores. SSL cifra los datos previamente a su transmisión, y una vez recibidos en la otra parte, son descifrados antes de ser procesados. Este proceso ocurre en los dos sentidos, lo cual significa que tanto la aplicación cliente como la aplicación servidor cifran todo el tráfico antes de enviar datos por el canal de comunicación SSL. Este protocolo se compone de dos capas y funciona de la siguiente manera:

- La primera capa se encarga de encapsular los protocolos de más alto nivel.
- La segunda capa se llama SSL Handshake Protocol y se encarga de la negociación de los algoritmos de cifrado y también de la autenticación entre el cliente y el servidor.

Este método proporciona un nivel de seguridad adecuado, ya que por cada conexión que se hace, el servidor envía una clave diferente. En el caso de que alguna entidad se haga fraudulentamente con la clave de la conexión, únicamente podrá cerrar la conexión correspondiente a dicha clave.

Cuando se logra este primer proceso que únicamente crea la sesión, actuarán a partir de ese momento los protocolos de capa 7 del OSI (capa de Aplicación), aunque todas las transacciones se realizarán sobre el SSL.

Otra característica importante del protocolo SSL es la autenticación. En los momentos iniciales de negociación de una nueva conexión segura, el servidor presenta un certificado con un conjunto de credenciales que permiten al cliente comprobar la identidad del servidor. En

ciertos casos, el servidor puede solicitar también un certificado que pruebe la identidad del cliente, lo cual se conoce como autenticación del cliente.

SSL se basa en criptografía asimétrica y en el concepto de los certificados. La versión estandarizada por el Internet Engineering Task Force (IETF) se conoce como Transport Layer Security (TLS).

Una de las ventajas más conocidas de SSL es que funciona entre la capa TCP ("Transmisión Control Protocol") y la capa de aplicación, por lo que es muy fácil emplearlo en la protección de protocolos de la capa aplicación (HTTP, FTP, SMTP, etc.) sin tener que realizar cambios importantes en los mismos.

1.6.2.4. Grid Security Infrastructure (GSI)

La infraestructura GSI [56][57] es la infraestructura para la seguridad que utiliza Globus en todas sus versiones (Globus 2, Globus 3, Globus 4). GSI posibilita a las infraestructuras desplegadas garantizar la autenticación de usuarios y privacidad de datos en las comunicaciones que se realicen a través de la red. GSI requiere de una infraestructura de clave pública PKI, puesto que los mecanismos que establece se basan en la correcta gestión de certificados X509.

GSI asigna a cada recurso y cliente Grid una serie de credenciales mediante certificados X509. Los clientes crean, a la hora de realizar una comunicación, unas credenciales temporales a partir de las que disponen, válidas para un tiempo determinado y mucho menor que el tiempo de vida del certificado. Estas credenciales son también un certificado X509 llamado "Proxy". Con estos certificados se crean comunicaciones seguras entre los clientes y los recursos, además de garantizar la autenticación de los usuarios. Para ello se utiliza Transport Layer Security protocol (TLS) que soporta las comunicaciones seguras a través de HTTPs y que está basado en SSH.

Además de estas dos características también permite configurar para cada recurso una política de seguridad según el usuario que accede y las credenciales que presente, aspectos que serán expuestos en la siguiente subsección referente a los sistemas de autorización Grid.

1.6.2.5. Aplicación de la Autenticación de Usuarios y Privacidad de datos

En los apartados anteriores se han expuesto los conceptos y tecnologías referentes a la autenticación y privacidad de datos utilizados en la actualidad. En este apartado se va a desarrollar la aplicabilidad de estas tecnologías y conceptos dentro de los desarrollos de esta tesis.

Las tecnologías y conceptos que se aplican son los referentes a las PKI, certificados x509, SSL y GSI. GSI [56][57] proporciona mecanismos de autorización y autenticación de usuarios, mediante la emisión de certificados X509, para todos los recursos y usuarios Grid involucrados en un despliegue Grid. Esta será la tecnología a aplicar en los desarrollos de esta memoria. Para utilizar GSI se requiere de una gestión de certificados x509, y por tanto de una PKI. Por otro lado, las conexiones mediante GSI son seguras porque se utiliza TSL, que es la versión estandarizada por el Internet Engineering Task Force (IETF).

Por tanto, GSI será la infraestructura utilizada en los desarrollos de esta tesis para garantizar la autorización y autenticación de usuarios. Es el mecanismo utilizado por todas las Toolkit de Globus (GT2, GT3 y GT4), además GT4 cumple con las especificaciones OGSA/WSRF, que

son las empleadas para los desarrollos de ésta tesis, tal y como se ha comentado en el apartado 1.4.3.3 referente al estado del arte de las tecnologías Grid.

1.6.3. Sistemas de Autorización Grid

Tal y como se ha mencionado en la introducción de la sección 1.6, el segundo bloque referente a la seguridad es el correspondiente a los sistemas de autorización Grid. Una definición de autorización Grid podría ser la concesión o denegación de permisos a un usuario Grid perteneciente, a una OV, para llevar a cabo una determinada acción sobre un recurso determinado, lo cual es aplicable tanto a los usuarios como a todos los procesos involucrados en operaciones seguras. Los dos conceptos básicos de los sistemas de autorización Grid son:

- Organización Virtual (OV). Entidad compuesta por un conjunto de usuarios, instituciones y recursos, en la que todos ellos pertenecen a un mismo dominio administrativo virtual [59].
- Proveedor de Recursos (RP). Entidad o dispositivo que ofrece recursos (CPUs como recursos computacionales, recursos de almacenamiento, etc.) a otros miembros pertenecientes a una misma OV, de acuerdo con unas políticas de colaboración.

En un despliegue Grid, el concepto de autorización es muy importante para facilitar la administración y proporcionar la coherencia del sistema. Para ello se despliegan sistemas que gestionan la autorización de los usuarios Grid a los recursos desplegados en la infraestructura. La autorización de un usuario Grid a un recurso está ligada estrechamente con el concepto OV, ya que generalmente los permisos se definen a partir de ésta. Los administradores de las OVs conceden los permisos a nivel de OV a los usuarios Grid, y además se encargan de establecer acuerdos con los RPs para el acceso a los recursos. Los RPs, a su vez, implantan sus propias políticas de seguridad local que aplicarán en función de las credenciales que el usuario Grid presente y los acuerdos establecidos con las OVs correspondientes.

En un sistema de autorización Grid se requiere una serie de requisitos básicos. En primer lugar, uno de los principales requisitos sería definir la estructura jerárquica de las OVs (grupos, subgrupos, etc.) con las que se va a trabajar y, en segundo lugar, definir ciertas características (roles y capacidades) de los usuarios Grid en las OVs. Estos requisitos serán los que definirán las credenciales que se presentarán a los RPs por parte de los usuarios Grid.

En los apartados que vienen a continuación, se describen más detalladamente aquellos requerimientos básicos que un sistema de autorización Grid debe tener en cuenta. Posteriormente se define cómo deben estructurarse las OVs, los atributos de usuarios Grid que deben definirse, y los formatos actuales para presentar las credenciales de los usuarios Grid a los RP. Por último se incluye una breve explicación de cómo estos han sido implementados en sistemas de autorización Grid actuales, mostrándose también aquellas deficiencias y ventajas que presentan con respecto a las necesidades del presente trabajo.

1.6.3.1. Requerimientos básicos de los sistemas de Autorización Grid

En este apartado se realiza un análisis preliminar de los principales requerimientos de un sistema de autorización Grid [58]. La primera condición, para el acceso a los recursos que un usuario Grid debe cumplir, es la de pertenencia a una OV. En general, un usuario Grid puede pertenecer a varias OVs, asignándole en cada OV unos roles determinados. La pertenencia a una OV no debe ser relevante para otras OVs.

Una vez el usuario Grid es autenticado, el usuario puede ser incluido o excluido de una OV por los administradores correspondientes. En este proceso se le asignan los roles correspondientes para cada OV. Esto implica que, una vez el administrador admita a un usuario en una OV, los permisos y los requerimientos de seguridad deben de ser concedidos de manera automática. Por otra parte, los administradores de los RPs deben ser capaces de conceder una autorización local a estos usuarios basándose en las credenciales que éste presente (pertenencia a una OV, pertenencia de grupos dentro de la OV, roles, su propia identidad. etc.). Este mecanismo de autorización debe comprobar la identidad del usuario Grid en el RP donde se ejecuta. También es importante que todas las características presentadas en las credenciales puedan ser verificadas.

Los requerimientos que debe cumplir un sistema de autorización Grid vienen condicionados por el gran número de usuarios y recursos que puede tener un despliegue Grid. La infraestructura a implantar debe controlar a éstos a nivel de OV, y de una manera centralizada para ser escalable. Por otra parte, la autorización de los usuarios Grid en los diferentes RP deberá ser acordada entre ambas partes (OV y RP), y a su vez los RP deberán implementar dentro de su dominio la autorización correspondiente del usuario Grid al recurso.

1.6.3.2. Estructura de una Organización Virtual

Tal y como se ha comentado en apartados anteriores, el concepto de OV es muy importante en los sistemas de autorización Grid, ya que a nivel de OV se definen las características que determinarán las credenciales que el usuario Grid presentará a las RPs y condicionarán sus derechos de acceso.

Una de las características importantes que determinará las credenciales a presentar a las RP para su acceso será la pertenencia del usuario dentro de la propia estructura de la OV. Cada OV puede tener una estructura compleja y jerárquica, organizada mediante grupos y subgrupos con el fin de diferenciar los usuarios en función de las tareas que vayan a realizar dentro de la OV. Desde el punto de vista administrativo, y por razones de escalabilidad, cada grupo puede tener delegada la administración, de manera independiente, a diferentes administradores. El administrador de cada grupo puede crear subgrupos y conceder derechos de administración a estos subgrupos, pero no pueden modificar las características de otros subgrupos sobre los cuales no tenga derechos administrativos.

Se asume que la pertenencia a un grupo implica la pertenencia a todos los grupos antecesores, hasta llegar al nivel de grupo más alto dentro de la jerarquía, es decir, la propia OV. Por tanto, sólo el administrador de la OV será el que tenga permisos para incluir nuevos usuarios Grid en la OV. Además, con el fin de permitir a los RPs asociar correctamente los permisos a los miembros de las OV, un usuario Grid debe ser miembro de algún grupo. Para implementar esta característica, todos los grupos son anexados a la información que el usuario presenta a los RPs.

Por tanto, en general, la estructura de una OV se puede representar como un grafo acíclico dirigido (DAG), en donde los grupos son los vértices del grafo y la pertenencia a los subgrupos en los grupos serían las aristas con una dirección (“Pertenece a”). Se puede considerar como caso especial, que la propia OV sea un grupo que contiene a todos los grupos de la OV.

1.6.3.3. Atributos de Usuario en la Organización Virtual

Como se ha visto en el apartado anterior, un grupo es básicamente un conjunto de usuarios o subgrupos. En general, un usuario Grid puede pertenecer a tantos grupos como desee dentro de la jerarquía de una OV. Además de la pertenencia a una OV, existen otras características dentro de la jerarquía de una OV que definen las credenciales que un usuario Grid presenta a los RPs, aunque vienen condicionadas por las OVs. Entre estas características, destacan los roles y las capacidades de los usuarios Grid dentro de la OV, que pueden definirse tanto a nivel de grupo como a nivel de OV (que sería el grupo que englobaría a todos los demás) y dan una mayor flexibilidad al modelo.

Los roles son utilizados para presentar capacidades de usuario más específicas, como por ejemplo, capacidades de administración. Por esta razón los roles están limitados a grupos determinados, pudiendo, un usuario tener un rol para un grupo determinado que no es válido para otros grupos de la OV. La principal diferencia entre roles y grupos, es que un usuario Grid puede seleccionar los roles a ser incorporados en sus credenciales, siempre que se especifiquen todos sus grupos. Esto hace posible tener grupos que rebajen los privilegios de usuarios.

Las capacidades pueden ser expresadas con texto libre, describiendo características especiales del usuario Grid, como por ejemplo, que es el usuario más antiguo de la OV.

Definidos la estructura de una OV (grupos y subgrupos) y los conceptos de roles y capacidades, se define un atributo como la información que incluye datos sobre la pertenencia a un grupo, roles y capacidades de un usuario Grid en una OV. Estos atributos pueden tener un carácter indefinido o temporal (se podría especificar que un usuario sólo pudiera ejecutar un proceso en un recurso en una franja horaria determinada).

La implementación de los manejadores de los atributos de un usuario Grid perteneciente a una OV en un RP, debe reflejar los acuerdos entre la OV y los RPs correspondientes. Sin embargo, debería ser posible para los RPs tener el control final sobre los permisos del recurso, superponiendo la concesión de permisos dados por OVs si se da el caso (Ej. prohibición de usuarios no deseados). Como consecuencia, para conseguir la traza de un usuario tanto a nivel de OV como a nivel local de un RP, los usuarios deben presentar las credenciales a los RPs junto con la información de autorización mediante los atributos correspondientes.

La traducción de estas políticas en el nivel de OV mediante los atributos correspondientes (grupos, roles, capacidades) debe de traducirse a políticas de seguridad locales de los recursos mediante los acuerdos entre las OVs y los RPs. Además debe ser posible anteponer las políticas de los RPs a las de las OV para así poder mantener a nivel del administrador local el control final del acceso a los recursos.

Otro tipo de requerimiento para los sistemas de autorización es que los servidores de autorización no deben ser, en la medida de lo posible, únicos en el despliegue, evitando puntos únicos de ruptura a nivel de la OV y, por supuesto, todas las comunicaciones con estos servidores deben ser seguras, tanto a nivel de autenticación de usuario como a nivel del uso de canales seguros de comunicación.

1.6.3.4. Formato de Credenciales

Tal y como se ha visto en los apartados anteriores, cada usuario Grid perteneciente a una OV presenta sus credenciales mediante una serie de atributos (grupos, roles y capacidades). Los valores combinados de todos los atributos formados por las tuplas (grupos, roles y capacidades) forman el llamado FQAN [60] ("Fully Qualified Attribute Name").

Los FQANs para un usuario Grid pueden representarse como una secuencia de nombres de grupos a los que pertenece, en la que para cada grupo el usuario puede tener una serie de roles que se presentan junto con la credencial, además de las capacidades del usuario para cada grupo y rol. En general un FQAN, basado en el RFC 3281-style Attribute Certificates [61], tiene la estructura:

/OV[/group[/subgroup(s)]][/Role = role][/Capability = cap]

Un ejemplo mas concreto podría ser un usuario Grid que tiene el rol de administrador en el grupo GRyCAP de la OV UPV:

/UPV/GRyCAP/Role=Administrador.

1.6.3.5. Arquitecturas de Sistemas de Autorización Grid Actuales

En la actualidad existen diferentes enfoques, con respecto a la arquitectura de los sistemas de autorización que cumplen los requisitos expuestos en los apartados anteriores, referentes a los requerimientos básicos de un sistema de autorización Grid, la estructura de una OV, los atributos de usuario en la OV y el formato de las credenciales.

Además se definen en los entornos Grid dos categorías de información referentes a la autorización en los sistemas Grid de los recursos, que son:

- Información general respecto a las relaciones de los usuarios Grid definidos en una OV. Esta información incluye los grupos a los que pertenecen los usuarios Grid definidos y aquellos roles y capacidades que definirían las credenciales administrativas de los usuarios, y que éstos deberán presentar a la hora de interactuar con un RP para resolver sus necesidades.
- Información relativa a lo que se permite hacer a un usuario Grid en un RP en función de las credenciales que presente. Esta información dependerá de su pertenencia a una OV y los contratos establecidos entre la OV y el RP implicados.

Una de las principales diferencias entre los sistemas de autorización Grid actuales, se encuentra en el carácter distribuido o centralizado de los componentes. Los privilegios de acceso a los recursos para un usuario Grid pueden estar definidos y guardados, bien en un servicio especifico de una manera completamente centralizada, o bien en el propio RP. La ubicación de estos dos tipos de información es lo que ha provocado diferentes enfoques a la hora de implementar diferentes sistemas de autorización Grid en la actualidad.

En los apartados siguientes se presentarán las diferentes arquitecturas para sistemas de autorización Grid que se están utilizando en la actualidad, intentando reflejar las diferencias más significativas entre éstas. Además se describirán las deficiencias con respecto a las necesidades actuales, así como aquellos puntos relevantes para los desarrollos en la ésta tesis.

1.6.3.5.1. Sistema de Autorización Grid del European DataGRID Project (EDG)

El proyecto European DataGrid [62] (EDG) se basa en GSI e implementa sobre esta infraestructura un sistema de autorización Grid. En EDG, al basarse sobre GSI, los usuarios Grid generan un certificado de corta duración "Proxy" que es utilizado para acceder a los recursos, en nombre del usuario.

En las primeras versiones de EDG, cada OV guardaba la información general respecto a los atributos de los usuarios Grid definidos en una OV en un servidor LDAP, implicando toda la información referente a la pertenencia de usuarios a diferentes OVs, roles y capacidades. En la implementación EDG test-bed 1 no se contemplaba la posibilidad de guardar información referente a los roles y capacidades de los usuarios, a pesar de que la distinción entre los diferentes usuarios Grid a la hora de acceder a un RP era manejada en los propios RP. Los RPs, periódicamente preguntaban a los servidores LDAP la información relativa a los usuarios de las OVs, generando una lista de usuarios Grid no deseados o restringidos, y un mapeo de los usuarios Grid permitidos a usuarios locales del RP a través del fichero "grid-mapfile". Este mapeo se realiza a partir de la identidad del usuario Grid obtenida del certificado "proxy" que presenta, y asociándolo a usuarios locales del RP correspondientes. El sistema de autorización tal y como se definió era "booleano", es decir, el RP permitía o no el acceso. Por tanto, el grado de autorización era muy simple y la diferenciación a nivel de usuario sólo se podía hacer a nivel local del recurso. Este mecanismo no cumple las condiciones básicas de un sistema de autorización Grid adecuado, en el que se debería permitir el control fino de políticas de seguridad tanto a nivel de OV como a un nivel local.

Los problemas principales que tenía este sistema de autorización Grid son los siguientes:

- No se soportan grupos y roles a nivel de OV, considerándose simplemente la pertenencia.
- Los usuarios de la OV son mapeados directamente a usuarios locales del RP mediante el fichero grid-mapfile.
- La granularidad de autorización es muy gruesa, al ser un mapeo directamente a usuarios locales y no considerarse los grupos y roles a nivel de OV.
- El manejo del fichero grid-mapfile es poco flexible y escalable en entornos con un gran número de usuarios y OVs.
- La organización de políticas globales es compleja.

Todos los problemas que se planteaban en este primer sistema de autorización de EDG eran referentes a la escasez de control de los usuarios a nivel de OV, ya que en los RP las políticas de seguridad son propias y no dependen de las OV sino de los contratos que se establezcan con éstas. Por ello, se desarrolló el servicio Virtual Organization Membership Service (VOMS), el cual es un enfoque centralizado para almacenar información referente a los atributos de los usuarios Grid a nivel de OV, además de permitir una mayor escalabilidad ante el gran número de usuarios Grid que pueden existir en un despliegue.

Al igual que sucedía con el primer sistema de autorización planteado en EDG, en este segundo sistema se debe generar un certificado "Proxy" [63], el cual tendrá las credenciales que se utilizarán para el acceso a los RPs. En el caso de VOMS, el certificado se genera mediante un servidor, el cual incluye toda la información referente al usuario Grid a nivel de la OV de

acuerdo el formato RFC3281-style atribute Certificates [57][64]. Este certificado es firmado por el propio servidor VOMS para probar la autenticidad de los atributos. Utilizando el "proxy" se accede a los RPs en los cuales se ha implementado un "Gatekeeper" que se encargará de interpretar las credenciales correspondientes presentadas por el "proxy" para mapearlas a las políticas locales. Esta interpretación toma en cuenta la información anexada al certificado correspondiente a los grupos, roles y capacidades del usuario Grid en la OV, con la cual se esta accediendo. Para ello se ha creado para cada RP un servicio de gestión de las políticas locales llamado Local Credential Authorization Service [65] (LCAS) encargado de coger toda la información y mapearla a políticas locales.

Por ello, en los siguientes apartados se explicarán las características más importantes del servicio VOMS desarrollado en EDG así como las diferencias con otros sistemas alternativos cuyo objetivo es similar, con diferentes aproximaciones.

1.6.3.5.2. Virtual Organization Membership Service (VOMS)

Virtual Organization Membership Service (VOMS) fue desarrollado por los proyectos European DataGrid y DataTAG [66][67]. Tal y como se ha descrito anteriormente se desarrollo para solucionar los problemas y limitaciones que existían en el uso de servidores LDAP para la implementación de sistemas de autorización completos, al menos en la parte referente al manejo de información de los atributos de los usuarios Grid a nivel de OV.

VOMS utiliza la autenticación y autorización basada en GSI. Todos los datos manejados por VOMS son almacenados en un sistema relacional de base de datos (RBMS). Concretamente, el backend que se utiliza en EDG es MySQL [68].

El sistema VOMS contempla los siguientes perfiles de usuario:

- Usuarios Clientes: Contactan con el servidor presentando un certificado de usuario y obtienen la lista de grupos, roles y capacidades del usuario.
- Usuario Servidor: Recibe peticiones de los clientes y retornan información sobre el usuario. Por ejemplo, cuando un usuario cliente presenta una petición con un certificado, el usuario servidor le retorna toda la información asociada al certificado que ha presentado (roles, grupos y capacidades).
- Administradores Clientes: Administran la OV agregando usuarios, creando nuevos grupos, cambiando roles, etc.
- Administradores Servidor: Aceptan las peticiones desde los administradores clientes y actualizan la base de datos.

Las operaciones del sistema VOMS tanto para los usuarios como para los administradores son los siguientes:

Por parte del usuario

La primera operación que un usuario debe realizar para entrar en el sistema es la generación de un certificado "proxy" [69] que será utilizado para acceder a los recursos desplegados en el Grid. Para ello se utiliza el comando "voms-proxy-init" el cual recoge información de un usuario Grid de un servidor VOMS presentando un certificado de usuario. Toda la información es incluida en un certificado tipo RFC 3281-style Attribute Certificates [57][61] firmado por el propio servidor VOMS.

El procedimiento para la realización de esta operación es la siguiente:

1. El usuario y el servidor VOMS se autentican mutuamente utilizando sus certificados (Existen APIS que soportan este tipo de acciones).
2. El usuario envía una petición firmada al servidor VOMS.
3. El servidor VOMS verifica la identidad del usuario y chequea sintácticamente la petición realizada.
4. El servidor VOMS retorna al usuario la información pedida, firmada por el propio servidor VOMS.
5. El usuario chequea la validez de la información recibida.
6. Opcionalmente, el usuario repite esta operación en diferentes servidores VOMS.
7. El usuario crea un certificado “proxy" conteniendo toda la información recibida del VOMS Server en una extensión no crítica. Estas extensiones están disponibles para su uso por parte de terceros y son ignoradas por los sistemas más simplificados que no las implementan.

A nivel de RP, toda la información referente a la autorización necesita ser extraída desde el certificado “proxy" y combinarse con las políticas locales del RP con el fin de tomar las decisiones de autorización pertinentes. En los sistemas Grid basados en Toolkits de Globus, el “Gatekeeper”, es el interfaz Grid que primero chequea la validez del certificado ”proxy", y posteriormente procesa la información referente a la autorización.

En EDG y DataTAG se utiliza el sistema LCAS para tomar las decisiones de autorización, de manera que se puede utilizar el sistema de los ficheros “grid-mapfile” combinado con atributos de certificados firmados por servidores VOMS.

Por parte del administrador

Para la administración del servidor se soporta el protocolo SOAP [70] como protocolo de conexión, por lo que resulta fácil cumplir las especificaciones OGSA. La parte de administración consta de cinco rutinas agrupadas en servicios, que son los siguientes:

1. Core. Funcionalidad básica para clientes. (Ejemplo editar y listar usuarios).
2. Admin. Funciones para administrar la base de datos VOMS (p.e. altas, bajas y modificaciones de usuarios).
3. History. Proporciona funciones para auditar todos los cambios que se puedan realizar en la base de datos VOMS.
4. Request. Permite manejar peticiones de nuevos usuarios y cambios.
5. Compatibility. Provee un acceso simple a la lista de usuarios.

Actualmente se dispone de dos interfaces para interactuar con el servidor VOMS, uno mediante Web y otro mediante línea de comandos.

1.6.3.5.3. Community Authorization Service (CAS)

El “Community Authorization Service” [72] (CAS) fue desarrollado por el equipo de Globus con un objetivo similar al de VOMS en EDG. CAS está soportado por toolkits de Globus (GT2, GT3 y GT4).

Las dos grandes diferencias entre VOMS y CAS son:

- CAS no emite certificados con atributos, pero crea un nuevo “proxy" que contiene el servidor CAS como “subject” del certificado y la información referente al usuario (DN y atributos). El DN y los atributos son incluidos como una extensión. Como consecuencia, los servicios que no son compatibles con CAS no pueden determinar la identidad del usuario para quien el servidor CAS ha generado el certificado. Por el contrario, VOMS utiliza un certificado “proxy" completamente estándar, agregando en una extensión no crítica la información referente a la autorización. Por esta razón, sus certificados son también compatibles con los servicios que no son capaces de leer este tipo de información, ignorando la información de la parte no crítica.
- CAS guarda permisos de usuario (como el control de acceso a un fichero específico) como contraposición a los atributos. Esto significa que la última decisión sobre el acceso del usuario a los recursos no la tiene el administrador de los mismos sino el administrador del CAS. En este caso, CAS no cumple la estructura general de los sistemas de seguridad Grid, en los que el control y las decisiones finales de cada recurso respecto a la política de acceso recaen finalmente en el administrador local del recurso.

1.6.3.5.4. Privilege and Role Management Infraestructure Standars Validation (PERMIS)

El Privilege and Role Management Infraestructure Standars Validation [71] (PERMIS) es un sistema desarrollado por Saltford University, e implementa un sistema basado sobre roles. PERMIS es un sistema similar a VOMS, aunque presenta importantes diferencias:

- VOMS considera un atributo como un dato compuesto por una tupla de tres componentes (grupo, rol y capacidades). Por el contrario PERMIS solo tiene en cuenta el rol. Esto en principio no es una limitación, porque cualquier tupla VOMS podría ser simulada usando los roles PERMIS, aunque suponga una perdida de flexibilidad y escalabilidad, sobre todo si hablamos de organizaciones virtuales muy grandes y complejas, en la que PERMIS podría resultar excesivamente complejo a nivel experimental.
- Una segunda diferencia respecto a VOMS, es que VOMS distribuye los atributos de certificados a los propios usuarios, quienes son los responsables de presentar sus certificados, cuantas veces ellos quieran, para acceder a un sistema particular, o incluso pedir un subconjunto de estos atributos, al estar estos en disposición del propio usuario. Por el contrario, PERMIS genera previamente los atributos de certificado (ACs) y los guarda en la base de datos del propio servidor PERMIS, desde la cual son recuperados cuando los necesita. Este sistema hace casi imposible para un usuario pedir un subconjunto de sus permisos.
- La tercera diferencia importante respecto a VOMS, es que PERMIS incluye un manejador basado en ficheros para la interpretación de los certificados que contienen los atributos que definen la política administrativa a aplicar. Por el contrario, VOMS deja la interpretación de los certificados que contienen los atributos a otros componentes.

1.6.3.5.5. AKENTI

AKENTI [73] es un sistema de autorización Grid desarrollado por Lawrence Berkeley Nacional Laboratory, para facilitar la configuración de las políticas de acceso a organizaciones independientes y a través de organizaciones virtuales. AKENTI es un sistema basado en atributos de certificados (ACs) y se diferencia principalmente de VOMS en los siguientes aspectos:

- AKENTI utiliza un modelo pull. El usuario no tiene la necesidad de pedir una credencial específica a un servidor de autorización externo. El contacto del servicio con el servidor de autorización externo es independiente, una vez el usuario ha sido autenticado satisfactoriamente. Esto provoca problemas de flexibilidad porque hace imposible para los usuarios recibir sólo un subconjunto de sus capacidades sin procesar una segunda identidad independiente.
- Como ocurría con PERMIS, AKENTI también contiene un manejador para interpretar la información de los certificados referente a las políticas de acceso, algo que no ocurría con el VOMS y limita la independencia de administración de permisos.
- El formato de los datos utilizados por AKENTI para los atributos contenidos en los certificados es el ASN.1 [64], un formato no estándar que no soporta XML.

1.6.3.6. Aplicación de Sistemas de Autorización Grid

El sistema de autorización adecuado para el trabajo de ésta tesis, debe ser un sistema orientado a servicios, pues lo que se pretende es definir una arquitectura SOA, con lo cual el sistema de autorización de EDG no es adecuado. Además las implementaciones de los servicios deben seguir las especificaciones OGSA/WSRF, tal y como se ha justificado anteriormente. Más aún, el sistema de autorización de EDG utiliza servidores LDAP, lo que implica la utilización de protocolos y formatos que no se encuadran dentro de la arquitectura SOA. Otro de los inconvenientes de este sistema es el acceso booleano, es decir, la información que se presenta a los recursos sólo permite la identificación del usuario, con lo que el acceso será aceptado o denegado en función del usuario.

CAS es un sistema que cubre las carencias más importantes que presentaba el sistema de autorización de EDG, pues está orientado a servicios y además está soportado por la toolkit de GT4 cumpliendo las especificaciones OGSA/WSRF. También, puede organizar los usuarios de una OV mediante atributos, con lo que el control de acceso a los recursos deja de ser booleano, es decir, en función de los atributos que se presenten, es posible otorgar unos privilegios u otros. A pesar de esto, tampoco será el sistema a emplear para los desarrollos de ésta tesis, pues presenta un inconveniente suficientemente importante como para descartarlo. CAS guarda los permisos de usuario en el propio servidor, por lo que la última decisión sobre el acceso del usuario a los recursos no la tiene el administrador de los mismos sino el administrador del CAS. En este caso, CAS no cumple con la estructura general de los sistemas de seguridad Grid, en los que el control y las decisiones finales de cada recurso respecto a la política de acceso recaen finalmente en el administrador local del recurso.

Los sistemas VOMS, PERMIS y AKENTI también cubren las carencias del sistema EDG referentes a las especificaciones OGSA/WSRF y a la organización de usuarios mediante

atributos. Por un lado, AKENTI se descarta para los desarrollos de ésta tesis principalmente por el uso de un formato no estándar a la hora de guardar los atributos en los certificados. Por otro lado, en lo que respecta a VOMS y PERMIS, se podría considerar que PERMIS es un sistema más completo y superior que VOMS en lo referente al manejo de políticas administrativas, al tener un manejador incluido. Pero el principal problema de PERMIS es su poca flexibilidad para manejar grandes organizaciones con muchos usuarios, con lo que se descarta también PERMIS. Por tanto, se propone usar VOMS porque se integra perfectamente en un entorno orientado a servicios OGSA/WSRF y organiza las OV mediante atributos. Además, estos atributos pueden ser guardados mediante un extensión estándar de los certificados x509. Por el contrario, la principal desventaja de esta componente es la inexistencia de un manejador completo para la interpretación de los certificados que contienen los atributos. Por ello, también se establece como tarea a realizar ésta tesis, la creación de dicho manejador.

1.6.4. Cifrado de datos

Los sistemas anteriores resuelven, en mayor o menor medida, los problemas de autenticación y definición de permisos de acceso a los recursos en entornos organizados mediante OVs. Sin embargo, en el uso de entornos Grid distribuidos, existen otros factores a considerar en el marco de la seguridad y privacidad de los datos, y más aún si consideramos el carácter especialmente sensible de los datos provenientes de los entornos médicos, por los niveles de protección que tienen los datos personales (nivel 3 de la LOPD).

En los entornos Grid, el servicio de almacenamiento de datos se establece sobre servidores de datos distribuidos que son no necesariamente conocidos y no totalmente confiables. La localización de los datos queda oculta en el Middleware en función del rendimiento y la disponibilidad de los mismos. En este escenario, es muy posible (incluso a veces deseable o necesario) que los datos sean almacenados fuera de los límites de la organización que los generó, e incluso que sean replicados varias veces fuera de su ámbito. Cuando esto ocurre y los datos necesitan ser guardados en diferentes recursos (bien para su almacenamiento, bien temporalmente para su proceso) éstos quedan protegidos de usuarios no autorizados en el entorno Grid, pero quedan expuestos a los usuarios con permisos de administrador de los recursos locales. Para asegurar una correcta prevención de la privacidad, se debería asegurar que ni los usuarios con privilegios de administración local del recurso puedan ser capaces de acceder o modificar los datos. En el ámbito de la imagen médica analógica, la autenticidad es percibida más fácilmente por los usuarios. Sin embargo, la confianza en la inmutabilidad de los datos en formato digital, y más aún en entornos Grid en los que la información puede guardarse en diferentes recursos e incluso administrada por administradores desconocidos, es más difícil de transmitir (aunque en la práctica la privacidad pueda ser mayor).

Sin embargo, la autenticidad y la confidencialidad no son solamente problemas técnicos o éticos. Los sistemas de información de salud se desarrollan en un marco legislativo muy estricto, que obliga a garantizar la protección de los datos personales en relación su almacenamiento y procesamiento, y que define las condiciones y reglas bajo las cuales éste está permitido. Existen muchas regulaciones a nivel europeo [75][76] y legislaciones adicionales implementadas en estados miembros (REF a la LOPD, el RD 994/99, la LSICCE y las regulaciones del Ministerio de Sanidad). De acuerdo a la directiva 95/46/EC, si un proveedor de

servicios de salud mantiene datos personales de sus pacientes, entonces es identificado como un controlador de datos y es responsable de la protección de los mismos contra el uso no autorizado. La forma habitual en que el proveedor de servicios de salud implementa la legislación, es a través de una política de seguridad, que rige las prácticas laborales y los requerimientos tecnológicos (tamaño de las claves que se utilizan para cifrar datos, algoritmos de cifrado de los datos). Si un proveedor de servicios de salud desea acceder a datos personales custodiados por otra organización, entonces es identificado como un procesador de datos. Para que ocurra la comunicación entre un controlador y un procesador de datos, es necesario obtener un consentimiento del paciente y debe existir un contrato entre ambas partes que defina condiciones tales como el objetivo del proceso, el tipo de procesamiento de datos, y el tiempo que los mismos pueden ser almacenados por el procesador.

Actualmente, los mecanismos de control de acceso garantizan que los datos no sean expuestos, siempre y cuando el atacante no gane acceso físico o de administración, pero una vez traspasados los límites de la organización, hacer cumplir esta condición queda fuera del alcance del controlador de los datos. Se hace necesario entonces tomar medidas adicionales para proteger la confidencialidad de los datos médicos en el Grid.

El cifrado de datos es utilizado en el campo de datos médicos digitales para asegurar la confiabilidad de los datos digitales. Las técnicas de cifrado de datos basadas en claves y las de estampado digital (firmas digitales y MAC) han sido ampliamente utilizadas para estos fines [74]. Por tanto, una de las soluciones que se están aplicando en la actualidad, es la de guardar los datos cifrados mediante claves, reduciendo considerablemente el riesgo de exposición. Sin embargo, es necesario proporcionar mecanismos que aseguren que las personas autorizadas tengan acceso a las claves de descifrado. El modelo a desarrollar en ésta tesis se centra en un modelo de cifrado de datos con distribución de claves. Existen en la literatura modelos como los publicados por Seitz [81], que propone una arquitectura que permite el almacenamiento y acceso a los datos cifrados en este tipo de entornos. Un modelo similar, lo encontramos en el trabajo de Torres [92], el cual define un modelo de cifrado de datos mediante claves distribuidas.

En el siguiente apartado se desarrollan los conceptos básicos que se requieren en los modelos de cifrado de datos en entornos distribuidos mediante el uso de claves distribuidas, además de describir algunos trabajos actuales.

1.6.4.1. Manejo de Claves para el Almacenamiento de Datos Cifrados en Sistemas Distribuidos

Tal y como se ha comentado anteriormente, los dispositivos de almacenamiento masivo requieren de una protección especial, debido a que en la actualidad no se puede garantizar que un usuario no pueda tomar poder administrativo sobre estos dispositivos a través de ingeniería social, claves débiles o agujeros de seguridad en los sistemas de control de acceso. Existen diariamente avisos de agujeros de seguridad en aplicaciones bien establecidas que son aprovechadas incluso por usuarios no expertos. Estas tareas son relativamente fáciles si se comparan con la intercepción e interpretación de comunicaciones de red [84], las cuales ya tienen soluciones más fiables, tal y como se comentó en el apartado 1.6.2.

Los mecanismos de control de acceso del Grid, representados por [85][86][87] y comentados en los apartados anteriores referentes a la autenticación de usuarios y privacidad de datos (apartado 1.6.2) y los sistemas de autorización Grid (apartado 1.6.3), solamente protegen los datos mientras no se gane acceso físico o administrativo a los recursos donde se almacenan. Tal y como se ha comentado anteriormente, los datos se pueden almacenar en dominios administrativos diferentes al del usuario que maneja el dato, por lo que en el caso de datos médicos este problema es crítico.

El cifrado de los datos mediante claves proporciona medios para disminuir este riesgo. Existen varios sistemas que proporcionan estos servicios, tanto para el cifrado mediante claves de archivos simples (por ejemplo, PGP [88]) o el cifrado mediante claves de sistemas de archivos completos [89][90]. Sin embargo, el manejo de claves constituye un grave problema al compartir archivos cifrados. La principal razón es que se necesita mantener las claves de cifrado un tiempo incomparablemente más largo que los tiempos de vida de las claves de comunicación que se utilizan en protocolos basados en SSL, los cuales sólo deben perdurar el tiempo que dura la comunicación. Como consecuencia, las claves necesitan ser almacenadas de forma segura y los usuarios autorizados necesitan de un mecanismo de acceso a ellas. Por ello, se necesita un esquema de manejo de claves adaptado a la arquitectura Grid, donde los usuarios autorizados deben ser capaces de acceder a las claves de descifrado, incluso cuando los propietarios de los datos sean de un dominio administrativo diferente. Por otra parte, los propietarios de los datos no deben tener la necesidad de confiar en los servidores de almacenamiento de datos porque éstos pueden estar en dominios administrativos diferentes.

Seitz [84] y Torres [91] presentan sistemas de manejo de claves para el almacenamiento de datos cifrados en sistemas distribuidos. La arquitectura propuesta por estos autores, permite a los usuarios almacenar y compartir datos en estos entornos. Una de las principales ventajas, introducidas por este modelo de seguridad, es que puede ser operable con un número considerable de mecanismos de control de acceso sin necesidad de cambios funcionales. Adicionalmente, este modelo minimiza el uso de terceras partes, lo cual disminuye los puntos de fallo, además de tener un impacto positivo en la utilización del ancho de banda y la transmisión de mensajes por la red. En este modelo de seguridad, los usuarios clientes tienen la responsabilidad de descifrar los objetos. Esta condición hace incluso innecesaria la utilización de canales seguros para trasmitir los datos desde los servidores de datos hasta los clientes, con una disminución de carga asociada al hecho de que no hay que codificar las trasmisiones porque todos los objetos viajarán codificados. Por otra parte, se elimina el riesgo de ataque a los objetos temporales descifrados en los servidores de datos y proceso, y se evita la aparición de cuellos de botella debido al aumento de la carga que presupone las operaciones de cifrado-descifrado. Por último, los servicios de administración de claves tienen acceso solamente a partes de la clave, pero no a claves completas y siempre a través de canales seguros. Se utiliza el esquema para compartir claves de Shamir y se distribuyen las partes entre dominios administrativos diferentes. De esta forma disminuye considerablemente el riesgo de que un usuario, con privilegios de acceso físico o de administración sobre un dispositivo de almacenamiento, pueda exponer datos confidenciales, porque necesitaría para ello reconstruir la clave de descifrado a partir de partes distribuidas entre servidores de claves pertenecientes a dominios administrativos diferentes, sobre los que debería ganar privilegios administrativos.

En los siguientes apartados se comentan diferentes conceptos relacionados con la compartición de claves, como los tipos de esquemas para compartir claves existentes, incluyendo primero una definición detallada del concepto de esquema para compartir claves.

1.6.4.1.1. Esquemas para Compartir Claves

Un esquema para compartir claves es un método mediante el cual se divide una clave, comúnmente una clave criptográfica, en múltiples partes llamadas shadows, fragmentos o simplemente partes, que son distribuidas entre un grupo de custodios. El ente encargado de llevar a cabo la división se conoce comúnmente con el nombre de distribuidor, y en muchos esquemas se asume que el distribuidor es perfectamente confiable. En los esquemas simples, la clave es dividida en N partes y solamente puede ser recuperada reagrupando las N partes. Este esquema se conoce como un esquema de límite (N, N). La ventaja reside en que se necesita el acuerdo de todos los custodios para recuperar la clave, pero si una de las partes se pierde, se pierde también la clave. Por esta razón hay muchos mecanismos para compartir claves que se basan en un umbral (k, N), con k < N, de forma tal que la clave es dividida en N partes, pero puede ser totalmente recuperada a partir de cualquier conjunto de k partes. Adicionalmente, el conocimiento de k-1 partes, en un esquema perfecto, no debe proporcionar más información que la contenida en una sola parte. Esto significa que la dificultad de cualquier ataque, incluyendo la fuerza bruta, debe ser la misma independientemente del número de partes del que se disponga, mientras que no se alcance el umbral k. En términos de seguridad, los esquemas de umbral son más débiles que los que requieren todas las partes, pero ofrecen la posibilidad de recobrar el clave incluso ante la pérdida de varias partes. Una variante final del esquema para compartir claves consiste en mantener un esquema de umbral (k, N) en el cual los custodios han sido divididos en conjuntos autorizados de k miembros, de forma tal que la clave solamente puede ser recuperada si todos los custodios que contribuyen con su parte forman uno de los conjuntos autorizados. De esta forma no todos los conjuntos de k partes conducen a la clave. Esta variante es la de alcance más general, porque abarca el esquema simple mencionado al principio cuando k=N y el esquema anterior (k, N) cuando todas las posibles combinaciones de k partes forman un conjunto autorizado.

1.6.4.1.2. Modelos de esquemas para compartir claves

- **Método Trivial**. El método trivial para compartir claves ocurre cuando ésta es dividida en *N* partes y se requieren todas las partes para reconstruirla. La manera más simple de llevar a cabo este esquema es utilizar *N*-1 números aleatorios y un número que represente la diferencia entre la clave y la suma de todas las *N*-1 partes: $S_n = K - (S_1 + S_2 + ... + S_{n-1})$. De esta forma, la clave puede ser hallada adicionando las *N* partes. Este esquema puede ser implementado en sistema binario utilizando la operación XOR, en lugar de la adición y la sustracción.
- **Esquemas Perfectos**. Se dice que un esquema para compartir claves es perfecto si cualquier grupo de hasta *k*-1 custodios no puede determinar más información de la clave que cualquier sujeto que no sea un custodio. La robustez de un algoritmo para producir un esquema perfecto se hace cumplir por demostración matemática, mientras que los esquemas imperfectos basan su condición en la dificultad computacional para recuperar la clave con menos de *k* partes. Los esquemas imperfectos pueden ser rotos en el futuro con el aumento

del poder de cómputo, pero, sin embargo, requieren partes de menor tamaño y a menudo su implementación es más rápida y fácil.

- **Esquemas Ideales**. Los esquemas perfectos requieren que el tamaño de las partes individuales sea tan grande o mayor que el tamaño de la clave misma. Pero, cuanto más grande es el tamaño de la parte, más difícil es de manejar y más problemas de seguridad puede causar a los custodios. A tamaño mayor de las partes individuales, mayor es la memoria y la capacidad de procesamiento necesaria para manipular las partes. Un esquema ideal es aquel en el que el tamaño de las partes individuales es exactamente igual al tamaño de la clave.
- **El esquema de Shamir** [93]. Es un esquema perfecto que puede ser implementado como un esquema ideal. El método de Shamir reside en el hecho de que para reconstruir un polinomio de grado N, se necesita conocer N+1 puntos del polinomio. De esta forma, cuando es aplicado al problema de dividir una clave K en N fragmentos de forma tal que se necesitan M fragmentos para recuperarlo, el algoritmo de Shamir procede creando un polinomio de grado M-1, $S(x)= K+S_1X^1+S_2X^2+\ldots+S_{m-1}\ X^{m-1}$. Nótese que la clave aparece como el término independiente del polinomio y que $S(0)$ devuelve K directamente. Los coeficientes S_i del polinomio tienen una importancia directa poco significativa y son generados de forma aleatoria por el ordenador. Una vez que el polinomio ha sido creado, las N partes son generadas como las coordenadas $(x, y=S(x))$ del polinomio. Comúnmente se utilizan los puntos $(1, S(1))$, $(2, S(2))$, …. $(N, S(N))$, pero se puede utilizar cualquier conjunto de N valores generado tomando X al azar, siempre y cuando X sea distinta de 0 y se encuentre en el "shadow" o fragmento. Un distribuidor confiable reparte las partes entre los custodios.

1.6.4.2. Aplicación del Cifrado de Datos

La aplicación del modelo de cifrado en ésta tesis se basa en el trabajo de Torres [92], el cual define un modelo de cifrado de datos mediante claves distribuidas, adaptando éste al esquema de la arquitectura SOA e implementándolo sobre WSRF. La elección de este modelo como base para la arquitectura, es debido a las ventajas que nos aporta la distribución de claves y que han sido descritas en apartados anteriores. El modelo original deberá ser adaptado a las necesidades de la arquitectura que se va a definir, con lo que va a ser tarea de esta tesis el redefinir este modelo.

1.7. Proyectos en el Ámbito de la Tesis

En este apartado se pretende dar una visión de todos aquellos proyectos que tienen objetivos similares, o usan tecnologías comunes, y que están relacionados con el tratamiento de datos médicos y particularmente en la parte correspondiente a la imagen digital, tanto en lo relativo a su almacenamiento como a su proceso. En este marco se analizarán proyectos con una base más tecnológica, como los referentes al proceso de imágenes médicas y el manejo de workflows (IXI, TAVERNA y MOTEUR), proyectos relacionados con la compartición y proceso de imágenes médicas (CaGRID, ARTEMIS, MEDIGRID y MDM) y proyectos relacionados con áreas médicas específicas como la neuroimagen (NeuroBase y BIRN) o los programas de detección precoz de cáncer de mama (eDiamond, MammoGrid y GPCALMA). Esta sección pretende analizar el estado actual de los proyectos más importantes que tienen objetivos comunes con el trabajo de la presente tesis y describir aquellas aportaciones que pueden ser interesantes en este marco.

Los puntos clave en estos proyectos, sobre los cuales se hará hincapié, son los relacionados con los workflows, proceso, almacenamiento distribuido, organización semántica y seguridad. Todos estos puntos serán parte importante de los objetivos que se plantean en ésta tesis.

1.7.1. Information eXtraction from Images (IXI)

Information eXtraction from Images (IXI) propone una infraestructura [96][97] que permite procesar imágenes médicas utilizando algoritmos complejos de posproceso (segmentación, registración, etc…) mediante un interfaz de Servicios Grid. Por otra parte, IXI permite interconectar diferentes procesos que se pueden ejecutar en diferentes recursos distribuidos, dependiendo del algoritmo que se pretenda ejecutar, prestando especial atención a la especificación de workflows de procesos sobre imágenes médicas.

La infraestructura definida en IXI se basa en una arquitectura SOA, en la que todos los recursos son ofrecidos mediante servicios Grid. Los servicios Grid implementados se desarrollan bajo las especificaciones OGSA y, más concretamente, utilizando su infraestructura OGSI. IXI define un lenguaje llamado “Medical Imaging Component Language” (MICL), que proporcionan a cada servicio Grid desarrollado un interfaz WSDL para que los clientes interactúen. Este lenguaje describe, para cada servicio, el tipo de algoritmo que implementa junto con todas aquellas características que el servicio ofrece, separando lo que es la interfaz de interacción de su implementación.

En el proyecto IXI también se ha especificado un lenguaje capaz de definir los pipelines de los procesos o workflows entre los diferentes procesos a ejecutar en diferentes recursos. IXI define los workflows mediante servicios Grid, que contendrán los workflows especificados a través de documentos XML, que encauzarán y dirigirán el flujo de tareas a través de los servicios desplegados en el Grid.

Los puntos a destacar en IXI respecto a ésta tesis son los siguientes:

- IXI se encarga del proceso de imágenes médicas y proporciona un lenguaje para especificar los workflows de los procesos, pero no contempla el manejo de otro tipo de información

que podría incluirse en los objetos DICOM. En esta tesis se pretende extender el uso de DICOM a todas las modalidades existentes (videos, señales, documentos estructurados).

- IXI plantea una arquitectura SOA, en la que los servicios se basan en la infraestructura OGSA/OGSI actualmente en desuso. En esta tesis se plantea el uso de la infraestructura OGSA/WSRF.
- IXI plantea, como hipótesis de trabajo, que las imágenes con las que se trabaja se encuentran en una base de datos centralizada [98], de forma que los usuarios que pueden acceder a ésta son controlados. IXI no plantea el problema de seguridad de la información, más allá de lo que ofrecen las infraestructuras Grid, confiando en GSI. No se plantea el problema en el que la información fluye por recursos cuyos entornos de administración son variados y no controlados, por lo que incumpliría las normativas de protección de datos en su uso en producción. Por ello en esta tesis plantearemos este tipo de problema y aportaremos un modelo de seguridad que lo solucione.

1.7.2. TAVERNA

TAVERNA [101] es un proyecto en el que colaboran el European Bioinformatics Institute (EBI), el Rosalind Franklin Centre for Genomic Research (RFCGR), las Facultades de Ciencias de la Computación de Newcastle y de Manchester, el Newcastle Centre for Life y el Nottingham University Mixed Reality Lab, especialmente dirigido a resolver los problemas de organización de procesos en la comunidad bioinformática.

El proyecto TAVERNA es un desarrollo realizado en el proyecto myGrid [102], el cual tiene como principal objetivo desarrollar un lenguaje y diversas herramientas software para facilitar el uso de workflows, y el manejo de tecnologías de computación distribuida dentro de la comunidad bioinformática.

TAVERNA permite crear y lanzar worflows de una manera sencilla, utilizando un lenguaje propio llamado Simplified Conceptual workFlow Languaje (Scufl), además de proporcionar un entorno gráfico y unas APIs. Se centra específicamente en la generación y manejo de workflows, y no es específico para ningún tipo de datos, con lo que es aplicable a objetos DICOM e imagen digital, y aporta una solución completa para experimentos científicos dentro de área de la biología y bioinformática.

Los puntos a destacar en TAVERNA, respecto a ésta tesis, son los siguientes:

- Al igual que ocurre con IXI, provee un marco para generar workflows de procesos, además define un lenguaje para ello.
- En TAVERNA no se plantea el problema del almacenamiento de información en recursos administrados por usuarios no confiables, aunque sea de manera temporal, tal y como ocurre en los workflows.

1.7.3. Home-made Optimised SCUFL Enactor (MOTEUR)

El proyecto hoMe-made OpTimisEd scUfl enactoR (MOTEUR) esta desarrollado en Java y disponible bajo la licencia pública CeCILL en http://www.i3s.unice.fr/~glatard. Es un proyecto para la creación de workflows. El lenguaje que se utiliza para la descripción de los workflow es

el Scufl, usada por el proyecto TAVERNA y que actualmente es un estándar de facto en la comunidad.

MOTEUR puede utilizar los interfaces de lanzamiento de trabajos de diferentes infraestructuras Grid, como por ejemplo la de EGEE [103] o el grid experimental GRID5000 [104]. Además, para la ejecución de trabajos ligeros permite utilizar recursos locales. MOTEUR puede someter diversas tareas que se ejecuten en diversas infraestructuras durante una sola ejecución del workflow. Los interfaces que ofrece MOTEUR actualmente son mediante Servicio Web y GridRPC.

MOTEUR tiene tres niveles diferentes de paralelismo, optimizando de esta manera la ejecución del código del workflow. Estos tres niveles son:

- Paralelismo de workflows inherente a la topología del workflow.
- Paralelismo de datos. Diferentes entradas de datos pueden ser procesadas independientemente en paralelo.
- Paralelismo de servicios. Diferentes servicios procesan diferentes datos que son independientes y pueden ejecutarse en paralelo.

Los puntos a destacar en TAVERNA, respecto a ésta tesis, son los siguientes:

- MOTEUR es el primer workflow basado en servicios que tiene todas las optimizaciones comentadas implementadas.
- MOTEUR, al igual que IXI y TAVERNA no se plantea el problema del almacenamiento de información en recursos administrados por usuarios no confiables, aunque sea de manera temporal, tal y como ocurre en los workflows.

1.7.4. An Information System for Managing Distributed Knowledge and Data Bases in Neuroimaging (NeuroBase)

El proyecto An Information System for Managing Distributed Knowledge and Data Bases in Neuroimaging [102] [106] (NeuroBase), tiene como objetivo establecer las condiciones necesarias para la federación de neuroimágenes distribuidas en diferentes bases de datos y que se localizan en diferentes centros de investigación, en departamentos clínicos de neurología o centros de investigación de neurociencia. Este proyecto pretende especificar como conectar y acceder a información distribuida en diferentes bases de datos de neurología, para definir una arquitectura de proceso de datos que permita el acceso y la compartición de resultados de los diferentes estudios que se vayan realizando, o incluso el proceso de datos en diferentes sitios distribuidos. Este proyecto permite definir diferentes ontologías en función del grupo de usuarios que pretenda acceder a la información.

Mediante el uso de NeuroBase es posible conseguir, a partir de las bases de datos distribuidas, imágenes similares, imágenes que contengan singularidades o realizar búsquedas transversales para resaltar posibles regularidades.

Neurobase surgió para cubrir las necesidades de las áreas relacionadas con la Neurocirugía, siendo uno de los principales objetivos la construcción de mapas funcionales cerebrales, tanto bajo condiciones normales como patológicas. Los investigadores actualmente trabajan para encontrar correlaciones entre estructuras anatómicas, donde la activación neurológica tiene

lugar en función de la actividad cerebral. Neurobase considera imágenes multimodales (PET, MRI, EEG y MEG).

Los aspectos a destacar de NeuroBase, respecto a ésta tesis, son las siguientes:

- NeuroBase se centra en la Neuroimagen y proporciona componentes genéricos para este tipo de imágenes. En esta tesis se pretende abarcar todas las modalidades DICOM, siendo las componentes válidas para todas ellas.
- Al igual que IXI, TAVERNA y MOTEUR, NeuroBase no plantea el problema de seguridad de la información más allá de lo que ofrecen las infraestructuras Grid (Cifrado SSL y certificación X509), no considerándose que la información fluya por recursos cuyos entornos de administración son variados y no controlados.

1.7.5. Biomedical Informatics Research Network (BIRN)

Biomedical Informatics Research Networks [107] (BIRN) es un entorno colaborativo para compartir datos y herramientas de proceso en el marco de la Neuroimagen. En esta red participan, a finales de 2006, 39 grupos de investigación de los Institutos Nacionales de Salud Norteamericanos (NIM). BIRN se estructura en cuatro áreas, que se dedican al mantenimiento de la infraestructura de investigación en imágenes de ratones, primates no humanos, Morfometría cerebral e imagen funcional.

Los principales objetivos de BIRN son:

- Organización de datos: Creación de DataGrids virtuales basados en el middleware Storage Resource Broker (SRB). SRB es un sistema cliente-servidor diseñado para manejar conjuntos de ficheros en entornos heterogéneos y distribuidos. Todos los ficheros de un entorno forman parte de un único sistema de ficheros Grid, donde la localización lógica de los ficheros dentro del sistema es representada independientemente de su localización física. El middleware SRB es capaz de manejar grandes conjuntos de datos.
- Seguridad y privacidad: La autenticación y autorización de los usuarios en el sistema se basa en GSI, asumiendo que el entorno es confiable, lo que podría entrar en conflicto con otras normativas más restrictivas. Los datos se pseudoanonimizan.
- Proceso: Los servicios se instalan en sitios donde se proporcionan además recursos de proceso.

Los aspectos a destacar de NeuroBase, respecto a esta tesis, son las siguientes:

- BIRN es un sistema cuya implantación es un referente mundial. Es un sistema en producción y bien recibido por los usuarios, combinando el almacenamiento distribuido con la disponibilidad de recursos de proceso. Sin embargo, el entorno de almacenamiento distribuido no cifrado puede plantear problemas legales, como también ocurre con los anteriores proyectos descritos.
- En BIRN hay una estructuración por comunidades virtuales, pero los datos no se organizan en un marco ontológico, tal y como planteamos en esta tesis.

1.7.6. MammoGrid, eDiamond y GPCALMA

Estos tres proyectos comparten orientación, objetivos e incluso tienen socios comunes. Los tres están orientados hacia los programas de cribado mamográfico, con algunas diferencias específicas.

MammoGrid [108] es un proyecto que tiene como principal objetivo el uso de las tecnologías Grid para el desarrollo de una gran base de datos de mamografías a escala europea, para que sean utilizadas para la investigación mediante el uso de un conjunto de aplicaciones para la asistencia médica, así como utilizar el potencial del Grid para soportar de forma efectiva la cooperación.

El proyecto busca concentrarse sobre las aplicaciones construidas sobre tecnologías Grid y no enfocarlo sobre otros desarrollos. Entre los beneficios que se esperan de Mammogrid cabe citar los siguientes:

- Tener almacenados un número significativo de casos diferentes registrados y normalizados para permitir su compartición.
- Tener mayor diversidad epidemiológica, mayor variedad de diagnósticos por imagen cubriendo diferentes áreas geográficas y población.
- Tener un interfaz homogéneo para el acceso a bases de datos heterogéneas.
- Tener una combinación de proceso local y remoto.
- Permitir el descubrimiento de conocimiento referente a los diagnósticos y entendimiento de los cánceres de mama.

Las aplicaciones objetivo en el proyecto se enmarcan en el diagnóstico precoz del cáncer de mama y su tratamiento. Estas aplicaciones tratan de suplir los siguientes problemas:

- Variabilidad de imágenes debido a las diferencias en la adquisición de imágenes en los diferentes centros, además de los diferentes paquetes software asociados.
- Población variable, lo que provoca diferencias entre diferentes regiones a la hora de elegir los tratamientos de los cánceres de mama detectados.

Los datos tratados mediante esta infraestructura son anonimizados, y se utiliza la infraestructura GSI para garantizar la privacidad de los datos y la autenticación de los usuarios.

eDiaMoND [109] es un proyecto colaborativo financiado por Engineering and Physical Sciences Research Council (EPSRC) e IBM. Los objetivos planteados en este proyecto son muy similares a los de MammoGrid, pero con un ámbito de actuación reducido al Reino Unido y utilizando la infraestructura del National e-Science (NeSC) para su ejecución. Finalmente, GPCALMA [110] es un proyecto que transfiere la experiencia en física de altas energías, en la que se usan las tecnologías Grid para el uso de computación distribuida y el manejo de los datos asociados. Este proyecto se ha enfocado al desarrollo de aplicaciones comunes para la detección temprana de los cánceres de mama y en un futuro de cánceres de pulmón. En términos generales, el proyecto es similar a los anteriores relacionados con el cáncer de mama.

Los aspectos a destacar de estos proyectos, respecto a ésta tesis, son las siguientes:

- La solución que proponen estos proyectos se orienta a los programas de cribado mamográfico. Si bien algunos de los módulos podrían ser utilizados de forma más general, no queda claro que su arquitectura sea extensible a cualquier objeto DICOM.

- Al igual que la mayoría de proyectos presentados, no tiene en cuenta el cifrado de datos, utilizando únicamente GSI para implementar la autenticación y autorización de los usuarios.

1.7.7. Cancer Biomedical Informatics Grid (CaBIG)

Cancer Biomedical Informatics Grid [111] (CaBIG) es una red dedicada al avance de los estudios de cáncer. Dentro de CaBIG hay diferentes comunidades de usuarios, una de las cuales está dedicada a la imagen médica. En el marco de esta red se ha desarrollado una serie de componentes bajo el nombre de CaGRID, que tiene las siguientes características:

- Organización de datos: Utiliza Open Grid Service Arquitectura/Data Acces Infraestructure [112] (OGSA/DAI) para la gestión de datos distribuidos. OGSA/DAI es una iniciativa cuyo principal objetivo es desarrollar un middelware para el acceso e integración de datos de diferentes fuentes mediante tecnologías Grid. El proyecto fue concebido por el UK Database Force y trabaja conjuntamente con el grupo de trabajo Open Grid Forum (DAIS-WG).
- Seguridad y Privacidad: Implementa un mecanismo de autorización y autenticación basado en PERMIS y dos componentes propias, el Grid User Management System [113] (GUMS) y caGrid Atributte Management System (CAMS). Estas componentes permiten un modelo de delegación de credenciales para el acceso a datos locales e implementar políticas globales de autorización. GUMS es un servicio de mapeo de identidades Grid, es decir, los recursos Grid no utilizan credenciales nativas del Grid con lo que se tienen que identificar mediante sus identificadores de usuario (cuentas LINUX o Kerberos principalmente) y asignarles una credenciales que se encuentran en el servidor GUMS. El Servicio CAMS se encarga de almacenar los atributos Grid de los diferentes usuarios.
- Para el proceso implementa una API para servicios Grid.

Los aspectos a destacar de caBIG, respecto a ésta tesis, son las siguientes:

- Si bien el esquema de autenticación y autorización es más avanzado que GSI, aún puede plantear problemas dependiendo del grado de disociación de los datos personales.
- Tiene una organización de datos distribuida destacable mediante OGSA/DAI, pero no soporta la gestión de los datos mediante ontologías. El uso de OGSA/DAI para la distribución de los datos, facilita la integración en arquitecturas Grid SOA, pues ello implica el uso de tecnologías Grid que siguen las especificaciones OGSA.

1.7.8. ARTEMIS

ARTEMIS [114][115] presenta una infraestructura de interoperabilidad segura para sistemas de información de salud. Esta arquitectura permite la comunicación de datos médicos entre proveedores de salud por medio de una mediación entre seguridad y políticas de privacidad sobre las condiciones que los contratos entre las partes y el consenso del paciente permitan. La utilización de descripciones semánticas de servicios Web posibilita resolver las diferencias entre los requerimientos de seguridad y capacidades de los diferentes proveedores. Hasta el momento, una infraestructura de autorización permite el control de acceso basado en la mediación entre roles clínicos definidos por diferentes organizaciones. Sin embargo la autorización basada en

roles ha demostrado ser insuficiente en la mayoría de los escenarios clínicos y las perspectivas futuras apuntan a la incorporación del flujo de trabajo en las decisiones de autorización.

Los aspectos a destacar de ARTEMIS, respecto a ésta tesis, son las siguientes:

- ARTEMIS proporciona un middleware que permite la conectividad entre proveedores utilizando redes P2P. Dentro de esta arquitectura, los mediadores actúan como traductores entre solicitantes y proveedores que utilizan diferentes políticas de seguridad. Por tanto ARTEMIS carece de un esquema de cifrado seguro en la actualidad, aunque hay que destacar el esquema P2P que ofrece un esquema semántico de autorización.
- ARTEMIS no proporciona un entorno de ejecución eficiente para la implementación de servicios de proceso.

1.7.9. MEDIGRID

MEDIGRID [99] es un proyecto relacionado con Grid y financiado por el Ministerio de Investigación Francés en el Programa "Action Concertée Iniciative Globalisation des Resources Informatique", del año 2002.

El principal objetivo de MEDIGRID es explorar el uso de las tecnologías Grid para realizar el proceso de grandes bases de datos de imagen médica, disponibles en los hospitales de hoy en día. Actualmente, las imágenes médicas digitales están cada vez más accesibles y existen una gran cantidad de herramientas que sirven para su análisis en las comunidades científicas. Además, la gran cantidad de datos médicos, y la complejidad de algunos algoritmos para modelar y procesar éstos, crean la necesidad de una enorme potencia de proceso y capacidad de almacenamiento. Grid debería proporcionar la infraestructura necesaria para dar solución a estas necesidades.

MEDIGRID utiliza las tecnologías Grid para responder a estas necesidades. Está dirigido a los campos de aplicaciones médicas, en las cuales puede dar la tecnología Grid un soporte valioso, estos campos son los siguientes:

- Algoritmos de proceso complejo para imagen médica que requieran de una implementación paralela en términos de recursos computacionales y memoria requerida (p.e. Modelado de órganos fisiológicos y anatómicos).
- Herramientas para procesar y manejar bases de datos de imagen médica (p.e. Búsquedas de imágenes por contexto [100]).

Los aspectos a destacar de MEDIGRID, respecto a ésta tesis, son los siguientes:

- ARTEMIS no pretende dar una solución global, sino aplicar las tecnologías Grid a una serie de áreas de la ciencia, concretamente en la imagen médica y los sistemas de información asociados. En ésta tesis se plantea como objetivo el manejo de objetos DICOM en general.
- A pesar de dar soluciones a la compartición de información, no profundiza en el tema de selección de ésta en función de ontologías determinadas.
- No ofrece unas componentes de alto nivel orientado a objetos, para la gestión de las funcionalidades del proyecto.

1.7.10. Medical Data Manager (MDM)

Medical Data Manager (MDM) es un sistema de propósito general para guardar y compartir datos DICOM usando tecnologías GRID. MDM afronta los retos de la imagen digital considerando:

- Organización de datos: Los datos son pseudoanonimizados y se ponen a disposición del Grid mediante el interface SRM y el sistema de catálogo de gLite 1.5. [46]. Gestiona la información privada mediante AMGA, servicio integrado en gLite para la gestión del metadata.
- Privacidad y seguridad: Los datos son cifrados y descifrados en los recursos de almacenamiento. Soporta servicios de almacenamiento de claves mediante servidores Hydra que vienen integrados en las infraestructuras que soporta gLite 1.5.
- Procesamiento: Utiliza los servicios Workload Management Service (WMS) de gLite para la ejecución de tareas de postproceso de imágenes.

Los aspectos a destacar de MDM, respecto a ésta tesis, son los siguientes:

- MDM almacena los datos atendiendo principalmente a criterios administrativos, mas que semánticos.
- No proporciona a los programadores de un interfaz de alto nivel orientado a objetos para el manejo de los servicios ofrecidos en la arquitectura mediante el middleware gLite, aunque sí proporciona un interfaz en línea de comandos.

CAPÍTULO II. Objetivos e Hipótesis de Trabajo

El objetivo principal de la tesis, es la definición e implementación de una arquitectura genérica SOA que provea de servicios suficientes para la gestión, manejo y proceso de información en formato DICOM (imágenes, señales, videos, documentos estructurados, etc.) de forma segura y organizada semánticamente a partir de ontologías. A esta arquitectura la llamaremos TRENCADIS[1] (Towards a gRid ENvironment to proCess and shAre DIcom objectS).

Una vez implementada la arquitectura en forma de un middleware Grid, TRENCADIS proporcionará a los desarrolladores un interfaz de alto nivel, orientado a objetos, para la creación de aplicaciones en diferentes ámbitos médicos. Tanto la arquitectura como el middleware no pretenden dar una solución concreta a un área médica determinada, ni a un problema específico referido al manejo de información DICOM, sino desarrollar una herramienta que permita a los desarrolladores e investigadores abstraerse de todo lo relacionado con el carácter distribuido de los objetos DICOM y las diferencias entre las fuentes, permitiendo así una mayor productividad en sus desarrollos, estudios o investigaciones.

Para la consecución de este objetivo genérico, se detallan, en un primer apartado, los cinco bloques en los que se enmarcan los diferentes objetivos específicos que se derivan del objetivo general, y en un segundo apartado, se expondrán las hipótesis de trabajo sobre las que se basará ésta tesis.

[1] TRENCADIS es la denominación valenciana de un cierto tipo de mosaico en el que las piezas que lo componen son irregulares. La presente tesis persigue la construcción de conocimiento a partir de fragmentos de información heterogéneos y diferentes, pero que tras su integración se consigue construir un objeto de valor superior.

2.1. Objetivos

En este apartado se describen los objetivos específicos que se pretenden alcanzar en ésta tesis. En la primera sección presenta los objetivos referentes a la estructura de la arquitectura TRENCADIS; la segunda sección corresponde a los objetivos que el modelo de seguridad debe cumplir al aplicarse éste sobre la arquitectura; la tercera sección corresponde a las funcionalidades que deben ofrecer los componentes de alto nivel del middleware a desarrollar sobre la arquitectura TRENCADIS y que afectarán directamente a los servicios a implementar sobre la arquitectura; la cuarta sección describe las componentes de alto nivel a desarrollar; la quinta sección presenta las aplicaciones que se utilizarán para validar todos los objetivos planteados en las secciones anteriores.

2.1.1. Arquitectura TRENCADIS

Los objetivos referentes a la arquitectura a desarrollar son los siguientes:

- **La arquitectura estará basada en tecnologías Grid.** Se utilizarán tecnologías Grid estándar basadas en servicios Web para la definición de los servicios dentro de la arquitectura, ya que ésta es la tecnología más extendida actualmente y mejor orientada a desarrollos de Internet para la compartición de recursos entre diferentes organizaciones. Se opta por las especificaciones de la Open Grid Services Architecture (OGSA) para las definiciones de Servicios Grid. Respecto a la implementación de los servicios, se opta por los Web Services Resources Framework (WSRF) al estar basados en Servicios Web.
- **La arquitectura debe ser orientada a servicios, Service Oriented Architecture (SOA)**. Se tomarán como base las especificaciones OGSA. Todos los recursos que deben ser integrados en la arquitectura (proceso, almacenamiento, datos, programas, etc.), como servicios en las capas correspondientes. Además, la arquitectura debe proporcionar componentes de alto nivel para el manejo de dichos servicios.
- **La arquitectura debe estar basada en capas.** TRENCADIS debe de proporcionar una estructura bien definida en capas, donde en cada capa se ubiquen los servicios y componentes necesarios que faciliten la incorporación de nuevas funcionalidades futuras.
- **Definir los servicios básicos de la arquitectura.** Definir lo servicios básicos de TRENCADIS necesarios para cubrir en su especificación la funcionalidad propuesta en el apartado siguiente y otras futuras. La arquitectura debe definir los protocolos que utilizarán los servicios, además de la forma en que se especificarán las estructuras de datos a emplear en los desarrollos de los diferentes servicios a la hora de intercambiar la información.
- **La arquitectura debe ser abierta.** TRENCADIS debe permitir ampliaciones de su funcionalidad, traduciéndose en la creación de nuevos servicios perfectamente identificables en las diferentes capas, en la estructura definida. La compatibilidad de los servicios se define a nivel de interfaz y protocolos de comunicación y no a nivel de implementación [116].
- **La arquitectura utilizará Internet como red de comunicación.** El uso de Internet facilitará el despliegue de la red y requerirá considerar diferentes políticas específicas en materia de transmisión de datos y seguridad.

2.1.2. Modelo de Seguridad

En este apartado se presentan los objetivos de seguridad de la arquitectura TRENCADIS. El modelo debe tener en cuenta la propia naturaleza vulnerable de Internet. Internet es una red pública, a la que todo el mundo puede conectarse de forma sencilla, con lo que se incrementan los riesgos de sufrir intrusiones. Dado que este trabajo se orienta a la utilización de datos médicos referentes a pacientes, hay que tener en consideración los aspectos legales comentados en la sección 1.6 referente a la infraestructura de seguridad en el Grid. Por tanto, los objetivos respecto de la seguridad son los siguientes:

- **Autenticación de usuarios y privacidad de datos en las comunicaciones.** Se debe garantizar la privacidad de los datos y la integridad de los mismos en las transmisiones. La integridad de los canales de comunicación deben garantizar que los datos estén también protegidos contra modificaciones accidentales o deliberadas durante la transmisión. Por otra parte, el middleware debe aportar mecanismos de autenticación de usuarios que garanticen que cuando un usuario se conecte a un servicio, éste sea realmente quien dice ser y pueda ser comprobado.
- **Gestión de políticas de seguridad de Organizaciones Virtuales (OVs) compuestas por diferentes organizaciones reales (Hospitales, Centros médicos, etc.) mediante grupos y roles de usuarios.** El middleware debe incluir la gestión completa de usuarios a nivel de OV mediante la creación y manejo de grupos de usuarios y roles, permitiendo una organización y administración de los recursos o servicios desplegados en la arquitectura a nivel de OV. El sistema deberá tener en cuenta los acuerdos alcanzados entre los responsables de la OV y los responsables de las organizaciones reales, sin quitar el control a los responsables de los diferentes dominios administrativos de las organizaciones reales involucradas en la OV.
- **Privacidad de datos guardados en dominios administrativos ajenos.** Se debe garantizar que los datos sean accesibles únicamente a usuarios autorizados y permanezcan confidenciales incluso a usuarios con privilegios de administración. El middleware Grid debe ser capaz de garantizar la privacidad de los datos incluso en entornos administrativos diferentes de los originales e incluso de los administradores de los recursos donde se guardan. Esto debe conseguirse mediante el cifrado de la información, por la cual, sólo los usuarios autorizados deben ser capaces de descifrar dicha información y leerla. Por tanto dentro de la arquitectura se deben definir también aquellos servicios necesarios para el manejo de la privacidad.

2.1.3. Funcionalidad

En esta sección se describe la funcionalidad que debe ofrecer el middleware Grid, compuesto por los componentes de alto nivel de la arquitectura, y que deben estar dirigidos a los desarrolladores e investigadores encargados de crear las aplicaciones. Los objetivos funcionales que se proponen son los siguientes:

- **Compartición de información en formato DICOM.** Uno de los objetivos principales es la capacidad de crear almacenes de información DICOM virtualizados, es decir, formados por almacenes de diferentes organizaciones reales y presentados como si se tratara de un único

almacén. Estos almacenes se formarán para una OV específica, en función de una ontología definida previamente. Los almacenes virtuales deben tener un carácter dinámico, permitiendo incluir o eliminar nuevos estudios o recursos proveedores, en función de los servicios de almacenes que se vayan desplegando en la infraestructura Grid.

- **Definición de ontologías sobre la información en formato DICOM.** Estas ontologías definen el tipo de información que se puede manejar en los almacenes virtuales. Se debe permitir la creación de diferentes almacenes virtuales, que trabajen sobre los mismos almacenes fuente pero utilizando ontologías diferentes.
- **La organización semántica mediante ontologías.** Esto permite un mejor acceso y consulta de los datos. Sobre la totalidad de los datos se definen tres niveles: comunidad de usuarios, experimento y vista. El nivel comunidad de usuario permite identificar los objetos referentes a una comunidad médica, el nivel experimento permite filtrar los datos relevantes para un estudio y el nivel vista se refiere al resultado de una operación de búsqueda.
- **Transferencia de información DICOM.** Debe permitir la transferencia de objetos DICOM de forma eficiente, permitiendo diferentes modalidades de descarga (download) y transferencia (upload) de información, incluyendo la descarga y transferencia progresiva.

2.1.4. Componentes

En este apartado se proponen los objetivos referentes a los componentes de alto nivel que se van a desarrollar, utilizando los servicios definidos en la arquitectura TRENCADIS, que proporcionarán a los investigadores la funcionalidad expuesta anteriormente para desarrollar aplicaciones. Los objetivos que deben cumplir los componentes a desarrollar son los siguientes:

- **Componentes de alto nivel orientados a objetos.** Proporcionan al usuario un interfaz mediante componentes de alto nivel que abstrae al programador del carácter distribuido de los objetos DICOM. Este interfaz debe ser orientado a objetos y debe proporcionar al programador toda la funcionalidad de la arquitectura (proceso sobre objetos DICOM, descarga de objetos DICOM, manejo de almacenes virtuales de objetos DICOM, etc.).
- **Biblioteca de componentes desarrollada sobre Globus Toolkit 4 (GT4).** Se pretende desarrollar una biblioteca o repositorio de componentes middleware sobre Globus 4 y cuya especificación se corresponda con WSRF. En la actualidad, los componentes middleware desarrollados están disponibles en paquetes Java, aunque se espera para un futuro realizar desarrollos en C++ y otros lenguajes.

2.1.5. Aplicaciones

En este bloque se platean una serie de aplicaciones clientes, que se construyen bajo las componentes middleware desarrolladas y cuyo principal objetivo es verificar su funcionalidad, además de demostrar la reutilización de dichos componentes. También se pretende demostrar la utilidad de estos componentes a la hora de mejorar la productividad en su desarrollo. Las aplicaciones propuestas son las siguientes:

- **Aplicación para la Creación de Almacenes Virtuales.** Esta aplicación se encarga de utilizar los componentes del middleware especificados para el almacén de información DICOM. La aplicación debe ser capaz de crear almacenes virtuales, es decir, el usuario debe

ser capaz de definir qué tipo de información pretende manejar de acuerdo a una ontología definida previamente. Cada almacén virtual creado debe contener implícitamente todos los almacenes de información DICOM desplegados como servicios en la arquitectura, y que correspondan a la ontología seleccionada. Estos almacenes distribuidos se manejan por el usuario como un único almacén virtual, a la hora de realizar las operaciones de búsqueda, almacenamiento, recuperación, mantenimiento y administración.

- **Aplicación para la Descarga de Imágenes**. Esta aplicación deberá ser capaz de realizar descargas de imágenes médicas en formato DICOM mediante los componentes definidos en el middleware, permitiendo configurar diferentes modalidades de descarga (descarga progresiva, batch, etc.), además de permitir organizar descargas por prioridades.
- **Aplicación de Decisión Clínica basado en Informes Diagnósticos Radiológicos.** En términos generales, se pretende desarrollar una aplicación Web que se encargue del control de informes de estudios de imagen médica digital en formato DICOM y que corresponda a un área médica o grupo de investigación, seleccionada previamente, es decir, que pertenezca a una ontología definida. La aplicación utilizará las componentes de alto nivel desarrolladas en el middleware, para la gestión tanto de los estudios DICOM como de los informes estructurados asociados a éstos.

  Respecto a la gestión de estudios de imágenes médicas DICOM, la aplicación debe tener un control de aquellos estudios almacenados en las bases de datos compartidas mediante los servicios Grid definidos en la arquitectura. Este control diferenciará estudios que han sido informados de los estudios pendientes de informar. Estos recursos de almacenamiento compartido deben contener estudios previamente seleccionados, es decir, se incluirán aquellos estudios que puedan ser interesantes, o tengan algún tipo de información relevante que pueda ser utilizado a posteriori por herramientas de consulta, a la hora de la toma de decisiones.

  Respecto a los informes estructurados, la aplicación debe ser capaz de generar informes estructurados que sigan el estándar DICOM (DICOM-SR), utilizando las codificaciones estándar. Los informes que se generen deben corresponder a plantillas que definan de una manera formal la estructura y contenidos de los informes. Estas plantillas pueden estar o no especificadas mediante el estándar DICOM. Los informes que se generen deberán ser guardados junto con las imágenes con las que están relacionados, para poder ser consultados posteriormente.

  Como último punto a destacar, la aplicación deberá permitir la búsqueda de los informes estructurados generados y guardados por criterios de contenido. De esta manera se proporciona a los usuarios de una herramienta de consulta muy útil a la hora de la toma de decisiones. Los informes buscados podrán ser descargados junto con las imágenes con las que estén relacionadas. La información sobre la cual se permitirá realizar estas búsquedas vendrá definida por una ontología, especificada previamente para los informes estructurados.

2.2. Hipótesis de Trabajo

En este apartado se describe la principal hipótesis de trabajo sobre la que se basarán los trabajos a desarrollar en ésta tesis, así como una breve explicación del trabajo que conlleva el validar esta hipótesis.

Tal y como se ha planteado en la sección anterior, el objetivo principal es desarrollar una arquitectura SOA, que permita definir e implementar los servicios necesarios para compartir recursos que manejan información DICOM, de una manera segura y definiendo diferentes ontologías, así como la implementación de componentes de alto nivel sobre estos servicios. Fijada la línea de investigación principal, la hipótesis de trabajo que subyace del propio objetivo planteado anteriormente y que se establece como punto de partida para las tareas a realizar, es la siguiente:

Se considera que las arquitecturas Grid orientadas a servicios (SOA) son idóneas para compartir recursos que manejan cualquier tipo de información DICOM (imágenes, documentos estructurados, videos, señales etc…) de una manera segura, garantizando la autenticidad y la integridad de la información en todo momento y permitiendo la definición de diferentes ontologías que estructuren dicha información, de manera que se puedan crear diferentes vistas sobre los mismos conjuntos de objetos DICOM.

Por supuesto, el fijar esta hipótesis de trabajo, también supone la verificación y estudio de los siguientes puntos como justificación de esta hipótesis. Los puntos a verificar y estudiar son los siguientes:

- Demostrar que una arquitectura SOA es la más apropiada para dar soporte al manejo, compartición y proceso de objetos DICOM en general, mediante componentes de alto nivel y de una forma segura entre diferentes organizaciones, utilizando Internet como red de comunicación.
- Demostrar que los problemas de seguridad y ancho de banda de Internet se pueden resolver para el problema específico de la arquitectura mediante las soluciones propuestas.
- Demostrar que el modelo ontológico cubre la semántica necesaria para la organización basada en el informe radiológico de los repositorios distribuidos DICOM.
- Demostrar que las tecnologías Grid son suficientemente apropiadas, en términos de robustez, seguridad y eficiencia, para solucionar los problemas que se plantean, en los objetivos propuestos, a la hora de definir la arquitectura.

Aunque la relación con las hipótesis de trabajo sea indirecta, también cabe realizar la verificación de los siguientes puntos:

- El estudio del modelo de seguridad de la arquitectura TRENCADIS definida, de manera que la información que se maneje sea sólo manipulable por los usuarios autorizados para ello.
- Demostrar la conveniencia de unos componentes de alto nivel para el manejo de los objetos DICOM de manera sencilla, en el desarrollo de aplicaciones de alto nivel.

CAPÍTULO III. Arquitectura TRENCADIS

En este capítulo se aborda la especificación de la arquitectura TRENCADIS, planteada en la sección 2.1 referente a los objetivos de esta tesis. Para ello se va a tomar como punto de partida las hipótesis de trabajo descritas en el capítulo anterior en la sección 2.2, es decir, la definición de la arquitectura debe basarse en tecnologías Grid, orientada a servicios (SOA) y utilizando Internet como red de comunicación.

El objetivo principal de esta arquitectura es dar un soporte para la creación de un middleware Grid, orientado a la gestión y proceso seguro de información en formato DICOM sobre un marco ontológico, sobre el cual desarrollar aplicaciones. Como cualquier arquitectura Grid, tal y como se ha descrito en el apartado 1.4.1 referente a las tecnologías Grid, la arquitectura debe proporcionar una gestión compartida de los recursos, un entorno seguro a nivel de usuarios, recursos y procesos, una transmisión de datos rápida y fiable, y todo esto utilizando tecnologías basadas en estándares abiertos.

En la primera sección de este capítulo se muestra la estructura general de TRENCADIS, dando una definición detallada de la misma basándose en los puntos fundamentales de una arquitectura Grid. Primero se identifican los elementos que componen la arquitectura, a continuación se detallan las diferentes capas en las que se ubican dichos elementos. Por último, se da una descripción detallada de la estructura de los elementos identificados, en función de la capa donde se alberga.

En el segundo apartado, se discute lo referente al sistema de monitorización e información de TRENCADIS, además de describir aquellos elementos requeridos para desplegar este sistema, que en el presente caso es el MDS4, descrito en el apartado 1.5.1.3.

En el tercer apartado, se describirán todos los aspectos relacionados con el modelo de seguridad, además de todos aquellos elementos requeridos en la arquitectura para su implementación. Los puntos a tener en cuenta en este modelo son la autenticación de usuarios y la privacidad de datos en las comunicaciones, descrita en el apartado 1.6.2, los sistemas de autorización Grid descritos en el apartado 1.6.3 y el cifrado de datos descritos en el apartado 1.6.4.

Por otra parte, toda la infraestructura se desarrollará siguiendo las especificaciones OGSA, además de realizar las implementaciones mediante Globus 4, basado en OGSA/WSRF, tal y como se ha comentado en el apartado 1.4.3 referente a las tecnologías Grid.

3.1. Estructura General

En este apartado se define la estructura general de la arquitectura TRENCADIS, tomando como punto de partida los requerimientos marcados en los objetivos y en las hipótesis de trabajo, es decir, que sea una arquitectura organizada por capas, orientada a servicios, abierta, utilizando tecnologías Grid basadas en las especificaciones OGSA y que la red de comunicación sea Internet.

En una arquitectura Grid, las capas que deben componer ésta son cinco: capa de Infraestructura; capa de Sistemas de Monitorización e Información; capa de Conectividad; capa de Seguridad y capa de Aplicaciones. Además, al ser una arquitectura SOA, los diferentes recursos o dispositivos a desplegar (almacenes de datos, recursos de proceso) en la arquitectura serán implementados como servicios y se ubicarán en la capa de Infraestructura.

3.1.1. Elementos de la Arquitectura

Uno de los principales objetivos marcados para la arquitectura TRENCADIS es definir las capas donde ubicar los diferentes elementos que se integran en un despliegue Grid. Por ello, la primera tarea a realizar en la definición de la arquitectura ha sido la definición e identificación de dichos elementos, para así posteriormente ubicarlas en las diferentes capas que la componen.

Los elementos identificados son recursos (servicios), los componentes middleware, las aplicaciones y los protocolos de comunicación, los cuales son descritos detalladamente en los siguientes apartados.

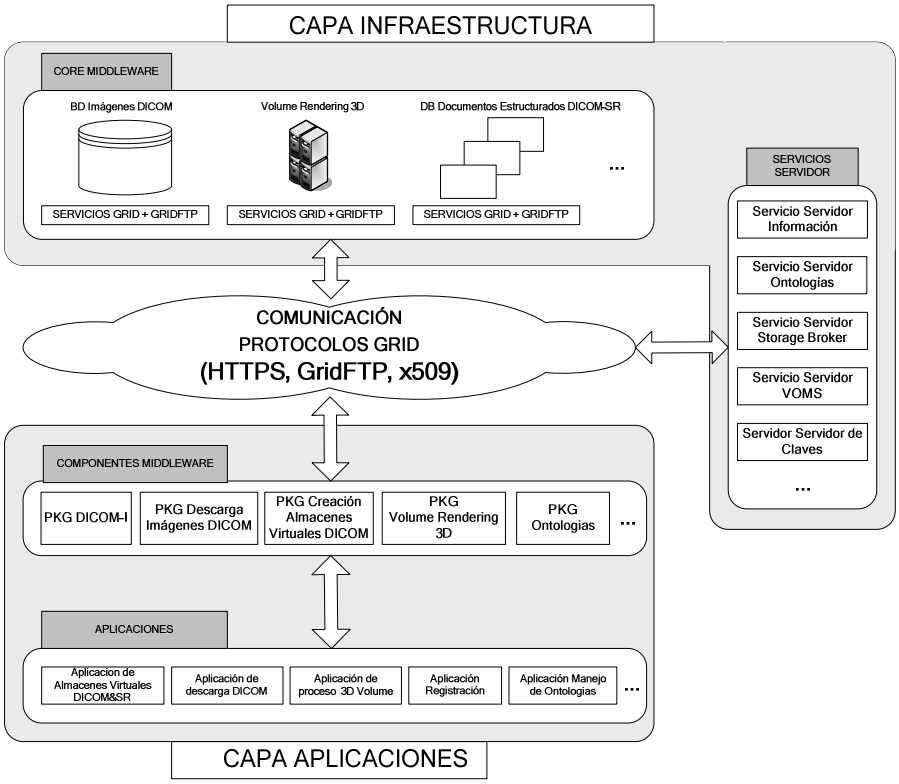

figura 11. Esquema General de la arquitectura TRENCADIS estructurada en capas.

3.1.1.1. Servicios Funcionales y Lógicos

Todos los recursos que se incorporan en la arquitectura TRENCADIS son desplegados como servicios. Los recursos constituyen los elementos de la arquitectura de más bajo nivel, que son aprovechados por la arquitectura para ofrecer a los usuarios Grid diferentes funcionalidades mediante componentes middleware de alto nivel con un interfaz amigable.

Los servicios son, por una parte, el potencial del despliegue Grid, ya que proporcionan la funcionalidad a la arquitectura mediante la interacción directa con los dispositivos (sistemas de información, clusters, supercomputadores, bases de datos, sistemas de almacenamiento en red etc.) que se quieren desplegar en el Grid. Por ejemplo, en la figura 11 se muestra la estructura general de la arquitectura Grid en la que se muestran tres servicios diferentes que interactúan con dispositivos reales; estos son: BD Imágenes DICOM, Volume Rendering 3D y BD Documentos Estructurados DICOM-SR.

Por tanto este tipo de servicios se definen como servicios funcionales. Destacar que los servicios funcionales ofrecen un interfaz homogéneo. Por ejemplo, en el caso del servicio funcional para el recurso BD Imágenes DICOM, se ofrece el mismo interfaz independientemente del sistema gestor de Base de Datos que se utilice para almacenar las imágenes (sistemas de ficheros, base de datos relacionales).

Por otra parte, los servicios además de ser los que interactúan directamente con los dispositivos que dotan al despliegue de funcionalidad, se encargan de gestionarlos de una manera lógica y organizada. Por tanto, pueden existir servicios que no interactúen con ningún dispositivo concreto, sino que se encarguen de organizar los servicios existentes para hacer mas eficiente y coherente su utilización. Por ejemplo, en la figura 11 en la que se muestra la estructura general de la arquitectura TRENCADIS, se muestran cinco ejemplos de estos servicios: servicio Servidor Información, servicio Servidor Ontologías, servicio Servidor Storage Broker, servicio Servidor VOMS y servicio Servidor de Claves, que serán descritos de forma detallada mas adelante.

Este tipo de servicios se definen como servicios lógicos, al no ofrecer funcionalidad de los dispositivos sino que se encargan de tareas de administración de todos los servicios, tanto funcionales como lógicos.

3.1.1.2. Componentes Middleware

El segundo nivel que forma parte de la arquitectura TRENCADIS son los Componentes Middleware. Estos componentes proporcionan a los programadores un nivel de abstracción mayor de los servicios funcionales, ayudados por los servicios lógicos y permitiendo una mayor productividad en el desarrollo de aplicaciones sobre este middleware y explotando las funcionalidades.

En la figura 11 se muestran estas componentes en cinco paquetes, los cuales se encargan de interactuar con los servicios para ofrecer a los desarrolladores de aplicaciones un interfaz orientado a objetos. Los paquetes son los siguientes: PKG DICOM-I para el tratamiento de imágenes, PKG Descarga Imágenes DICOM para la gestión de transferencias, PKG Creación de almacenes Virtuales DICOM para gestionar almacenes virtuales, PKG Volume Rendering 3D para generar imágenes 3D y PKG Ontologías para la gestión de ontologías.

3.1.1.3. Aplicaciones

El tercer nivel son las aplicaciones, que se desarrollan sobre los componentes middleware ofrecidos en los diferentes paquetes, tal y como se muestra en la figura 11. Las aplicaciones nunca interactuarán directamente con los servicios, sino a través de las componentes middleware.

3.1.1.4. Protocolos de comunicación

Como último elemento de la arquitectura, se encuentran los protocolos de comunicación. Estos, tal y como se ve en la figura 11, se encargan de enlazar los servicios con los componentes middleware de una manera segura, atendiendo a los requisitos de seguridad propuestos en el apartado 2.1.2 referente a los objetivos, y que corresponden a la autenticación de usuarios y privacidad de datos en las comunicaciones.

3.1.2. Capas de la Arquitectura

Identificados los elementos que deben formar parte de un despliegue de la arquitectura Grid TRENCADIS, se detallan en este apartado las diferentes capas que conformarán la arquitectura, basándose en las permisas establecidas en el apartado 1.4.2 referente a las tecnologías Grid, estableciendo además el tipo de elemento que albergan.

Tal y como se ha especificado en el apartado 1.4.2 referente a las arquitecturas Grid, las capas que debe ofrecer cualquier arquitectura Grid deben ser cinco: capa de Infraestructura, capa de Sistemas de Monitorización e Información, capa de Conectividad, capa de Seguridad y capa de Aplicaciones.

Con respecto a la capa de Infraestructura, los elementos que se albergan en ésta son los servicios (recursos), tanto funcionales como lógicos. En el diseño de la arquitectura propuesta en ésta tesis, la capa de Infraestructura se subdivide en dos subcapas, en función del tipo de servicio. Por un lado, la subcapa Core Middleware constituye el menor grado de abstracción y aloja a los servicios funcionales. Por otra parte, la subcapa Servicios Servidor aloja a los servicios lógicos.

La división de la capa Infraestructura en dos subcapas es puramente conceptual, pues alberga servicios cuyas tareas son diferentes. Los servicios funcionales interactúan con los dispositivos y los servicios servidor se encargan de tareas de organización y administración, aunque estructuralmente ambos grupos de servicios sean prácticamente iguales.

Respecto a la capa del sistema de Monitorización e Información, ésta será desplegada mediante servicios dentro de las subcapas definidas en la capa Infraestructura. Un sistema de Monitorización e Información requiere de servicios de soporte a nivel de la capa Core Middleware y a nivel de la capa Servicios Servidor. Para este cometido se partirá del sistema MDS4, descrito en el apartado 1.5.1.3 referente a sistemas de Monitorización e Información. Por tanto, en la arquitectura TRENCADIS, la capa de Sistema de Monitorización e Información se traduce en el despliegue de diferentes servicios en la Capa Core Middleware y la capa Servicios Servidor.

La capa de conectividad viene muy condicionada por la utilización de Internet. En la arquitectura TRENCADIS, todo lo referente a la conectividad será incluida en la capa de Comunicación, en la que se definirán los protocolos a utilizar para las comunicaciones entre

todos los elementos de la arquitectura de las diferentes capas, así como la manera de definir los interfaces de comunicación.

Con respecto a la capa referente a la seguridad, la arquitectura TRENCADIS debe garantizar los puntos descritos en la sección referente a la infraestructura de seguridad, es decir, autenticación de usuarios, privacidad de datos en las comunicaciones, sistemas de autorización Grid y cifrado de datos. La solución adoptada en este apartado se basa principalmente en el uso de protocolos basados en SSL y certificados digitales x509, utilizando una infraestructura de clave pública (PKI) que permita emitir los certificados tanto de los servicios a desplegar en la capa de infraestructura como de los usuarios del entorno Grid. Por lo tanto, se requerirá, en la capa de Infraestructura, de servicios que sustenten la PKI para la emisión de los certificados.

La autenticación de usuarios y privacidad de datos en las comunicaciones se garantizará por el uso de la infraestructura Grid de seguridad GSI, descrita en el apartado 1.6.2.4 referente a la infraestructura de seguridad Grid, y cuyos certificados requeridos para el despliegue serán proporcionados por los servicios de la PKI.

Respecto al sistema de autorización Grid, tal y como se ha descrito en el apartado 1.6.3 referente a la infraestructura de seguridad Grid, se emplea VOMS. Para su gestión se necesitan, a partir de sus certificados, servicios para la gestión de los grupos, roles y capacidades de los usuarios Grid. Por otra parte, la autorización se realiza en el propio recurso a partir de esta información, por lo que se debe definir un componente dentro de cada servicio desplegado en la capa Infraestructura que gestione la interpretación de las credenciales que se presentan. En el entorno Grid, se debe asegurar que el administrador local de cada servicio mantenga el control en todo momento. Este componente (“Gatekeeper”) será descrito en los apartados siguientes, cuando se describa la estructura de los servicios.

Por último y respecto a la seguridad, se requerirán servicios en la capa de Infraestructura que implementen el modelo de compartición de claves necesario para el cifrado de los datos. El modelo a emplear se basa en el descrito en el apartado 1.6.4, referente a la infraestructura de seguridad, y se describirá en el apartado 3.3.3 referente al modelo de seguridad.

Por lo tanto, la capa de seguridad se sustenta gracias a la capa de infraestructura mediante servicios en las subcapas de Core Middleware y Servicios Servidor, en la que se despliegan los servicios requeridos para cumplir los aspectos de seguridad planteados anteriormente, además de utilizar los protocolos que ofrece la Capa de Comunicación.

Finalmente, se encuentra la capa de aplicación que se divide en dos subcapas para dar un grado máximo de abstracción de los servicios. Por un lado, la Capa de Componentes Middleware ofrece a los desarrolladores de software un interfaz orientado a objetos para interactuar con los servicios ofrecidos por la capa de Infraestructura, y con los que se comunicará mediante los protocolos definidos en la capa de comunicación. Por otro lado, las aplicaciones se sitúan en la capa de Aplicaciones, desarrolladas a partir de los componentes de alto nivel de la capa de Componentes Middleware.

En resumen, la arquitectura TRENCADIS se estructura en cinco capas mostradas en la figura 11 (capa Core Middleware, capa Servicios Servidor, capa Comunicación, capa Componentes Middleware y capa Aplicaciones). En donde se ubicarán los servicios, componentes, aplicaciones y protocolos requeridos para proporcionar las funcionalidades descritas en los

objetivos planteados. El enfoque de la arquitectura pretende abstraer al usuario de todo lo relacionado con la naturaleza de los datos, dispositivos y sistemas con los que se interactúan, dando una visión de recursos lógicos de alto nivel, interactuando de manera análoga con los recursos que tienen las mismas funcionalidades, independientemente de su ubicación y naturaleza real.

En los apartados siguientes se da una descripción detallada de la estructura de los elementos que se albergan en las cinco capas definidas de la arquitectura TRENCADIS, profundizando en las partes que conforman los servicios de la capa Core Middleware y capa Servicios Servidor, los protocolos a utilizar en las comunicaciones (capa comunicaciones) y las partes en las que se estructura los componentes de alto nivel de la capa de Componentes Middleware Grid. En estas especificaciones también se plantearán aquellos aspectos que afectan a la estructura de los componentes o protocolos necesarios para el cumplimento de los aspectos de seguridad planteados en el apartado 2.1.2 referente a los objetivos.

3.1.3. Estructuras de los Elementos por Capas

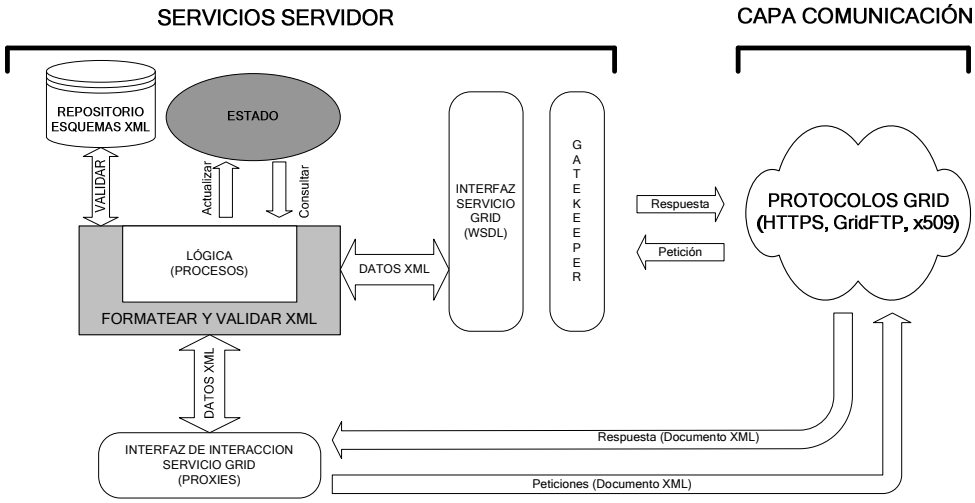

figura 12. Esquema general de un Servicio de la capa Infraestructura.

3.1.3.1. Capa Infraestructura

Tal y como se ha descrito en los apartados anteriores, la capa Infraestructura se encargará de albergar todos los servicios de la arquitectura dentro de las subcapas de Core Middleware y la Capa de Servicios Servidor.

Independientemente de la subcapa donde se ubiquen los servicios (servicios funcionales y lógicos) su estructura es prácticamente igual, excepto en algunos componentes que serán explicados en los apartados siguientes. La figura 12 muestra el diagrama de todos aquellos componentes comunes que forman parte de la estructura de un servicio.

Hay que recordar que las especificaciones de la arquitectura TRENCADIS deben ajustarse a OGSA, lo cual afecta a los servicios tanto en la forma de definir los interfaces para invocarlos, como en su estructura interna. Por un lado OGSA define como deben ser exactamente los interfaces de los servicios genéricos de cualquier aplicación Grid, como por ejemplo, los servicios de un Sistema de Monitorización e Información. En caso de no ser servicios genéricos, OGSA también especifica una serie de requerimientos para definir los interfaces de dichos

servicios. Por otra parte, tal y como se ha comentado en el apartado 1.4.3 referente a las tecnologías Grid, OGSA requiere de una funcionalidad que no puede ser proporcionada por los servicios web, por ello WSRF extiende los servicios web dando respuestas a algunas de estas carencias, como notificaciones de eventos, manejo del ciclo de vida de un servicio, transacciones y gestión del estado, entre otras, con lo que OGSA afecta directamente a la estructura de los servicios. Siguiendo la nomenclatura OGSA, se referirá a los servicios a partir de ahora como servicios Grid.

A continuación se describen aquellos elementos estructurales que cualquier servicio Grid, en la capa Infraestructura de la arquitectura, dispone, y que son los siguientes:

- **Estado del Servicio Grid.** El estado del Servicio Grid se define mediante atributos, el valor de estos atributos determinan la funcionalidad del servicio y persisten tras su invocación. La funcionalidad se realiza mediante procesos lógicos definidos en el propio servicio, tal y como se muestra en la figura 12.

  En el caso de los servicios funcionales, el estado será definido de la misma forma en todos los servicios que compartan la misma funcionalidad, independientemente del dispositivo con el que actúe. Por ejemplo, volviendo al ejemplo del servicio de almacén de imágenes DICOM mostrado en la figura 11, este servicio puede interactuar con diferentes dispositivos reales (bases de datos relacionales, manejadores de disco duro) ofreciendo el mismo interfaz en todos los servicios. En este caso, el estado también tendría la misma estructura independientemente del dispositivo con el que interactúe. Concretamente, el estado de este servicio funcional se compondrá de la información de las imágenes DICOM que se albergan en los dispositivos.

  Obviamente, en el caso de los servicios lógicos, el estado tampoco varía si la tarea de organización o administración que realizan es la misma. Por ejemplo, todos los servicios servidor VOMS tendrán un estado que vendrá dado por los usuarios Grid dados de alta con la información adicional referente a estos (grupos, roles, capacidades).

- **Procesos lógicos del Servicio Grid.** El concepto de proceso también es el mismo, tanto para los servicios funcionales como lógicos. Al igual que ocurría con el estado, estos procesos son independientes del dispositivo con el que se interactúa, si la funcionalidad que ofrece es la misma en el caso de servicios funcionales, o de las tareas de organización o administración en el de servicios lógicos. Esto es debido a que los procesos lógicos tienen como objetivo modificar o consultar el estado definido en el servicio Grid. Para ello definen unos parámetros de entrada, a partir de los cuales realizarán las consultas y modificaciones del estado en función de una lógica de proceso implementada, y unos parámetros de salida que serán el resultado de estas operaciones. Estos parámetros de entrada y salida utilizan unas estructuras de datos predefinidas mediante documentos XML, tal y como se muestra en la figura 12.

  Siguiendo con el ejemplo del servicio funcional del servicio de almacén de imágenes DICOM, un proceso lógico es el encargado de búsquedas en función de unos criterios determinados y especificados mediante un documento XML de entrada, retornando un resultado en un documento XML atendiendo al dato buscado.

Los procesos lógicos solo podrán ser ejecutados previa validación de los parámetros de entrada y por medio de los interfaces definidos en el servicio Grid, o bien por el componente de interacción en el caso de servicios funcionales.

- **Componente validación de los formatos de entrada/salida de los procesos lógicos.** Este componente, tal y como se observa en la figura 12, se encuentra entre los procesos lógicos y aquellos interfaces que pueden disparar estos. En este componente se guarda la definición de los formatos de entrada y salida mediante esquemas XML que se almacenan en un repositorio, formalizando las estructura de datos, tal y como se describió en al apartado 1.1.2.1 referente a los metalenguajes. El objetivo de este componente es homogeneizar el formato de los datos, de manera que puedan ser procesados por los procesos lógicos implementados en el servicio Grid, independientemente del dispositivo con quien interactúe, si hablamos se servicios funcionales de una misma funcionalidad o de la tarea de organización o administración para los servicios lógicos.

  Continuando con el ejemplo del servicio de almacén de imágenes DICOM, cabría definir los interfaces de entrada/salida para los procesos lógicos que se encargan de cambiar el estado del servicio (p.e. insertar o borrar información DICOM) o bien los parámetros de una petición de un usuario mediante el interfaz de cliente (p.e. búsquedas de información mediante criterios de búsqueda con una determinada ontología).

- **Gestor de Seguridad (Gatekeeper).** El acceso e interacción con los servicios Grid debe ser solo posible por los usuarios Grid debidamente autenticados y autorizados. Además, en el caso de la imagen médica, se requiere realizar transferencias de grandes volúmenes de datos, para los que se utiliza el protocolo GridFtp, el cual también debe autorizar adecuadamente a los usuarios si estos están debidamente autenticados y autorizados. Tal y como se muestra en la figura 12, todas las peticiones que se realizan a través de los interfaces pasan por el componente GateKeeper.

  Para todo esto, debe existir un componente que se integre dentro de los requerimientos de las infraestructuras de seguridad Grid, especificados en la sección 1.6 referente a la infraestructura de seguridad, tanto para la parte de autenticación de usuarios y privacidad de datos en las comunicaciones como en lo relativo a la autorización Grid basada en certificados VOMS.

  La autenticación de usuarios y privacidad de datos se basa en GSI. Esta infraestructura se apoya en los protocolos de la capa de comunicación, basados en SSL y certificados x509. Tal y como se ha expuesto en el apartado 1.6.2.3 referente a la infraestructura de seguridad en el Grid, se consigue crear un canal seguro entre las dos partes mediante el uso de criptografía asimétrica que se utiliza para el cifrado y firmado de los datos que se envían en las comunicaciones. El encargado de realizar todas estas tareas en el servicio Grid será el Gatekeeper.

  La figura 13 ilustra el esquema general de acceso a un servicio a través de un Gatekeeper. El primer paso a realizar por parte del usuario Grid, a la hora de interactuar con un servicio Grid, es la creación de un certificado x509 y su presentación al sistema de autorización Grid (paso 1). El sistema de autorización Grid retorna al usuario Grid la credencial (Proxy) con los atributos (grupo, roles, capacidades) del usuario (paso 2). Una vez recuperados los

atributos, estos son presentados al Gatekeeper (paso 3) mediante los protocolos ofrecidos por la infraestructura GSI, que además de garantizar el canal seguro, también implementa la identificación de usuarios mediante la lectura de certificados x509 mediante su Distinguish Name (DN). El Gatekeeper primero comprueba que el certificado x509 que presenta el usuario es de confianza, es decir, está firmado por una entidad certificadora con la que se confía (integrada en la PKI a desplegar mediante los servicios de la capa infraestructura). En caso positivo se identifica al usuario que realiza la conexión mediante los datos incluidos en el certificado (este paso se incluye dentro del paso 3).

Respecto a la integración de un sistema de autorización Grid, una vez aceptada la conexión y establecida la comunicación de manera segura identificando al usuario mediante su DN, se procede a la lectura de los atributos proporcionados por el sistema de autorización Grid, extrayendo los grupos, roles y capacidades que posee el usuario a nivel de OV. Estos permisos son administrados y guardados en una base de datos local del propio recurso. A partir de estos permisos, el administrador deberá, en función de los acuerdos que haya alcanzado con la OV correspondiente a la hora de compartir el servicio (paso 3.1), denegar o aceptar la conexión con el usuario Grid (paso 3.2).

Para las operaciones que requieran la descarga de datos de gran dimensión, se identifica en primer lugar al usuario que la inicia y se le otorgan privilegios de uso de acuerdo a los pasos anteriores. Posteriormente, el servicio le da permiso para la utilización del servidor GridFTP (paso 3.3). Esta componente se utiliza sólo en algunos servicios Grid, requiriendo de un fichero grid-mapfile en el que el administrador del recurso, anota los DNs de los usuarios Grid que pueden utilizar el GridFTP (paso 3.3). El Gatekeeper se encarga de actualizar de manera automática el grid-mapfile, insertando el DN en el grid-mapfile cuando se requiera una descarga mediante GridFTP (paso 4), y eliminando éste cuando dicha descarga haya finalizado.

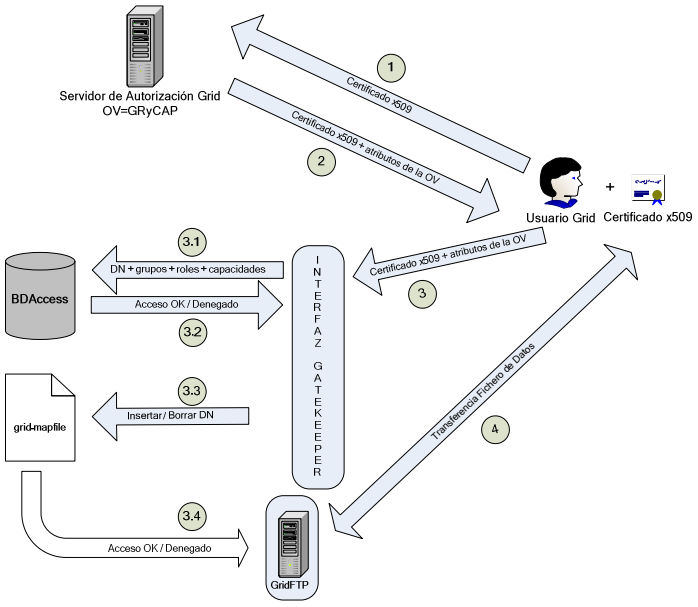

figura 13. Esquema general de acceso a un servicio Grid a través de un Gatekeeper.

- **Interfaz de Interacción con los Servicios Grid (Proxies).** Los procesos lógicos, en la implementación de la propia lógica de proceso, pueden requerir otros Servicios Grid

definidos y desplegados en la infraestructura de la arquitectura. Por ello, TRENCADIS define un interfaz mediante el cual se implementa la comunicación entre servicios Grid. Este interfaz sólo puede ser utilizado por los procesos lógicos del servicio Grid que los necesitan.

Siguiendo con el ejemplo del servicio de almacén de imágenes DICOM, existirán procesos lógicos que necesitarán recuperar información sobre una determinada ontología con la que el proceso está operando. Por ello, este servicio necesita interaccionar con el servidor de Ontologías y el servidor de Índices.

- **Interfaz del Servicio Grid.** El objetivo principal de esta parte del servicio es ofrecer un mismo interfaz para los servicios que tengan la misma funcionalidad (en el caso de los servicios funcionales), o las mismas tareas de organización y administración en el caso de servicios lógicos. Este interfaz deberá ser igual tanto a nivel de especificación como a nivel de protocolo de comunicación. Los protocolos y la manera de especificar los interfaces de comunicación vienen definidos en la capa de Comunicación. La definición de estos interfaces se realiza mediante el lenguaje descriptivo Web Service Descripion Language (WSDL), que es el mismo que el empleado por la tecnología de los servicios Web y entra dentro de las especificaciones OGSA. De esta manera cualquier entidad capaz de interpretar WSDL será capaz de interactuar con el servicio Grid si es aceptado por el GateKeeper.

  Las interfaces ofrecidas siempre van ligadas a un proceso lógico que se encargará de consultar, procesar o modificar el estado del servicio Grid, previa validación de los parámetros de entrada y salida mediante los esquemas XML, según el tipo de petición.

  Estos interfaces son utilizados por los componentes de alto nivel definidos en la capa de Componentes Middleware, o bien por los procesos lógicos de otros servicios Grid que requieran de determinada información o proceso.

  Por ejemplo, en el servicio de almacén de imágenes DICOM se requerirá la definición de un interfaz que realice las búsquedas de imágenes, dados unos criterios de búsqueda determinados y una ontología concreta.

Además de todos estos componentes, la estructura de los servicios de la capa de infraestructura dispone de componentes específicos, que sólo se dan en los servicios funcionales o en los servicios lógicos. Los siguientes apartados describen los aspectos particulares de los servicios funcionales y los servicios lógicos.

3.1.3.1.1. Servicios Funcionales

En este apartado se describen aquellos componentes particulares de los servicios funcionales que se albergan en la capa Core Middleware. Tal y como se ha comentado en el apartado anterior, en los servicios funcionales que proporcionan la misma funcionalidad no varían la especificación de los componentes, independiente del dispositivo con el que interactúan. Para conseguir esta abstracción se requiere del siguiente componente:

- **Componente de interacción.** Este componente es capaz de interactuar con el dispositivo que se quiera integrar en el entorno Grid. Este modulo de interacción depende de las características del dispositivo y será necesario reimplementarlo, al menos en parte, cada vez

que se cambie de dispositivo. Este componente lanzará procesos lógicos para la consulta o modificación del estado del servicio Grid.

Siguiendo con el ejemplo del servicio de almacén de imágenes DICOM, en la figura 14 se muestra un recurso para almacenar objetos DICOM mediante una base de datos relacional SQL. El componente de interacción debe traducir las peticiones o cambios en la base de datos SQL a estructuras XML entendibles por los procesos lógicos encargados de modificar el estado del servicio Grid y, por otra parte, debe también traducir las peticiones de los procesos lógicos a sentencias SQL para abstraer la información que se necesite. Además, en este caso concreto, debe implementar los requisitos necesarios para que se cumpla la legislación vigente referente a la protección de datos, al ser la información a compartir protegida por diversas resoluciones en materia de protección de datos.

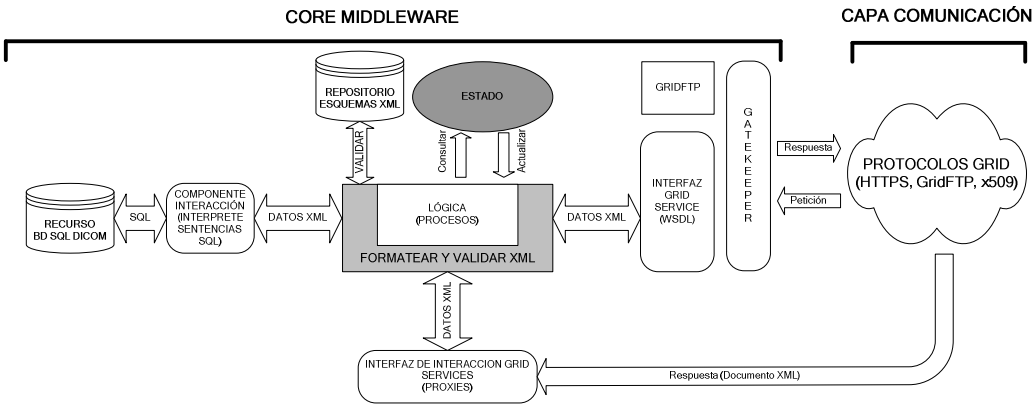

figura 14. Esquema de un Servicio Funcional, que corresponde a un Almacén de Objetos DICOM gestionado mediante una base de datos SQL.

3.1.3.1.2. Servicios Lógicos

La diferencia fundamental de estos servicios, con respecto a los servicios funcionales de la capa Core Middleware, es que no interactúan con ningún recurso de forma directa. Estos servicios se encargan de realizar tareas de organización y administración para gestionar a nivel lógico los servicios Grid desplegados.

Las partes que conforman los servicios ubicados en esta capa son las mismas que las presentadas para la capa infraestructura en el apartado 3.1.3.1, a excepción de los siguientes matices:

- **Gestor de Seguridad (GateKeeper).** El gestor de seguridad de las componentes de la capa de Servicios Servidor tiene una parte análoga al descrito en el apartado 3.1.3.1. Además de esta funcionalidad, este componente también debe filtrar los permisos de acceso a un nivel de grano mas fino, mas allá de las credenciales que se presenten al recurso mediante los certificados x509 y los atributos correspondientes. En el caso de los servicios funcionales se permitía el acceso o no al recurso en función de las credenciales presentadas (grupos, roles y capacidades). En este caso también se puede requerir de otras restricciones de seguridad referentes a los datos que maneja, debido a que su función se basa en tareas administrativas de gestión de mas alto nivel como, por ejemplo, en el servicio lógico donde se almacene información de las ontologías con las que se quiere trabajar, será necesario controlar qué

usuarios pueden o no manejar determinadas ontologías, además de garantizar los puntos de seguridad mencionados anteriormente.

3.1.3.2. Capa Comunicación

La capa de Comunicación proporciona los protocolos y la manera de definir los interfaces de comunicación, que utilizarán los proveedores y clientes de los servicios Grid en las capas Core MiddleWare y Servicios Grid Servidor. Además, los componentes desarrollados en la capa de Componentes MiddleWare son también clientes de los servicios desplegados en las capas de Infraestructura, con lo que también utilizarán estos protocolos.

Tal y como se ha descrito en la sección 2.2 referente a las hipótesis de trabajo, la red de conexión sobre la cual se va a desplegar la arquitectura es Internet, con lo que se utilizarán protocolos y lenguajes comunes y estándares en la medida de lo posible. Por otra parte, la capa de comunicación debe garantizar los puntos de seguridad descritos en el apartado 1.6.2 referentes a la infraestructura de seguridad Grid, que hacen mención a la autenticación de usuarios y la privacidad de datos en las comunicaciones, basados en la infraestructura Grid de seguridad GSI descrita en el apartado 1.6.2.4.

Más concretamente, los protocolos utilizados son, por un lado SOAP, que constituye un protocolo de más alto nivel basado en tecnologías XML y puede ser soportado por otros protocolos de más bajo nivel (http, https, ftp, ftps, etc.). Para la comunicación con los servicios Grid implementados se utiliza SOAP bajo https, ya que https es un protocolo típico de Internet y está basado en SSL y certificados digitales que garantizan la autenticación de usuarios y privacidad de datos en las comunicaciones.

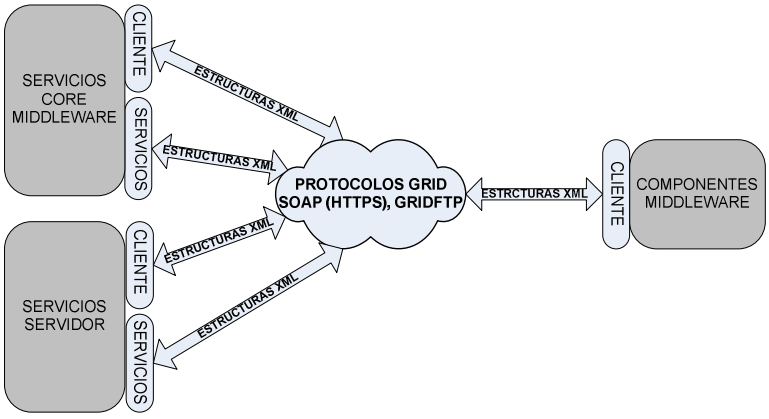

figura 15. Esquema de la Capa Comunicación.

Por otra parte, se utiliza el protocolo GridFTP para la transferencia de datos. Este protocolo también se basa en SSL y certificados digitales x509, siendo una extensión del protocolo FTP que incorpora nuevas funcionalidades, entre las que cabe destacar el control de terceros o la capacidad de multiplexar varios canales simultáneos entre otros. GridFTP se utiliza para la transferencia de grandes volúmenes de datos (como la transferencia de estudios de imágenes médicas DICOM).

Tanto SOAP/https como GridFTP son protocolos que se adaptan perfectamente a las necesidades definidas de la arquitectura, pues funcionan de forma segura y eficiente en Internet y en redes TCP/IP no dedicadas.

Los interfaces de los diferentes servicios serán descritos mediante el lenguaje Web Service Description Language (WSDL), que es el mismo que utilizan los servicios Web y es aceptado por infraestructuras que implementa OGSA como WSRF.

3.1.3.3. Capa Componentes Middleware

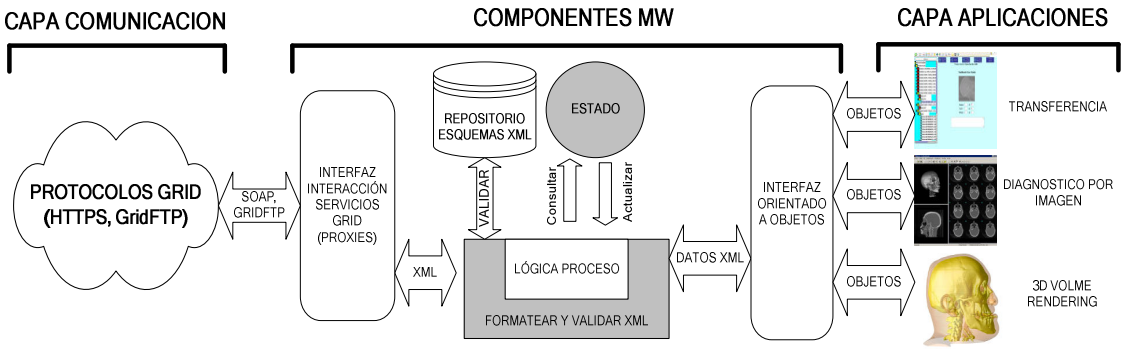

figura 16. Esquema de un componente Middleware de la capa Componentes Middleware.

El principal objetivo de esta capa es el desarrollo de componentes para cubrir los objetivos funcionales comunes a diferentes aplicaciones, y que implican un nivel de abstracción mayor del concepto de recurso, como lo será el almacén virtual de objetos DICOM o la transferencia de objetos DICOM. Esta capa se sitúa sobre las capas que proporcionan los servicios Grid (Core Middleware y Servicios Servidor), utilizando estos servicios para cubrir la funcionalidad. Estas componentes suponen el nivel de abstracción más alto dentro de la arquitectura, ocultando a los desarrolladores las características referentes al formato, protocolos de comunicación, ubicación, etc…, de los servicios Grid que se quieren utilizar, y aportando un manejo local y homogéneo de los recursos que se despliegan en el Grid.

Todo componente está compuesto por las siguientes cinco partes diferenciadas:

- **Estado del Componente Middleware.** El estado del Componente Middleware, al igual que ocurre con los servicios Grid, se define mediante atributos, describiendo, por una parte, el estado en el que se encuentra el componente y, por otra, el comportamiento que va a tener el servicio en función de las acciones que se realicen sobre éste. Dichas acciones se realizan mediante procesos lógicos definidos en el propio componente, tal y como se muestra en la figura 16.
- **Procesos lógicos del Componente Middleware.** Los procesos lógicos tienen como único objetivo modificar o consular el estado definido en el Componente Middleware, para lo que se definen unos parámetros de entrada a los procesos, a partir de los cuales se realizará la consulta y modificación del estado en función de una lógica de proceso implementada, y unos parámetros de salida que serán el resultado de estas operaciones. Estos parámetros de entrada y salida, tienen unas estructuras de datos definidas mediante documentos XML, tal y como se muestra en la figura 16.
- **Componente de validación de los formatos de entrada/salida de los procesos lógicos.** Este componente, tal y como se observa en la figura 16 se encuentra entre los procesos lógicos y aquellos interfaces que pueden ejecutar éstos (interfaz orientado a objetos e interfaz interacción con servicios Grid). En este componente se guarda la definición de los

formatos de entrada y salida mediante esquemas XML que se almacenan en un repositorio, formalizando las estructuras de datos, tal y como se describió en al apartado 1.1.2.1 referente a los metalenguajes. El objetivo de este componente es homogeneizar el formato de los datos, de manera que puedan ser procesados por los procesos lógicos implementados en el Componente Middleware.

- **Interfaz orientado a objetos.** Se proporciona a los desarrolladores de aplicaciones un interfaz definido como un modelo de objetos de alto nivel, que el desarrollador utilizará directamente como una biblioteca software de clases. Los objetos darán a los desarrolladores una funcionalidad por medio de los atributos y métodos que ofrecen. Por ejemplo, en el caso de almacenes virtuales, el desarrollador sólo tendrá que declarar un objeto almacén para virtualizar una serie de repositorios locales distribuidos de objetos DICOM, que publicarán solamente la información relevante atendiendo a los criterios de selección en la creación. Una vez creado, este almacén virtual manejará un conjunto de servicios almacén como si se trataran de un único almacén.
- **Interfaz de Interacción de Servicios Grid (Proxies).** Los componentes middleware utilizan los formatos definidos en XML para comunicarse con los servicios Grid de la capa Infraestructura. Para esta comunicación se utilizan los protocolos ofrecidos por la capa de Comunicación. Los componentes middleware pueden interaccionar con uno o varios servicios Grid de diferentes capas y diferentes funcionalidades, en función del componente de alto nivel que se desee implementar. Por ejemplo, en el caso de un componente de creación de almacenes virtuales de objetos DICOM, éste se comunicará con todos aquellos servicios Grid desplegados relacionados con el almacén de objetos DICOM, para virtualizar estos almacenes como si fueran un único almacén.

Esta capa ofrecerá todas las funcionalidades a los desarrolladores, la aparición de nuevas funcionalidades implicará el desarrollo de nuevos servicios Grid en las capas inferiores.

3.1.3.4. Capa Aplicaciones

Tal y como se ha comentado en el apartado 3.1.2, la capa Aplicación de una arquitectura Grid se encuentra subdividida en dos subcapas, la capa de componentes Middleware expuesta en el apartado anterior, la cual se encarga de su funcionalidad y la capa de Aplicaciones que, aunque tenga el mismo nombre en la definición de la arquitectura TRENCADIS, sólo se refiere a la parte de interfaz que interactúa con los Componentes middleware a través del interfaz orientado a objetos de alto nivel que ofrecen. Esta capa se sitúa en el nivel más alto de la arquitectura.

Las aplicaciones pueden ofrecer cualquier tipo de interfaz, y seguir diferentes filosofías de diseño software como cliente-servidor, aplicaciones web, aplicaciones J2EE etc. En este trabajo se desarrollan una serie de aplicaciones enfocadas a la validación y testeo de las componentes middleware desarrolladas y que serán comentadas más adelante. Dos de ellas serán aplicaciones Java y una será una aplicación Web que interactúa con los usuarios mediante navegadores Web. Estas aplicaciones pueden a su vez ser módulos, con los que se constituyen aplicaciones de mayor nivel.

3.2. Sistema de Información y Monitorización MDS4-Extended

Tal y como se ha comentado en el apartado 1.4.2 referente a las tecnologías Grid, un Sistema de Monitorización e Información es una parte imprescindible para cualquier arquitectura Grid. En la actualidad, existen diversos sistemas de Monitorización e Información que se están utilizando en diferentes entornos como GMA/R-GMA, MDS, MDS2, MDS4 y DiDA. Todos estos sistemas han sido descritos en la sección 1.5 referente a sistemas de Monitorización e Información, donde se plantean las ventajas e inconvenientes de cada una de estas aproximaciones respecto a las hipótesis de trabajo y los objetivos planteados en la tesis.

Todo sistema de Monitorización e Información tiene unas características fundamentales, que pueden ser implementadas de diferentes maneras. Estas características son:

- Se debe gestionar un nombre de servicio único para la identificación inequívoca de recursos.
- Se debe permitir un acceso ubicuo al sistema para que pueda ser accesible desde cualquier punto.
- El funcionamiento del sistema debe ser transparente.
- El sistema debe ser escalable y permitir mantener información de nuevos recursos de una manera fácil y eficiente.

Por otra parte, existen otras características que, si bien no son necesarias, sí son recomendables a la hora de implementar estos sistemas, como:

- Carácter distribuido del sistema. La información no debería concentrarse en un único punto y debería poder ser manejada de manera eficiente.
- Acceso a toda la información. Cualquier recurso desplegado en la arquitectura debe poder interrogar el sistema completo.
- Fácil accesibilidad. En el sistema puede existir diferentes puntos de acceso.
- Tolerancia a fallos. El fallo de un componente no debe provocar un fallo completo en el sistema.
- Fácil interrogación del sistema. El lenguaje de interrogación debe ser simple y efectivo.
- Interfaces de interacción estándar con el sistema. Para facilitar la integración del sistema en diferentes despliegues de arquitecturas Grid.

Atendiendo a estas características, el sistema de Monitorización e Información desarrollado en este trabajo, se basa en MDS4, expuesto en el apartado 1.5.1.3 referente a los sistemas de Monitorización e Información. Uno de los principales problemas de MDS4 es la escalabilidad en la gestión de grandes volúmenes de datos. Otro de los problemas que se presenta en este sistema es que las consultas se realizan mediante el lenguaje XPATH, al guardarse la información mediante ficheros XML, que a pesar de ser un lenguaje estándar puede resultar un tanto problemático a la hora de realizar consultas complejas al sistema. Por tanto, en ésta tesis se propone extender MDS4, de forma que se mejoren los aspectos de escalabilidad, proporcionando además al usuario el lenguaje de interrogación SQL, más sencillo que XPATH para consultas complejas. A este sistema le llamaremos MDS4-Extended.

Como primer paso para extender MDS4, se ha procedido a la identificación de los tipos de información que puede manejar el Sistema de Monitorización e Información. Por una parte se encuentra la información básica que será manejada por el sistema MDS4, y por otra parte está la información extendida, que será manejada primero por el MDS4 y después por un nuevo componente que se definirá dentro de MDS4.

La información básica corresponde a la información esencial de los servicios, y es accesible por el sistema (URL, tipo de servicio), de manera que cuando un usuario pregunte al sistema la ubicación de los servicios de un determinado tipo, éste contesta con la dirección básica. Para este tipo de información se mantiene la funcionalidad de MDS4, pues la información a almacenar en el sistema puede ser manejada de una manera eficiente mediante ficheros XML en los diferentes IDXSRV, sin afectar a la escalabilidad. Además, las consultas de este tipo de interacción son sencillas y pueden ser realizadas mediante XPATH.

Por otra parte, la información extendida corresponde a aquella información que hay que publicar en el sistema de Monitorización e Información y que, debido a su volumen, necesita de un tratamiento especial para su manejo y consulta, siendo ineficiente el uso de ficheros XML en los IDXSRV. Por ejemplo, uno de los objetivos marcados en ésta tesis es la creación de almacenes virtuales, de manera que se puedan manejar diferentes servicios que gestionan y guardan objetos DICOM como un único almacén. A la hora de crear este almacén virtual, se requiere interrogar al sistema sobre los criterios de creación, es decir, donde están los servicios que almacenan los objetos DICOM que contienen información DICOM que cumplan los criterios de creación. Un caso concreto sería, por ejemplo, que se desee crear un almacén virtual de todos aquellos objetos DICOM cuya modalidad sea resonancia magnética, pertenecientes a estudios anteriores al año 2000. Para ello, el MDS4 debería obtener y publicar dicha información, por lo que si los servicios de almacén contuvieran muchos estudios, la información a publicar (fechas de los estudios y los tipos de estudio) podría tener un volumen de datos que haría ineficiente la utilización del sistema MDS4. Por ello, se ha decidido extender el sistema MDS4 manteniendo la base para consultas sencillas tales como la ubicación de servicios mediante los IDXSRV, y definiendo al componente IDXSRV-Extended, el cual será un servicio lógico ubicado, al igual que el IDXSRV, en la capa Servicios Servidor de la capa Infraestructura, y que guardará la información extendida mediante una base de datos relacional, de manera que puede ser consultada mediante el lenguaje SQL. La información de los IDXSRV-Extended es publicada en los IDXSRV correspondientes, para que pueda ser consultada en el sistema MDS4 estándar mediante consultas XPATH.

Por tanto, la forma de proceder en el Sistema MDS4-Extended, que se plantea para la arquitectura TRENCADIS, varía en función del tipo de información que se quiera manejar. Por una parte, si se pretende consultar información básica se interrogan los IDXSRV correspondientes mediante el lenguaje XPATH y, por otra parte, si se trata de información extendida, la consulta se realiza en dos pasos, primero se interroga el sistema a través de los IDXSRV para localizar los IDXSRV-Extended que se requieran mediante el lenguaje XPATH, y una vez localizados, se consulta la información correspondiente mediante el lenguaje SQL en el IDXSRV-Extended correspondiente. En el ejemplo anterior, si se pretende crear un almacén virtual, primero se preguntará a los IDXSRV sobre los servicios que indexan la información de los servicios que almacenan de objetos DICOM mediante una consulta simple XPATH (ver

figura 17), y una vez localizados, se les preguntaría a éstos mediante SQL las ubicaciones de los servicios de almacén que contienen los estudios que cumplen los criterios especificados en la ontología creada.

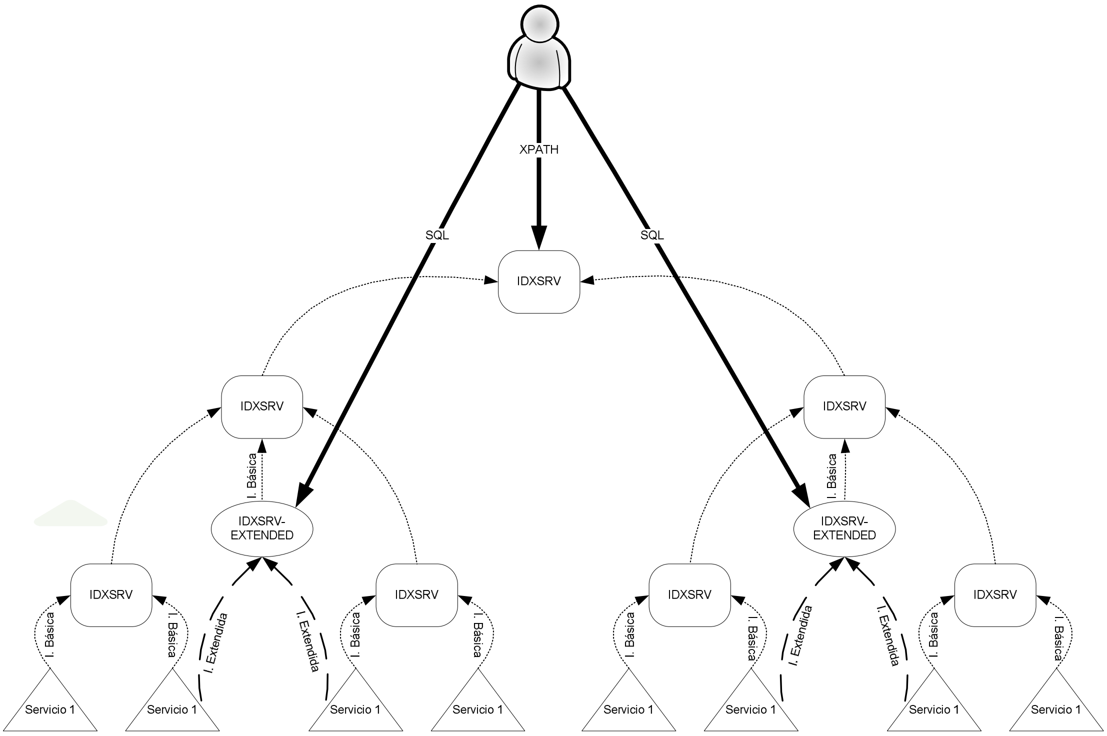

figura 17. Esquema de General de los servicios de MDS4-Extended.

3.2.1. Infraestructura

Respecto al despliegue en la arquitectura TRENCADIS, para la implementación de este sistema de Monitorización e Información (MDS4-Extended), se requieren de los servicios lógicos (IDXSRV e IDXSRV-Extended) en la capa Infraestructura. Estos servicios realizan tareas de gestión y no requieren de interacción con ningún dispositivo concreto, tratándose de servicios puramente lógicos. Al ser estos servicios lógicos siguen la estructura descrita en el apartado 3.1.3.1.

3.2.1.1. Servicio de Índice (IDXSRV)

Este servicio forma parte de MDS4 y se encarga de registrar la información referente de los diferentes servicios a desplegar en la capa Infraestructura, para que ésta sea accesible desde cualquier punto del sistema.

Por otra parte, también se encargan de proporcionar información registrada en el Sistema de Monitorización e Información mediante consultas con sintaxis XPath, al estar la información guardada mediante documentos XML.

Este servicio se incorpora dentro de los servicios implementados para el Sistema de Monitorización e Información MDS4, el cual está incluido dentro de GT4.

3.2.1.2. Servicio de Índice Extended (IDXSRV-Extended)

Este servicio no forma parte de MDS4, es un componente definido en la tesis para resolver los problemas de escalabilidad y facilitar la interrogación del sistema. Se encarga de registrar la

información extendida, para que ésta sea accesible desde cualquier punto del sistema y pueda ser consultada mediante el lenguaje SQL. Al ser éste un servicio lógico, sigue la estructura planteada en la arquitectura en el apartado 3.1.3.1 referente a la estructura general de la arquitectura.

Los diferentes servicios IDXSRV-Extended, que se definen para cubrir los objetivos funcionales planteados, serán descritos en el capítulo correspondiente a la funcionalidad de la arquitectura, donde se incluyen más detalles referentes a las implementaciones realizadas para cubrir las funcionalidades planteadas.

3.3. Modelo se Seguridad

La capa de Seguridad es otra de las partes imprescindibles de cualquier despliegue Grid, tal y como se ha expuesto en el apartado 1.4.2 referente a las tecnologías Grid. En la arquitectura TRENCADIS, se cubren los tres aspectos claves referentes a la seguridad, planteados en el apartado 2.1.2 referente a los objetivos, que son:

- Autenticación de usuarios y privacidad de datos en las comunicaciones.
- Gestión de políticas de seguridad de Organizaciones Virtuales (OVs) compuestas por diferentes organizaciones reales (Hospitales, Centros médicos, etc.) mediante grupos y roles de usuarios.
- Privacidad de datos en dominios administrativos ajenos.

Para los dos primeros aspectos se utiliza como base los entornos del GSI [56] y VOMS, aplicándolos y desarrollando nuevos componentes para su adecuación a la arquitectura TRENCADIS. Respecto a la privacidad de los datos en dominios administrativos ajenos, se han extendido los desarrollos descritos en [92], modificando por una parte el modelo original y por otra desarrollando nuevos componentes, también para adecuar el modelo a la arquitectura TRENCADIS.

En los siguientes apartados, se describen diferentes elementos que se han aplicado y desarrollado para abordar los aspectos referentes a la seguridad dentro de la infraestructura TRENCADIS, es decir, los protocolos y los servicios Grid, de forma que la arquitectura cumpla con los objetivos planteados en el capítulo segundo. Finalmente, se exponen los componentes middleware desarrollados sobre los servicios Grid de la infraestructura, y los protocolos para ofrecer al usuario todos los aspectos de seguridad de una forma transparente y sencilla.

3.3.1. Autenticación de usuarios y privacidad de datos en las comunicaciones

La autenticación de usuarios y privacidad de datos es un problema común en las comunicaciones, que se agudiza en redes públicas como Internet, debido a su naturaleza insegura. Tal y como se ha comentado en el apartado 1.6.2 referente a la infraestructura de seguridad, en la actualidad este tipo de problema se aborda mediante protocolos basados en SSL, que se apoyan en el uso de certificados digitales x509. Tanto el algoritmo SSL como el estándar x509 han sido explicados respectivamente en los apartados 1.6.2.3 y 1.6.2.2 referentes a la infraestructura de seguridad.

3.3.2. Gestión de políticas de seguridad de Organizaciones Virtuales

El segundo punto a tener en cuenta en el modelo de seguridad, es el referente a la gestión de los permisos de acceso de los miembros de una OV, es decir, incluir dentro del despliegue Grid un sistema de autorización Grid que gestione diferentes dominios administrativos reales como un único dominio virtual. Un sistema de autorización Grid debe ser capaz de conceder o denegar permisos a un usuario Grid perteneciente a una OV, para llevar a cabo una determinada acción sobre un recurso o servicio determinado.

En el apartado 1.6.3 referente a la infraestructura de seguridad Grid, se han expuesto diferentes sistemas de autorización Grid que en la actualidad se están utilizando en diferentes

entornos. Como en el apartado anterior, en ésta tesis se va a utilizar un sistema de autorización ya existente, concretamente el VOMS, requiriéndose algunos desarrollos para acoplar este sistema a la arquitectura TRENCADIS. El componente que se ha definido y desarrollado para dar soporte a VOMS es el Gatekeeper, y será el encargado de autenticar los usuarios y establecer las conexiones seguras, pero extendiendo su funcionalidad. Este componente, una vez realizadas las tareas de autenticación y garantizar el canal seguro en la comunicación, se encarga de dar permisos al usuario Grid autenticado en función de las credenciales que presenta mediante los atributos VOMS (grupos, roles y capacidades) que presenta. El Gatekeeper es capaz de interpretar la información de los certificados emitidos por el VOMS. El Gatekeeper se ha descrito en la definición de la arquitectura TRENCADIS en los apartados 3.1.3.1 y 3.1.3.1.2.

Como se ha comentado en el apartado anterior, cada dominio administrativo posee su propia CA que certifica a sus propios recursos y usuarios. VOMS se encarga de unificar todos estos dominios diferentes e integrarlos en una única OV, organizando los usuarios en diferentes grupos y subgrupos y asignando a cada uno de estos determinados roles dentro de la OV.

3.3.3. Privacidad de datos en dominios administrativos ajenos

Como último apartado referente a la seguridad respecto a la arquitectura TRENCADIS, se encuentra la gestión de la privacidad de los datos en dominios administrativos ajenos, información médica DICOM en entornos Grid, en donde los datos pueden ser guardados en entornos administrativos diferentes a los entornos desde donde se originan. Tal y como se comentó en el apartado 1.6.4 referente a la infraestructura de seguridad Grid, para asegurar un correcto esquema de protección de la privacidad de los datos, se debe asegurar que ni siquiera los usuarios con privilegios de administración local de los recursos donde se tratan los datos, puedan ser capaces de leerlos ni manipularlos si no están autorizados para ello.

Los modelos más utilizados en la actualidad, se basan en esquemas de cifrado de información mediante claves. Concretamente, el modelo que se aplica en ésta tesis [92], consiste en un modelo de cifrado de datos mediante claves, divididas mediante un esquema de Shamir para compartir secretos. Las partes de la clave son distribuidas en diferentes servidores de claves que deben pertenecer a dominios administrativos (hospitales, centros médicos, etc.) diferentes. En los siguientes apartados se describe el modelo de seguridad aplicado a la arquitectura, además de las modificaciones respecto al modelo original, para integrarlo dentro de los objetivos planteados en ésta tesis.

Por tanto, los apartados que se van a presentar a continuación referentes al modelo de seguridad son: en primer lugar se explica el tipo de clave que se utiliza para el cifrado-descifrado de los datos; en segundo lugar se describe la forma en la que el modelo consigue una mejor garantía en la autenticidad y en la confidencialidad de los datos; en tercer lugar se describe la forma en la que se distribuyen las claves de cifrado o descifrado, así como la información que se deberá distribuir junto a éstas, y por último, se explicará el proceso de reconstrucción de claves y la relación de éstas con la definición de las ontologías, haciendo hincapié en la forma en que esta relación mejora la seguridad del proceso de reconstrucción de las claves, pues éstas sólo podrán ser recuperadas por grupos de usuarios Grid que estén autorizados para trabajar con unas determinadas ontologías y a las que esté asociado el dato cifrado a tratar.

3.3.3.1. Modelo de Seguridad

3.3.3.1.1. Cifrado y Descifrado de datos

El modelo que sigue la arquitectura se basa en un cifrado de clave simétrica, en el que tal y como se ha comentado en el apartado 1.6.1.2 referente a la infraestructura de seguridad Grid, los datos son cifrados y descifrados utilizando una misma clave. El propio cliente se encarga de generar la clave (en principio, la longitud de las claves es de 128 bits, aunque su longitud no está restringida en la arquitectura) mediante el algoritmo Avanced Encrypted Standard [117] (AES), y se realizan en éste las operaciones de cifrado o descifrado de datos, evitando así sobrecargas en los servidores donde se encuentran los datos. Para cada operación de cifrado de datos se genera una clave diferente.

3.3.3.1.2. Autenticidad y Confidencialidad de los datos

Tal y como se ha comentado en el apartado 1.6.3.6 referente a la infraestructura de seguridad Grid, el objetivo principal de preservar la privacidad de los datos, es la de mantener los datos plenamente confiables, definiendo la confianza en términos de autenticidad (detectar modificaciones no autorizadas) y de confidencialidad (prevenir la exposición no autorizada de datos).

Para garantizar la autenticidad de los datos cifrados, en el modelo se emplea un Código de Autenticación del Mensaje (MAC), calculado mediante el protocolo HMAC. HMAC emplea un algoritmo de resumen criptográfico, como MD5 o SHA, para calcular un resumen basado en una clave secreta y en los contenidos del objeto cifrado. El HMAC se incluye en la cabecera del objeto cifrado, de manera que al descifrar el dato se comprueba su integridad. El formato de objeto cifrado permite opcionalmente validar la autenticidad del objeto sin realizar el descifrado previo de los datos.

Por otra parte, la confidencialidad de los datos se consigue mediante la distribución de claves entre diferentes dominios administrativos reales. La distribución de una clave de cifrado/descifrado se realiza por el propio cliente, el cual divide la clave en N partes diferentes, mediante el uso de un esquema de Shamir para compartir secretos y en el que se necesitan k partes cualquiera de una clave (con $k < N$) para reconstruirla. Estas partes de la clave son guardadas en diferentes servidores de claves, junto con información adicional que será comentada más adelante, y en la que cada servidor de clave, encargado de guardar una parte de una clave determinada se ubicará en un dominio administrativo real diferente. De esta forma, el acceso a un dominio administrativo sólo permite el acceso a una parte de la clave, que por la naturaleza de los métodos [92] no reduce la seguridad del sistema.

3.3.3.1.3. Distribución de Claves

En este modelo de seguridad, la distribución de claves es el punto más importante. La distribución de claves asegura la protección contra accesos desde un nivel administrativo individual a los datos cifrados, aprovechándose además de esta distribución para garantizar su integridad, mediante el reparto de la MAC por los diferentes servidores de claves.

En lo referente a la confidencialidad, las claves se distribuyen en diferentes dominios administrativos reales para evitar que un administrador, de un dominio concreto, tenga acceso a la información necesaria para poder descifrar datos a los que no debería tener acceso.

Una de las variantes introducidas al modelo presentado en [92], en su integración en la arquitectura TRENCADIS, es precisamente la forma de considerar los diferentes dominios administrativos para el reparto de las claves. Como se ha comentado anteriormente, las partes de las claves son guardadas en diferentes servidores de claves ubicados en dominios administrativos diferentes. En el caso del modelo original, los diferentes dominios administrativos eran determinados por las diferentes OVs, gestionadas de manera independiente por un servidor VOMS. De esta forma, se consideraban dominios administrativos independientes diferentes a las diferentes OVs implicadas en el despliegue. En el caso de la arquitectura TRENCADIS, se ha definido una única OV, en la que los diferentes dominios administrativos que se pretenden utilizar para el reparto de las partes de las claves están integrados dentro de una misma OV. Por tanto, la variante introducida en la arquitectura se basa en que todos los recursos y usuarios de un dominio administrativo posean su propia CA capaz de emitir sus certificados, de manera que las CAs determinen el dominio administrativo, pudiendo de esta manera ser identificados los diferentes servidores de clave en los diferentes dominios administrativos a la hora de repartir las claves, encontrándose estos servicios certificados por diferentes CAs.

En lo referente a la autenticidad de los datos cifrados, al almacenar la MAC de los datos cifrados en todos los servidores de claves, en las que se almacene una parte de la clave que se ha utilizado para el cifrado de datos, es posible detectar si un dato cifrado se modifica, ya que el usuario que ha conseguido la clave y modifica el objeto, debería además modificar la MAC de todos los servidores de claves ubicados en los diferentes dominios administrativos, para que su modificación no fuera advertida. Esta acción requeriría corromper todos los dominios administrativos involucrados.

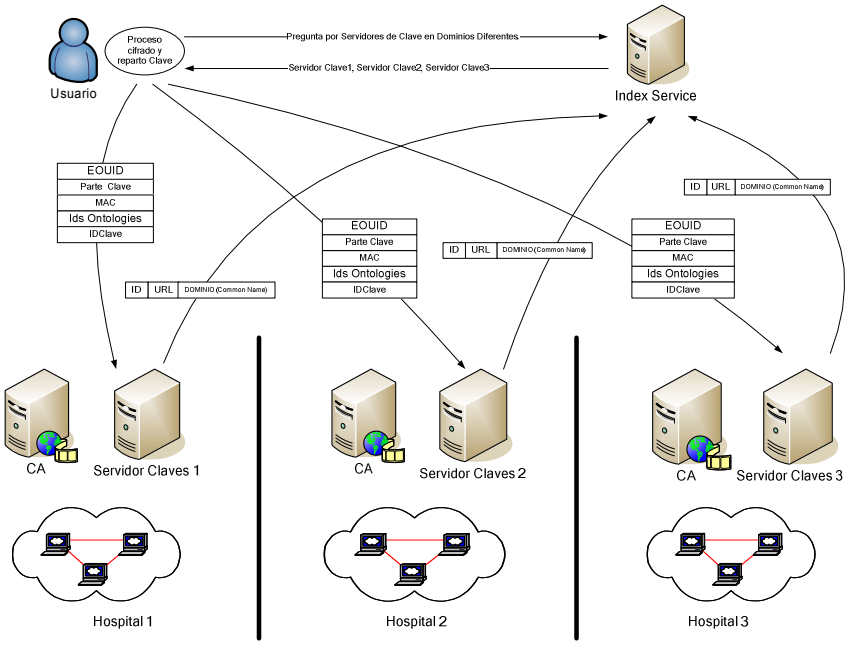

figura 18. Publicación de los dominios administrativos de los diferentes Servidores de Claves en el Sistemas de Información y Monitorización MDS4 y reparto de claves en el momento de cifrado de los datos de un objeto DICOM, en el que primero consulta los dominios administrativos existentes y después realiza el reparto de las claves junto con la información asociada entre los diferentes Servidores Claves de diferentes dominios administrativos.

3.3.3.1.4. Publicación en el Sistema de Monitorización e Información

Otro punto muy importante en el modelo de seguridad, es la información a publicar por parte de los servidores de claves. Los servidores de claves deben publicar la suficiente información en el Sistema de Monitorización e Información de la arquitectura, para que al ser consultado sea posible identificar a los servidores de claves requeridos a la hora de distribuir y recuperar las partes de una clave asociada a un determinado objeto de forma adecuada.

La información a publicar por un servidor de claves es, por una parte, la URL donde se ubica y, por otra, un manejador que identifica al servidor de clave dentro de su dominio administrativo. Además se almacena en el dominio administrativo al que pertenece, obtenido a partir de su "common name" extraído de la CA con la que el servidor ha sido certificado. De esta manera, se simplifica el proceso de reparto y recuperación de las partes de una clave, ya que sólo se necesita interrogar al Sistema de Monitorización e Información por los servidores de claves disponibles, escogiendo aquellos que se encuentran ubicados en sistemas de administración diferentes.

3.3.3.1.5. Información en Servidores de Claves

Otra consideración muy importante en el proceso de distribución de las partes de una clave, además de la selección de los servidores de clave, es la información a guardar en estos servidores junto a la parte de la clave correspondiente. En este modelo se considera que, además de guardar las partes de la clave y su correspondiente identificador, se guarde información adicional que permita al sistema comprobar la autenticidad de los datos cifrados, garantice su ubicuidad y proporcione un control de seguridad de acceso organizado en grupos ontológicos, es decir, a partir de roles definidos por un modelo semántico.

Para ello, se propone almacenar los siguientes elementos:

- En lo referente a la autenticidad de los datos cifrados, se considera relevante almacenar la MAC de los datos cifrados en todos los servidores de clave, además de almacenarla en el propio objeto cifrado. De esta manera, un usuario malintencionado debería corromper todos los dominios administrativos involucrados.
- Respecto la independencia de la ubicación de los datos cifrados respecto a las claves, cuando los datos son cifrados, a estos se les asigna un Encrypted Object Unique Identifier (EOUID). Este EOUID es un identificador global único asignado mediante servicios desplegados en la infraestructura Grid, que generan este EOUID utilizando el generador de UUID de Apache Axis. Este identificador garantiza que no sea necesario modificar la información almacenada en los servidores de claves, si se desea mover los datos cifrados de un almacén a otro, pues el objeto identificado por el EOUID no depende de su ubicación física en el Grid.
- Con respecto a la parte referente a la seguridad mediante un control de acceso a claves organizado en grupos, se ha optado por guardar además los identificadores de las ontologías a las cuales pertenecen los datos cifrados por la clave. El control de seguridad de acceso a claves organizado en grupos ontológicos, viene determinado por los objetivos definidos en ésta tesis, en la que los datos pueden manejarse en diferentes ontologías, por lo tanto, el control de claves por grupos de trabajo que manejan ontologías es una extensión al modelo original.

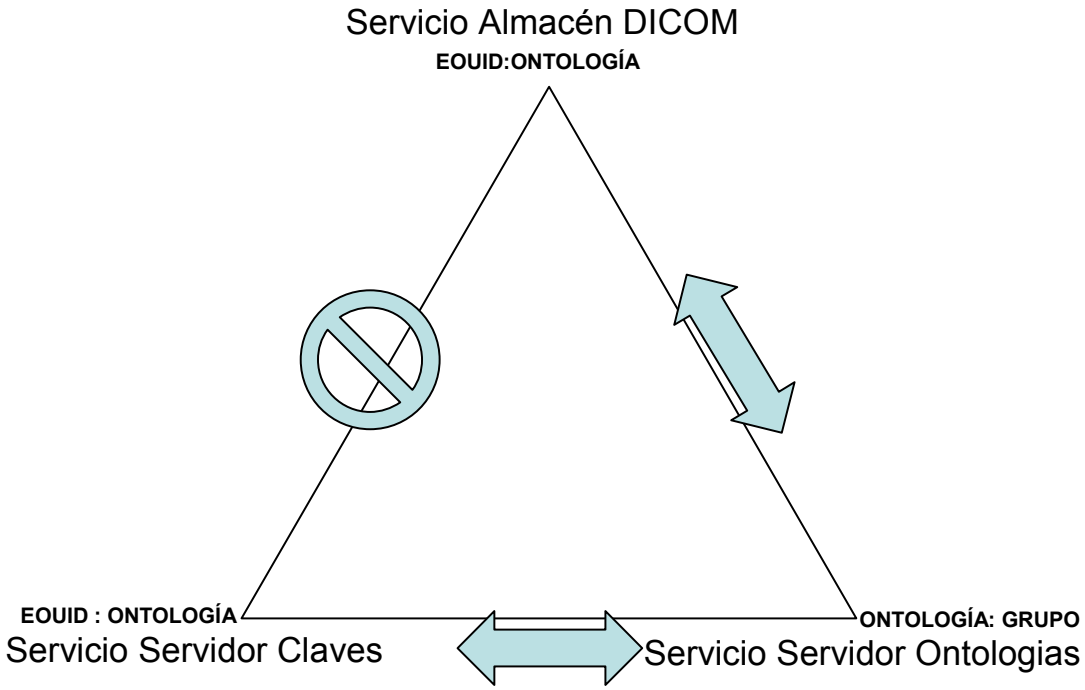

figura 19. Relación entre Ontología – Grupo – EOUID para permitir la recogida de una parte de una Clave de un servidor de Claves.

Para la recuperación de toda la información guardada en un recurso desplegado en la infraestructura, el usuario requerirá de un certificado VOMS, el cual indicará los grupos en los cuales esta incluido el usuario a nivel de OV. Las ontologías pueden ser manejadas por grupos de OV que han sido autorizados, definiéndose esta información en el servidor de ontologías. Los diferentes servidores de claves pueden almacenar claves de datos cifrados por usuarios de diferentes grupos que pueden manejar ontologías diferentes, con lo que el Gatekeeper de estos servidores de claves debe permitir el acceso a los diferentes grupos que manejan estas ontologías. Esto permitiría el acceso de un grupo a las partes de claves almacenadas por otro grupo, aunque las ontologías a las que tienen acceso sean diferentes. Con esto, cualquier administrador de un dominio local con acceso a los datos cifrados en un almacén DICOM podría extraer de las cabeceras las direcciones de los servidores de claves donde se ubican las partes de ésta para el descifrado, y un certificado VOMS que contuviera un grupo con acceso al servidor, accediendo de esta forma a datos de ontologías a las que no tiene acceso. Por ello se debe insertar, por cada parte de clave, además de la MAC y el EOUID, las ontologías a las que pertenece el dato cifrado. De esta manera, el propio Gatekeeper además de comprobar el acceso del grupo al servicio, comprueba que la ontología a la que pertenece la parte de la clave pertenece a la ontología que presentará el propio usuario, y que a su vez pueda ser manejada por un grupo al que pertenezca el usuario.

3.3.3.1.6. Cabecera de Datos de los Datos Cifrados

Una vez los datos han sido cifrados mediante la clave correspondiente, y esta clave ha sido dividida y distribuida por los diferentes servidores, se le añade una cabecera a los datos cifrados, que contendrá el EOUID, además de los dominios administrativos donde se ubican los

diferentes servidores de claves que guardan las partes repartidas con las que se han cifrado los datos.

3.3.3.1.7. Reconstrucción de Claves y Descifrado de datos

Todo usuario del Grid puede recuperar los datos si tiene autorización para ello mediante un certificado VOMS válido. El proceso normal de descifrado de datos empieza en la búsqueda de información contenida en objetos DICOM. Para realizar esta búsqueda se requiere que el certificado VOMS presente las credenciales adecuadas, para poder lanzar las consultas en los diferentes almacenes que maneja la ontología en la que se esté trabajando. Una vez localizado el dato, se extrae de su cabecera los dominios administrativos donde se ubican los servidores de claves que almacenan las partes de la clave para descifrar el dato. Una vez recuperadas las *k* partes necesarias, se reconstruye la clave que descifra el dato, y se comprueba la MAC. Si las verificaciones han sido satisfactorias, el cliente ya tiene acceso a toda la información descifrada.

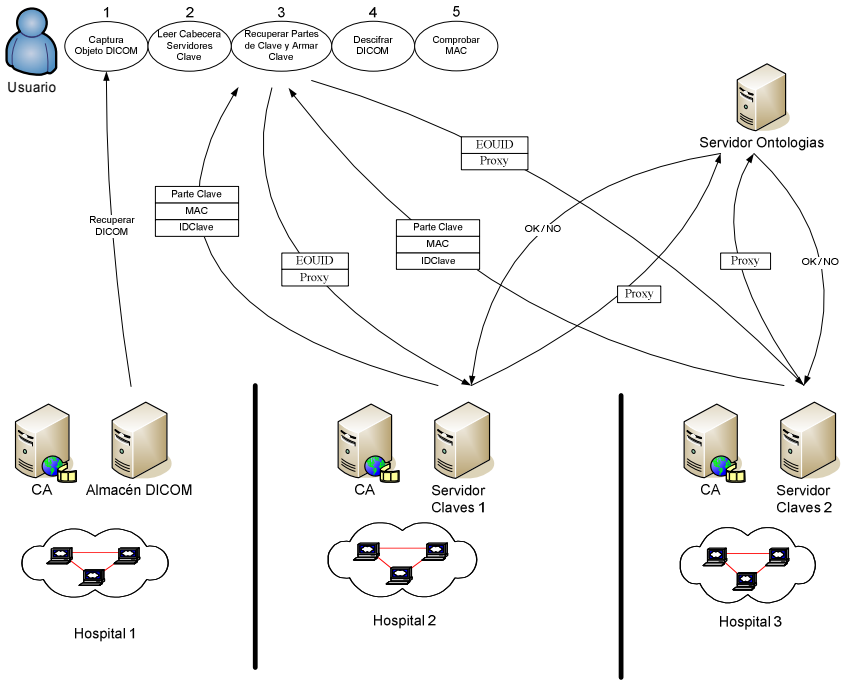

figura 20. Esquema de Reconstrucción de Claves a la hora de descifrar los datos de un objeto DICOM cuyas claves se encuentran repartidas en diferentes dominios administrativos.

3.3.4. Infraestructura del Modelo de Seguridad

Por tanto, para garantizar la autenticación de usuarios y privacidad de datos en las comunicaciones, en la arquitectura se han definido protocolos basados en SSL y certificados digitales x509. Para ello, se requiere del uso de los servicios de una infraestructura PKI, descritos en el apartado 1.6.2.1 referente a la infraestructura de seguridad Grid, la cual se encarga de generar y gestionar todos los certificados x509 de todos los servicios a desplegar en la infraestructura, y todos los usuarios Grid que deseen utilizar estos servicios. Los servicios que proporciona una PKI se encargan de la firma de las peticiones de certificados provenientes de estos usuarios Grid y servicios. Cabe decir que se pueden utilizar diferentes entidades certificadoras, de manera que cada una de ellas pueda estar administrada por un dominio

administrativo diferente, y de forma que los recursos y usuarios de cada centro tengan que ser certificados por la CA del propio dominio administrativo local.

Por otra parte, los elementos y componentes que se requerirán para garantizar la autenticación de usuarios y privacidad de datos en las comunicaciones, dentro de la arquitectura, necesitarán ser implementados tanto en la capa de comunicación (protocolos de comunicación seguros basados en SSL) como en la capa de infraestructura, con la integración de un componente en los servicios que se encargue de garantizar la privacidad de los datos cuando alguien se conecte con el servicio, además de autenticar el usuario para validar la conexión. Para ello se ha tomado como solución el uso de GSI, descrito en el apartado 1.6.2.4 referente a la infraestructura de seguridad Grid. Los protocolos utilizados por GSI son SOAP/HTTPs para la interacción con los servicios y el protocolo GridFTP para el manejo de grandes volúmenes de datos. Estos protocolos son los que se han ubicado en la capa de comunicación de la arquitectura TRENCADIS, tal como se ha descrito en el apartado 3.1.3.2 referente a la estructura general de la arquitectura. Por otra parte, el componente a integrar en los diferentes servicios es el Gatekeeper, encargado de garantizar la privacidad de los datos y autenticación de los usuarios. La función de esta componente ha sido explicada en el apartado 3.1.3, referente a la estructura general de la arquitectura, de forma detallada.

Los componentes a agregar en la infraestructura con respecto a los sistemas de autorización Grid son el servicio VOMS para la gestión de los certificados y los diferentes Gatekeeper de los servicios a desplegar. El servicio VOMS se ubicará en la capa se Servicios Servidor de TRENCADIS y se encargará de la gestión de los certificados de usuarios para el entorno Grid. El VOMS podrá organizar diferentes OV, dar de alta Usuarios, Grupos y Subgrupos de usuarios en una OV y asignar roles a estos grupos.

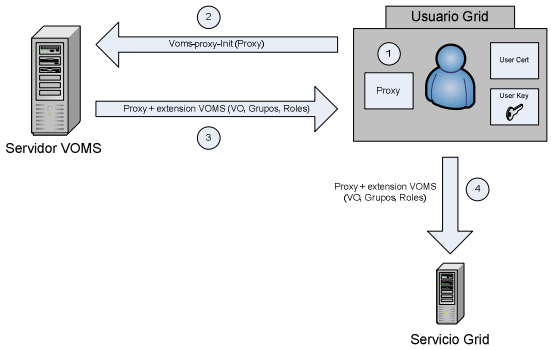

figura 21. Esquema obtención certificado VOMS.

Por otra parte, el Gatekeeper es un desarrollo realizado en ésta tesis, en la que su estructura general queda descrita en los apartados 3.1.3.1 y 3.1.3.1.2 referentes a la estructura general de la arquitectura. La parte más específica para cada servicio se describirá con posterioridad en el capítulo cuarto, referente a las funcionalidades, de una manera más detallada en cada uno de los servicios desarrollados a partir de la funcionalidad ofrecida.

Con respecto al modelo de seguridad para la privacidad de datos en dominios administrativos ajenos, se requiere de una infraestructura compuesta por servicios Grid correspondientes a los servidores de claves, así como un servidor de EOUID. Ambos servidores se despliegan en la subcapa de Servicios Servidor correspondiente a la capa de Infraestructura de TRENCADIS, al

realizar operaciones que no interactúan con ningún dispositivo de manera directa. A continuación se describen dichos servidores con más detalle.

3.3.4.1. Servicio Grid Servidor de Claves

Este servicio se encarga de guardar las partes de las claves e información adicional como el EOUID, la MAC y las ontologías a las cuales pertenecen las claves utilizadas para cifrar los datos. Por otra parte, este servicio proporciona a los clientes las partes de la clave y la información adicional asociada, cuando se requieren para el descifrado de datos.

Los clientes de este servicio son los encargados del cifrado y descifrado de los datos y que, en la arquitectura, serán los componentes de alto nivel de la capa middleware que se encarguen de guardar información en los almacenes DICOM.

3.3.4.1.1. Interacciones

Las interacciones que realiza este servicio Grid están relacionadas con el resto de los servicios implementados en el modelo de seguridad (servicio Grid Generador de EOUID) y con el servicio IDXSRV del sistema de Monitorización e Información MDS4-Extended.

Todas estas interacciones han sido descritas en el propio modelo de seguridad, en el apartado 3.3.3.

3.3.4.1.2. Gatekeeper

Todo cliente, cuando necesita ejecutar alguna operación del servicio de gestión de claves, debe presentar un certificado Proxy VOMS válido, el cual especificará los grupos a nivel de OV a los cuales pertenece el usuario. El Gatekeeper del servicio aceptará o denegará al acceso al recurso según esté o no accesible el recurso para los grupos presentados. Los grupos aceptados o denegados se guardan en una base de datos, implementada con el software de gestión de Bases de Datos Relacionales PostgreSQL. Además de controlar el acceso de los grupos, también puede otorgar permisos de acceso al recurso a usuarios individuales mediante su DN. La estructura de las bases de datos empleadas es la siguiente:

figura 22. Tablas de la Base de Datos del Gatekeeper del Servidor de Claves.

Por otra parte, otra de las tareas de las que se encarga el Gatekeeper, es comprobar que la parte de la clave que se pretende recuperar del servidor de claves pertenece a una ontología a la que el Proxy VOMS presentado por el cliente está autorizado. Esta autorización se interpreta a partir de los grupos del certificado VOMS, cuya asociación con la ontología a la que la clave pertenece se comprueba mediante la consulta directa al servicio Grid de Servidor de Ontologías, el cual posee este tipo de Información, validando o no la operación.

3.3.4.1.3. Estado del Servicio Grid

El estado de este Servicio Grid viene dado por un manejador único, que lo identifica, y un atributo que contiene el último error producido en éste. También define el estado una base de datos relacional e implementada en PostgreSQL. En ella se guardan las partes de las claves y

sus identificadores. Por cada parte de clave se guarda además el EOUID del dato cifrado, la MAC y las ontologías a las que pertenece la clave. El diagrama relacional de esta base de datos es el siguiente:

figura 23. Base de Datos del Estado del Servidor de Claves.

3.3.4.1.4. Procesos lógicos y Esquemas XML de entrada/salida

A continuación se presenta una breve descripción de los procesos más significativos que proporcionan al servicio la funcionalidad correspondiente del Servicio Grid Servidor de Claves, indicando si requieren de esquemas XML para las entradas y salidas correspondientes.

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Arranque	No	No
Descripción		
Proceso que arranca el servicio e inicializa el estado. Registra el servicio en el Sistema de Monitorización e Información y MDS4-Extended. Los datos a publicar son la dirección donde se ubica el servicio, su identificador y el dominio administrativo al cual pertenece.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Recuperar Error	No	RecuperarError.xsd
Descripción		
Proceso que retorna la descripción del último error que se ha producido en el Servicio Grid.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Guardar Parte Clave	InputSaveKey.xsd	ReturnSaveKey.xsd
Descripción		
Guarda una parte de la clave con la información correspondiente en el servidor de claves.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Recuperar Clave	InputRetrieveKey.xsd	ReturnRetrieveKey.xsd
Descripción		
Retorna una parte de la clave con la información adicional correspondiente.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Actualizar Clave	InputUpdateKey.xsd	ReturnUpdateKey.xsd
Descripción		
Actualiza una parte de la clave con la información adicional correspondiente.		

3.3.4.1.5. Interfaces definidas para el Servicio Grid

En este apartado se van a listar los interfaces, mediante los cuales se podrá interactuar con el Servicio Servidor de Claves, relacionando estos interfaces con los procesos lógicos a los cuales van ligados.

Interfaz	Descripción	Proceso Lógico
xmlGetError	Retorna el último error ocurrido en el servicio.	Recuperar Error
xmlSaveSubkey	Guarda una parte de la clave en el servidor de claves.	Guardar Parte Clave
xmlRetrieveSubkey	Recupera una parte de la clave en el servidor de claves.	Recuperar Clave
xmlUpdateSubKey	Actualiza una parte de la Clave	Actualizar Clave

3.3.4.2. Servicio Grid Generador de EOUID

Este servicio se encarga de generar los EOUID utilizados cuando se cifran los datos. Los clientes de este servicio serán todos aquellos procesos que se encarguen de la codificación de los datos. Este servicio se despliega junto con el servicio de servidor de claves y, al igual que este, suscribe la información necesaria en el MDS4-Extended. El objetivo de proporcionar más de un servicio generador de claves por OV, es disminuir los puntos de fallo únicos del modelo. Si sólo se dispusiera de un único generador de EOUID en la OV, el fallo de este servicio ocasionaría la caída del sistema completo al ser imposible producir nuevos objetos cifrados.

3.3.4.2.1. Interacciones

Al igual que ocurría con el servicio anterior, las interacciones que realiza este servicio Grid están relacionadas con servicios implementados en el modelo de seguridad (Servicio Grid Servidor de Claves) y con el servicio IDXSRV del MDS4-Extended.

Todas estas interacciones han sido descritas en el propio modelo de seguridad, en el apartado 3.3.3.

3.3.4.2.2. Gatekeeper

Al igual que ocurría con el servicio de servidor de claves, cuando un cliente necesita ejecutar alguna operación de este servicio, necesita presentar un certificado proxy VOMS válido, en el cual se especificarán los grupos a nivel de OV a los cuales pertenece el usuario. El Gatekeeper

del servicio aceptará o denegará al acceso al recurso según si está o no accesible para los grupos presentados o para el usuario en concreto. En principio el acceso a este servicio será general para todos los miembros de la OV.

3.3.4.2.3. Estado del Servicio Grid

El estado de este Servicio Grid viene dado por un código único que lo identifica, y un atributo que contiene el último error producido en éste.

3.3.4.2.4. Procesos lógicos y Esquemas XML de entrada/salida

A continuación se presenta una breve descripción de los procesos más significativos que proporcionan al servicio la funcionalidad correspondiente del servicio Grid Generador de EOUID, indicando si requiere de los esquemas XML para las entradas y salidas correspondientes.

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Arranque	No	No
Descripción		
Proceso que arranca el servicio e inicializa el estado. Registra el servicio en el Sistema de Monitorización e Información MDS4.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Recuperar Error	No	RecuperarError.xsd
Descripción		
Proceso que retorna la descripción del último error que ha ocurrido en el Servicio Grid.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
RecuperaEOUID	No	ReturnRecuperarEOUID.xsd
Descripción		
Retorna un EOUID válido.		

3.3.4.2.5. Interfaces definidas para el Servicio Grid

En este apartado se van a listar los interfaces, mediante los cuales se podrá interactuar con el Servicio Generador de EOUID, relacionando estos interfaces con los procesos lógicos a los cuales van ligados.

Interfaz	**Descripción**	**Proceso Lógico**
xmlGetError	Retorna el último error ocurrido en el servicio.	Proceso de Recuperado de Error.
xmlGetEOUID	Retorna un EOUID válido.	Proceso de Recuperar EOUID.

3.3.5. Componentes Middleware

Una vez especificados e implementados los elementos de la infraestructura de la arquitectura TRENCADIS, necesarios para abordar los objetivos de seguridad planteados, y definidos los protocolos a utilizar, en este apartado se va a especificar los Componentes Middleware que se encargarán de abstraer al usuario de todos los aspectos de más bajo nivel, ofreciendo un interfaz orientado a objetos más sencillo que se encargue de interactuar directamente con los elementos de la infraestructura Grid. Los componentes middleware se distribuyen en paquetes y concretamente para este punto se ha desarrollado el paquete Gestor de Sesiones.

3.3.5.1. Paquete Gestor de Sesiones

Estos componentes se encargan de gestionar toda la información necesaria para identificar y autenticar a un usuario Grid determinado. Por ello se encargan de interactuar con el servicio VOMS, para recabar toda la información referente al usuario para la OV a la que se quiere conectar.

3.3.5.1.1. Componente C_GRID_Session

Este componente se encarga de crear un objeto mediante el cual se guardan todas las credenciales correspondientes a un usuario Grid. Para ello interactúa directamente con el servicio Grid VOMS para recuperar toda la información referente al usuario (grupos, roles y capacidades).

Estado

El estado del componente vendrá determinado por el certificado VOMS, que recupera del Servicio Grid VOMS al iniciar con este una sesión.

Métodos

A continuación se listan los métodos que proporciona el componente middleware C_GRID_Session. A diferencia de lo que ocurría con los servicios Grid, estos componentes al ser de alto nivel, ofrecen para su interacción métodos con tipos de datos para la entrada o salida y no documentos XML.

Método	**Entrada**	**Salida**
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de iniciar el estado de objeto para la gestión de una sesión Grid.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error ocurrido
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error ocurrido.		

Método	Entrada	Salida
iInitSessionVO	strPathConfigFile → Fichero XML donde se encuentran los certificados a utilizar para interactuar con el VOMS, así como la OV con la que se quiere conectar. strPassword → Password de los certificados para iniciar la sesión.	C_GRID_Session → Componente con toda la información la sesión y las credenciales del usuario para acceder a ésta.
Descripción Proceso Lógico Asociado		
Método que se encarga de iniciar una sesión con un servidor VOMS determinado.		

CAPÍTULO IV.
Funcionalidad

En los capítulos anteriores se han definido las diferentes capas que componen la arquitectura TRENCADIS, se han identificado todos los elementos que pueden integrarse en un despliegue Grid según esta arquitectura (servicios, protocolos, componentes middleware y aplicaciones), se ha especificado la funcionalidad de los sistemas funcionales como el sistema de Monitorización e Información (MDS4-Extended) y se ha descrito el modelo de seguridad a aplicar según los objetivos planteados (autenticación de usuarios y privacidad de datos en las comunicaciones, autorización Grid y modelo de cifrado en dominios administrativos ajenos). En este capítulo se desarrollan los aspectos relativos a los objetivos funcionales, según se ha descrito en el apartado 2.1.3 referente a los objetivos. Los objetivos funcionales son el paso previo al desarrollo de aplicaciones, pues éstas se construyen a partir de la funcionalidad ofrecida por los componentes de alto nivel de la capa Componentes Middleware de la arquitectura, que se nutren a su vez de los servicios que ofrece la capa de Infraestructura.

El principal objetivo de este capítulo es describir los componentes de la arquitectura TRENCADIS que implementan estas funcionalidades, y cómo éstos se integran dentro de las diferentes capas descritas. Estos componentes son los servicios Grid y componentes middleware correspondientes.

La primera funcionalidad que se explica en este capítulo es el manejo de ontologías. En este aspecto, primero se describirá la información DICOM que se utilizará para su definición y la estructura del lenguaje creado para describir las ontologías mediante documentos XML. También se describirán y desarrollarán los componentes necesarios dentro de la arquitectura para integrar el manejo de ontologías. Los componentes de la infraestructura que interactúan directamente con las ontologías son el servicio Grid Servidor de Ontologías, donde se guardan las especificaciones de las ontologías y el servicio Grid Storage Broker perteneciente al MDS4-Extended y encargado de optimizar las búsquedas mediante la indexación de la información de las fuentes por los campos especificados en las ontologías. Además, se presentan los componentes de alto nivel que forman parte del paquete de manejo de ontologías.

La segunda funcionalidad descrita se refiere a la compartición y manejo de información DICOM, principalmente imágenes médicas y documentos estructurados. En este apartado se describen y desarrollan los dos componentes que integran estas funcionalidades, que son el servicio Grid Storage DICOM, encargado de almacenar y gestionar los objetos DICOM y su información asociada, y el servicio Grid Servidor de Códigos, encargado de albergar las codificaciones que se requieren para la creación de documentos estructurados DICOM-SR. Los componentes de alto nivel se estructuran en este apartado en cuatro paquetes diferentes: el paquete de gestión de Códigos para gestionar las codificaciones; el paquete de DICOM-I para

tratar con imágenes médicas DICOM; el paquete DICOM-SR para tratar con documentos estructurados; y el paquete de compartición de objetos DICOM para la creación de almacenes virtuales.

La última funcionalidad a abordar en este capítulo se refiere a la transferencia de datos DICOM desde los clientes a los almacenes DICOM (estos repositorios son implementados mediante servicios Grid Storage DICOM) y viceversa, de una manera segura, describiendo e implementando los componentes correspondientes de la arquitectura TRENCADIS, lo que implica en este caso, todos aquellos relacionados con el modelo de seguridad, manejo de ontologías y compartición de imágenes y que han sido presentados con anterioridad a este apartado. Por otra parte, los paquetes que albergan los componentes de alto nivel que soportan esta funcionalidad serán cuatro, el paquete de descarga de imágenes DICOM, el paquete de descarga de documentos estructurados DICOM-SR, el paquete de transferencia de imágenes médicas DICOM y finalmente el paquete de transferencia de documentos estructurados DICOM-SR.

En resumen, en todos estos apartados se describirán todos los componentes implicados en la arquitectura, tanto a nivel de infraestructura (capa Core Middleware y capa Servicios Servidor) como a nivel de componentes de alto nivel (capa de Componentes Middeware), situándolos en la capa de la arquitectura TRENCADIS que corresponden, y detallando su estructura en función de las especificaciones descritas en la arquitectura.

4.1. Manejo de Ontologías sobre Información DICOM

Tal y como se ha planteado en el apartado 2.1.3, referente a los objetivos, una de las funcionalidades que se quiere dar a los usuarios de los componentes que ofrece la arquitectura TRENCADIS, es la de manejar, como un único almacén virtual, información DICOM guardada en diferentes repositorios o almacenes ubicados en diferentes entidades reales (hospitales, centros médicos, etc…), pero con diferentes vistas en función del área de trabajo a la que pertenece cada usuario. Esto implica la necesidad de definir componentes que se encarguen del manejo de ontologías, y gestionar la información que un usuario puede o no acceder en función de las ontologías a las cuales tienen acceso. Necesariamente, los componentes implicados en esta funcionalidad interactuarán directamente con otros componentes que son requeridos en otras funcionalidades y que también serán descritas en ésta tesis. Por ejemplo, la funcionalidad de manejo de ontologías sobre información DICOM, está fuertemente relacionada con la funcionalidad de compartición de objetos DICOM mediante ontologías, con lo cual habrá componentes que se verán implicados en ambas. Por ello, cuando se plantee el uso de un componente compartido por varias funcionalidades, sólo se describirá la parte implicada en la funcionalidad que se esté describiendo.

En lo relativo al manejo de ontologías, uno de los puntos importantes es definir las relaciones entre ontología y área de trabajo. Las diferentes áreas de trabajo se han implementado mediante la creación de grupos dentro de una OV, donde se integran las diferentes entidades que comparten la información DICOM. Tal y como se ha definido en el apartado 3.3.2 referente al modelo de seguridad, la gestión de grupos, roles y capacidades de la OV en la arquitectura TRENCADIS, se realiza mediante un servidor VOMS. En este servidor se dan de alta los grupos asociados a las áreas de trabajo. Todo usuario del despliegue Grid perteneciente a una OV, pertenecerá a uno o varios grupos ya definidos. Además, todos los grupos se relacionan con una o varias ontologías. Esta relación no se almacena en el servidor VOMS, sino en otro componente que será descrito más adelante y al que llamaremos Servicio Grid Servidor de Ontologías. Estas ontologías determinan las vistas o partes de la información DICOM accesibles por el usuario.

Por ello, todo usuario podrá generar diferentes almacenes virtuales en función del grupo al que pertenezca dentro de la OV, ya que podrá tener tantos almacenes virtuales con información diferente como ontologías accesibles por el grupo al que pertenece. Por ejemplo, en el caso de la infraestructura CVIMO [120], cinco hospitales con diferentes bases de datos DICOM se han unido para crear una OV para compartir la información, donde se han definido tres grupos de trabajo, cada uno de ellos identificado con un área médica, como el cáncer de pulmón, el cáncer de hígado y el cáncer del sistema nervioso central. Estos grupos se dan de alta en el servidor VOMS, por lo que todos los usuarios que se den de alta pertenecerán a uno o varios de los grupos. Estos tres grupos estarán interesados en información DICOM diferente, además de que no deberían acceder a las imágenes propias de los otros grupos de trabajo, aunque se encuentren en los mismos almacenes o repositorios fuente. Para ello se define una ontología por grupo de trabajo. En este ejemplo, la asociación de grupo de trabajo y ontología es una relación uno a uno, aunque también se ha considerado la posibilidad de que un grupo de trabajo pueda tratar con diferentes ontologías, siendo la relación de uno a varios. El usuario Grid, una vez

identificada, de entre las ontologías permitidas por su grupo de trabajo, la ontología con la que se pretende trabajar , debe ser capaz de poder crear un almacén virtual con la información DICOM que necesita, para realizar consultas únicamente sobre la información que se especifica en la ontología seleccionada.

En el apartado 1.1.2.2, referente a los metalenguajes, se propone una definición de ontología, descrita para objetos DICOM mediante el lenguaje XML. Por ello, este apartado primero explica cómo se puede identificar la información dentro de los objetos DICOM para definir las diferentes ontologías a manejar (aunque parte de este apartado viene descrito en los apartados 1.2.3 y 1.2.4, referentes al estándar DICOM). A continuación se describirá, detalladamente, el lenguaje que se ha desarrollado para definir las ontologías mediante documentos XML. Por último, se plantearán los diferentes componentes de la infraestructura que se requieren para el despliegue de esta funcionalidad en la arquitectura TRENCADIS, además de los componentes de alto nivel implementados también dentro de la arquitectura.

4.1.1. Estructura DICOM para la Definición de Ontologías

Tal y como se ha comentado en el apartado 1.1.2.2, referente a lo metalenguajes, las ontologías se definen mediante XML. El principal motivo por el cual se usa XML, y no otros lenguajes más avanzados como OWL, estriba en la necesidad de una especificación de los datos y no de las relaciones semánticas entre éstos. Sólo es necesario expresar las relaciones sintácticas, identificando los campos en los propios objetos DICOM.

Los campos a emplear en estos documentos XML e identificar en los objetos DICOM a la hora de definir las ontologías, son principalmente dos. Por una parte, los campos que se encuentran en los diferentes “Data Element” correspondientes a los IOD de los objetos DICOM involucrados y que han sido descritos en el apartado 1.2.3, referente al estándar DICOM. En estos elementos se encuentra información referente al objeto que alberga, tal como el nombre del paciente, el hospital donde se ha generado el objeto, su modalidad de imagen, etc. Profundizando más en la forma de acceso a este tipo de información, los datos pueden ser localizados a partir de un alias predefinido, que corresponde con un par de códigos definidos en el propio estándar DICOM. Este par de códigos se denominarán “Data Element”, y son específicos para cada concepto según el estándar. Por ejemplo, los códigos 0x0010/0x0010 corresponden al “Data Element” cuyo alias es “Nombre de Paciente”.

Por otra parte, el otro campo que se utiliza para la instanciación y especificación de las ontologías se encuentra en los documentos estructurados DICOM-SR. En este caso, la información se organiza en forma de árbol, en el que cada uno de los nodos (campos) del árbol viene identificado por un “Concept Name”. Este “Concept Name” no es más que una codificación mediante tres valores (Code Scheme, Code Value, Code Meaning) que identifica el campo de manera inequívoca. El “Code Scheme” corresponde a la codificación utilizada, que puede ser propia o acorde a algún estándar, el “Code Value” contiene el valor que tiene el campo dentro de la codificación y el “Code Meaning” contiene una breve descripción del significado que adquiere el “Code Value” para el “Code Scheme” seleccionado.

4.1.2. Lenguaje de definición de Ontologías sobre Objetos DICOM

En este apartado se describe el lenguaje de especificación para la definición de las ontologías. Entendemos que la ontología de un objeto DICOM corresponde con aquella información sobre la cual los usuarios se basan para realizar sus tareas, así como las características y restricciones que se quieran definir. En los objetos DICOM, tal y como se ha comentado en el apartado anterior, se puede albergar diferente tipo de información e información relacionada con el contenido del objeto. La información requerida por los usuarios depende de varios factores, como el grupo de investigación o área médica en la que se trabaje, los diferentes experimentos o estudios que se pretendan realizar y los criterios de filtrado que se requieran en el estudio o experimento.

El propio lenguaje de especificación de ontologías puede ser definido mediante un documento XML, que recoja los campos para caracterizar todos estos tipos de información. Los campos a especificar en el documento XML son los siguientes:

- **IDOntology:** Identificador único de la ontología. Se define mediante la etiqueta XML <IDOntology> </IDOntology>.
- **Description:** Breve descripción de la ontología. Se define mediante la etiqueta XML <Description> </Description>.
- **TypeOntology:** Tipo de ontología que se define. Se consideran todos los tipos de objetos sobre los cuales se pueden definir las ontologías, tales como imágenes DICOM, documentos estructurados, señales, etc. Se define mediante la etiqueta XML <TypeOntology> </TypeOntology >.
- **Restrictive:** Son los campos DICOM que vendrán definidos por una etiqueta <FIELD>…</FIELD> en el caso que se referencie a un "Data Element", o bien por una etiqueta <CONCEPT_NAME>…</CONCEPT_NAME> en el caso que se referencie a un nodo de un árbol y que corresponda a un documento estructurado. Dentro de la propia etiqueta se indican los criterios de restricción de los valores. Por ejemplo:

```
<IDONTOLOGY>1</IDONTOLOGY>
<CERTIFICATE>1</CERTIFICATE>
<TYPEONTOLOGY>1</TYPEONTOLOGY>
<DESCRIPTION>Ontologia para Estudios de Imagenes en General
</DESCRIPTION>
<CREATION>
    <FIELD>
            <CODE1>0x0008</CODE1>
            <CODE2>0x0060</CODE2>
            <ALIAS>MODALITY</ALIAS>
            <DESCRIPTION>Modality Type of Image
            </DESCRIPTION>
            <TYPE>STRING</TYPE>
    </FIELD>
    <FIELD>
            <CODE1>0x0008</CODE1>
            <CODE2>0x0030</CODE2>
            <ALIAS>DATE_STUDY</ALIAS>
            <DESCRIPTION>Study Date</DESCRIPTION>
            <TYPE>DATE</TYPE>
    </FIELD> …
</CREATION> …
```

```
<IDONTOLOGY>4</IDONTOLOGY>
<TYPEONTOLOGY>2</TYPEONTOLOGY>
<DESCRIPTION>Ontologia Cancer Pulmón</DESCRIPTION>
<CREATION>
    <CONCEPT_NAME>
            <CODE_SCHEME>CVIMO_CONCEPT
            </CODE_SCHEME>
            <CODE_VALUE>0098</CODE_VALUE>
            <CODE_MEANING>Tipo Informe
            </CODE_MEANING>
            <TYPE>CODE</TYPE>
            <RANGE_VALUES>
                    <CODE_SCHEME>cvimo_value
                    </CODE_SCHEME>
                    <CODE_VALUE>00006XXX
                    </CODE_VALUE>
            </RANGE_VALUES>
    </CONCEPT_NAME> …
</CREATION> …
```

figura 24. Documentos XML que definen una Ontología. En el ejemplo de la derecha se hace referencia a los campos CREATION de un DICOM-SR para un diagnóstico de cáncer de Pulmón y en el ejemplo de la izquierda los campos CREATION de una imagen DICOM de un estudio genérico.

Estos campos vendrán dentro de la etiqueta <RESTRICTIVE>…</RESTRICTIVE> y determinarán los criterios que deben cumplir los objetos DICOM para ser enmarcados dentro de la ontología definida para un área médica o grupo de investigación.

- **Creation:** Al igual que ocurre con los campos de "Restrictive", esta parte se define mediante las etiquetas XML <FIELD>…</FIELD> y <CONCEPT_NAME>…</CONCEPT_NAME> pero, en este caso, sin las etiquetas XML <CRITERIA> y <VALUE>.

  Estos campos vendrán dentro de la etiqueta <CREATION>…</CREATION> y determinan los campos por los cuales se podrán crear almacenes virtuales compuestos por subconjuntos o vistas de datos, definidos en función de los experimentos o estudios de un grupo de trabajo o área médica.

- **Filter:** Similar a los campos anteriores, se define mediante los TAGS <FIELD>…</FIELD> y <CONCEPT_NAME>…</CONCEPT_NAME>.

  Estos campos vendrán dentro de la etiqueta <FILTER>…</FILTER > y determinan los campos que definen los campos de búsqueda o filtrado, sobre los cuales será posible realizar las búsquedas y filtrados de los almacenes virtuales ya creados.

En el Anexo I se incluyen ejemplos de documentos XML de ontologías, referentes a imagen médica DICOM y a documentos estructurados DICOM-SR.

4.1.3. Infraestructura

En este apartado se exponen aquellos componentes de la arquitectura TRENCADIS que se han desarrollado en la capa infraestructura, para dar soporte a la funcionalidad de gestión de ontologías.

4.1.3.1. Servicio Grid Servidor de Ontologías

Para dar soporte a la funcionalidad de la gestión de ontologías, se requiere de un servidor donde guardar las ontologías definidas mediante el lenguaje expuesto en el apartado anterior. Además, este servidor guardará las correspondencias entre grupos de la OV definidos en el servidor VOMS y las propias ontologías guardadas, que tal y como se ha comentado en apartados anteriores, es una relación de una a varias, pudiendo un grupo de trabajo manejar diversas ontologías. Por ello se plantea la necesidad de crear un servicio Grid, al que llamaremos servicio Grid Servidor de Ontologías, que realice estas funciones. Por otra parte, este servicio debe proporcionar las especificaciones de las ontologías a quienes las necesiten, y dar acceso a los usuarios que quieran acceder a información propia de una ontología, en función de los grupos de trabajo a los que pertenezcan.

Las tareas que realiza éste servicio corresponden a tareas de servidor, lo que implica que éste servicio no interactúa con ningún dispositivo que se ofrezca como recurso en el despliegue de la arquitectura, realizando tareas puramente lógicas, con lo cual será ubicado en la capa de servicios Servidor de la arquitectura TRENCADIS.

4.1.3.1.1. Interacciones

Este servicio interactúa con diferentes componentes que se van a desplegar en la arquitectura TRENCADIS. Por una parte, este servicio interactúa con los servicios encargados de gestionar

los almacenes DICOM (servicio Grid Storage DICOM), por otra parte con el servicio de indexación (servicio Grid Storage Broker) y, finalmente, con el servicio IDXSRV del MD4-Extended para su registro y para la consulta de la ubicación de los diferentes servicios con los que interactúa. El servicio Grid Storage Broker corresponde a un IDXSRV-Extended del MDS4-Extended utilizado por la arquitectura TRENCADIS, y es el encargado de guardar información para la creación de almacenes virtuales. Este servicio será descrito más detalladamente en este mismo apartado. Por el contrario, el servicio Grid Storage DICOM será comentado en la parte correspondiente a la funcionalidad referente a la compartición de objetos DICOM mediante ontologías, incluido en este mismo capítulo. Todas estas interacciones con los servicios Grid se producen de forma bidireccional.

Por otra parte, este servicio también interactúa con el componente de alto nivel C_GRID_OntologyServer, ubicado en la capa de Componentes Middleware de la arquitectura TRENCADIS, aunque en este caso la interacción solo se produce en una dirección, ya que las peticiones se producen siempre desde el componente al servicio Grid. En este apartado, primero se van a describir las interacciones que se producen desde otros componentes de la arquitectura TRENCADIS al servicio Grid Servidor de Ontologías (figura 25), a las que llamaremos interacciones de entrada, posteriormente se describen las interacciones que se producen desde el Servidor de Ontologías al resto de componentes, a las que llamaremos Interacciones de Salida.

Interacciones de Entrada

Las peticiones que realiza el servicio Grid Storage DICOM (punto 1 de la figura 25) al servicio Grid Servidor de Ontologías, se producen cuando éste se activa. En su actuación, el servicio necesita saber la información completa de todas las ontologías existentes, para actualizar y organizar sus datos considerando las nuevas ontologías y modificaciones realizadas sobre éstas, mientras el servicio ha estado inactivo. También se realizan peticiones desde el servicio Grid Storage DICOM, cuando un usuario accede al servicio Grid Storage DICOM para saber los grupos que tienen acceso a una determinada ontología. Estas interacciones se describirán, más detalladamente, cuando se desarrolle más adelante, en este capítulo, el servicio Grid Storage DICOM correspondiente a la sección de compartición de objetos DICOM.

Las peticiones que realiza el servicio Grid Storage Broker (punto 2 de la figura 25) al servicio Grid Servidor de Ontologías, se producen cuando un usuario crea una nueva ontología, o realiza alguna modificación sobre alguna ya existente, ya que el servicio Storage Broker necesita recuperar la nueva información para actualizar sus bases de datos. Estas interacciones se describirán más detalladamente, en este mismo apartado, cuando se desarrolle el servicio Grid Storage Broker.

También existe un componente Middleware encargado de interactuar directamente con el servicio, y será el encargado de mostrar a los usuarios un interfaz para realizar las altas, bajas y modificaciones de las ontologías (punto 3 de la figura 25) que se almacenan en el servicio Grid Servidor de Ontologías. Este tipo de interacción provoca que el servicio Servidor de Ontologías lance peticiones a otros servicios Grid, denominadas interacciones de salida y que serán descritas en el apartado siguiente.

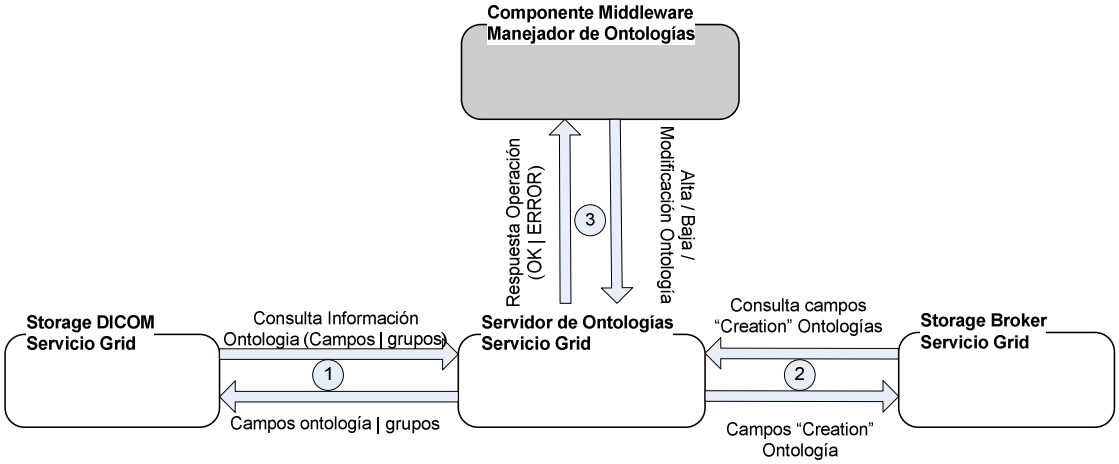

figura 25. Interacción de los servicios de almacén de imágenes DICOM (Storage DICOM) y de indexación (Storage Broker) con el servicio Grid Servidor de Ontologías. Además se muestra la interacción del componente middleware manejador de ontologías con el servicio Grid Servidor de Ontologías (operaciones de alta, baja y modificación).

Interacciones de Salida

La interacción de salida que, en primer lugar, se produce en el servicio Grid Servidor de Ontologías cuando se activa el servicio, provoca que el Servicio Grid se registre en el MDS4-Extended.

Además, cuando un usuario da de alta, baja o modifica una ontología en el servicio Grid Servidor de Ontologías, el servicio notifica a los servicios Grid Storage DICOM activos y al servicio Grid Storage Broker que ha realizado la operación, para que estos actualicen los datos de acuerdo con las ontologías existentes en el momento de la actualización. Por ejemplo, en la figura 26 se muestra el flujo de información que se produce cuando se da de alta una nueva ontología, en el servicio Grid Servidor de Ontologías, a través de un cliente. El flujo de información para una modificación es similar, pues las interacciones de salida son prácticamente las mismas, a excepción de la información que se intercambia. Siguiendo con el ejemplo de alta de una ontología, primero el cliente envía los datos de la nueva ontología al servidor de Ontologías (paso 1 de la figura 26) o, en caso de modificación, los datos de los campos modificados. En caso de una operación de alta, el servidor de Ontologías recoge los datos de la nueva ontología y la inserta en su base de datos, mientras que en el caso de modificación el servidor sólo opera sobre los campos modificados. Realizada la operación correspondiente, el servidor de ontologías pregunta al servicio IDXSRV por la ubicación del servicio Grid Storage Broker (paso 2 de la figura 26). Una vez tiene la ubicación, el servidor envía los campos definidos en la ontología como "Creation" al servicio Grid Storage Broker (paso 4 de la figura 26). A continuación, el servicio Storage Broker recoge estos campos y actualiza su base de datos, para poder realizar indexaciones con la nueva ontología (en el caso de una operación de alta) o bien modificar las existentes (en el caso de modificación). Una vez actualizada la base de datos del servicio Grid Storage Broker, éste envía al servicio Grid Servidor de ontologías un código que indica si la operación ha sido completada de forma satisfactoria (paso 5 de la figura 26). En caso de que la operación haya sido satisfactoria, el servicio Grid Servidor de Ontologías vuelve a preguntar al servicio IDSRV por la ubicación de todos los servicios Grid Storage DICOM activos (paso 6 de la figura 26). Una vez haya

recogido las ubicaciones, el servidor envía la nueva ontología, o las ontologías modificadas, a todos los servicios Grid Storage DICOM activos (paso 8 de la figura 26). Los servicios Storage DICOM recogen la información, y actualizan sus bases de datos con la información de las ontologías recibidas y la información de los objetos DICOM que tienen guardados. A continuación, los servicios Grid Storage DICOM activos preguntan al servicio IDXSRV por la ubicación del Servicio Grid Storage Broker (paso 9 de la figura 26), y envían la información de los objetos DICOM referente a los campos "Creation" de las ontologías, al servicio Grid Storage Broker (paso 11 de la figura 26), quien recoge la información y la guarda en su base de datos. Finalmente, el servicio Grid Storage Broker comunica al Servicio Grid Servidor de Ontologías el resultado de la actualización (paso 12 de la figura 26).

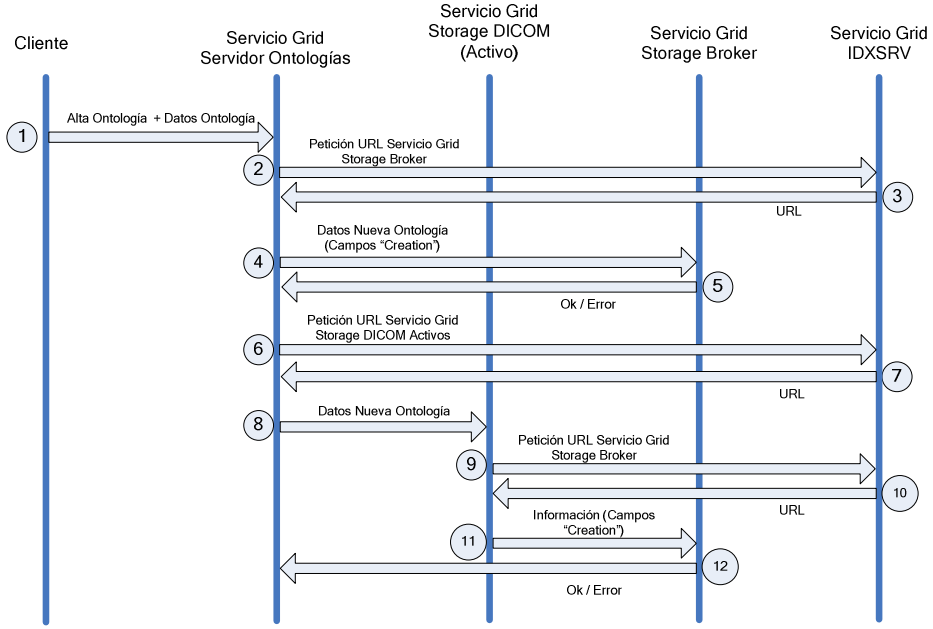

figura 26. Interacción del Servicio Grid Servidor de Ontologías con los servicios de almacén de imágenes DICOM (Storage DICOM) y de indexación (Storage Broker) cuando se realiza el alta de una nueva ontología.

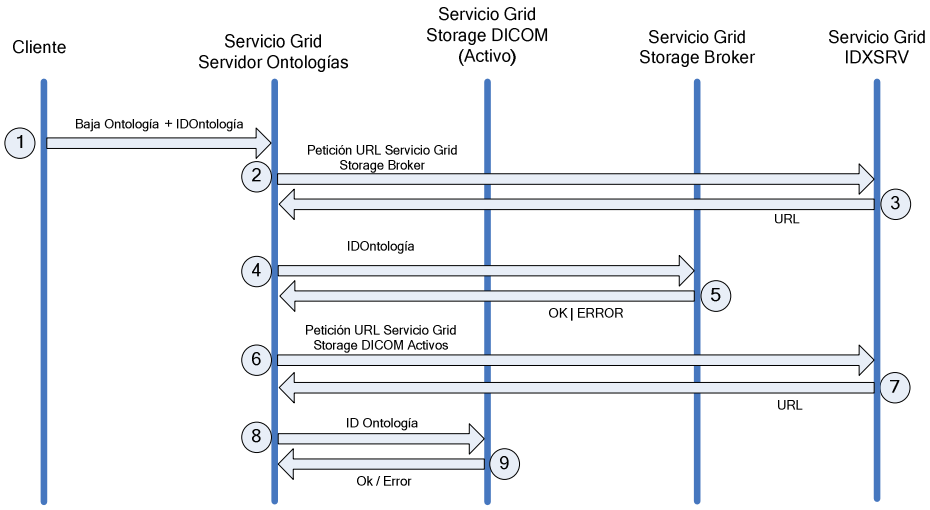

figura 27. Interacción del Servicio Grid Servidor de Ontologías con los servicios de almacén de imágenes DICOM (Storage DICOM) y de indexación (Storage Broker) cuando se realiza la baja de una ontología.

Con respecto a la baja de una ontología, las interacciones son similares al alta o modificación. Cuando un usuario envía una petición de baja al servicio Grid Servidor de ontologías (paso 1 de la figura 28), éste debe comunicar a los servicios Grid Storage DICOM activos y al servicio Grid Storage Broker, que se ha dado de baja una ontología para que éstos actualicen sus bases de datos (paso 2 y 4, figura 27). Estos servicios eliminan de sus bases de datos los datos correspondientes a la ontología eliminada, y comunican al servicio Grid Servidor de Ontologías si la operación ha sido completada de forma satisfactoria (paso 3, figura 27).

4.1.3.1.2. GateKeeper

Tal y como se ha comentado en el apartado 3.1.3.1.2 referente a la estructura general de la arquitectura, todos los servicios de la capa Servicios Servidor tienen una parte análoga a los de la capa Core Middleware, ya que éstos deben cumplir con los requerimientos de seguridad establecidos, tanto para la parte de autenticación de usuarios y privacidad de datos en las comunicaciones como en la parte de sistemas de autorización Grid. Por ello, este componente interpreta las credenciales que se presentan al servicio mediante los certificados VOMS x509.

Por otra parte, se requieren otras restricciones adicionales de seguridad referentes a las ontologías, debido a que al ser un servicio de la capa Servicios Servidor realiza tareas administrativas de gestión de más alto nivel. En este caso concreto, es necesario controlar qué usuarios Grid pueden o no manejar determinadas ontologías en función del grupo de trabajo al que pertenecen. Por ello, cuando un usuario accede a este servicio, se identifican, mediante las credenciales que presenta su certificado VOMS, los grupos a los que pertenece, y se consultan las ontologías que están relacionadas con los grupos presentados para permitir o denegar el acceso.

4.1.3.1.3. Estado del Servicio Grid

Como todo Servicio Grid, el servicio de ontologías tiene asignado un identificador único que lo identifica y un atributo que contiene el último error producido. El estado se almacena en una base de datos relacional que se implementa en Postgres SQL. En ella se guarda la información de todas las ontologías que se van a emplear en la infraestructura Grid desplegada, así como de las relaciones con los grupos de trabajo.

El diagrama relacional de esta base de datos es la siguiente:

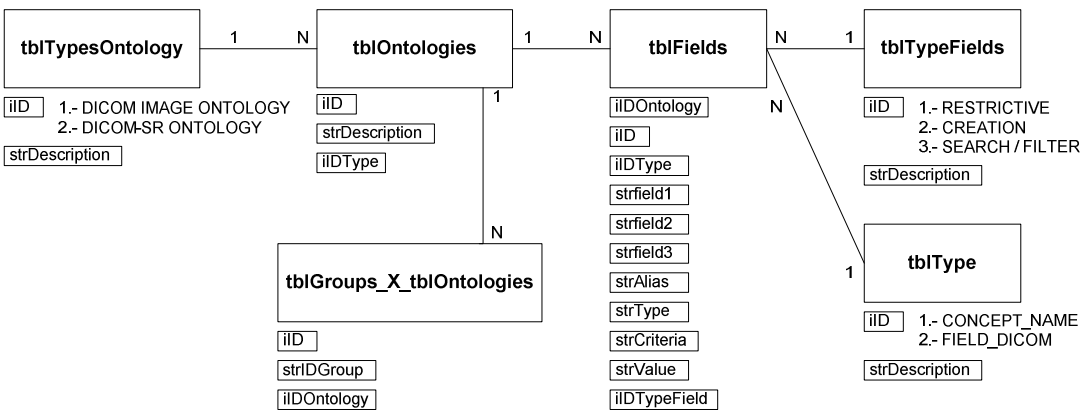

figura 28. Diagrama Relacional de la Base de datos del Ontology Server.

Ésta base de datos, muestra el diagrama relacional en el que se almacena el estado del servicio. La tabla tblOntologies almacena los registros que identifican las ontologías existentes. Estas ontologías pueden ser referentes a imagen médica o documentos estructurados (tblTypesOntology), los campos de cada ontología se almacenan en la tabla "tblFields" y pueden ser de 3 tipos diferentes (tblTypeFields), tal y como se indica en el lenguaje de especificación de ontologías descrito en apartados anteriores. Además, estos campos pueden referenciar a dos tipos de información (tblType). Por un lado, los campos correspondientes a la cabecera de un objeto DICOM (FIELD_DICOM) y por otro los campos que corresponden a un nodo del árbol que representa a un documento estructurado DICOM-SR (CONCEPT_NAME). La relación entre grupos de trabajo y ontologías se guarda en la tabla "tblGroups_X_tblOntologies".

4.1.3.1.4. Procesos lógicos y Esquemas XML de entrada/salida

A continuación se presenta una breve descripción de los procesos más significativos que proporcionan la funcionalidad correspondiente, indicando si requiere de esquemas XML para las entradas y salidas correspondientes.

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Arranque	No	No
Descripción		
Proceso que arranca el servicio e inicializa el estado. Se registra en el MDS4-Extended		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Recuperar Error	No	RecuperarError.xsd
Descripción		
Proceso que retorna la descripción del último error que ha ocurrido en el servicio Grid.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Recuperar Ontología	input_getontology.xsd	return_getontology.xsd
Descripción		
Proceso que retorna la ontología completa en un documento XML.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Crear Ontología	input_createontology.xsd	return_createontology.xsd
Descripción		
El proceso crea una nueva ontología en la base de datos del servicio. Cuando se inserta una ontología se interactúa con todos los Storage DICOM desplegados, para crear de forma distribuida las vistas de la nueva Ontología, al igual que con el Storage Broker.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Eliminar Ontología	input_remove_ontology.xsd	return_remove_ontology.xsd
Descripción		
Proceso que borra una ontología determinada de la base de datos de ontologías. Cuando se borra una ontología, este proceso se comunica con todos los servicios Grid Storage DICOM implicados, y elimina las vistas creadas para la ontología eliminada, al igual ocurre con el Storage Broker.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperar tipos de ontología	No	return_getidstypesontologies.xsd
Descripción		
Proceso que retorna todos los tipos de las ontologías registradas, encapsuladas en un documento XML. Consulta, en la base de datos de ontologías, los tipos de las ontologías.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperación de los identificadores de todas las Ontologías	No	Return_getidsontologies.xsd
Descripción		
Proceso que retorna todos los IDs de las ontologías registradas, encapsuladas en un documento xml. Consulta, en la base de datos de ontologías, los IDs de las ontologías activas.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperación de los datos de las Ontologías de un tipo	input_getOntologiesType.xsd	return_getOntologiesType.xsd
Descripción		
Proceso que consulta a la base de datos de ontologías y retorna toda la información de ontologías, de un determinado tipo, encapsulada en un documento XML.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperación de los grupos de una Ontología	input_getgroupsOntology.xsd	return_getgroupsOntology.xsd
Descripción		
Método que consulta en la base de datos de ontologías y retorna todos los grupos de una ontología determinada.		

4.1.3.1.5. Interfaces definidas para el Servicio Grid

En este apartado se listan los interfaces a través de los que se interactúa con el servicio Grid Servidor de Ontologías, relacionando estos interfaces con los procesos lógicos a los cuales van ligados.

Interfaz	Descripción	Proceso Lógico
xmlGetError	Retorna el último error ocurrido en el servicio.	Recuperar error.
xmlCreateOntology	Crea una nueva ontología en el entorno Grid.	Crear ontología.
xmlRemoveOntology	Elimina una ontología en el entorno Grid.	Eliminar ontología.
xmlGetOntology	Retorna una ontología.	Recuperar una ontología.
xmlGetTypesOntologies	Retorna todos los tipos de ontologías disponibles.	Recuperar los tipos de ontologías.
xmlGetIDsOntologies	Retorna todos los IDs de las ontologías registradas.	Recuperar los identificadores de todas las ontologías.
xmlGetDataOntologiesType	Retorna todas las ontologías de un tipo.	Recuperar los datos de las ontologías de un tipo.
xmlGetGroupsOntology	Retorna las ontologías a las cuales tiene acceso un grupo determinado.	Recuperar grupos Ontología.

4.1.3.2. Servicio Grid Storage Broker (IDXSRV-Extended)

Este componente de la arquitectura ya ha sido introducido en apartados anteriores. El servicio está estrechamente relacionado con el servicio Grid Servidor de Ontologías, pues cualquier cambio que se produce en las ontologías acaba repercutiendo sobre éste. Por otra parte, también tiene una fuerte relación con los servicios Grid Storage DICOM que se presentarán en el

siguiente apartado, pues cualquier alteración de los datos también puede afectar a este componente.

El principal objetivo de este servicio es optimizar la creación de almacenes virtuales, mediante la indexación de los campos "creation" definidos en las ontologías. Este servicio Grid se encarga de indexar, para cada ontología, los campos "Creation" mediante los datos guardados en los diferentes almacenes fuente (servicios Grid Storage DICOM). De esta manera, las búsquedas dentro de las diferentes fuentes, a la hora de crear un almacén virtual, se realizan en paralelo y de manera eficiente. De esta forma no se requiere realizar una consulta broadcast a todos los almacenes fuentes, evitando así consultas innecesarias en almacenes que no contienen la información que se requiere, pues este servicio Grid indicará mediante sus índices dónde está la información que se busca.

Dado que este componente maneja información del sistema, se integrará dentro del sistema de Monitorización e Información de TRENCADIS, MDS4-Extended, y se definirá como un IDXSRV-Extended. A este servicio Grid lo llamaremos servicio Grid Storage Broker.

Por otra parte, al igual que ocurría con el servidor de Ontologías, este servicio también realiza tareas de Servidor sin interactuar directamente con ningún dispositivo, siendo un recurso puramente lógico y por tanto será ubicado en la Capa de Servicios Servidor.

4.1.3.2.1. Interacciones

Este servicio Grid se comunica con otros servicios Grid mediante interacciones de entrada, a excepción del registro en el MDS4-Extended. Los servicios Grid con los que interactúa son el servicio Grid Servidor de Ontologías, cuando éste realiza alguna operación de insertado, modificado o baja de ontología y con el servicio Grid Storage DICOM cuando en ella se producen alteraciones en los datos que afecten a los campos definidos en las ontogías como "Creation".

Por otra parte, también existe un componente de alto nivel de la capa Componentes Middleware de la arquitectura TRENCADIS que interactúa con este servicio, y se encarga de la creación de almacenes virtuales.

Interacciones de Entrada

Las peticiones que realizan los diferentes Servicios Grid Storage DICOM (punto 1 de la figura 29) se produce cuando se introducen, eliminan o modifican datos sobre los objetos DICOM que albergan. Cuando esto ocurre, se revisan las ontologías que hay definidas y se determina de qué manera quedan afectados los campos definidos en el apartado de "Creation". Tanto si se incluye un nuevo campo, como si se elimina o se modifica un campo existente, se debe actualizar el servidor de índices. De esta manera el servicio Grid Storage Broker dispone en todo momento de la información actualizada sobre las distintas ontologías de los datos que almacenan los servicios Storage DICOM.

Las peticiones que realiza el componente de alto nivel para la creación de almacenes virtuales (punto 2 de figura 29) se produce cuando un usuario crea un almacén virtual a partir de este componente. En primer lugar se realiza una consulta al servicio Grid Storage Broker para obtener la ubicación de los servicios Grid Storage DICOM que contienen la información que corresponde con la ontología que se consulta.

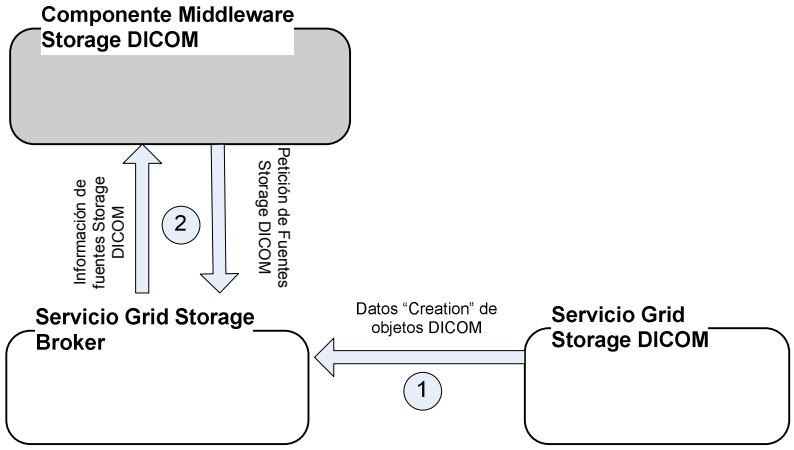

figura 29. Interacción del servicio grid Storage Broker con los servicios de almacén de imágenes DICOM (Storage DICOM) cuando se produce alguna alteración en los datos definidos en las ontologías como de "Creation". Además se muestra la interacción con el componente Middleware que interactúa con este servicio para la creación de Almacenes virtuales.

Interacciones de Salida

La única interacción de salida de este servicio Grid se produce cuando se activa el servicio, ya que éste se registra en el servicio IDXSRV del MDS4-Extended, como cualquier otro servicio Grid.

4.1.3.2.2. GateKeeper

De forma análoga a los casos anteriores, este componente se encarga de cumplir con los requerimientos de seguridad especificados, tanto para la parte de autenticación de usuarios y privacidad de datos en las comunicaciones como en la parte de sistemas de autorización Grid. Por ello este componente analiza los atributos de las credenciales que se presentan al recurso mediante los certificados VOMS x509, permitiendo o denegando el acceso al recurso.

4.1.3.2.3. Estado del Servicio Grid

Todos los servicios Grid tienen asignado un código único que los identifica y un atributo que contiene el último error producido. El estado está además almacenado en el servicio Grid Storage Broker en una base de datos relacional implementada en Postgres SQL. En ella se guarda la información por ontología de los campos de todos los objetos DICOM que hay en los servicios Grid Storage DICOM, que han sido definidos en el apartado "Creation" de la ontología. La información guardada en la base de datos varía según la ontología, pues cada una define los campos por los cuales se crearán los índices y se guardarán los valores correspondientes.

El diagrama relacional de esta base de datos es la siguiente:

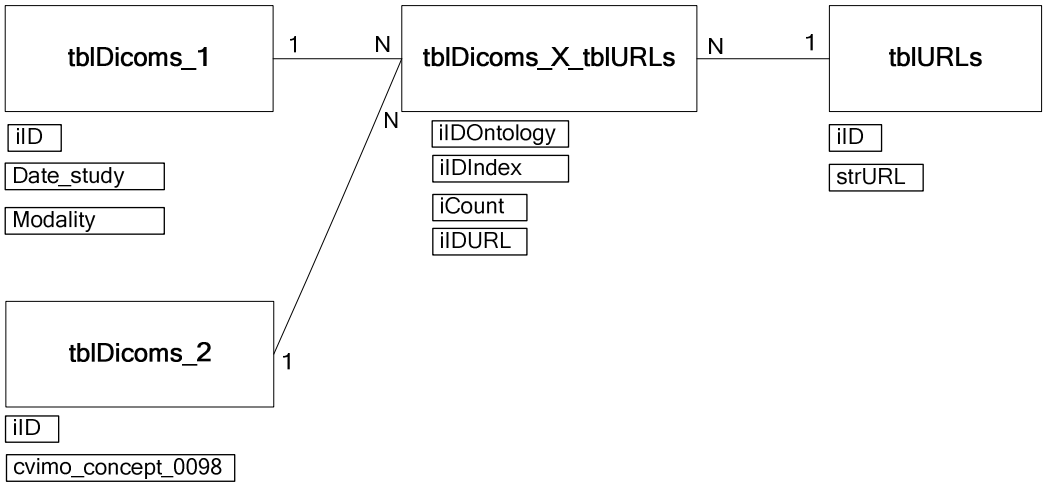

figura 30. Diagrama Relacional de la Base de datos del servicio Grid Storage Broker.

La base de datos que se muestra en la figura 30, muestra el diagrama relacional en la que se estructura y almacena el estado del servicio. Las tablas tblDicoms_1, tblDicoms_2 contienen los valores de los campos definidos en las ontologías 1 y 2 como "Creation" y que corresponden a los valores que guardan los servicios Grid Storage DICOM. La ubicación de estos servicios Grid se almacena en la tabla tblURLs y la relación entre los datos y las ubicaciones de los servicios Grid Storage DICOM en la tabla tblDicoms_X_tblURLs.

4.1.3.2.4. Procesos lógicos y Esquemas XML de entrada/salida

A continuación se presenta una breve descripción de los procesos más significativos que proporcionan al servicio de la funcionalidad correspondiente del servicio Grid Storage Broker, indicando si requiere de los esquemas XML para las entradas y salidas correspondientes.

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Arranque	No	No
Descripción		
Proceso que arranca el servicio e inicializa el estado. Se registra en el MDS4-Extended.		

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Recuperar error	No	RecuperarError.xsd
Descripción		
Proceso que retorna la descripción del último error que se ha producido en el servicio Grid.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperar almacenes DICOM	input_getstoragedicom.xsd	return_getstoragedicom.xsd
Descripción		
Retorna toda la información referente a los servicios Grid Storage DICOM que cumplan una serie de criterios de búsqueda. Consulta, en la base de datos del Storage Broker, las URLs de los servicios Grid Storage DICOM que están registrados y cumplen los criterios de búsqueda establecidos en el documento XML de entrada.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperar Todos Los Almacenes DICOM	No	return_getallstoragedicom.xsd
Descripción		
Retorna toda la información referente a todos los servicios Grid Storage DICOM registrados. Consulta, en la base de datos del Storage Broker, las URLs de los servicios Grid Storage DICOM que están registrados.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Actualizar datos	input_updatestoragebroker.xsd	return_updatestoragebroker.xsd
Descripción		
Proceso que actualiza la base de datos del Storage Broker con la información del Storage DICOM que viene codificada en la entrada por un documento XML.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Agregar ontología	input_addontology.xsd	return_addontology.xsd
Descripción		
Proceso que agrega una ontología del servicio Storage Broker.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Eliminar ontología	input_removeontology.xsd	return_removeontology.xsd
Descripción		
Proceso que elimina una ontología del servicio Storage Broker.		

4.1.3.2.5. Interfaces definidas para el Servicio Grid

En este apartado se van a listar los interfaces mediante los cuales se podrá interactuar con el servicio Grid Storage Broker, relacionando estos interfaces con los procesos lógicos a los cuales van ligados.

Interfaz	Descripción	Proceso Lógico
xmlGetError	Retorna el último error producido en el servicio.	Recuperar error.
xmlGetStorageDICOMs	Retorna toda la información referente a los Servicios Grid Storage DICOM que cumplan una serie de criterios de búsqueda.	Recuperar almacenes DICOM.
xmlGetAllStorageDICOMs	Retorna toda la información referente a todos los Servicios Grid Storage DICOM.	Recuperar todos los almacenes DICOM.
xmlUpdateStorageBroker	Proceso que actualiza la base de datos del Storage Broker.	Actualizar datos.
xmlAddOntology	Agrega una ontología en el StorageBroker Service.	Agregar ontología.
xmlRemoveOntology	Elimina una ontología en el StorageBroker Service.	Eliminar Ontología.

4.1.4. Componentes Middleware

Con respecto a la funcionalidad del manejo de ontologías, se ha desarrollado un único paquete encargado de ofrecer a los usuarios un interfaz de más alto nivel para el manejo de ontologías, e interactuar con los componentes de la infraestructura de una manera transparente.

4.1.4.1. Paquete Ontologías

Este paquete se encarga de gestionar las ontologías (altas, bajas y modificaciones) que se quieran introducir en el entono Grid desplegado mediante objetos de alto nivel. Cuando mediante cualquier objeto de este paquete se realiza una operación con una ontología, el paquete se encarga de disparar las interacciones entre aquellos servicios Grid implicados, que principalmente son el servicio Grid Servidor de Ontologías y el servicio Grid Storage Broker.

4.1.4.1.1. Diagrama de Objetos

Se ha desarrollado una clase C_Service_Server encargada de albergar toda la funcionalidad genérica de cualquier servicio Grid que se ubique en la capa de servicios Servidor de la arquitectura TRENCADIS. La clase C_GRID_OntologyServer hereda directamente de la clase C_Service_Server, la cual tiene toda la funcionalidad génerica de los servicios servidor implementada. La clase C_GRID_Ontology se utiliza para manejar ontologías.

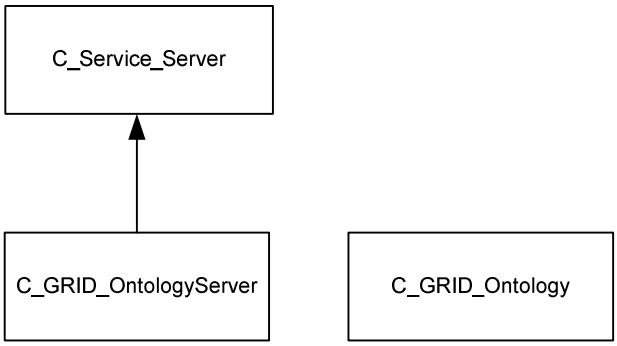

figura 31. Diagrama de Clases del Paquete de Ontologías.

4.1.4.1.2. Componente C_GRID_OntologyServer

Este componente se encarga de crear instancias de objetos, capaces de gestionar el servicio Grid Servidor de Ontologías, para la creación, eliminación o consulta de datos, respecto a las ontologías que se encuentran guardadas en el servidor de ontologías desplegado en la infraestructura Grid.

Estado

El estado de esta componente vendrá determinado por la información sobre las ontologías que almacena el servicio Grid Servidor de Ontologías con el que interactúa, definido durante la creación de una instancia de este componente.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_OntologyServer. A diferencia de lo que ocurría con los servicios Grid, al ser estos componentes de alto nivel, ofrecen para su interacción métodos con tipos de datos para la entrada o salida y no documentos XML. Estos tipos de datos pueden ser básicos (como cadenas o enteros) o bien tipos complejos representados por otros componentes definidos en el propio paquete, como por ejemplo un C_GRID_Ontology.

Método	**Entrada**	**Salida**
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de iniciar el estado del objeto para la gestión del servicio Grid servidor de Ontologías.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
iInsertOntology	C_GRID_Ontology→Objeto Ontología a insertar.	Int →Código de error. Un valor distinto de 0 indica la presencia de un error.
Descripción Proceso Lógico Asociado		
Método que inserta la ontología en el servidor de ontologías, de manera que todos los servicios implicados se verán afectados.		

Método	Entrada	Salida
iRemoveOntology	iIDOntology→Identificador de la Ontología a eliminar del Servidor.	Int→Código de error. Un valor distinto de 0 indica la presencia de un error.
Descripción Proceso Lógico Asociado		
Método que elimina la ontología que corresponde al código de ontología especificado en la entrada, de manera que todos los servicios implicados se verán afectados.		

Método	Entrada	Salida
oGetOntology	iIDOntology →Identificador de la ontología.	C_GRID_Ontology → Objeto ontología.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la ontología guardada en el servidor, que corresponde a el identificador indicado en el parámetro de entrada.		

Método	Entrada	Salida
vGetTypesOntologies	No	Vector→ Vector con los identificadores y descripciones de todos los tipos de ontologías especificados. Un valor nulo indica que se ha producido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna información de todos los tipos de ontologías registradas en el servidor de ontologías.		

Método	Entrada	Salida
vGetIDsOntologies	No	Vector→ Vector con los identificadores de todas las ontologías especificadas. Si es null indica que se ha producido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna información de todos los IDs de las ontologías registradas en el servidor de ontologías.		

Método	Entrada	Salida
vGetDataOntologiesType	iIDTypeOntology→ Identificador del tipo de vector.	Vector→Vector de objetos de C_GRID_Ontology. Si es null indica que ha ocurrido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna información referente a todos los datos referentes a las ontologías de un determinado tipo definido en la entrada.		

4.1.4.1.3. Componente C_GRID_Ontology

Este componente crea instancias de objetos que se encargan de ofrecer un interfaz para la definición de ontologías y poder manejarlas a través de los métodos que publica sin necesidad de generar los documentos XML de una manera directa.

Estado

El estado de este componente vendrá determinado por los diferentes campos que definan la ontología a la que representan, es decir, los campos etiquetados como "Restrictive" con sus condiciones, los campos "Creation" y los campos "Filter".

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_Ontology.

Método	Entrada	Salida
Constructor	C_GRID_Session→Credencial de la session.	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar los campos de las ontologías a definir.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
vGetFieldsRestricted	No	Vector→Vector compuesto por 4 vectores. [Nombres Campos] [Criterios] [Valores] [Operadores]. Si es null indica que se ha producido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna un vector que está compuesto por cuatro vectores, uno con los nombres de campos restrictivos, otro con los criterios, otro con los valores y finalmente con los operadores.		

Método	Entrada	Salida
vGetFieldsCreation	No	Vector→Vector con el nombre de los campos de creación de la ontología. Si es null indica que se ha producido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna un vector con el nombre de los campos de creación de la ontología.		

Método	Entrada	Salida
vGetFieldsFilter	NO	Vector→Vector con el nombre de los campos de filtro de la ontología. Si es null indica que se ha producido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna un vector con el nombre de los campos de filtrado de la ontología.		

Método	Entrada	Salida
iAddFieldRestrictive	String→Nombre del campo.	int→Código de error. Un valor distinto de 0 indica la presencia de un error.
	String→Criterio.	
	String→Valor del campo.	
	String→Operador del campo.	
Descripción Proceso Lógico Asociado		
Método que agrega un campo restrictivo a la ontología.		

Método	Entrada	Salida
iAddFieldCreation	String→Nombre del campo de creación.	int→Código de error. Un valor distinto de 0 indica la presencia de un error.
Descripción Proceso Lógico Asociado		
Método que agrega un campo de creación de la ontología.		

Método	Entrada	Salida
iAddFieldFilter	String→Nombre del campo de filtrado.	int→Código de error. Un valor distinto de 0 indica la presencia de un error.
Descripción Proceso Lógico Asociado		
Método que agrega un campo de filtrado a la ontología.		

4.2. Compartición de Objetos DICOM mediante Ontologías

En este apartado se desarrolla todos los componentes relacionados con la compartición y manejo de objetos DICOM mediante ontologías, componentes que corresponden al objetivo planteado en el apartado 2.1.3, referente a los objetivos, a excepción de aquellos componentes que se encargan del manejo de ontologías, que han sido descritos en el apartado anterior. Concretamente, en este apartado se plantea, como objetivo principal, un modelo mediante el cual los usuarios Grid pueden crear almacenes virtuales a partir de diversas fuentes de objetos DICOM, de manera que estos almacenes virtuales sean capaces de proporcionar a los usuarios diferentes vistas de la información DICOM, en función del área de trabajo en la que se integren los usuarios. De esta forma, el usuario puede generar almacenes virtuales en función de la ontología que seleccione, siempre y cuando ésta sea accesible al usuario. La relación de las áreas de trabajo de un usuario Grid con las ontologías, y la manera de especificar y guardar las ontologías, han sido descritas en el apartado anterior. En este contexto, es importante destacar la fuerte relación entre los componentes de la arquitectura requeridos para la compartición de objetos DICOM mediante ontologías (que serán descritos en este apartado) y las componentes requeridas para manejo de ontologías (que ya han sido descritas).

En esta sección se describe, en primer lugar, la infraestructura necesaria para compartir objetos DICOM mediante almacenes virtuales. En esta infraestructura se definen dos servicios Grid, que son el servicio Grid Storage DICOM y el servicio Grid Code Server. El servicio Grid Storage DICOM se encarga de interactuar con los repositorios DICOM integrados en los sistemas hospitalarios, ofreciendo diferentes vistas de la información en función de las ontologías existentes. Por otra parte, el servicio Grid Servidor de Códigos cobra importancia cuando lo que se quiere realizar es la compartición de documentos estructurados DICOM-SR, al albergar éste todas las codificaciones implícitas y explícitas en los datos DICOM-SR. A continuación, se expondrán las componentes de alto nivel para el manejo de imágenes médicas mediante el paquete DICOM-I, y para el manejo de documentos estructurados mediante el paquete DICOM-SR. Por último se expondrá el paquete de manejo de códigos, así como el paquete de creación de almacenes virtuales de objetos DICOM.

4.2.1. Infraestructura

En este apartado se van a exponer aquellos componentes de la arquitectura TRENCADIS que se han desarrollado en la capa infraestructura, para dar soporte a la funcionalidad de la compartición de objetos DICOM mediante ontologías.

4.2.1.1. Servicio Grid Storage DICOM

Para compartir los objetos DICOM, se requiere de un servicio Grid que se encargue de interactuar directamentente con los dispositivos que manejan los repositorios DICOM (Bases de datos SQL, discos duros, Sistemas PACS, etc.). El que el servicio interactúe con los repositorios, no significa que sea el propio servicio el que alberge físicamente a los objetos DICOM, sino que ofrece un interfaz de manera que la información almacenada en estos repositorios DICOM pueda ser consultada y manejada en función de la ontología que se emplee, e independientemente del dispositivo o sistema de información que maneja el repositorio de objetos DICOM. La información que se almacena en el servicio Grid esta separada por

ontologías, de manera que se guardan todos los valores de los objetos DICOM del repositorio que correspondan con los campos "Creation" y "Filter" de las ontologías existentes, relacionando estos valores con la ubicación física de los Objetos DICOM en los repositorios. Todos estos valores se utilizarán para indexar las búsquedas de los objetos a través del servicio Grid Storage DICOM, de manera que éstos puedan ser consultados sólo por los campos "Creation" y "Filter" de las ontologías, o bien prepararlos para su descarga y transferencia utilizando el componente de interacción, el cual se encarga de interactuar con los diferentes dispositivos e implementa sus particularidades, por lo que se requiere uno por cada tipo de repositorio que se integre en el despliegue.

Este servicio encapsula el almacén como un recurso lógico que ofrece una funcionalidad determinada, en la que podemos destacar funciones básicas tales como búsquedas o descargas. En el desarrollo actual, el servicio soporta una implementación de un repositorio de objetos DICOM mediante un directorio de disco o una base de datos relacional SQL, por lo que se han implementado dos componentes de interacción distintos, uno por cada dispositivo. El componente de interacción hace al servicio independiente del dispositivo con el que interactúa, con lo que el estado que define el servicio, los procesos lógicos y el interfaz WSRF no varían para ninguna de las dos versiones implementadas

Al interactuar con un dispositivo o sistema, este servicio se ubicará en la capa Core Middleware de la arquitectura TRENCADIS, tal y como se ha especificado en el apartado 3.1.1.1, referente a la estructura general de la arquitectura TRENCADIS.

4.2.1.1.1. Interacciones

Este servicio Grid, debido a su ubicación en la capa Core Middleware de la arquitectura TRENCADIS, tendrá dos tipos de interacción diferentes, a diferencia de los servicios que se ubican en la capa Servicios Servidor que solo tienen interacciones de entrada y salida. Por una parte, el servicio requerirá de las interacciones propias que se realizan a través del componente de interacción con los dispositivos que manejan los repositorios DICOM, y que serán detalladas en el apartado correspondiente al componente de interacción. Por otra parte, el servicio requerirá de las interacciones que se producirán con otros componentes de la arquitectura TRENCADIS (servicios Grid y componentes Middleware), es decir, las interacciones de entrada y salida. Estas interacciones de entrada y salida son principalmente tres, por una parte con el servidor de índices (servicio Grid Storage Broker) cuando haya alguna modificación en los datos del repositorio DICOM, por otra parte con el servicio Grid servidor de ontologías cuando se produzca un cambio en alguna ontología, ya que el servicio Grid Storage DICOM debe actualizar sus datos en función de las modificaciones realizadas sobre éstas. Por último, también con el servicio IDXSRV del sistema de monitorización e información MDS4-Extended, para su registro y para la consulta de la ubicación de los diferentes servicios con los que interactúa.

Interacciones de Entrada

Las peticiones que realiza el servicio Grid Servidor de Ontologías (punto 1 de la figura 32) al servicio Grid Storage DICOM, se producen cuando existe un alta, baja o modificación de alguna ontología, pues el servicio Grid Storage DICOM debe actualizar sus índices en función de los datos guardados en el repositorio y las definiciones de las ontologías actualizadas. El

flujo de las interacciones ha sido descrito detalladamente en el apartado anterior, referente a las interacciones de salida del servicio Grid Servidor de Ontologías (figura 26 y figura 27).

También se producen interacciones de entrada por parte del Componente Middleware Storage DICOM (punto 2 de la figura 32), encargado de la gestión de los almacenes virtuales, ya que cuando un usuario realiza una consulta, ésta es trasladada a todos los servicios Grid Storage DICOM activos que forman parte del almacén Virtual, de manera paralela, y recoge la información correspondiente a la consulta, si el usuario que ha solicitado la información para una ontología determinada tiene privilegios suficientes.

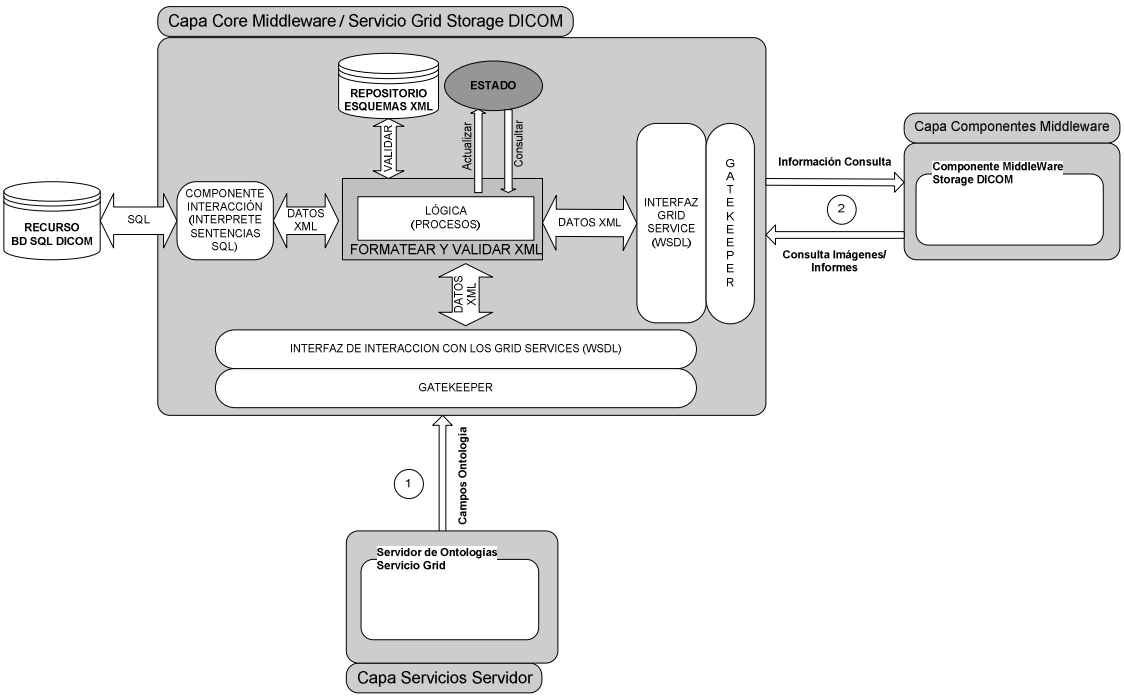

figura 32. Interacción del servicio Grid servidor de Ontologías (altas, bajas o modificaciones de ontologías) con el servicio Grid Storage DICOM. Además, muestra la interacción del componente Middleware Storage DICOM (Consultar Información) con el servicio Grid Storage DICOM.

Interacciones de Salida

La interacción de salida que se produce en primer lugar, en el servicio Grid Storage DICOM, ocurre cuando se activa el servicio, lo que provoca que el servicio Grid se registre en el servicio de Monitorización e Información MDS4-Extended.

Otra de las interacciones de salida, ocurre de forma continuada al registro y se produce con el servicio Grid Servidor de Ontologías, cuando el servicio Grid Storage DICOM se activa, ya que tiene que actualizar sus vistas en función de las ontologías existentes, pues pueden haber existido altas, bajas o modificaciones sobre las ontologías en el periodo inactivo. En el momento que se activa el servicio Grid Storage DICOM, el servicio comprueba si ha habido modificaciones en las ontologías existentes. Para ello realiza una petición al servicio IDXSRV para consultar la ubicación del servidor de ontologías (pasos 1 y 2 de la figura 33). A continuación consulta al servidor de ontologías sobre los datos de todas las ontologías existentes (paso 3 de la figura 33), el servidor de ontologías responde (paso 4 de la figura 33) para que el servicio Grid Storage DICOM actualice sus datos de acuerdo con las altas, bajas o

modificaciones realizadas sobre las ontologías existentes en los campos “Creation” y “Filter”. Una vez actualizados los datos del servicio Grid Storage DICOM, se actualizan los índices del servidor de índices (servicio Grid Storage Broker) también de acuerdo con las ontologías, por lo que primero se consulta al IDXSRV sobre la ubicación del servidor de índices (pasos 5 y 6 de la figura 33). Una vez localizado éste, se envían los datos correspondientes a los campos “Creation” de las ontologías que han sido modificadas para que se actualicen (paso 7 de la figura 33). Finalmente, el servidor de índices retorna un código al servicio Grid Storage DICOM indicando si la operación se ha realizado satisfactoriamente o no (paso 8 de la figura 33).

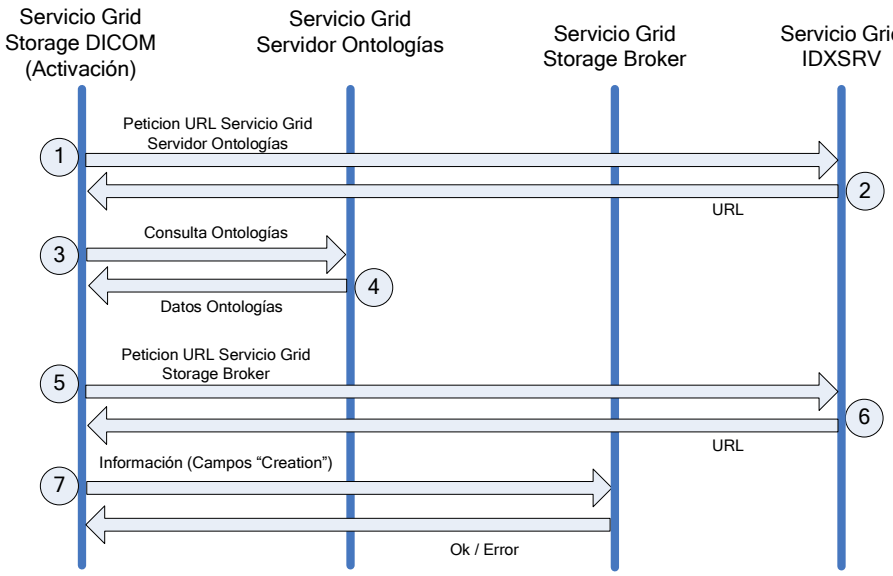

figura 33. Interacción del servicio Grid Storage DICOM con los servicios Grid servidor de ontologías, Storage Broker e IDXSRV en el momento de su activación.

Por último, también se produce una interacción de salida con el servidor de índices (Servicio Grid Storage Broker) cuando existe alguna modificación sobre los datos guardados en el repositorio, pues se deben actualizar los índices con respecto a los datos guardados. Las modificaciones de los datos pueden ser provocadas por los propios usuarios Grid, a través de los componentes middleware, o bien por el propio funcionamiento interno del dispositivo, ya que éstos repositorios DICOM pueden estar integrados en sistemas hospitalarios (PACS, RIS etc.) de forma que el propio funcionamiento de estos sistemas permita la inserción, eliminación y modificación de los objetos DICOM que albergan. Cuando se produce una alteración de los datos DICOM del repositorio (paso 1 de la figura 34), el servicio Grid Storage DICOM localiza el servidor de índices (pasos 2 y 3 de la figura 34) para actualizar los índices en función de las alteraciones que se han producido en los campos “Creation” de las ontologías (pasos 4 y 5 de la figura 34).

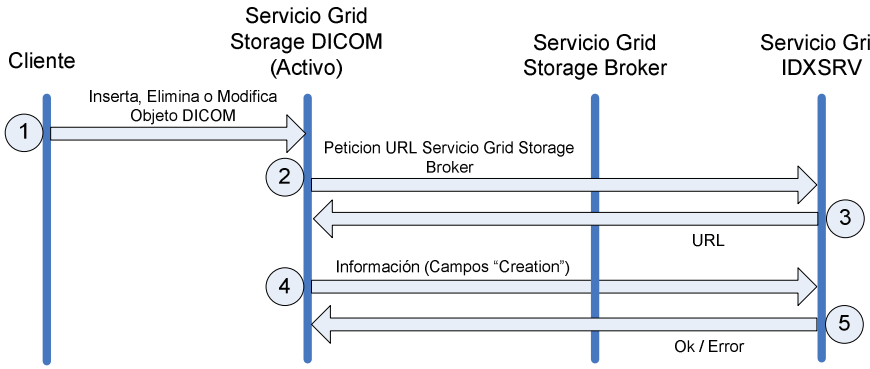

figura 34. Interacción del servicio Grid Storage DICOM con los servicios Grid Storage Broker e IDXSRV cuando se realizan modificaciones sobre los datos del repositorio DICOM.

4.2.1.1.2. Componente de Interacción

Como se ha mencionado anteriormente, las diferentes versiones del servicio Grid Storage DICOM tendrán una parte común (funcionalidad, estado, procesos lógicos e interfaces) y una parte dependiente del dispositivo. La parte dependiente se implementa en el componente de interacción en función del recurso con el que se interactúa. Los objetos DICOM se almacenarán en el repositorio y no en el servicio Grid, que sólo almacenará la información referente a las diferentes ontologías, es decir, los valores correspondientes a los campos "Creation" y "Filter", relacionando éstos con la ubicación física en los repositorios.

El estado de los repositorios se actualiza dinámicamente, es decir, los objetos DICOM que se almacenan pueden ser insertados, modificados o eliminados directamente por los usuarios, con lo que a la hora de gestionar las diferentes vistas ontológicas en el servicio Grid, éstas operaciones deben de ser tenidas en cuenta para realizar las actualizaciones correspondientes. Por ello, las operaciones de insertado, modificación y borrado en los repositorios deben interactuar, a través del componente de interacción, con los procesos lógicos del servicio Grid para modificar su estado, y que las diferentes vistas ontológicas recojan los cambios realizados. Generalmente, estas operaciones requieren de un preproceso de los datos y su anonimización, evitando publicar información médica de carácter personal.

Para realizar estas operaciones, los métodos a reimplementar en el componente de interacción son los siguientes:

Método	**Entrada**	**Salida**
InsertarObjetoDICOM	Objeto DICOM	Código de error.
Descripción		
Este método inserta los datos de un objeto DICOM referente a las diferentes ontologías existentes en el servicio Grid. Previamente a la inserción en el servicio Grid, estos datos se preprocesan y anonimizan si lo requiere el caso de uso.		

Método	Entrada	Salida
BorrarObjetoDICOM	Objeto DICOM	Código de error.
Descripción		
Este método borra los datos de un objeto DICOM referente a las diferentes ontologías existentes en el servicio Grid.		

Método	Entrada	Salida
ModificarObjetoDICOM	Objeto DICOM	Código de error.
Descripción		
Este método modifica los datos de un objeto DICOM referentes a las diferentes ontologías existentes en el servicio Grid.		

Por otra parte, el servicio Grid Storage DICOM también ofrece la posibilidad de descargar objetos DICOM desde el interfaz del usuario. Para ello el servicio Grid recupera directamente del repositorio el objeto y lo deja disponible para la descarga en el servidor GridFTP del servicio, de manera que el usuario pueda descargárselo mediante el componente middleware correspondiente. Para ello, también se utiliza el componente de interacción, el cual dispone de un método que depende de la implementación física del repositorio con el que se interactúa. El método a utilizar para este cometido es el siguiente:

Método	Entrada	Salida
RecuperarObjetoDICOM	Objeto DICOM	Código de error.
Descripción		
Este método recupera un objeto DICOM para la descarga, desde el recurso donde se almacena físicamente, y lo deja preparado en el servidor GridFTP del servicio Grid.		

Por último, los usuarios Grid también pueden generar objetos DICOM (por ejemplo, documentos estructurados) a partir de los componentes middleware, y almacenarlos en los repositorios. Este caso es similar al anterior, con la diferencia de que la interacción es en sentido contrario. En el caso anterior, el repositorio cambiaba su contenido y notificaba al servicio Grid. En este caso, el usuario crea el objeto DICOM y mediante el servicio Grid, modifica el contenido del repositorio agregando el objeto. Para ello, el usuario dispara el proceso lógico de transferencia de objetos DICOM mediante el interfaz correspondiente. Este proceso a su vez se encarga de actualizar el estado del servicio Grid en función de las ontologías, y utilizar el componente de interacción para guardar el objeto en el repositorio. Para ello es necesario también reimplementar el método correspondiente del componente de interacción.

Método	Entrada	Salida
GuardarObjetoDICOM	Objeto DICOM	Código de error
Descripción		
Este método guarda un objeto DICOM en el repositorio donde se almacenan físicamente los objetos DICOM.		

En términos generales, el componente de interacción se encargará de realizar peticiones a los procesos lógicos del servicio Grid por medio de los interfaces WSRF ofrecidos al recurso, realizando los prepocesos pertinentes y codificando los parámetros en las estructuras XML compatibles con las especificaciones de procesos lógicos.

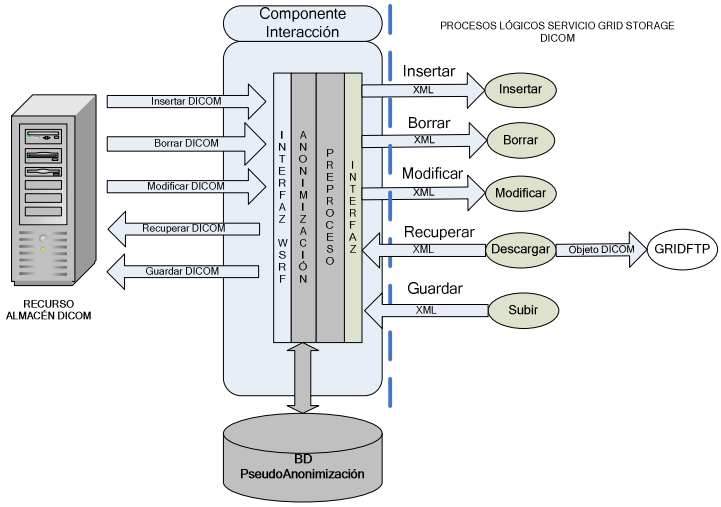

figura 35. Esquema General del Componente Interacción de un servicio Grid Storage DICOM. Se muestra el componente interfaz WSRF que interactúa directamente con el recurso, la componente de pseudos-anonimización de los datos, la componente de preproceso para la adecuación y normalización de la imagen y la componente de interfaz con los procesos lógicos mediante tipos de datos definidos con documentos XML.

4.2.1.1.3. GateKeeper

Además de realizar las tareas de seguridad descritas en el apartado 3.1.3.1, referente a la estructura general de la arquitectura TRENCADIS, el Gatekeeper contacta con el servicio Grid servidor de Ontologías para validar el acceso de las credenciales VOMS presentadas por un usuario. Estas credenciales indican los grupos de trabajo que presenta un usuario, el cual podrá o no acceder a la información de la ontología con la que se quiere trabajar. El servicio servidor de ontologías guarda las relaciones entre áreas de trabajo y las ontologías que se pueden manejar.

4.2.1.1.4. Estado del Servicio Grid

El servicio Grid viene identificado de forma única, dentro de un despliegue Grid, con una clave que identifica de forma inequívoca al servicio Grid dentro de una OV. Además, contiene un atributo donde se alberga la descripción del último error producido en el servicio Grid.

El estado de este recurso viene determinado por las diferentes ontologías definidas en el despliegue Grid almacenadas en el servidor de ontologías. Para este cometido, se ha definido una base de datos relacional bajo Postgress SQL, cuya definición se muestra en la figura 36.

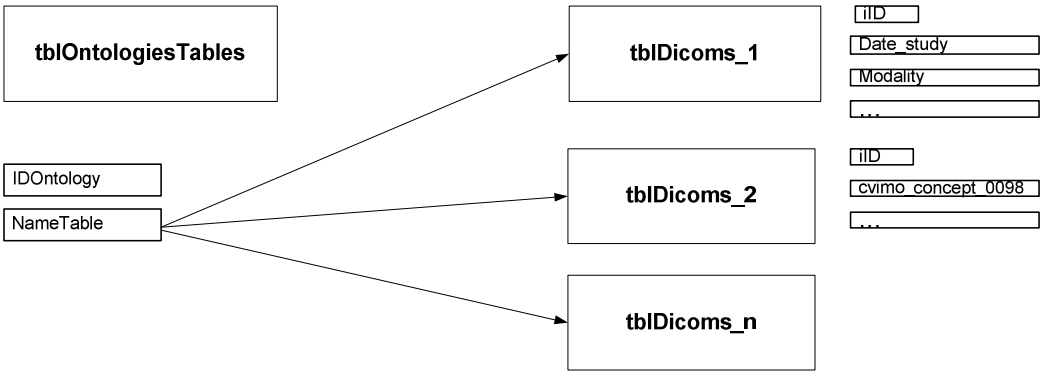

figura 36. Diagrama Relacional de la Base de datos del servicio Grid Storage DICOM.

La base de datos de la figura 36, muestra el diagrama relacional en el que se estructura y almacena el estado del servicio. La tabla tblOntologiesTables registra las ontologías y contiene los nombres de las tablas donde se encuentra la información relacionada con una determinada ontología, es decir, los valores de los campos definidos en las ontologías como "Creation" y "Filter", junto al identificador interno que se le asocia. La creación de estas tablas se realiza dinámicamente en la creación y eliminación de las ontologías. Además el contenido de estas tablas también se actualiza dinámicamente en función del insertado y borrado de los objetos DICOM, tanto por parte del repositorio como por parte de los usuarios Grid.

4.2.1.1.5. Procesos lógicos y Esquemas XML de entrada/salida

A continuación se presenta una breve descripción de los procesos más significativos que proporcionan la funcionalidad correspondiente del servicio Grid Storage DICOM, indicando si requiere de esquemas XML para las entradas y salidas correspondientes.

Proceso	**Esquema XML de Entrada**	**Esquema XML de Salida**
Arranque	No	No
Descripción		
El proceso de arranque se produce al activar el servicio Grid y se encarga de inicializar los valores del estado. Primero interacciona con el servicio Grid del MDS4-Extended, registrando el servicio como activo y retornando a éste el identificador del recurso que se le ha asignado. Una vez asignado el recurso, solicita la información de las ontologías al servidor de ontologías, registrando las nuevas ontologías en la tabla de la base de datos tblOntologiesTables y creando las tablas pertinentes. También comprueba las ontologías eliminadas, para así borrar la información correspondiente de la tabla tbOntologiesTables y borrando las tablas implicadas. Por otra parte, también interactúa con el servicio Grid Storage Broker, enviando la información que requiere de las nuevas ontologías para la indexación de los campos definidos en éstas. Una vez realizadas todas estas tareas, el servicio espera peticiones del componente de interacción o del interfaz para ejecutar los procesos lógicos que se requieran.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperar Error	No	RecuperarError.xsd
Descripción		
Proceso que retorna la descripción del último error que se ha producido en el servicio Grid.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Proceso Inicializar Descarga Objeto DICOM	input_initcializeDownload.xsd	return_initcializedownload.xsd
Descripción		
Proceso que se encarga de preparar las descargas de las imágenes solicitadas. Esta información se introduce a través de un documento XML conforme al esquema correspondiente. Este servicio comprime la imagen en formato JPEG2000 sin perdidas asegurando que no se pierde la validez diagnóstica, reduciendo su tamaño y optimizando el tiempo de transferencia, al permitir descargas progresivas. Retorna, a quien lo solicita, la información relativa a la imagen comprimida en el documento XML, para que se proceda a su descarga mediante el protocolo de comunicación GridFTP.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Proceso Subida Objeto DICOM	input_upload.xsd	return_upload.xsd
Descripción		
Proceso que se encarga de preparar las subidas de los objetos DICOM. Esta información viene dada a través de un documento XML que cumple el esquema correspondiente.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Búsqueda Objetos DICOM	input_search_DICOM.xsd	return_search_DICOM.xsd
Descripción		
Se encarga de recoger las peticiones de búsqueda, codificadas en el documento XML, para posteriormente traducir éste al lenguaje de consulta del gestor de vistas de almacén del recurso. Por ejemplo, en el caso de que el repositorio sea una base de datos, la petición se codifica en SQL. Se retorna el resultado de dichas búsquedas también de forma codificada en el documento XML de salida.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Crear Nueva Ontología	input_newontology.xsd	return_newontology.xsd
Descripción		
Este proceso se encarga de crear una nueva vista en la base de datos a partir de una nueva ontología, registrando en la tabla tblOntologiesTables la información correspondiente y creando la nueva tabla que guardará los datos definidos en la Ontología. Finalmente se debe informar al servicio Grid Storage Broker de la nueva ontología para que éste pueda actualizar la nueva información referente a la nueva ontología, indexando los campos que se hayan definido en ella.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Eliminar Ontología	input_removeontology.xsd	return_removeontology.xsd
Descripción		
Proceso que elimina toda la información referente a una ontología determinada, tanto de la tabla tblOntologiesTables como eliminando la tabla correspondiente a la ontología. Finalmente se debe informar al Storage Broker de la desaparición de una vista de Ontología, para que éste elimine los campos de indexación referentes a este recurso.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Inserta Objeto DICOM	input_insertDICOM.xsd	Return_insertDICOM.xsd
Descripción		
Proceso que inserta un objeto DICOM, insertando la información correspondiente del objeto DICOM en aquellas ontologías en la que esté implicado. Este proceso también debe de informar los datos agregados al servicio Storage Broker para la indexación de los campos.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Borra Objeto DICOM	input_removeDICOM.xsd	Return_removeDICOM.xsd
Descripción		
Proceso que elimina un objeto DICOM, eliminando la información correspondiente del objeto DICOM en aquellas ontologías en las que esté implicado. Este proceso también debe informar al Storage Broker para la actualización de la indexación de los campos.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Modifica Objeto DICOM	input_updateDICOM.xsd	Return_updateDICOM.xsd
Descripción		
Proceso que modifica un objeto DICOM, modificando la información correspondiente del objeto DICOM en aquellas ontologías en las que esté implicado. Éste proceso también debe informar al Storage Broker para la actualización de la indexación de los campos.		

4.2.1.1.6. Interfaces definidas para el Servicio Grid

En este apartado se listan los interfaces mediante los que se podrá interactuar con el servicio Grid Storage DICOM, relacionando estos interfaces con los procesos lógicos a los cuales van ligados.

Interfaz	Descripción	Proceso Lógico
xmlGetError	Retorna el último error producido en el servicio.	Recuperar Error.
xmlSearch	Se encarga de realizar las búsquedas de objetos DICOM en el almacén de objetos DICOM.	Búsqueda Objetos DICOM.
xmlInicializeDownload	Se encarga de preparar las descargas de los objetos DICOM en formato JPEG2000.	Inicializar Descarga Objetos DICOM.
xmlAddOntology	Crea en el Storage DICOM una nueva ontología.	Crear nueva Ontología.
xmlRemoveOntology	Elimina en el Storage DICOM una nueva ontología.	Eliminar Ontología.
xmlUpload	Se encarga de subir objetos DICOM al repositorio.	Subida Objeto DICOM.

4.2.1.2. Servicio Grid Servidor de Códigos

Este componente de la arquitectura está dedicado al tratamiento de documentos estructurados DICOM-SR. Este servicio surge con la necesidad de definir codificaciones en la generación de los documentos estructurados y tener éstos guardados en un repositorio. Por una parte, las codificaciones se necesitan para conceptualizar los nodos del árbol que componen el documento estructurado (es decir, los “concept name”, tal y como se explicó en el apartado 4.1.1, referente a la funcionalidad de manejo de ontologías sobre información DICOM) y por otra parte, también se utiliza para los valores correspondientes a estos campos.

Por tanto, el servicio Grid Servidor de Códigos se encarga de guardar todas las codificaciones y valores para su uso. Este servicio realiza tareas de servidor puramente lógicas, por lo que se ubica dentro de la arquitectura TRENCADIS en la capa de Servicios Servidor.

4.2.1.2.1. Interacciones

Las interacciones de este componente con otros componentes de la arquitectura se reducen a dos. Por una parte, la única interacción de salida que se produce es con el IDXSRV del MDS4-Extended, como ocurre con cualquier otro servicio Grid a la hora de registrarse. El resto de las interacciones son de entrada, y son provocadas por el Componente Middleware encargado de tratar con este servicio para dar de alta, baja o modificar los datos referentes a las codificaciones.

4.2.1.2.2. GateKeeper

Este componente solo se encarga de cumplir con los requerimientos genéricos de seguridad comentados en el apartado 3.1.3.1, referentes a la estructura general de la arquitectura TRENCADIS.

4.2.1.2.3. Estado del Servicio Grid

Al igual que ocurre con el resto de servicios, el servicio Grid servidor de códigos posee un identificador único, mediante el cual el servicio se identifica de manera inequívoca. Por otra parte también contiene una variable donde se guarda la descripción del último error producido.

El estado de este recurso viene determinado por las diferentes codificaciones que se quieran utilizar en la generación y consulta de los DICOM-SR, y que son almacenadas en el servicio Grid. Se ha definido una base de datos relacional, implementando en Postgress SQL, cuya definición se muestra en la figura 37.

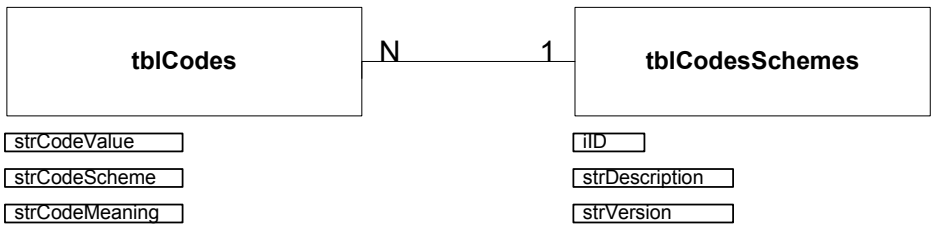

figura 37. Diagrama Relacional de la Base de datos del servicio Grid servidor de Códigos.

La base de datos de la figura 37, muestra el diagrama relacional en la que se alberga el estado del servicio. En la tabla tblCodesSchemes se encuentran las codificaciones dadas de alta y en la tabla tblCodes las codificaciones correspondientes con la tripleta “Code Scheme”, “Code Value” y “Code Meaning” correspondientes.

4.2.1.2.4. Procesos lógicos y Esquemas XML de entrada/salida

A continuación se presenta una breve descripción de los procesos más significativos que proporcionan al servicio de la funcionalidad correspondiente del servicio Grid servidor de códigos, indicando si requiere de los esquemas XML para las entradas y salidas correspondientes.

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Arranque	No	No
Descripción		
El proceso de arranque se produce al arrancar el servicio Grid que se encarga de inicializar los valores del estado.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Recuperar Error	NO	RecuperarError.xsd
Descripción		
Proceso que retorna la descripción del último error que se ha producido en el servicio Grid.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Consulta Código	input_getcode.xsd	return_getcode.xsd
Descripción		
Proceso que retorna toda la información de un código determinado.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Consulta Esquemas	input_getschema.xsd	return_getschema.xsd
Descripción		
Proceso que retorna todos los esquemas guardados en el servidor.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Consulta Código Esquema	input_getcodeschema.xsd	return_getcodeschema.xsd
Descripción		
Proceso que retorna la información de un código para un esquema concreto.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Agregar Código	input_addcode.xsd	return_addcode.xsd
Descripción		
Proceso que agrega un código de un esquema determinado.		

Proceso	Esquema XML de Entrada	Esquema XML de Salida
Eliminar Código	input_removecode.xsd	return_removecode.xsd
Descripción		
Proceso que elimina un código de un esquema determinado.		

4.2.1.2.5. Interfaces definidas para el Servicio Grid

En este apartado se van a listar los interfaces mediante los que se podrá interactuar con el servicio Grid servidor de códigos, relacionando estos interfaces con los procesos lógicos a los que van ligados.

Interfaz	Descripción	Proceso Lógico
xmlGetCodes	Se encarga de retornar toda la información de un código determinado para un esquema concreto.	Consulta Código Esquema.
xmlGetAllCodesSchemes	Se encarga de retornar el nombre de todos los esquemas de codificación albergados en el servidor.	Consulta Esquemas.
xmlGetCodeScheme	Se encarga de retornar todas las codificaciones de un esquema concreto.	Consulta Código Esquema.

4.2.2. Componentes Middleware

En este apartado se presentan los componentes middleware, encargados de ofrecer a los usuarios un interfaz de más alto nivel, para que puedan utilizar las funcionalidades de compartición de objetos DICOM mediante ontologías, ofrecidas por los servicios Grid de la capa infraestructura de una manera sencilla. Los paquetes que ofrecen la funcionalidad de compartición de objetos DICOM mediante ontologías son el paquete de gestión de códigos, el paquete DICOM-I, el paquete DICOM-SR y el paquete de Compartición DICOM que se detallan en los apartados siguientes.

4.2.2.1. Paquete de Gestión de Códigos

En este paquete están implementadas todas las clases que se encargan de gestionar todo lo referente a la consulta, insertado, actualización y borrado, de los esquemas de codificaciones que se requerirán para la gestión y creación de los DICOM-SR mediante objetos de alto nivel.

4.2.2.1.1. Diagrama Objetos

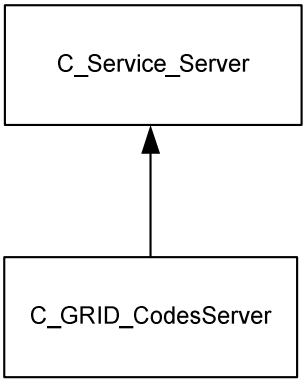

figura 38. Diagrama de clases del Paquete Gestión de Códigos.

La clase C_GRID_CodesServer hereda directamente de la clase C_Service_Server al igual que ocurría con la clase C_GRID_OntologyServer, al ubicar este servicio en la capa servicios Servidor de la arquitectura TRENCADIS. A través de C_GRID_CodesServer se realizan las altas, bajas y modificaciones de las codificaciones.

4.2.2.1.2. Componente C_GRID_CodesServer

Este componente se encarga de crear instancias de objetos que interactúan directamente con el servicio Grid Servidor de códigos, para realizar las consultas, insertado, modificaciones y borrado de los esquemas y codificaciones que se encuentran guardados en el servidor.

Estado

El estado de este componente vendrá determinado por los diferentes esquemas y códigos que se definan en el servicio Grid servidor de códigos donde estos son guardados.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_CodesServer. Estos componentes ofrecen, para su interacción, métodos con tipos de datos básicos o complejos para la entrada o salida.

Método	**Entrada**	**Salida**
Constructor	C_GRID_Session→Credenciales de la session.	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar los esquemas de los códigos almacenados en el gestor de códigos.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
htGetAllCodesScheme	String →Código del esquema del cual se pretende extraer sus códigos.	Hashtable→ Hashtable con los códigos como clave y el significado de los códigos como valor.
Descripción Proceso Lógico Asociado		
Este método se encarga de construir una hashtable con todos los códigos y su significado.		

Método	Entrada	Salida
htGetCodesScheme	String→ Código del esquema sobre el cual se pretende extraer información. String→ Valor del esquema del cual se pretende extraer la información. Admite expresiones regulares mediante el carácter '*'.	Hashtable→ Hashtable con los códigos como clave y el significado de los códigos como valor.
Descripción Proceso Lógico Asociado		
Este método se encarga de construir una hashtable con uno o varios códigos y sus significados.		

Método	Entrada	Salida
htGetAllCodesSchema	No	Hashtable→ Hashtable con los códigos de esquemas como clave y una breve descripción de estos como valor.
Descripción Proceso Lógico Asociado		
Este método se encarga de construir una hashtable con todos los códigos de los esquemas existentes en el servicio Grid Servidor de Códigos y una breve descripción.		

4.2.2.2. Paquete DICOM-I

Este paquete se encarga de la gestión de imágenes médicas DICOM que están guardadas en los servicios Grid Storage DICOM. Mediante este paquete se puede tratar las imágenes médicas de manera individual o conjunta (series, estudios, conjunto de imágenes, etc…).

4.2.2.2.1. Diagrama Objetos

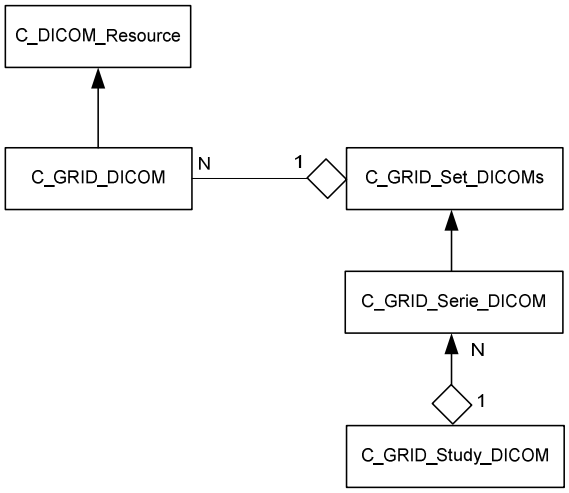

figura 39. Diagrama de Clases del Paquete DICOM-I.

C_GRID_DICOM hereda directamente de la clase C_DICOM_Resource, el cual contiene toda la funcionalidad genérica de cualquier recurso. Las clases C_GRID_Set_DICOMs, C_GRID_Serie_DICOM y C_GRID_Study_DICOM manejan un conjunto de imágenes en general, una serie determinada o un estudio determinado, respectivamente. La clase C_GRID_Serie_DICOM es una especialización de la clase C_GRID_Set_DICOMs, de manera que todas las imágenes que alberga son de una misma serie. La clase C_GRID_Study_DICOM está formada por una o varias series de un mismo estudio.

4.2.2.2.2. Componente C_GRID_Image_DICOM

Este componente del middleware se encarga de crear instancias que representen a una imagen médica DICOM. La componente gestionará todo lo referente a una imagen DICOM (datos de cabecera y píxeles de la imagen), limitando el acceso a la información a los campos que se definan en la ontología.

Realmente, la ubicación física del fichero se encuentra en el repositorio del servicio Grid Storage DICOM correspondiente, el componente middleware tendrá siempre asociado este servicio Grid y utiliza su interfaz WSRF para gestionar los datos del fichero (transferencias o consultas). Este objeto hace transparente la conexión y gestión de la imagen para el desarrollador como si se encontraran localmente.

Estado

El estado del componente viene definido por los valores de la cabecera de la imagen y de la referencia del servicio Grid Storage DICOM que contiene realmente ésta imagen, así como la ontología con la que se haya accedido a la imagen. Es la ontología de partida la que definirá los datos de interés de la imagen médica que se pueden manejar.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Image_DICOM.

Método	Entrada	Salida
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar una imagen médica ubicada en un servicio Grid Storage DICOM.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
strGetTAGHeader	String→Alias de la etiqueta de la cabecera de la imagen DICOM.	String→Valor de la etiqueta de la cabecera de la imagen DICOM. Si es null indica algún error.
Descripción Proceso Lógico Asociado		
Método que retorna el valor de un campo especificado en la entrada de la cabecera de la imagen DICOM.		

Método	Entrada	Salida
xmlGetDICOMHeader	No	String→Documento XML que contiene los datos de la cabecera en función de la ontología definida.
Descripción Proceso Lógico Asociado		
Método que retorna un documento XML con los datos de la cabecera definidos según la ontología con la que se trabaja.		

Método	Entrada	Salida
iSetTAGHeader	String→ Alias de la etiqueta.	int→ Código de error. Un valor diferente de cero indica la presencia de un error.
	String →Valor a asignar.	
Descripción Proceso Lógico Asociado		
Método que asigna un valor a la cabecera DICOM dado el nombre del campo de cabecera que se quiera actualizar.		

Método	Entrada	Salida
asGetImage	No	Array de Shorts→ Array que contiene los puntos de la imagen representada mediante una escala de grises de 12 bits. La dimensión principal es el ancho.
Descripción Proceso Lógico Asociado		
Método que retorna los puntos de la imagen como un array de shorts que representan las diferentes filas de la imagen. Estos puntos se han descargado anteriormente mediante los componentes de descarga correspondientes.		

Método	Entrada	Salida
iSetImage	Array de Shorts→Array de shorts que representa una imagen en escala de grises de 12 bits. La dimensión principal es el ancho.	int→ Código de error. Un valor diferente de cero indica la presencia de un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de asignar una imagen codificada en escala de grises de 12 bits.		

4.2.2.2.3. Componente C_GRID_Set_Images_DICOMs

Este componente se encarga de la gestión y manejo de un conjunto de imágenes DICOM en general. Cada una de las imágenes DICOM del conjunto podrá ser gestionada de manera individual, mediante el componente C_GRID_Image_DICOM.

Estado

El estado de componente viene definido por los diferentes C_GRID_Image_DICOM que componen este objeto.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_Set_Images_DICOMs.

Método	Entrada	Salida
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar un conjunto de imágenes médicas, ubicadas en uno o varios servicios Grid StorageDICOM.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
oGetImageDICOM (Sobrecargado)	Int→Índice de la imagen DICOM dentro del conjunto que alberga el objeto.	C_GRID_Image_DICOM → Imagen DICOM. Si es null indica que se ha producido un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar un objeto C_GRID_image_DICOM que corresponde con la imagen cuyo índice, dentro del conjunto de imágenes, se ha especificado en la entrada.		

Método	Entrada	Salida
oGetImageDICOM (Sobrecargado)	String→UID del estudio de la imagen.	C_GRID_Image_DICOM → Imagen DICOM. Si es null indica que se ha producido un error.
	String→UID de la serie a la que pertenece la imagen.	
	Int→Número de la imagen DICOM.	
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar un objeto C_GRID_Image_DICOM que corresponde con un número de estudio, serie y número de imagen determinado.		

4.2.2.2.4. Componente C_GRID_Serie_DICOM

Este componente se encarga de la gestión y manejo de un conjunto de imágenes DICOM que corresponden a una misma serie de un estudio DICOM determinado. Cada una de las imágenes DICOM en el conjunto, se gestiona mediante el componente C_GRID_Image_DICOM.

Estado

El estado del componente vendrá definido por los diferentes C_GRID_Image_DICOM que componen este objeto.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Serie_DICOM.

Método	Entrada	Salida
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto que gestiona un conjunto de imágenes médicas, ubicadas en uno o varios servicios Grid Storage DICOM de una misma serie de un estudio DICOM concreto.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
oGetImageDICOM	Int→ Índice de la imagen DICOM dentro del conjunto que alberga el objeto.	C_GRID_Image_DICOM → Imagen DICOM. Si es null indica que se ha producido un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar un objeto C_GRID_Image_DICOM que corresponde al índice de entrada dentro del conjunto de imágenes.		

Método	Entrada	Salida
oGetImageDICOM	Int→Número de la imagen DICOM.	C_GRID_Image_DICOM → Imagen DICOM. Si es null indica que se ha producido un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar un objeto C_GRID_Image_DICOM que corresponde con un número de imagen determinado dentro del estudio y serie a la que pertenece.		

4.2.2.2.5. Componente C_GRID_Study_DICOM

De forma análoga a los anteriores, este componente se encarga de la gestión y manejo de un conjunto de imágenes DICOM que corresponden a un mismo estudio DICOM. Cada una de las imágenes DICOM que se maneje en el conjunto, serán gestionadas mediante el componente C_GRID_Image_DICOM. Además, se ofrecerán métodos para actuar sobre el conjunto de imágenes C_GRID_Image_DICOM, al igual que ocurre con C_GRID_Set_Images_DICOM y C_GRID_Serie_DICOM.

Estado

El estado del componente vendrá definido por el contenido de los diferentes C_GRID_Image_DICOM que componen este objeto.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Study_DICOM.

Método	**Entrada**	**Salida**
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar un conjunto de imágenes médicas, ubicadas en uno o varios servicios Grid Storage DICOM de un mismo estudio.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	**Entrada**	**Salida**
oGetImageDICOM	Int→ Índice de la imagen DICOM dentro del conjunto que alberga el objeto.	C_GRID_Image_DICOM → Imagen DICOM. Si es null indica que se ha producido un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar un objeto C_GRID_image_DICOM que corresponde al índice de entrada dentro del conjunto de imágenes.		

Método	**Entrada**	**Salida**
oGetImageDICOM	String→UID (Codificación DICOM) de la Serie.	C_GRID_Image_DICOM → Imagen DICOM. Si es null indica que se ha producido un error.
	Int→Número de la imagen DICOM.	
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar un objeto C_GRID_Image_DICOM, que corresponde a una serie determinada y un número de imagen dentro del estudio DICOM al que pertenece.		

4.2.2.3. Paquete DICOM-SR

Este paquete se encarga de la gestión de informes estructurados DICOM-SR que se almacenan en los repositorios de los servicios Grid Storage DICOM. Mediante este paquete se puede tratar los DICOM-SR de manera individual o conjunta, al igual que ocurre con el paquete DICOM-I para imágenes médicas.

4.2.2.3.1. Diagrama Objetos

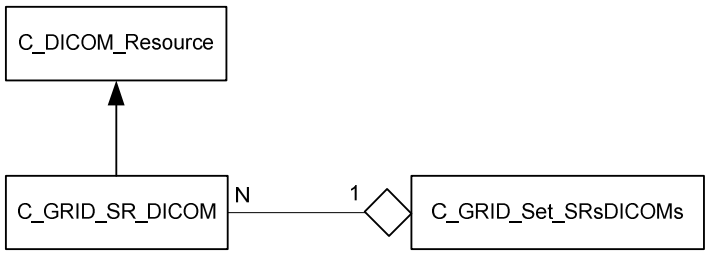

figura 40. Diagrama de Clases del Paquete DICOM-SR.

C_GRID_SR_DICOM hereda directamente de la clase C_DICOM_Resource, que es la clase genérica implementada para todo recurso Grid. La clase C_GRID_Set_SRsDICOMs maneja un conjunto de documentos estructurados en general.

4.2.2.3.2. Componente C_GRID_SR_DICOM

Este componente del Middleware se encarga de crear instancias que representan a un informe estructurado DICOM-SR. Este componente gestionará todo lo referente al DICOM-SR, es decir datos de cabecera de cualquier objeto DICOM y el árbol correspondiente del documento estructurado, en el que cada nodo del árbol viene indexado por un "concept name", tal y como se ha explicado en el apartado 1.2.4, referente al estándar DICOM. Todos estos datos serán accesibles en función de la ontología mediante la cual se cree la instancia del objeto, pues ésta indica los campos que pueden tratarse en el objeto.

Realmente la ubicación física del fichero se encuentra en un servicio Grid Storage DICOM. Este objeto hace transparente la conexión y gestión del documento estructurado para el desarrollador, independientemente de su ubicación.

Estado

El estado del componente viene definido por los valores de la cabecera del objeto DICOM y de la referencia del servicio Grid Storage DICOM que contiene realmente el DICOM-SR, además de la ontología en la que se esté trabajando y que define los datos de interés.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_SR_DICOM.

Método	**Entrada**	**Salida**
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar un informe estructurado ubicado en un servicio Grid Storage DICOM.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
strGetTagHeader	String→Alias del tag de la cabecera del DICOM-SR.	String→Valor del campo de la cabecera del DICOM-SR. Si es null indica que se ha producido algún error.
Descripción Proceso Lógico Asociado		
Método que retorna el valor de un campo especificado en la entrada de la cabecera del DICOM-SR.		

Método	Entrada	Salida
xmlGetDICOMHeader	No	String→Documento XML que contiene los datos de la cabecera en función de la ontología definida.
Descripción Proceso Lógico Asociado		
Método que retorna un documento XML con los datos de la cabecera definidos según la ontología con la que se trabaja.		

Método	Entrada	Salida
iSetTagHeader	String→ Alias del tag a asignar. El tag es el código DICOM del campo a asignar.	int→ Código de error. Un valor distinto de cero indica la aparición de un error.
	String →Valor a asignar al campo.	
Descripción Proceso Lógico Asociado		
Método que asigna un valor a la cabecera DICOM dado el nombre del campo de cabecera que se quiera asignar.		

Método	Entrada	Salida
strGetSR	No	String→ String que corresponde a un documento XML con los campos que componen el árbol del documento estructurado.
Descripción Proceso Lógico Asociado		
Método que retorna un documento XML del árbol que corresponde con el documento estructurado que alberga el objeto DICOM.		

Método	Entrada	Salida
iSetSR	String→Documento XML que representa a un documento estructurado.	int→ Código de error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de asignar un documento estructurado al objeto DICOM.		

4.2.2.3.3. Componente C_GRID_Set_SRs_DICOMs

Este componente se encarga de la gestión y manejo de un conjunto de documentos estructurados DICOM-SR. Cada uno de los DICOM-SR del conjunto se gestiona mediante el componente C_GRID_SR_DICOM.

Estado

El estado de componente vendrá definido por el contenido de los diferentes C_GRID_SR_DICOM que componen este objeto.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_Set_SRs_DICOMs.

Método	Entrada	Salida
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto que gestiona un conjunto de documentos estructurados, ubicados en uno o varios servicios Grid Storage DICOM.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
oGetSRDICOM	Int→Índice del DICOM-SR.	C_GRID_SR_DICOM → DICOM-SR. Si es null indica que se ha producido un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar el objeto C_GRID_SR_DICOM que corresponde con el índice de entrada dentro del conjunto de documentos estructurados.		

4.2.2.4. Paquete de Creación de Almacenes Virtuales de Objetos DICOM

Este componente se encargará de gestionar conjuntos de objetos DICOM que hay distribuidos en los diferentes servicios Grid Storage DICOM, desplegados en el entorno Grid en función de una ontología determinada. Los conjuntos de objetos se podrán manejar como si formarán parte de un único almacén virtual, aunque su ubicación física sea totalmente distribuida.

Los almacenes virtuales, dada una ontología determinada, proporcionarán una visión consolidada virtual de un conjunto de objetos DICOM, de manera que sólo los estudios que cumplan las restricciones de la ontología serán visibles y accesibles.

4.2.2.4.1. Diagrama Objetos

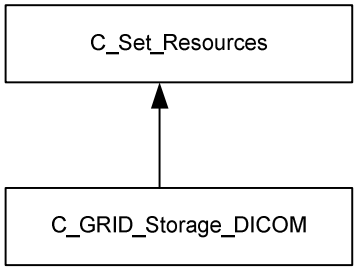

figura 41. Diagrama de Clases del Paquete Creación de Almacenes Virtuales de Objetos DICOM.

C_GRID_Storage_DICOM hereda directamente de la clase C_Set_Resources, que es la clase genérica implementada para manejar un conjunto de recursos. La clase C_GRID_Storage_DICOM maneja un conjunto de servicios Grid Storage DICOM como si fuera un único almacén.

4.2.2.4.2. Componente C_GRID_StorageDICOM

Este componente se encarga de virtualizar diferentes almacenes de objetos DICOM representados por servicios Grid StorageDICOM, dada una determinada ontología y dados unos

criterios de creación que deben estar definidos en la propia ontología. Mediante este componente es posible realizar búsquedas y filtrados de objetos DICOM como si fuera un único almacén.

Además de interactuar con el servicio Grid Storage DICOM, este componente también interactúa directamente con el servicio Grid Storage Broker en el momento de creación de un almacén virtual, ya que este servicio proporciona, al objeto creado, las direcciones donde se ubican los servicios Grid Storage DICOM que cumplen los criterios de creación contenidos en la ontología presentada.

Estado

El estado del componente vendrá definido por los diferentes servicios Grid de Storage DICOM que formarán el almacén virtual.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_StorageDICOM.

Método	**Entrada**	**Salida**
Constructor	C_GRID_Session→Credenciales de la session. Int→Identificador de la ontología a la que pertenece.	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar el almacén virtual DICOM. Al no darse criterios de creación, se presupone que son válidas todas las fuentes que albergan objetos DICOM y que tienen información de la ontología seleccionada. Esta información la proporciona el servicio Grid Storage Broker.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
Constructor	Int→Identificador de la ontología a la que pertenece.	No
	String[]→Campos de creación	
	String[]→Criterios de creación.	
	String[]→Valores de creación.	
	String[]→Operadores de creación.	
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar el almacén virtual DICOM. Este componente pregunta al servicio Grid Storage Broker con los criterios de entrada, y éste le responde con los servicios Grid Storage DICOM que contienen objetos DICOM según los criterios de entrada y la ontología seleccionada.		

Método	Entrada	Salida
oSearchDICOMImages	String[]→Campos de búsqueda.	C_GRID_Set_Images_DICOM → Conjunto de imágenes DICOM resultado de la búsqueda. Si es null indica que se ha producido error.
	String[]→Criterios búsqueda.	
	String[]→Valores de búsqueda.	
	String[]→Operadores búsqueda.	
Descripción Proceso Lógico Asociado		
Método que permite realizar una consulta, en paralelo, sobre todos los servicios Grid Storage DICOM que componen el almacén virtual, recuperando un conjunto de objetos C_GRID_Image_DICOM que cumplen los criterios de búsqueda, mediante el objeto C_GRID_Set_Images_DICOM.		

Método	Entrada	Salida
oSearchDICOM_SRs	String[]→Campos de búsqueda.	C_GRID_Set_SRs_DICOM → Conjunto de DICOM-SR resultado de la búsqueda. Si es null indica que se ha producido algún error.
	String[]→Criterios de búsqueda.	
	String[]→Valores de búsqueda.	
	String[]→Operadores búsqueda.	
Descripción Proceso Lógico Asociado		
Método que lanza una consulta, en paralelo, sobre todos los servicios Grid Storage DICOM que componen el almacén virtual, recuperando un conjunto de objetos C_GRID_SR_DICOM que cumplen los criterios de búsqueda, mediante el objeto C_GRID_Set_SRs_DICOM.		

4.3. Transferencia de Información DICOM

En este apartado se aborda la transferencia de información en formato DICOM, objetivo que se ha descrito en el apartado 2.1.3, referente a los objetivos. Se tiene en consideración tanto la descarga desde los repositorios fuentes donde se almacena los objetos DICOM hasta una aplicación local, como la transferencia de objetos DICOM a estos repositorios. Ambas operaciones (descarga y transferencia a repositorios) deben realizarse de manera segura y con un orden de prioridades si lo que se transfiere es un conjunto de objetos DICOM. Además, la transferencia a repositorios tiene implicaciones adicionales de seguridad, ya que la información puede ser guardada en dominios administrativos diferentes al del origen de la información DICOM. Incluso teniendo en cuenta que la información se pseudoanonimiza, las regulaciones legales actuales no garantizan que la identidad de la información médica (como imágenes, incluso pseudoanonimizadas) no pueda en el futuro desvelarse mediante las nuevas técnicas de cálculo o a través de nuevos procesos. Por ello, se aplican las restricciones que la LOPD y la ley europea [75][76] que exigen el tratamiento de datos más críticos (nivel 3) y el modelo se seguridad descrito en el apartado 3.3.3.1, referente al modelo de seguridad distribuida entre diversos dominios administrativos pertenecientes a la OV en la que se trabaja.

Cabe decir que en esta funcionalidad, se ven implicados diversos componentes de la arquitectura TRENCADIS ya descritos en apartados anteriores. A continuación se describirá la relación de estos componentes con la funcionalidad, y posteriormente se detallarán los componentes de alto nivel implementados a través de cuatro paquetes: el paquete de descarga de imágenes DICOM, paquete de descarga de documentos estructurados DICOM-SR, paquete de transferencia de imágenes médicas DICOM y paquete de transferencia de documentos estructurados DICOM-SR.

4.3.1. Infraestructura

Toda la infraestructura que soporta esta funcionalidad ha sido descrita y desarrollada en apartados anteriores. A continuación se va a describir de qué manera estos componentes están implicados.

Por una parte, en la transferencia se utilizan los protocolos de comunicación descritos en el apartado 3.1.3.2, referente a la estructura general de la arquitectura. El protocolo GridFTP es el más utilizado para las transferencias de los objetos DICOM, mientras que el protocolo SOAP se emplea en periodos de negociado entre el emisor y el receptor de la transferencia y su inicialización.

Por otra parte los servicios Grid implicados en las transferencias, permiten realizar descargas desde un repositorio de Objetos DICOM a una aplicación local, y transferencias de estos objetos desde una aplicación local a un repositorio de Objetos DICOM. En estos casos, aunque los servicios implicados son los mismos, los pasos que se siguen varían significativamente. El servicio Grid principalmente implicado es el servicio Grid Storage DICOM, pues es el servicio que interactúa con los repositorios de los objetos DICOM. Por otra parte, también se ven implicados los servicios que se utilizan para el cifrado de los objetos, descritos en el apartado 3.3.4, referente al modelo de seguridad, en el que se desarrolla el modelo de compartición de claves en la arquitectura. Concretamente los servicios involucrados son, el IDXSRV del sistema

de monitorización e información MDS4-Extended, en el cual se publican los servidores de claves, el servicio Grid EOUID y el dominio administrativo donde se albergan. En el servicio Grid servidor de claves se guardan las partes de las claves que se utilizan para cifrar la información, y el servicio Grid EOUID asigna un identificador único a los objetos para su cifrado.

4.3.2. Componentes Middleware

Con respecto a la funcionalidad de transferencia de información DICOM, se han desarrollado cuatro paquetes de componentes middleware que se encargan de gestionar las transferencias de objetos DICOM, principalmente las relacionadas con imágenes médicas y documentos estructurados DICOM-SR.

4.3.2.1. Paquete Descarga Imágenes DICOM

Los componentes desarrollados en este paquete se encargan de la descarga de las imágenes médicas DICOM, gestionadas de forma individual o conjunta, mediante componentes de tipo C_GRID_Image_DICOM. Las descargas podrán configurarse, pudiendo determinar diferentes modalidades de acuerdo con el orden mediante el cual se quieran realizar las descargas y el modelo progresivo. Además, la descarga implica la gestión de las claves de cifrado de datos en caso de que los objetos DICOM estén cifrados.

Por otra parte, también se pueden definir prioridades en la descarga de los diferentes objetos DICOM. El modelo de prioridades se basa en la definición de diferentes niveles, en el que el nivel más prioritario es el nivel 0. La información a transferir que corresponda al nivel (i) determinado, no empezará a transferirse hasta que toda la información definida en el nivel (i -1) esté completamente transferida. Toda la información que corresponda a un mismo nivel tendrá un orden de descarga en función del modelo seleccionado, que será descrito en los apartados siguientes.

En caso de que el usuario esté autorizado para el descifrado de un objeto, según el modelo de seguridad definido en la sección 3.3, el proceso de recuperación y unificación de las claves y el descifrado de datos es transparente para el usuario.

4.3.2.1.1. Diagrama Objetos

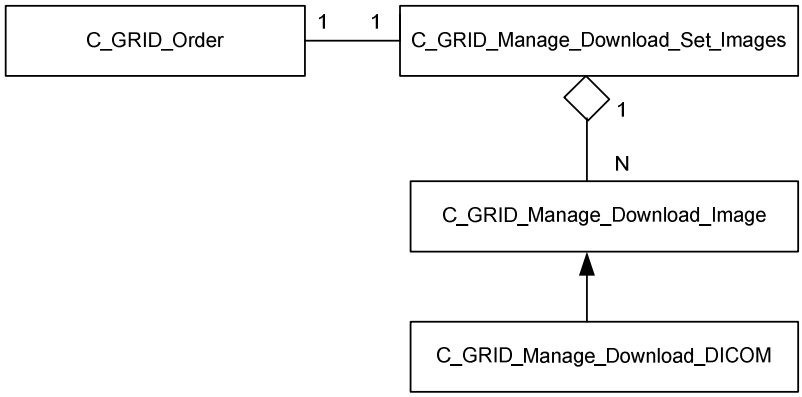

figura 42. Diagrama de Clases del Paquete de Descarga de Imagen Médica.

La clase C_GRID_Manage_Download_DICOM alberga toda la funcionalidad de descarga común a todo objeto DICOM. La clase C_GRID_Manage_Download_Image es una especialización de C_GRID_Manage_Download_DICOM para imágenes. Para el manejo de la descarga de un conjunto de C_GRID_Manage_Download_Image se define la clase C_GRID_Manage_Download_Set_Images, que tiene en cuenta las diversas modalidades de descarga y prioridades definidas en C_GRID_Order.

4.3.2.1.2. Componente C_GRID_Order

Este componente se encarga de organizar una serie de índices que corresponderán a objetos DICOM manejados por las componentes C_GRID_Image_DICOM o C_GRID_SR_DICOM a transferir. Este objeto establece el orden por prioridades descrito (desde 0 a infinito), de forma que para una transferencia conjunta de objetos DICOM, mediante los componentes C_GRID_Manage_Set_Imagenes o C_GRID_Manage_Set_SRs, la transferencia se realice en un orden determinado.

Estado

El estado del componente vendrá definido por las prioridades y el orden de transferencia de los objetos DICOM implicados en este componente.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Order.

Método	**Entrada**	**Salida**
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto que gestiona los índices de descarga con prioridades, de un conjunto de objetos que representa objetos DICOM.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	**Entrada**	**Salida**
vGetPriorities	No	Vector→Vector en el que cada elemento es otro vector. El elemento i con el nivel i y el elemento vector contiene los índices de los objetos a transferir.
Descripción Proceso Lógico Asociado		
Método que retorna un vector formado por diferentes vectores que contienen los índices de los objetos DICOM a descargar o subir de diferentes prioridades.		

Método	Entrada	Salida
vGetOrderPrioriries	Int→Índice de la prioridad.	Vector→Vector con los índices que representan los objetos DICOM de forma ordenada.
Descripción Proceso Lógico Asociado		
Método que retorna los índices de forma ordenada de una determinada prioridad.		

Método	Entrada	Salida
iGetPriority	Int→Índice que representa un objeto DICOM.	Int→Índice que representa la prioridad a la que pertenece el objeto DICOM.
Descripción Proceso Lógico Asociado		
Método que retorna el índice que representa la prioridad de un objeto DICOM, cuyo índice se determina en la entrada del método.		

Método	Entrada	Salida
iGetIndexOrder	Int→Índice del objeto DICOM.	Int→ Índice que representa el orden al que pertenece un objeto DICOM.
Descripción Proceso Lógico Asociado		
Método que retorna el índice que determina el orden de los objetos DICOM, identificados por su código en la entrada del método.		

Método	Entrada	Salida
iSetIndex	Int→Índice que indica prioridad.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
	Int→Índice que indica el Orden.	
	Int→Índice que indica el objeto DICOM	
Descripción Proceso Lógico Asociado		
Método que inserta un índice de un objeto DICOM con una determinada prioridad y orden dentro de la misma prioridad.		

Método	Entrada	Salida
iNextIndex	No	Int→Índice que indicará el próximo objeto DICOM a descargar o subir dentro del orden establecido.
Descripción Proceso Lógico Asociado		
Método que indica el índice del siguiente objeto DICOM a transferir.		

Método	Entrada	Salida
iRemoveIndex	Int→Índice que representa a un objeto DICOM.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que elimina del orden de transferencia a un determinado objeto DICOM a partir de su índice.		

4.3.2.1.3. Componente C_GRID_Manage_Download_Image_DICOM

Este componente permite la gestión de la descarga de imágenes médicas DICOM encapsuladas en componentes C_GRID_Image_DICOM. En caso de ser un objeto cifrado, se encarga además de recuperar todas las partes de la clave presentando las credenciales de usuario a los servicios implicados y descifrando el dato de una manera transparente al usuario.

Las funcionalidades de descarga que se definen en este componente son los siguientes:

- **Descarga Completa:** Descarga la imagen DICOM completamente en un una única operación.
- **Descarga paso a paso:** Descarga paso a paso la imagen en capas de menor a mayor resolución. Estas capas se generan mediante la transformación frecuencial wavelet y el empaquetamiento de diferentes conjuntos de armónicos. Para este proceso se siguen los pasos de la codificación en formato JPEG2000, que además de conseguir buenos resultados en la compresión permite realizar una visualización progresiva de menor a mayor resolución frecuencial. Esta visualización progresiva permite reducir los tiempos de espera improductivos, permitiendo el análisis preliminar desde que el usuario dispone de suficiente información, manteniendo de forma concurrente y en segundo plano la descarga progresiva.

Estado

El estado del componente vendrá definido por la cantidad de bytes transferidos desde el origen de la imagen médica hasta el componente C_GRID_Image_DICOM correspondiente.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GID_Manage_Download_Image_DICOM.

Método	Entrada	Salida
Constructor	C_GRID_Session→Credenciales de la sesión.	No
Descripción Proceso Lógico Asociado		
Este método inicializa el estado del objeto que gestiona la descarga de una imagen DICOM.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que retorna la descripción del último error ocurrido.		

Método	Entrada	Salida
iDownloadComplete	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que descarga completamente la imagen DICOM en un solo paso.		

Método	Entrada	Salida
iInicializeDownloadProgressive	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que inicializa la descarga progresiva mediante el formato JPEG2000 de la imagen DICOM.		

Método	Entrada	Salida
iDownloadProgressiveNextLayerDICOM	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que se encarga de descargar una capa de la imagen DICOM que previamente ha sido codificada para su descarga en formato JPEG2000.		

4.3.2.1.4. Componente C_GRID_Manage_Download_Set_Images

Este componente se encarga de gestionar la descarga de un conjunto de imágenes representado por componentes C_GRID_Image_DICOM. Al igual que ocurre con el componente C_GID_Manage_Download_Image_DICOM, el descifrado es transparente al usuario. La descarga de las imágenes se realizará mediante el orden establecido por el componente C_GRID_Order, permitiendo varias modalidades de descarga:

- **Descarga Imagen a Imagen:** Se realizará la descarga completa de imágenes consecutivas y siguiendo el orden establecido por C_GRID_Order.

- **Descarga Imagen a Imagen por Capas:** Realiza la descarga siguiendo el orden establecido en C_GRID_Order de capas consecutivas de cada imagen. Una vez descargada completamente la imagen, se procede a la descarga por capas de la siguiente.
- **Descarga por Capas:** Realiza la descarga por capas según el orden establecido por C_GRID_Order. Descarga en cada paso una capa de una imagen, y pasa a la imagen siguiente aunque la anterior no esté totalmente descargada. Se procederá a la descarga de las capas de las imágenes de la siguiente prioridad, cuando todas las imágenes de la misma estén completamente descargadas.

Estado

El estado vendrá definido por el conjunto de imágenes DICOM manejadas mediante los componentes C_GRID_Image_DICOM y el estado de descarga en el que se encuentren.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Manage_Download_Set_Images.

Método	**Entrada**	**Salida**
Constructor	C_GRID_Session→Credenciales de la sesión.	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de la descarga de una imagen DICOM.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	**Entrada**	**Salida**
iSetOrderDownload	C_GRID_Order → Objeto que establece el orden de descarga.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que asigna el orden definido en un componente C_GRID_Order para la descarga del conjunto de imágenes DICOM.		

Método	Entrada	Salida
strGetIsComplete	Int→Identificador de una imagen DICOM determinada	String→Retornará "completo" si la imagen se ha descargado completamente e "incompleto" si no se ha descargado completamente.
Descripción Proceso Lógico Asociado		
Método que permite consultar si una imagen representada por el índice de entrada ha sido descargada completamente o no.		

Método	Entrada	Salida
iNextDownloadComplete	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que realiza la descarga de las imágenes en función del orden establecido, si el modo de descarga no es progresivo.		

Método	Entrada	Salida
iSetModeDownload	Int→Modo de descarga. El valor 0 indica una descarga en modo no progresivo. El valor 1 en modo progresivo imagen a imagen y el valor 2 en modo progresivo por capas.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que asigna el modo de descarga a emplear.		

Método	Entrada	Salida
iNextDownloadLayer	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que realiza la descarga de las imágenes en función del orden establecido, si el modo de descarga está activado como progresivo		

4.3.2.2. Paquete para la Descarga de Documentos Estructurados DICOM-SR

Los componentes desarrollados en este paquete se encargan de la descarga de documentos estructurados DICOM-SR gestionados de forma individual o conjunta mediante componentes de tipo C_GRID_SR_DICOM. Como en el caso del paquete de descarga de imágenes médicas DICOM, las descargas podrán configurarse con las mismas modalidades y con el mismo modelo de prioridades. También gestionan las claves de cifrado de datos en caso de que los objetos DICOM estén cifrados.

4.3.2.2.1. Diagrama de Objetos

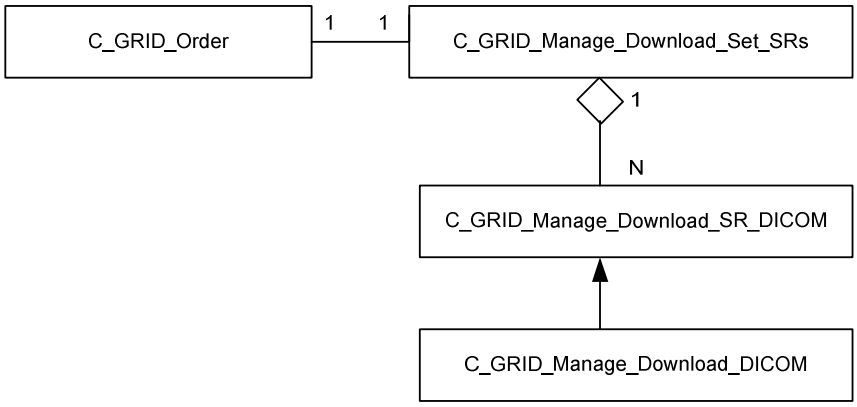

figura 43. Diagrama de Clases del Paquete de Descarga de Documentos Estructurados DICOM-SR.

Las clases que se han definido son análogas las comentadas en el paquete de descarga de imágenes médicas DICOM, pero adaptadas para manejar documentos estructurados DICOM-SR.

4.3.2.2.2. Componente C_GRID_Manage_Download_SR_DICOM

Este componente descarga los objetos que contienen documentos estructurados DICOM-SR encapsulados en componentes C_GRID_SR_DICOM. La funcionalidad es prácticamente la misma que para el C_GRID_Manage_Download_Image_DICOM, pero para documentos estructurados DICOM-SR, excepto que este componente solo permite la modalidad de descarga completa, en la que descarga los documentos estructurados DICOM-SR completamente en cada paso de descarga.

Estado

El estado del componente vendrá definido por la cantidad de bytes transferidos, desde el origen del documento estructurado hasta el componente C_GRID_SR_DICOM correspondiente.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Manage_Download_SR_DICOM.

Método	Entrada	Salida
Constructor	C_GRID_Session→Credenciales de la sesión.	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto que gestiona la descarga de un documento estructurado DICOM-SR.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
iDownload	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que descarga completamente el DICOM-SR en un único paso.		

4.3.2.2.3. Componente C_GRID_Manage_Download_Set_SRs

Este componente se encarga de gestionar la descarga de un conjunto de documentos estructurados representados por componentes C_GRID_SR_DICOM. La funcionalidad que ofrece este componente es equivalente al componente C_GRID_Manage_Download_Set_Images para imágenes, excepto que la descarga de documentos estructurados sólo permite la descarga de documentos completos consecutivos y siguiendo el orden establecido por C_GRID_Order.

Estado

El estado vendrá definido por el conjunto de documentos estructurados DICOM-SR gestionados mediante los componentes C_GRID_SR_DICOM y el estado de descarga en la que se encuentran.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_Manage_Download_Set_SRs.

Método	Entrada	Salida
Constructor	C_GRID_Session→Credenciales de la sesión.	No
Descripción Proceso Lógico Asociado		
Este método inicializa el estado del objeto que gestiona la descarga de un documento estructurado DICOM-SR.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
iSetOrderDownload	C_GRID_Order → Objeto que se encarga de gestionar el orden de descarga.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que asigna el orden definido en un componente C_GRID_Order para la descarga de un conjunto de documentos estructurados DICOM-SR.		

Método	Entrada	Salida
strGetIsComplete	Int→Índice que se refiere a un documento estructurado DICOM-SR determinado.	String→Retornará "completo" si el documento estructurado se ha descargado completamente e "incompleto" si no se ha descargado completamente.
Descripción Proceso Lógico Asociado		
Método que indicará si un documento estructurado DICOM-SR representado por el índice de entrada ha sido descargado completamente o no.		

Método	Entrada	Salida
iNextDownloadComplete	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que realiza la descarga de los documentos estructurados en función del orden establecido.		

4.3.2.3. Paquete de Transferencia de Imágenes Médicas DICOM a Repositorios

Los componentes desarrollados en este paquete se encargan de la transferencia de imágenes médicas DICOM, manejadas mediante componentes C_GRID_DICOM de forma individual o conjunta. La transferencia de las imágenes se realizará desde el propio componente hasta los servicios encargados de interactuar con los repositorios DICOM.

La transferencia, al igual que ocurre con las descargas, implica la gestión de las claves de cifrado de datos en caso de que los objetos DICOM a transferir deban ser cifrados. El proceso de creación de las claves y reparto entre diferentes dominios administrativos de la organización virtual y el cifrado de los datos es transparente al usuario, y sólo se realiza si está debidamente autorizado según el modelo de seguridad definido en TRENCADIS en el capítulo 3.

4.3.2.3.1. Diagrama Objetos

La clase C_GRID_Manage_Upload_DICOM alberga toda la funcionalidad de transferencia común a todo objeto DICOM. La clase C_GRID_Manage_Upload_Image es una especialización de C_GRID_Manage_Upload_DICOM para imágenes. La clase C_GRID_Manage_Upload_Set_Images maneja la transferencia de un conjunto de objetos C_GRID_Manage_Upload_Image, teniendo en cuenta las diversas modalidades de transferencia y las prioridades definidas en C_GRID_Order.

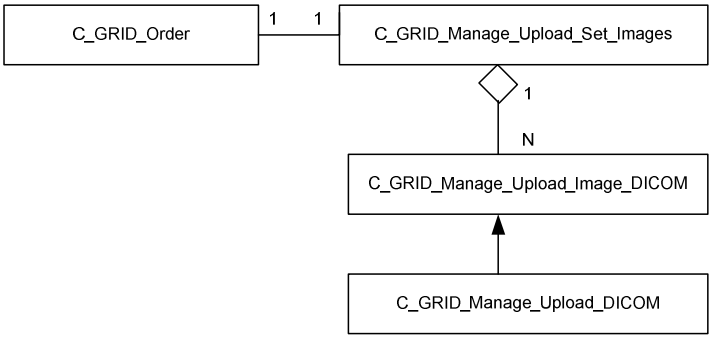

figura 44. Diagrama de Clases del Paquete de Transferencia de Imágenes Médicas DICOM.

4.3.2.3.2. Componente C_GRID_Manage_Upload_image_DICOM

Este componente gestiona las transferencias de los objetos de imágenes médicas DICOM a los repositorios DICOM correspondientes. Las imágenes se manejan a través de componentes C_GRID_Image_DICOM y sólo se ofrece la transferencia en modo de imagen completa. Este

componente gestiona además la creación, reparto y cifrado de los datos en caso necesario, haciendo estas operaciones transparentes al usuario.

Estado

El estado del componente vendrá definido por la imagen médica DICOM transferida desde el componente C_GRID_Image_DICOM hasta el repositorio correspondiente.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Manage_Upload_image_DICOM.

Método	**Entrada**	**Salida**
Constructor	C_GRID_Session→Credenciales de la sesión.	No
Descripción Proceso Lógico Asociado		
Este método inicializa el estado del objeto que gestiona la transferencia de una imagen DICOM hacia el repositorio TRENCADIS correspondiente.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	**Entrada**	**Salida**
iUpload	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que transfiere completamente la imagen DICOM en un solo paso.		

4.3.2.3.3. Componente C_GRID_Manage_Upload_Set_Images

Este componente gestiona la transferencia de un conjunto de imagenes DICOM, representado por componentes C_GRID_Image_DICOM, a los repositorios DICOM correspondientes. De forma análoga al componente anterior, la creación, reparto y cifrado de los datos es transparente al usuario. La transferencia de las imágenes se realizará mediante el orden establecido por el componente C_GRID_Order, aunque, a diferencia de la descarga, solo se soporta una modalidad, en la que en cada paso de la transferencia se envía una imagen completa.

Estado

El estado vendrá definido por el conjunto de imágenes médicas DICOM manejadas mediante los componentes C_GRID_Image_DICOM, y el estado de subida en la que se encuentran.

Métodos

En este apartado se listan los métodos que proporciona el componente Middleware C_GRID_Manage_Upload_Set_Images.

Método	**Entrada**	**Salida**
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método inicializa el estado del objeto que gestiona la transferencia de un conjunto de imágenes médicas DICOM a los repositorios TRENCADIS.		

Método	**Entrada**	**Salida**
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	**Entrada**	**Salida**
iSetOrderUpload	C_GRID_Order →Objeto que gestiona el orden de subida.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que asigna el orden definido en un componente C_GRID_Order para la transferencia de un conjunto de imágenes médicas DICOM.		

Método	**Entrada**	**Salida**
strGetIsComplete	Int→Índice que se refiere a una imagen médica DICOM determinada.	String→Retornará “completo” si la imagen médica se ha subido completamente e “incompleto” si no se ha transferido completamente.
Descripción Proceso Lógico Asociado		
Método que indica si una imagen médica DICOM, representada por el índice de entrada, ha sido transferida completamente o no.		

Método	Entrada	Salida
iNextUploadComplete	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que realiza la transferencia de las imágenes médicas en función del orden establecido.		

4.3.2.4. Paquete de Transferencia de Documentos Estructurados DICOM-SR

Los componentes desarrollados en este paquete, se encargan de la transferencia de documentos estructurados DICOM-SR a los repositorios DICOM correspondientes, gestionados de forma individual o conjunta mediante componentes de tipo C_GRID_SR_DICOM. Al igual que ocurre con el paquete de transferencia de imágenes médicas DICOM, es posible definir prioridades y se encarga también de la gestión de las claves de cifrado de una manera transparente para el usuario.

4.3.2.4.1. Diagrama Objetos

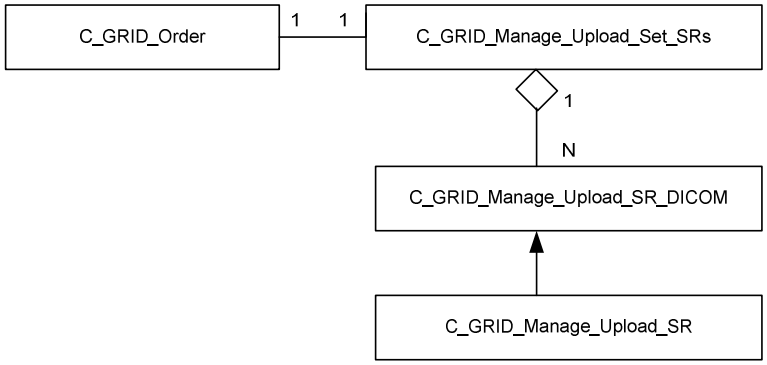

figura 45. Diagrama de Clases del Paquete de Transferencia de Documentos Estructurados DICOM-SR.

Las clases que se han definido son equivalentes a las comentadas en el paquete de transferencia de imágenes médicas DICOM, pero adaptadas a los documentos estructurados DICOM-SR.

4.3.2.4.2. Componente C_GRID_Manage_Upload_SR

Este componente transfiere los objetos que contienen documentos estructurados DICOM-SR encapsulados en componentes C_GRID_SR_DICOM. La funcionalidad es prácticamente la misma que para el C_GRID_Manage_Upload_DICOM, pero para documentos estructurados DICOM-SR.

Estado

El estado del componente vendrá definido por el documento estructurado DICOM-SR transferido desde el origen C_GRID_SR_DICOM hasta el servidor correspondiente.

Métodos

En este apartado se listan los métodos que proporciona el componente middleware C_GRID_Manage_Upload_SR.

Método	Entrada	Salida
Constructor	C_GRID_Session→Credenciales de la sesión.	No
Descripción Proceso Lógico Asociado		
Este método se encarga de inicializar el estado del objeto encargado de gestionar la subida de un documento estructurado DICOM-SR.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
iUpload	No	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que transfiere completamente el documento estructurado DICOM-SR en un solo paso.		

4.3.2.4.3. Componente C_GRID_Manage_Upload_Set_SRs

Como en el caso del componente C_GRID_Manage_Upload_Set_Images, este componente se encarga de la transferencia de un conjunto de datos, aunque en este caso se transfieren documentos estructurados DICOM-SR.

Estado

El estado vendrá definido por el conjunto de documentos estructurados DICOM-SR manejados, mediante los componentes C_GRID_SR_DICOM, y el estado de transferencia en que se encuentren.

Métodos

Método	Entrada	Salida
Constructor	No	No
Descripción Proceso Lógico Asociado		
Este método inicializa el estado del objeto encargado de gestionar la transferencia de un conjuntos de documentos estructurados DICOM-SR.		

Método	Entrada	Salida
strGetError	No	String → Descripción del último error producido.
Descripción Proceso Lógico Asociado		
Método que se encarga de retornar la descripción del último error producido.		

Método	Entrada	Salida
iSetOrderUpload	C_GRID_Order →Objeto que gestiona el orden de subida.	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que asigna el orden definido en un componente C_GRID_Order para la subida de un conjunto de documentos estructurados DICOM-SR.		

Método	Entrada	Salida
strGetIsComplete	Int→Índice que se refiere a un documento estructurado DICOM-SR determinado.	String→Retornará "completo" si la imagen médica se ha subido completamente e "incompleto" si no se ha transferido completamente.
Descripción Proceso Lógico Asociado		
Método que indica si un documento estructurado DICOM-SR, representado por el índice de entrada, ha sido transferido completamente o no.		

Método	Entrada	Salida
iNextUploadComplete	NO	Int→Código de Error. Un valor distinto de cero indica la aparición de un error.
Descripción Proceso Lógico Asociado		
Método que realiza la transferencia de los documentos estructurados en función del orden establecido.		

CAPÍTULO V. Aplicaciones

Tal y como se ha descrito en el apartado 2.1.5, referente a los objetivos, el último objetivo planteado, es la creación de aplicaciones construidas sobre los componentes de alto nivel definidos en la arquitectura TRENCADIS, y ofrecidos mediante los paquetes descritos en capítulos anteriores. La meta de estas aplicaciones es demostrar que la arquitectura TRENCADIS es capaz de dar la funcionalidad descrita en el apartado 2.1.3, referente a los objetivos, y que los componentes de alto nivel desarrollados permiten desarrollar de forma eficiente aplicaciones para la gestión de proceso de imágenes e informes estructurados almacenados de forma distribuida, además de demostrar que estos componentes son reutilizables.

Para este fin, se han desarrollado tres aplicaciones, que se describen en los siguientes puntos:

- **Aplicación para la creación de almacenes virtuales.** Esta aplicación permite acceder, mediante un interfaz desarrollado en Java, a imágenes DICOM distribuidas entre diferentes centros, creando almacenes virtuales que gestionan todos los almacenes fuentes como un único almacén, además de permitir el acceso únicamente a la información DICOM definida en las ontologías para la cual el usuario esté autorizado.
- **Aplicación para la descarga de imágenes.** Esta aplicación también permite acceder y visualizar imágenes DICOM e informes estructurados DICOM-SR, distribuidas en diferentes almacenes, mediante un interfaz desarrollado en Java. También permite crear almacenes virtuales y descargar las imágenes e informes en diferentes modalidades y prioridades.
- **Aplicación para el apoyo a la decisión clínica en informes radiológicos.** Esta aplicación ofrece todas las funcionalidades descritas en las dos aplicaciones anteriores, a través de un interfaz web, de manera que con un navegador se puede acceder y crear almacenes virtuales. Por otra parte, esta aplicación permite crear informes estructurados DICOM-SR a partir de plantillas predefinidas y transferirlas a los almacenes distribuidos, permitiendo utilizarlas para organizar semánticamente los contenidos.

Para cada una de estas aplicaciones se realiza, en los siguientes apartados, una descripción general de cada aplicación, los objetivos que se pretenden alcanzar, los elementos de la arquitectura TRENCADIS involucrados y la relación entre éstos, además de la infraestructura necesaria y los componentes middleware utilizados. Finalmente, se describe la funcionalidad real que ofrece cada aplicación de forma detallada mostrando los interfaces desarrollados y unos resultados experimentales que han sido realizados en entornos controlados.

5.1. Aplicación Para la Creación de Almacenes Virtuales

La aplicación que se describe en este apartado, constituye una herramienta básica de testeo y validación de algunos componentes middleware desarrollados en la arquitectura TRENCADIS, demostrando su productividad a la hora de crear aplicaciones y su reusabilidad para aplicaciones de diferente ámbito.

La aplicación ha sido desarrollada en Java. En ella se pueden crear y eliminar almacenes virtuales de imágenes médicas, utilizando diferentes ontologías que definirán las partes de información que se pueden utilizar. Todo almacén virtual que se cree en la aplicación, contiene implícitamente todos los repositorios de información DICOM desplegados como servicios Grid en la arquitectura TRENCADIS, y los gestiona como un único almacén a la hora de realizar las operaciones de búsqueda, indexación, almacenamiento y recuperación.

5.1.1. Objetivos

Los objetivos que se pretenden alcanzar en esta aplicación son:

- Crear una aplicación para el manejo de estudios de imágenes DICOM distribuidos en diferentes repositorios, es decir, manejar un conjunto de repositorios fuentes como si fueran un único almacén virtual.
- Definir los contenidos del almacén virtual a partir de los contenidos que especifique una determinada ontología.
- Demostrar que los componentes middleware aumentan la productividad en el desarrollo de aplicaciones, al separar la parte grid de la parte de desarrollo de la aplicación.
- Demostrar la reusabilidad de los componentes a la hora de crear aplicaciones en diferentes ámbitos.

5.1.2. Infraestructura Desplegada

Para el desarrollo de la aplicación se han desplegado en la capa de infraestructura (capa Core Middleware y capa Servicios Servidor) diferentes servicios Grid, necesarios para poder poner en marcha la aplicación. Todos los servicios se han desplegado en una intranet local:

- **Cuatro servicios Grid Storage DICOM.** Se han desplegado cuatro servicios de almacenamiento, implementados de manera que los dispositivos con los que interactúan (backend), en los cuatro casos, son manejadores de disco, donde se albergan las imágenes DICOM. Se utilizará un equipo para cada servicio Grid
- **Servicio Grid Storage Broker**. Este servicio se utiliza para la indexación de los campos "Creation" de las ontologías. Se utilizará un equipo diferente al de los servicios Grid Storage DICOM por razones de productividad.
- **Servidor VOMS.** Este servicio guarda los grupos de trabajo a los cuales pertenecerán los usuarios Grid. Se utilizará solo un equipo para este servicio.
- **Servicio Grid Servidor de Ontologías**. Este servicio guarda toda la información sobre las ontologías y la relación entre las ontologías y grupos de trabajo de los usuarios. Se albergará en el mismo equipo que el servicio Grid Storage Broker.

- **Servicios Grid IDXSRV.** Se instalarán cuatro servicios, uno por cada servidor donde se alberguen los servicios Grid Storage DICOM, otro para el servidor donde se albergue el servicio Grid Storage Broker y el servicio Grid Servidor de Ontologías. Finalmente, se desplegará el IDXSRV central del Sistema de monitorización e información MDS4-Extended.

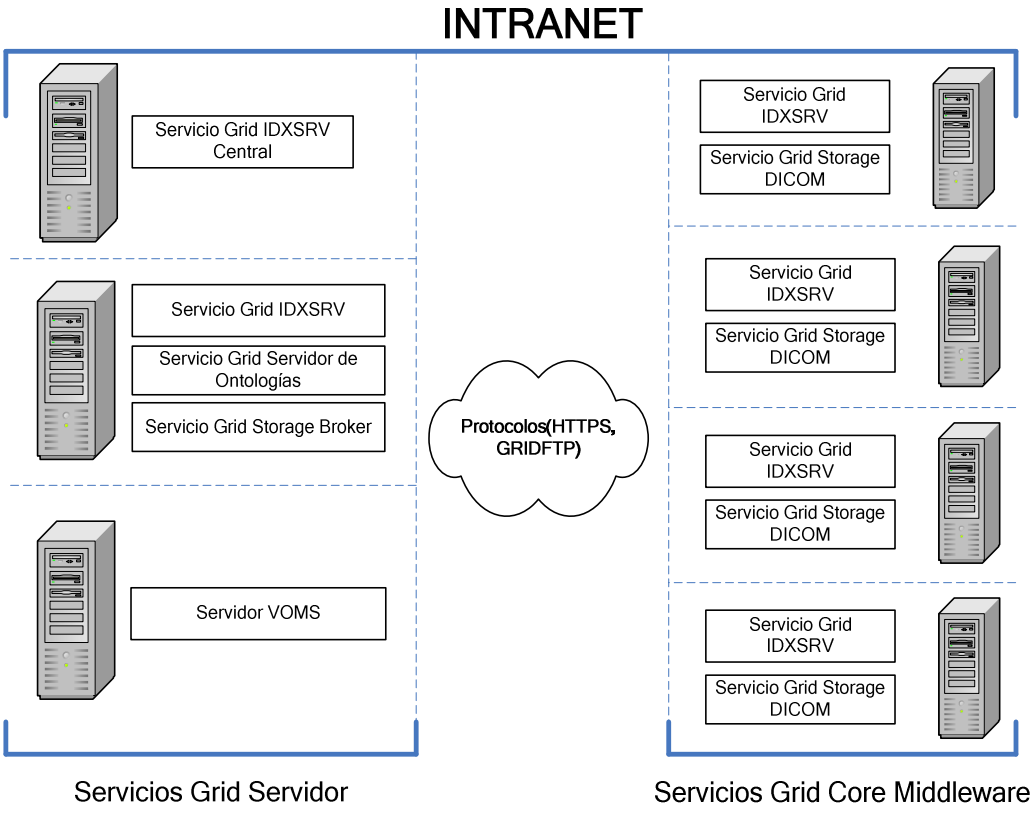

figura 46. Esquema general del despliegue de la infraestructura.

5.1.3. Componentes Middleware

Los diferentes componentes middleware utilizados, para la creación de esta aplicación, son los que se ubican en los siguientes paquetes middleware:

- **Paquete Gestor de Sesiones.** De este paquete se utilizará el componente C_GRID_Session, el cual se encarga de gestionar las credenciales del usuario en el momento que arranca la aplicación y los privilegios necesarios para acceder a los servicios de la infraestructura. Solo podrá existir una sesión abierta en la aplicación. (ver figura 47).
- **Paquete Ontologías.** De este paquete se utilizará el componente C_GRID_Ontology, el cual realizará las consultas de las imágenes médicas según la ontología seleccionada. Habrá una ontología por cada almacén virtual creado. (ver figura 47). También se utilizará el componente C_GRID_OntologyServer para consultar las ontologías existentes.
- **Paquete DICOM-I.** Este paquete se utilizará para manejar imágenes digitales en formato DICOM a través de los componentes C_GRID_Image_DICOM (ver figura 47). También se utilizan de este paquete el componente C_GRID_Set_Images_DICOMs para el manejo de un conjunto de imágenes, aunque este componente no se vea reflejado en la figura 47.

- **Paquete para la creación de Almacenes Virtuales de Objetos DICOM.** De este paquete se utilizará el componente C_GRID_StorageDICOM (ver figura 47) creando tantas instancias como almacenes virtuales.

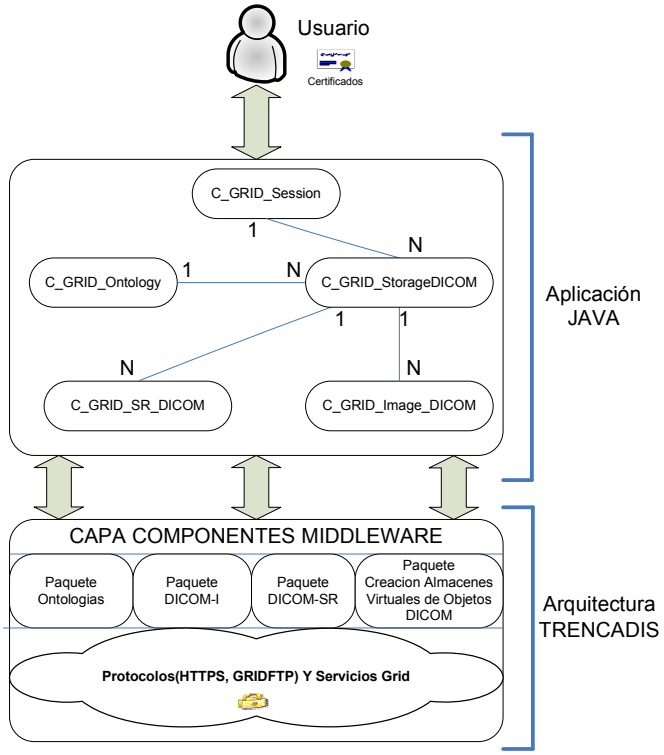

figura 47. Esquema general de la aplicación para la creación de almacenes virtuales.

5.1.4. Funcionalidad

La aplicación implementa las siguientes cuatro funcionalidades principales.

- **Iniciar Sesión.** Dado un fichero de configuración, donde se indica la ubicación de los certificados x509 del usuario y el servidor VOMS, y un password, esta funcionalidad permite iniciar una sesión para acceder a los diferentes recursos de la infraestructura a través de los componentes middleware. En esta operación se crea una instancia del componente C_GRID_Session.

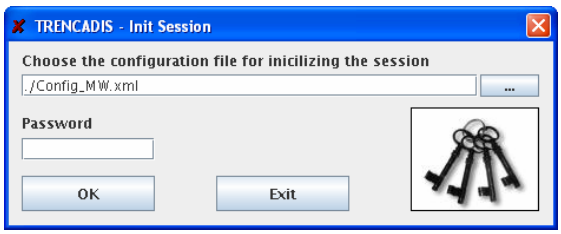

figura 48. Pantalla de Inicio de Sesión.

- **Seleccionar Ontologías.** Esta funcionalidad se encarga de crear una instancia de C_GRID_OntologyServer, para mostrar por pantalla las ontologías a través de las cuales se puede interactuar. El usuario selecciona la ontología activa entre las ontologías ofrecidas y crea una instancia de C_GRID_Ontology, que se va a utilizar tanto para el manejo de imágenes DICOM como para las consultas en los repositorios DICOM.

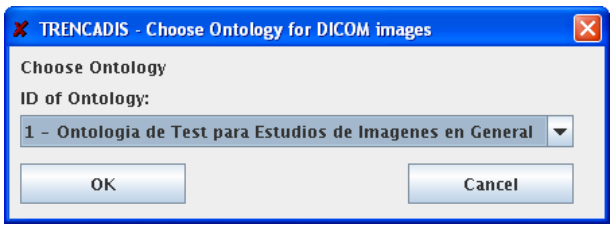

figura 49. Pantalla de selección de ontología para imágenes DICOM.

- **Crear/Borrar Almacenes Virtuales.** Esta funcionalidad permite al usuario crear almacenes virtuales en función de la ontología seleccionada. La ontología permite a partir de los campos "Creation" definidos, especificar los estudios accesibles y relevantes a la ontología. Para cada almacén virtual se genera una instancia C_GRID_StorageDICOM. En la figura 50 se muestra un almacén virtual compuesto por cinco estudios.

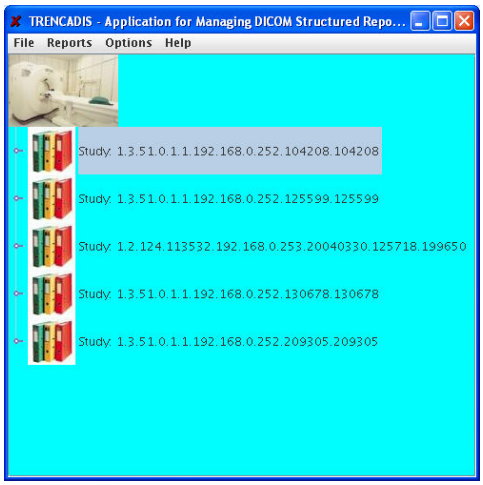

figura 50. Pantalla de creación de Almacenes Virtuales sobre imágenes médicas.

- **Realizar Búsquedas sobre Almacenes Virtuales.** Esta funcionalidad permite las búsquedas sobre los almacenes virtuales creados. Las búsquedas se realizan sobre los campos especificados en las ontologías como campos "Filter". El resultado de las búsquedas genera instancias del componente C_GRID_Set_Images_DICOMs, que contienen el conjunto de imágenes DICOM resultado de la búsqueda. La figura 51 muestra cómo se realiza una búsqueda sobre un almacén virtual a través del campo "FINDING" ("Hallazgo"), especificado previamente en la ontología seleccionada.

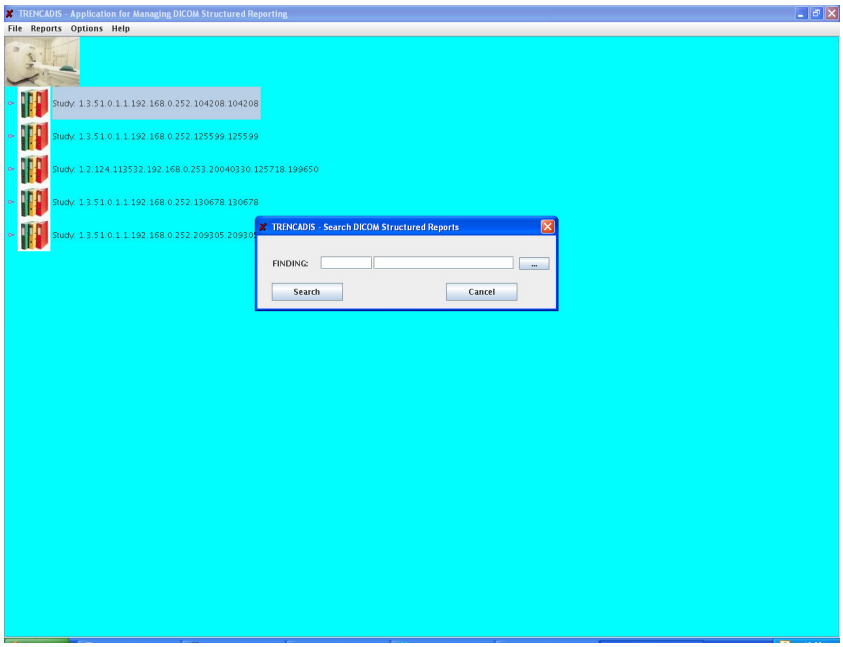

figura 51. Pantalla de Búsqueda sobre Almacenes Virtuales.

5.1.5. Resultados experimentales

Se ha realizado un experimento concreto sobre esta aplicación, cuyo principal objetivo es la evaluación de los componentes TRENCADIS de creación de almacenes virtuales y las operaciones de búsqueda de objetos DICOM sobre estos mismos componentes. Para ello, se ha utilizado el despliegue descrito anteriormente. Las características de los cuatro servicios Grid Storage DICOM y la información que contienen estos vienen reflejadas en la tabla 2.

Nombre	Arquitectura de la Máquina	Número de Imágenes	Número de imágenes por modalidad
SEKER	4 Procesadores Intel Xeon 2.0 Ghz 4 GB RAM	2154	OT → 1820 | CT → 334
AKER	2 Procesadores Pentium III 1 Ghz 512 MB RAM	933	CT → 558 | MR → 375
KEFREN	Biprocesador. Intel Xeon 2.0 Ghz. 1 GB RAM	54	CT → 42 | CR → 12
OSIRIS	AMD 1800 Mhz 512 MB RAM	67	OT → 67

tabla 2. Tabla resumen de las características de los servicios Grid Storage DICOM para la realización del experimento.

Se han utilizado más de 3000 imágenes, que se han distribuido entre los diferentes servicios Storage DICOM, tal y como se describe en la tabla 2. Para la realización del experimento, se ha empleado una red local de 100 Mbps TCP/IP y dos usuarios simultáneos.

El primer test realiza la creación de almacenes virtuales, posteriormente, el segundo test ejecuta las operaciones de búsqueda sobre los almacenes creados.

Los tiempos que se emplean en el sistema para la creación de almacenes virtuales se muestran en la tabla 3.

Nombre del Almacén Virtual	Número Imágenes	Tiempo Creación (segundos)	Descripción	Resources
ALL1	3208	5.4	Todas las imágenes.	4
ALL2		0.4		
CT1	934	5.0	Imágenes de tipo CT.	3
CT2		0.3		
OT1	1887	5.1	Imágenes de tipo OT.	2
OT2		0.2		

tabla 3. Tiempos de creación de almacenes virtuales.

En la tabla 3, se puede observar que la creación de un segundo almacén virtual requiere menos tiempo. El motivo de este comportamiento es debido al overhead introducido por la conexión con el servicio Grid Storage Broker, el cual sólo afecta en la primera conexión, que es cuando se produce el negociado.

En la tabla 4, se muestra el tiempo que utilizan los componentes que manejan los almacenes virtuales para las búsquedas, las cuales utilizan diferentes criterios. La información mostrada en esta tabla muestra el almacén virtual utilizado, el número de imágenes involucradas y el tiempo empleado.

Los tiempos de búsqueda, tal y como se observan en la tabla 4, sólo dependen de la base de datos o backend donde se guarde la información DICOM, el tamaño de los resultados de la búsqueda y del ancho de banda de la red. Todos los recursos desplegados en el Grid para el experimento tienen el mismo manejador de base de datos, tal y como se ha descrito en el apartado 5.1.2. Obviamente, cuanto mayor sea el volumen de imágenes involucradas en el resultado, los tiempos de búsqueda serán mayores.

Descripción de la búsqueda	Nombre del Almacén Virtual	Imágenes	Tiempos de Respuesta (segundos)
Todas las imágenes	ALL	3208	30.0
Solo imágenes CT	ALL	934	11.2
Solo imágenes OT	ALL	1887	15.6
Solo imágenes MR	ALL	375	3.0
Solo imágenes CR	ALL	12	0.2
Todas las imágenes	CT	930	11.1
Todas las imágenes	OT	1887	18.4
Todas las imágenes	MR	375	5.9
Todas las imágenes	CR	12	0.6
StudyUID= 1.3.12.2.1107.5….	ALL	24	1.1
StudyUID= 1.3.12.2.1107.5….	CT	24	1.1
StudyUID= 1.2.756.9999….	OT	62	1.8
StudyUID= 1.3.46.670589…	MR	24	1.1
StudyUID=1.3.46.670589…	CR	2	0.3

tabla 4. Tiempos de búsqueda sobre los almacenes virtuales.

5.2. Aplicación Para la Descarga de Imágenes DICOM

La aplicación que se describe en este apartado, al igual que ocurría en la aplicación para la creación de almacenes virtuales, es una aplicación de testeo y validación de componentes middleware de la arquitectura TRENCADIS.

La aplicación ha sido desarrollada en Java. En líneas generales, esta aplicación extiende la funcionalidad de la aplicación para la creación de almacenes virtuales, añadiendo la funcionalidad de descarga de imágenes médicas que las componentes middleware ofrecen.

5.2.1. Objetivos

Los objetivos principales que se plantean en esta aplicación son:

- Crear una aplicación para la descarga de estudios de imágenes DICOM, preparando el conjunto de imágenes seleccionadas en el almacén virtual para su posterior descarga.
- Utilizar y probar las diferentes modalidades de descarga existentes en los componentes.
- Demostrar que los componentes middleware aumentan la productividad en el desarrollo de aplicaciones, al separar la parte Grid de la parte de desarrollo de la aplicación.
- Demostrar la reusabilidad de los componentes a la hora de crear aplicaciones en diferentes ámbitos.

5.2.2. Infraestructura Desplegada

Para el desarrollo de la aplicación se ha desplegado la misma infraestructura que en la aplicación anterior (cuatro servicios Grid Storage DICOM, un servicio Grid Storage Broker, un Servidor Grid Servidor de Ontologías, un servidor VOMS y tres servicios IDXSRV). Además de estos servicios, se ha desplegado servicios para el cifrado y descifrado de la información. Estos servicios son:

- **Cinco servicios Grid servidores de Claves.** Estos servicios se utilizan para almacenar las partes de las claves que se generan al cifrar un objeto. Cada servicio estará certificado por una CA diferente, para reflejar dominios administrativos diferentes.
- **Un servicio Grid generador de claves EOUID.** Este servicio asignará a cada objeto cifrado un identificador único.
- **Servicios Grid IDXSRV.** Se instalará uno para cada servicio Grid servidor de Claves, y otro para el generador de claves EOUID.

Al igual que ocurría con la aplicación de creación de almacenes virtuales, la arquitectura se ha desplegado en una Intranet, en la que el reparto de las maquinas es la siguiente:

- Un equipo donde se instalará el servidor VOMS.
- Cuatro equipos, los cuales albergarán cada uno un servicio Grid Storage DICOM.
- Un equipo que alberga los servicios Grid Storage Broker, servicio Grid servidor de ontologías y servicio Grid generador de claves EOUID.
- Cinco equipos, los cuales albergan cada uno un servicio Grid servidor de claves.
- Un equipo para el servicio IDXSRV central del MDS4-Extended.

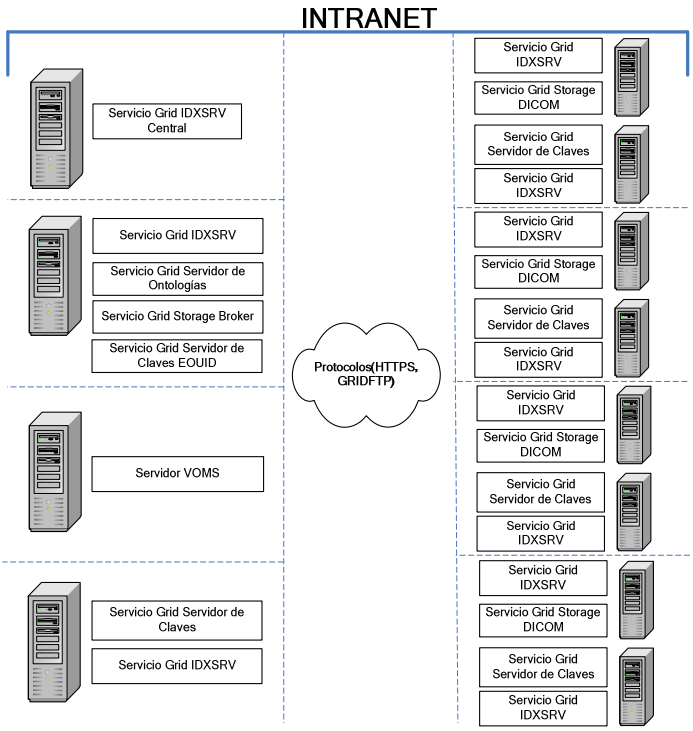

figura 52. Esquema general del despliegue de la infraestructura.

5.2.3. Componentes Middleware

Además de los componentes middleware utilizados para la aplicación de creación de almacenes virtuales, se utilizarán los componentes del paquete de descarga de imágenes médicas DICOM (ver figura 53). Concretamente se utilizarán instancias del componente C_GRID_Order (ver figura 53) para configurar el orden de descarga de las imágenes médicas seleccionadas. Las descargas se realizarán mediante instancias del componente C_GRID_Manage_Download_Set_Images (ver figura 53).

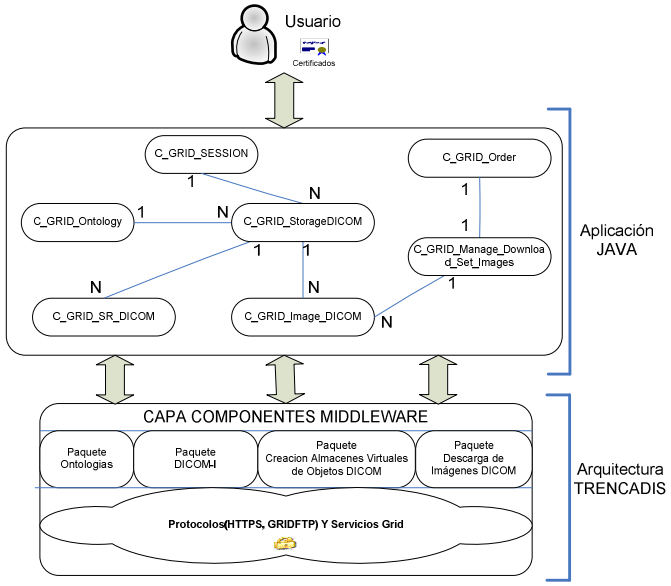

figura 53. Esquema general de la aplicación la descarga de imágenes DICOM.

5.2.4. Funcionalidad

Además de las funciones que implementa la aplicación y que son comunes a las presentadas en la aplicación de creación de almacenes virtuales (iniciar sesión y creación de almacenes virtuales), se realizan las siguientes operaciones:

- **Selección de un estudio para su descarga.** Esta funcionalidad se encarga de seleccionar aquellos estudios que se pretende descargar desde un almacén virtual DICOM. Se crea una instancia del componente C_GRID_Manage_Download_Set_Images, que gestionará la descarga, y una instancia del componente C_GRID_Order para establecer el orden por prioridades de la descarga.

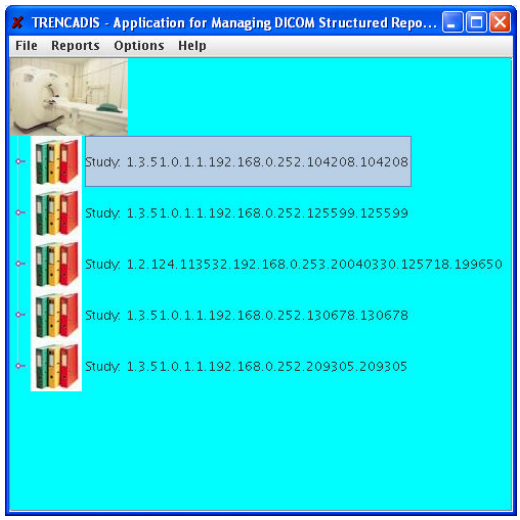

figura 54. Pantalla de Selección de descarga.

- **Configurar Prioridades de descarga.** Esta funcionalidad permite configurar por niveles las prioridades de descarga de las imágenes seleccionadas, modificando la instancia de C_GRID_Order creada. En la figura 55 se muestra, en la parte izquierda, las prioridades de descarga y, en la parte derecha, la visualización de las imágenes descargadas total o parcialmente.

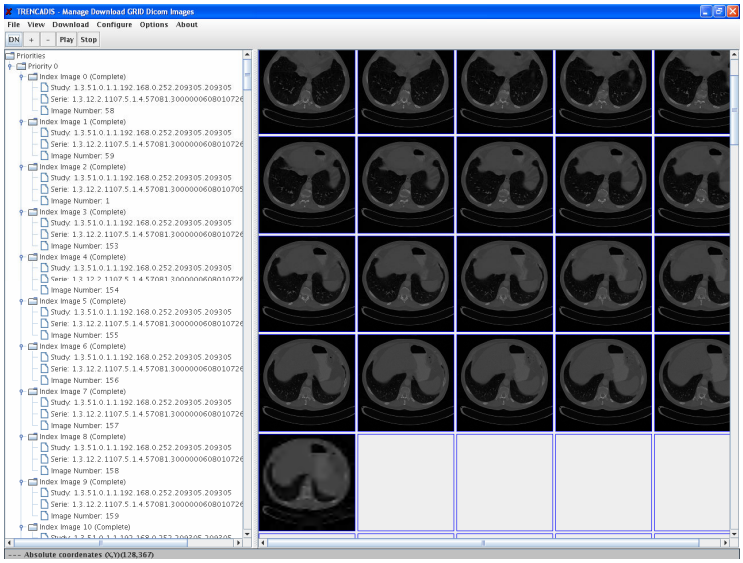

figura 55. Pantalla de Configuración de Prioridades de descarga.

- **Seleccionar las modalidades de Descarga.** Esta funcionalidad permite configurar las diferentes modalidades de descarga que permite el componente GRID_Manage_Download_Set_Images, que son: a) Descarga Imagen a Imagen; b) Descarga Imagen por capas y c) Descarga por capas.

5.2.5. Resultados experimentales

Se han realizado dos experimentos concretos sobre esta aplicación, cuyo principal objetivo es el chequeo de los componentes TRENCADIS de transferencia de estudios DICOM y la incidencia de las operaciones de cifrado y descifrado, en caso de que se utilicen.

Para el primer experimento se han utilizado los mismos Servicios Grid Storage DICOM que en la aplicación anterior. La figura 55 muestra la parte de la aplicación que corresponde con el manejo de descargas de estudios de imágenes DICOM. El usuario, primero crea los almacenes virtuales y filtra la información sobre los estudios que se desea descargar, posteriormente específica el orden de descarga estableciendo prioridades.

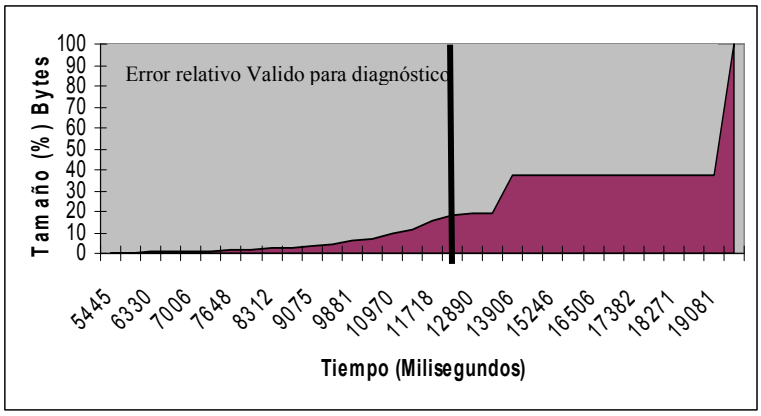

figura 56. Evolución de la descarga de una imagen con transmisión progresiva. La línea vertical muestra el instante en el cual el error relativo está por debajo del 2%.

La figura 56 muestra la evolución de tiempo y tamaño en una descarga progresiva de una imagen CT 512 x 512 x 12 Bits. La línea vertical marca el tiempo en el cual el error relativo está por debajo del 2%, siendo este el umbral definido por los expertos que participaron en el experimento. El tiempo de transferencia es mayor si se utiliza la transferencia progresiva, debido a las operaciones de compresión en JPEG2000 de las imágenes y las sucesivas descompresiones. Sin embargo, el tiempo de espera improductivo, se reduce desde el momento en que la imagen empieza a presentarse al usuario, al disponer éste de información con la que ya poder trabajar.

Con respecto al experimento concreto para medir la incidencia del cifrado y descifrado de los objetos DICOM, se han utilizado un conjunto de imágenes DICOM, concretamente cuatro imágenes de cuatro estudios diferentes, con diferentes tamaños (ver tabla 5).

Id.	Identificador del Estudio	Imágen Id.	Tamaño de la Imágen (MB)
1	1.2.840.10008.5.1.4.1.1.4	1	0.5
2	1.2.840.10008.5.1.4.1.1.7	1	2.5
3	1.2.840.10008.5.1.4.1.1.1	1	5.8
4	1.2.840.10008.5.1.4.1.1.3	1	7.7

tabla 5. Imágenes de los cuatro estudios de radiología utilizados para el experimento de cifrado y descifrado.

Todas las imágenes han sido cifradas y guardadas en un Servicio Grid Storage DICOM. La imagen original, sin cifrar, también ha sido guardada en el mismo servicio Grid. De esta manera se puede medir las diferencias entre los procesos que tratan con objetos cifrados y los procesos que manejan los objetos no cifrados.

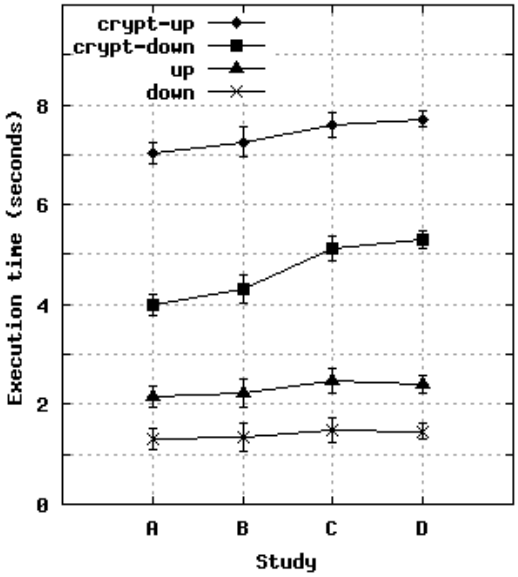

figura 57. Gráfica que muestra el tiempo de ejecución para el conjunto de los cuatro estudios.

En la figura 5, se muestran cuatro series de los resultados experimentales. Por una parte “crypt-up” y “crypt-down” muestran el tiempo utilizado para subir y descargar objetos DICOM, incluyendo los tiempos el cifrado y descifrado de éstos, y el manejo de las claves compartidas. Por otra parte, “up” and “down” muestra el tiempo utilizado para subir y descargar objetos sin cifrar. Cada punto en la gráfica, representa el promedio medido en segundos, utilizado para la subida o descarga, medido por el reloj del cliente.

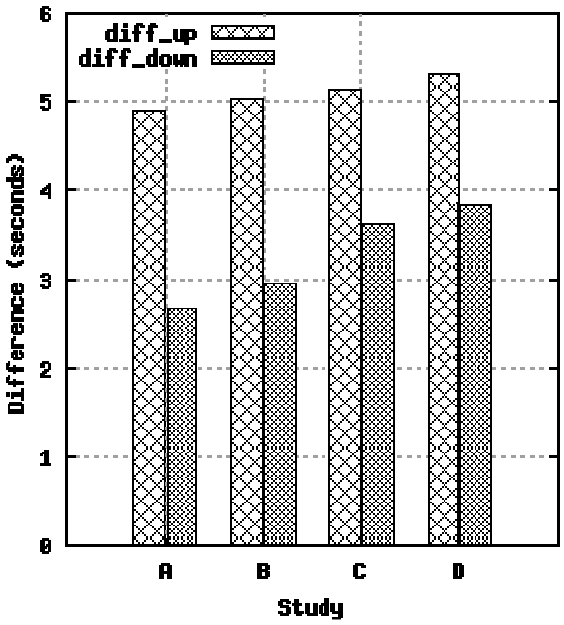

figura 58. Diferencias de tiempo entre la subida y descarga de los objetos utilizando y sin utilizar las operaciones de cifrado y descifrado.

La figura 58 muestra las diferencias de tiempo D, calculadas como:

D = *tiempo con cifrado – tiempo sin cifrado*

La importancia de esta gráfica, es mostrar que incluso cuando la diferencia entre la subida y descarga de objetos DICOM utilizando o no el cifrado, tiende a ser más grande para grandes objetos, además es posible estimar un nivel de prestaciones en un determinado intervalo. Por ejemplo, los resultados de este estudio muestran que para estos objetos en el intervalo de 0,5 a 7,7 MB, es posible anticipar una subida de 4 segundos adicionales para la descarga de un objeto si utilizamos el cifrado de la información y la comparamos sin el uso del cifrado. Esta característica es igual en todas las imágenes utilizadas, demostrando que la mejora en la seguridad introducida por el modelo de cifrado introduce un overhead que puede ser aceptado incluso en un uso interactivo.

5.3. Aplicación de Apoyo a la Decisión Clínica basado en Informes Radiológicos

A diferencia de las aplicaciones descritas en los apartados anteriores, esta aplicación pretende ser una aplicación en producción dentro del ámbito colaborativo entre un conjunto de hospitales. La aplicación se enmarca dentro del Proyecto de Ciberinfraestructura Valenciana de Imagen Médica Oncológica (CVIMO), financiado por la "Consellería d'Empresa, Universitat i Ciencia de la Generalitat Valenciana" (GVEMP06/004).

La herramienta ofrece un interfaz que de soporte a la gestión de un conjunto de estudios de imágenes médicas en formato DICOM, distribuidas entre varios repositorios DICOM ubicados en diferentes hospitales. Los estudios calificados como relevantes o de interés, son seleccionados y almacenados en los repositorios para que puedan ser utilizados a posteriori en la formación de nuevos radiólogos, o como soporte para la creación de nuevos informes o de toma de decisiones.

Las principales características que ofrece la aplicación son:

- **Interfaz Web.** Mediante el uso de un navegador Web, los usuarios pueden interactuar con todas las funcionalidades que ofrece el sistema.
- **Relevancia de los estudios.** El área en el que trabaja un usuario debe corresponder con un área médica o grupo de investigación determinado. Esta selección se realizará mediante las capas correspondientes de las ontologías anteriores.
- **Capacidad de gestión de los estudios DICOM.** Los estudios se agrupan entre aquellos que ya han sido informados y aquellos que están pendientes de informar. Se considera que un estudio ha sido informado cuando tiene un informe estructurado asociado.
- **Creación de informes estructurados.** El entorno permite crear informes estructurados. Dichos informes se generan a partir de su almacenamiento en plantillas previamente definidas. El objetivo de estas plantillas es formalizar la estructura de dichos informes y las codificaciones a utilizar, así como las relaciones semánticas de sus contenidos. Los informes estructurados que se generen deberán seguir el estándar DICOM, por lo que todos los informes deben ser codificados en DICOM-SR y guardados en los mismos almacenes en donde se encuentre el estudio DICOM al que referencien.
- **Búsquedas de informes estructurados generados.** La aplicación proporciona un interfaz para buscar y posteriormente descargar informes DICOM-SR generados. Estas búsquedas se realiza a partir de los criterios definidos por la ontología activa.

5.3.1. Objetivo

Los objetivos principales que se plantean en esta aplicación son:

- Crear una aplicación para el manejo de estudios DICOM y la generación de informes estructurados, que sigan el estándar DICOM utilizando a unas plantillas previamente definidas.

- Crear una herramienta de apoyo para la toma de decisiones en la generación de nuevos informes. La aplicación debe permitir el acceso a las bases de datos de los informes estructurados DICOM-SR ya generados, de manera que puedan ser consultados.
- Demostrar que la arquitectura TRENCADIS es una herramienta adecuada para facilitar la colaboración entre diferentes hospitales a la hora de compartir recursos.
- Al igual que las aplicaciones ya descritas, demostrar la reutilización de los componentes Middleware TRENCADIS en aplicaciones de diferentes ámbitos, y demostrar que los componentes Middleware aumentan la productividad en el desarrollo de aplicaciones.

5.3.2. Infraestructura Desplegada

Para el desarrollo de esta aplicación, se han desplegado todos los servicios Grid desarrollados en la tesis. Concretamente los servicios desplegados son:

- **Cuatro Servicios Grid Storage DICOM** implementados de manera que los dispositivos con los que interactúa son manejadores de disco donde se almacenan tanto las imágenes médicas como los documentos estructurados asociados.
- **Un servicio Grid Storage Broker** para la indexación de los campos "Creation" de las ontologías de los datos, almacenados sobre las imágenes y documentos estructurados en los servicios Grid Storage DICOM.
- **Un Servidor VOMS** para guardar los grupos de trabajo a los que pertenecen los usuarios Grid.
- **Un servicio Grid Servidor de Ontologías** que almacena toda la información sobre las ontologías y su relación con los grupos de trabajo.
- **Un servicio Grid Servidor de Códigos** que contiene las codificaciones que se utilizarán para los valores de los campos de los informes estructurados DICOM-SR, a partir de las plantillas predefinidas.
- **Cinco servicios Grid Servidores de Claves** que se utiliza para almacenar las partes de las claves que se generen al cifrar un objeto. Cada servicio está certificado por una CA diferente para implementar los diferentes dominios administrativos.
- **Un servicio Grid generador de claves EOUID** que asigna a cada objeto cifrado un identificador único.
- **Servidores IDXSRV** que instala en cada servidor que albergue algún servicio Grid.
- **Un servidor Web** donde se aloja la aplicación.

Al igual que con la aplicación de creación de almacenes virtuales, la arquitectura se ha desplegado en una Intranet, en la que el reparto de las maquinas es el siguiente:

- Un equipo donde se instalará el servidor VOMS.
- Cuatro equipos, cada uno con un Servicio Grid Storage DICOM.
- Un equipo que aloja el servicio Grid Storage Broker, el servicio Grid Servidor de Ontologías, el servicio Grid generador de claves EOUID y el servicio Grid Servidor de Códigos.
- Cinco equipos, cada uno de ellos con un servicio Grid Servidor de Claves.

- Un equipo para el Servicio IDXSRV central del MDS4-Extended.
- Un equipo que alojará el servidor web donde estará instalada la aplicación.

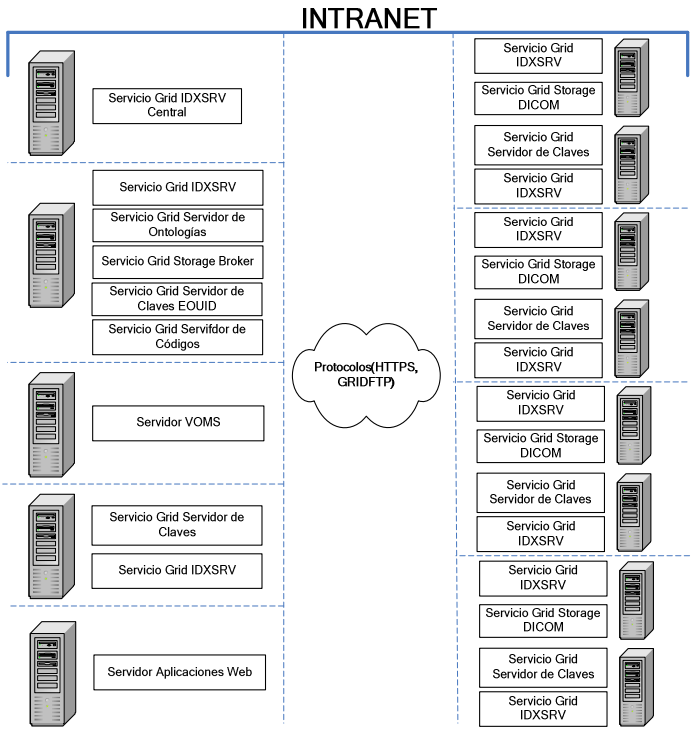

figura 59. Esquema general del despliegue de la infraestructura.

5.3.3. Componentes Middleware

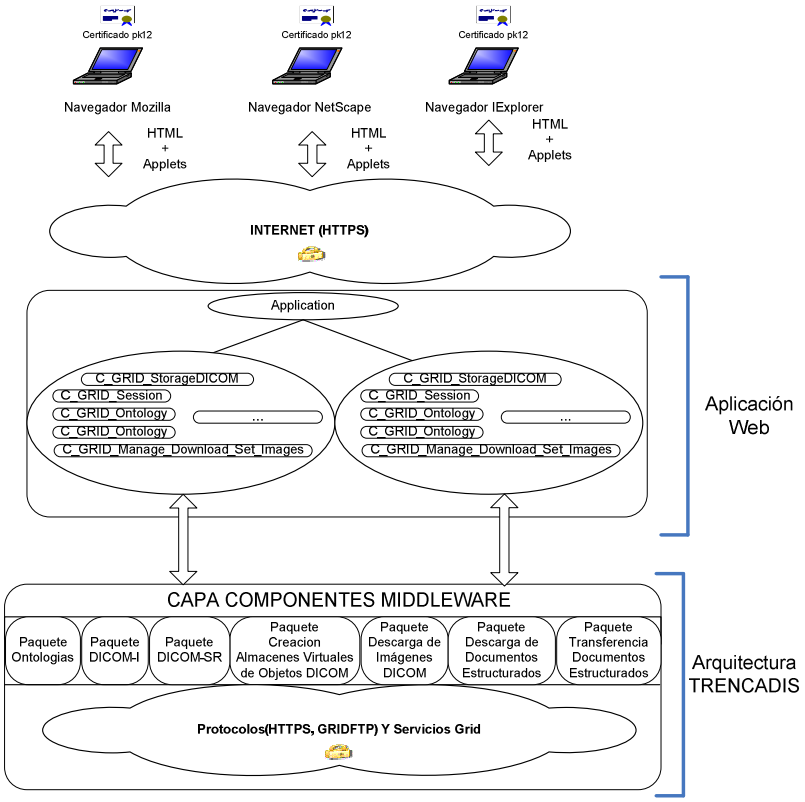

figura 60. Esquema General de la Aplicación de Apoyo a la Decisión Clínica basado en Informes Radiológicos.

En esta aplicación se han utilizado prácticamente la totalidad de los componentes Middleware desarrollados en la tesis. Los componentes de cada paquete utilizados son:

- **Paquete Gestor Sesiones.** Como en el caso de las aplicaciones anteriores, de este paquete se utilizará el componente C_GRID_Session. La diferencia fundamental con respecto a las aplicaciones anteriores es la creación de una instancia por cada sesión web que se abra en la aplicación.
- **Paquete Ontologías.** Como en las aplicaciones anteriores, se utilizan instancias de C_GRID_OntologyServer para consultar las ontologías existentes y C_GRID_Ontology para manejar una ontología determinada.
- **Paquete Gestión de Documentos Estructurados DICOM-SR.** Este paquete se utiliza para la creación y modificación de documentos estructurados DICOM-SR y la consulta de su contenido. Para ello se creará una instancia de C_GRID_SR_DICOM por cada DICOM-SR. También se utiliza el componente C_GRID_Set_SRs_DICOMs para el manejo de un conjunto de informes DICOM-SR.
- **Paquete DICOM-I.** Este paquete se utilizará para manejar imágenes digitales en formato DICOM a través del componente C_GRID_image_DICOM. También se utiliza el componente C_GRID_Set_Images_DICOMs para el manejo de un conjunto de imágenes DICOM.
- **Paquete para la creación de Almacenes Virtuales de Objetos DICOM.** De este paquete se utilizan instancias del componente C_GRID_StorageDICOM para crear los almacenes virtuales que se requieran.
- **Paquete para la Descarga de Imágenes Médicas.** De este paquete se utilizarán las instancias del componente C_GRID_Order para configurar el orden de descarga de las imágenes médicas seleccionadas. Las descargas se realizarán mediante instancias del componente C_GRID_Manage_Download_Set_Images para un conjunto de imágenes y C_GID_Manage_Download_Image_DICOM para la descarga de una sola imagen.
- **Paquete Descarga de DICOM-SR.** Este paquete se encarga de lo mismo que el paquete anterior pero para documentos estructurados. Los componentes Middleware que utiliza son el componente C_GRID_Manage_Download_Set_SRs para la descarga de un conjunto de informes DICOM-SR y C_GRID_Manage_Download_SR_DICOM para la descarga de un solo informe DICOM-SR.
- **Paquete de transferencia de DICOM-SR.** Este paquete se encarga de la transferencia de documentos estructurados, a través de la aplicación, a los repositorios DICOM desplegados. Los componentes que se utilizan son C_GRID_Order para configurar el orden de transferencia de los DICOM-SR, C_GRID_Manage_Upload_Set_SRs para la transferencia de un conjunto de informes DICOM-SR y C_GRID_Manage_Upload_SR_DICOM para la transferencia de un solo DICOM-SR.

5.3.4. Funcionalidad

Las principales funciones que implementa la aplicación son las siguientes:

- **Iniciar Sesión.** Este evento se produce cuando un usuario inicia una sesión en la aplicación a través de un navegador Web y los certificados digitales de cliente pk12, de manera segura

y mediante el protocolo HTTPS. En la parte de servidor Web se crea una sesión donde se guardarán credenciales necesarias para el acceso a los recursos o servicios Grid que ofrece la arquitectura TRENCADIS. Esta será generada a partir de los certificados de usuario correspondientes y a través del componente middleware Grid C_GRID_Session. El objeto será valido durante todo el tiempo de vida de la sesión de la aplicación Web.

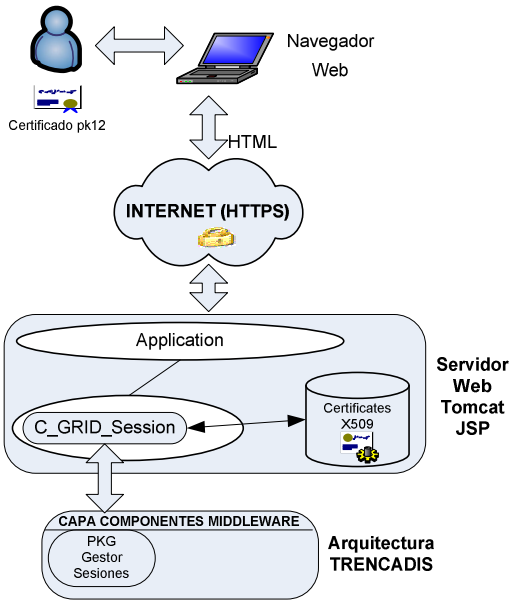

figura 61. Iniciar Sesión en Aplicación de Apoyo a la Decisión Clínica basada en Informes Radiológicos.

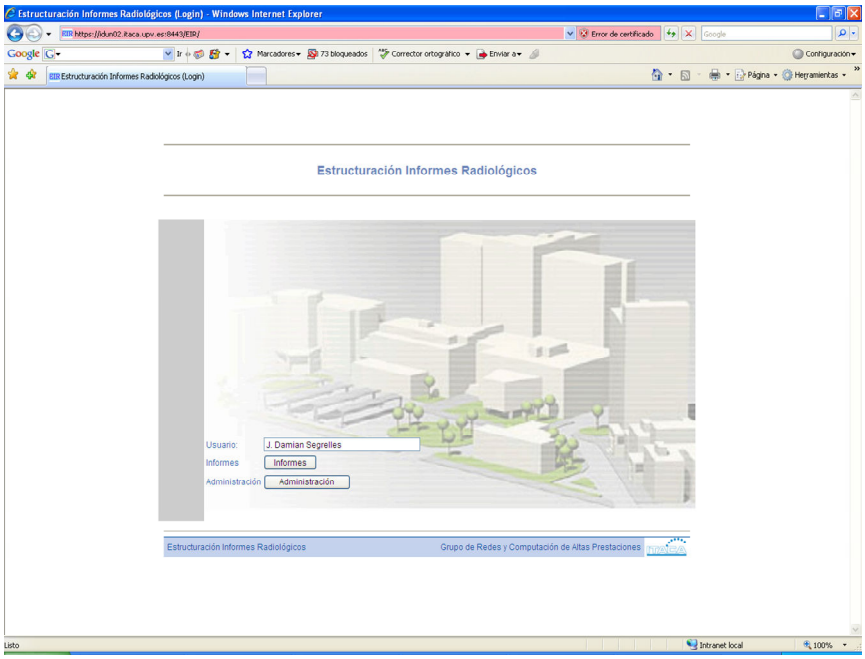

figura 62. Pantalla de Inicio de Sesión con el Navegador Web Internet Explorer.

- **Administración de Plantillas.** Esta funcionalidad no interactúa directamente con ningún componente middleware. Se encarga de crear las plantillas correspondientes a los informes estructurados que se generarán en el programa.

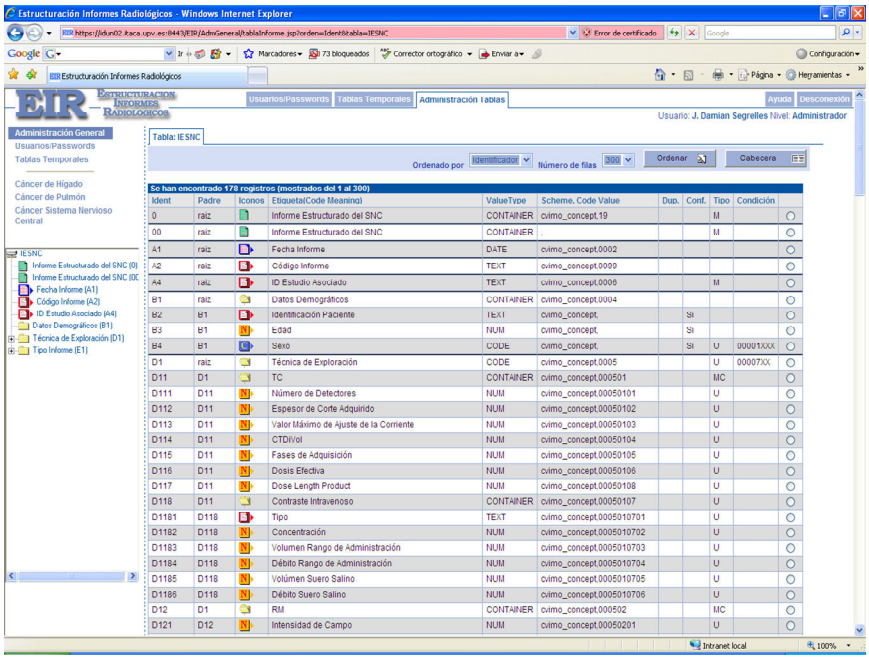

figura 63. Pantalla Creación de plantillas DICOM-SR.

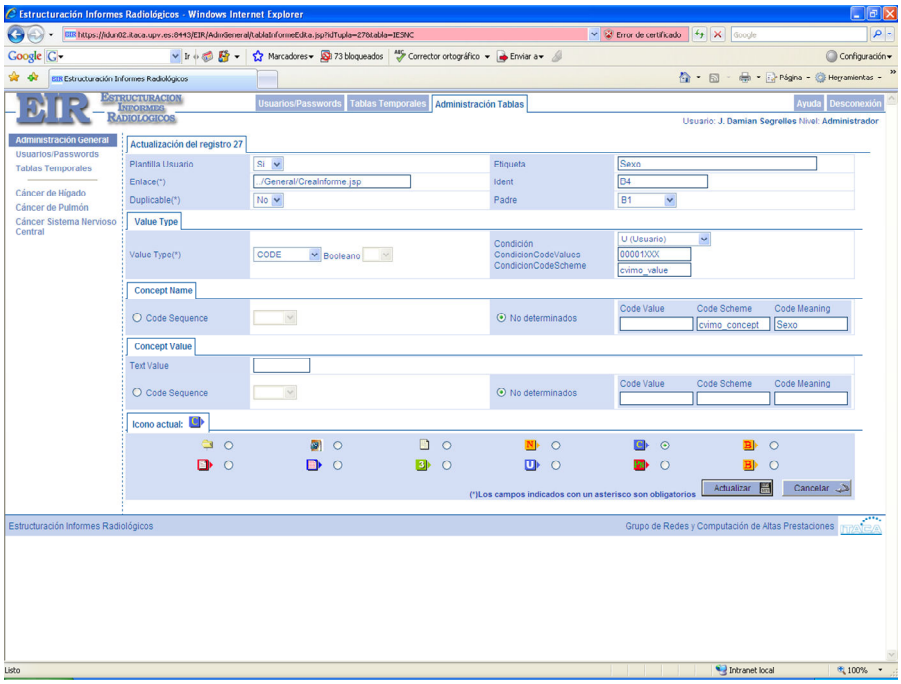

figura 64. Pantalla Creación de un ítem de las plantillas DICOM-SR.

- **Selección de Ontologías.** Esta es la primera operación que se realiza después de crear la sesión. Se encarga de seleccionar las ontologías que se utilizarán, por una parte, para agrupar los estudios sobre los que va a trabajar según el área médica a la cual pertenece el usuario, y por otra para tratar con el conjunto de informes estructurados DICOM-SR ya generados, y que se utilizan para realizar búsquedas por contenido. Las ontologías seleccionadas inicialmente deben estar activas el tiempo que dure la sesión Web iniciada

por el cliente mediante el navegador, y se manejan mediante instancias de C_GRID_Ontology.

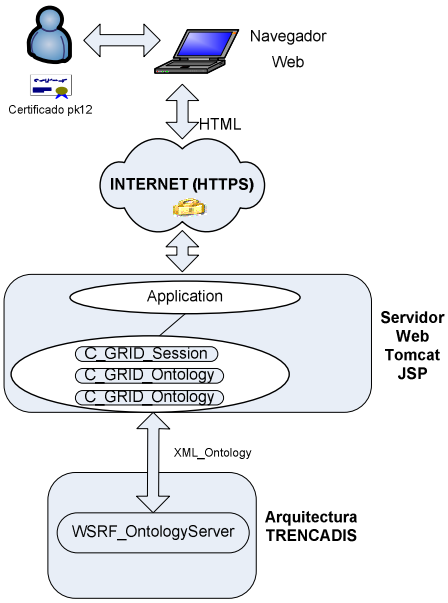

figura 65. Seleccionar Ontologías en la Aplicación de apoyo a la Decisión Clínica basada en Informes Radiológicos.

- **Crear Almacenes Virtuales de estudios DICOM.** Esta operación se realiza en el momento que se selecciona la ontología referida a los estudios de imágenes DICOM a tratar. La aplicación debe tener el control de aquellos estudios de imágenes que han sido informadas y de aquellas que están pendientes de informar, ofreciendo al usuario un interfaz donde se muestren por separado estos dos grupos. Los almacenes virtuales son instancias del componente C_GRID_StorageDICOM.

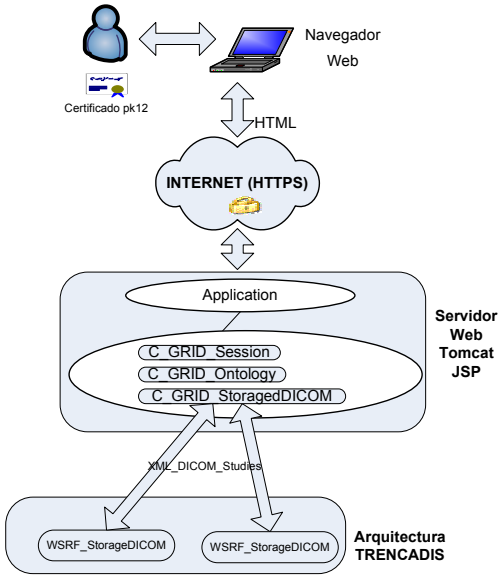

figura 66. Crear Almacenes Virtuales en la Aplicación de Apoyo a la Decisión Clínica basada en Informes Diagnósticos Radiológicos.

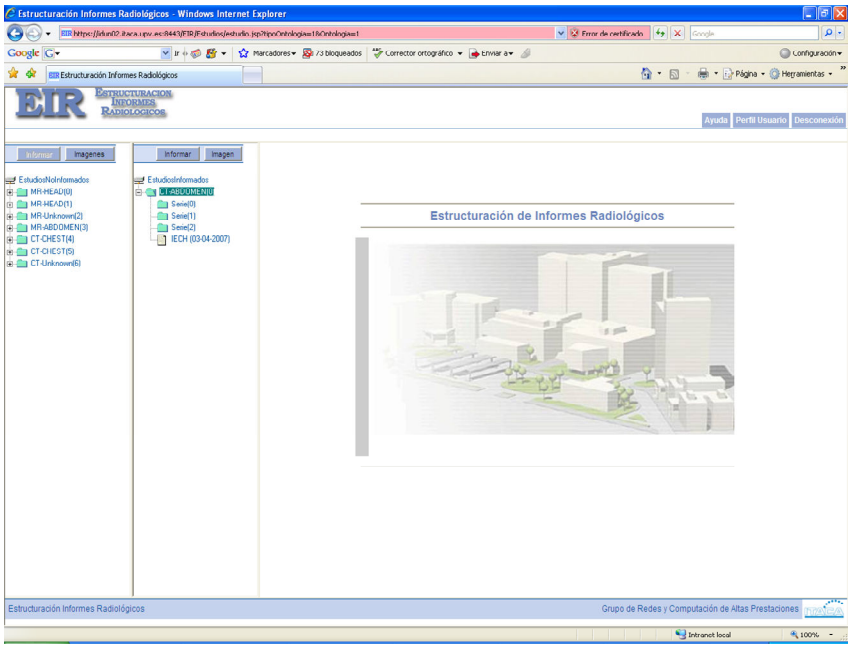

figura 67. Pantalla de visualización del almacén Virtual de estudios DICOM, separando los estudios informados de los no informados.

- **Crear Almacenes Virtuales sobre informes estructurados DICOM-SR**. Esta operación se realiza en el momento que se selecciona la ontología activa que determina a los DICOM-SR a tratar. Se establece para poder realizar consultas en función de una determinada ontología definida previamente, y también para catalogar a partir de este almacén virtual aquellos estudios sobre los cuales se ha realizado un informe (estudios informados) y aquellos sobre los que no (estudios pendientes de informar). Esta información será guardada en la aplicación Web mediante una base de datos. Los almacenes virtuales serán instancias de C_GRID_StorageDICOM.

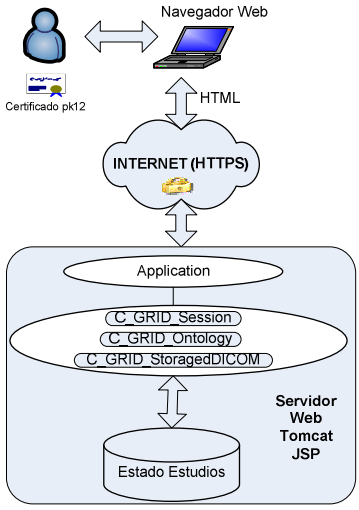

figura 68. Control de Estado de Estudios en la Aplicación de Apoyo a la Decisión Clínica basada en Informes Diagnósticos Radiológicos.

- **Generar informes estructurados DICOM-SR.** Esta funcionalidad se encarga de crear informes estructurados DICOM-SR a partir de una plantilla definida previamente y de un estudio pendiente de informar. Para ello se utilizarán instancias del componente C_GRID_DICOM_SR.

figura 69. Pantalla de Selección de las plantillas creadas para la generación de un informe radiológico DICOM-SR.

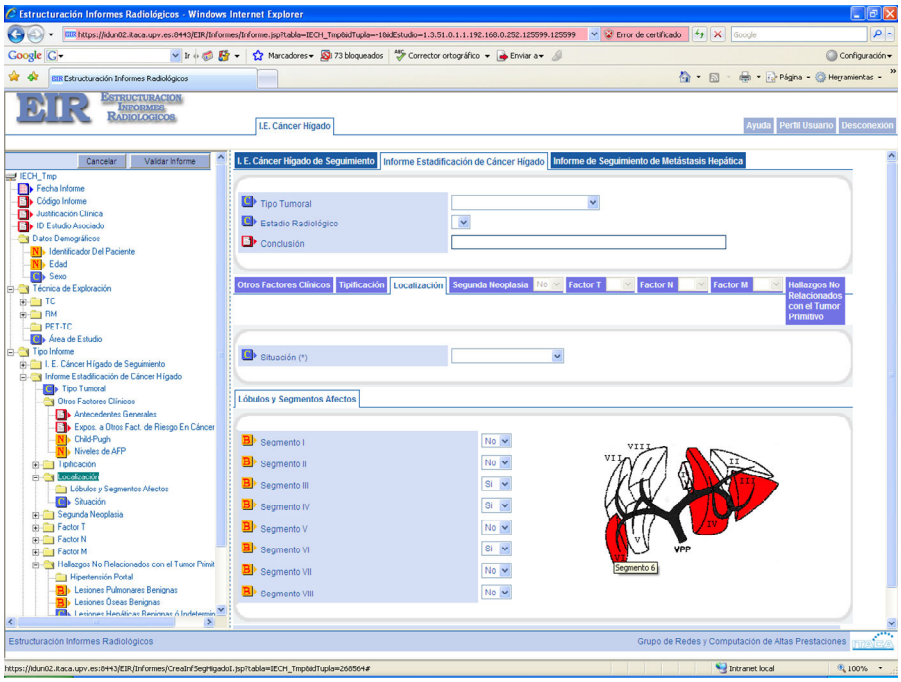

figura 70. Pantalla de generación de un informe estructurado DICOM-SR a partir de la plantilla seleccionada.

- **Guardar los DICOM-SR generados.** Los DICOM-SR que se generen se guardan en el mismo repositorio DICOM donde se encuentre el estudio informado con el que está relacionado. Esta transferencia se realizará mediante instancias de los componentes C_GRID_Manage_Upload_Set_SRs para un conjunto DICOM-SR y C_GRID_Manage_Upload_SR_DICOM para un único DICOM-SR.

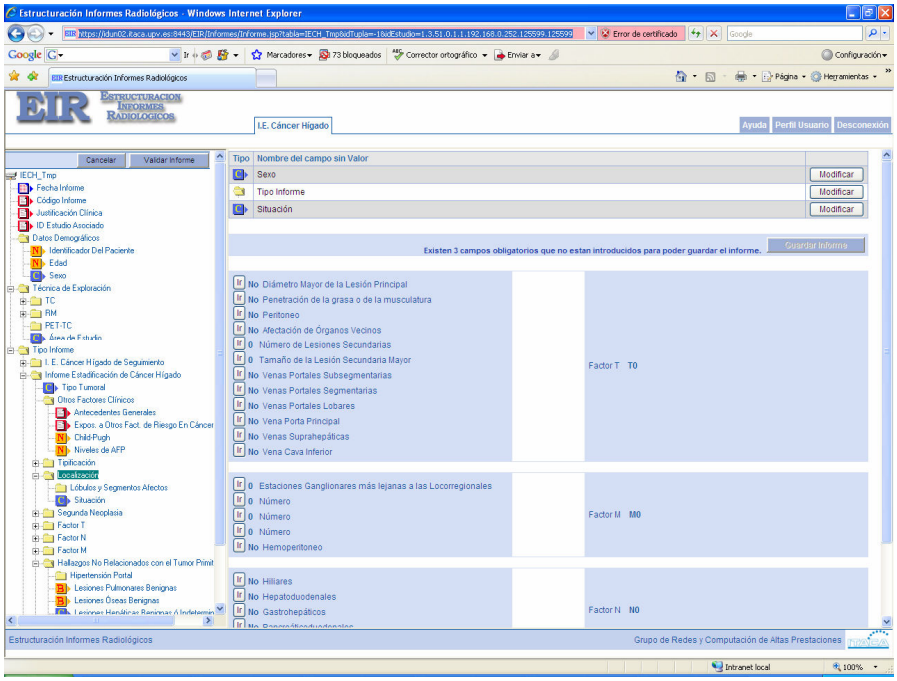

figura 71. Pantalla de validación para la transferencia de los DICOM-SR a los repositorios.

- **Búsqueda y descarga de informes DICOM-SR.** Esta funcionalidad permite la búsqueda de DICOM-SR existentes a través de los campos definidos en las ontologías. Las búsquedas se realizan utilizando las instancias del componente C_GRID_StorageDICOM creadas y las descargas mediante instancias C_GRID_Manage_Download_Set_SRs para un conjunto DICOM-SR y. C_GRID_Manage_Download_SR_DICOM para un único DICOM-SR.

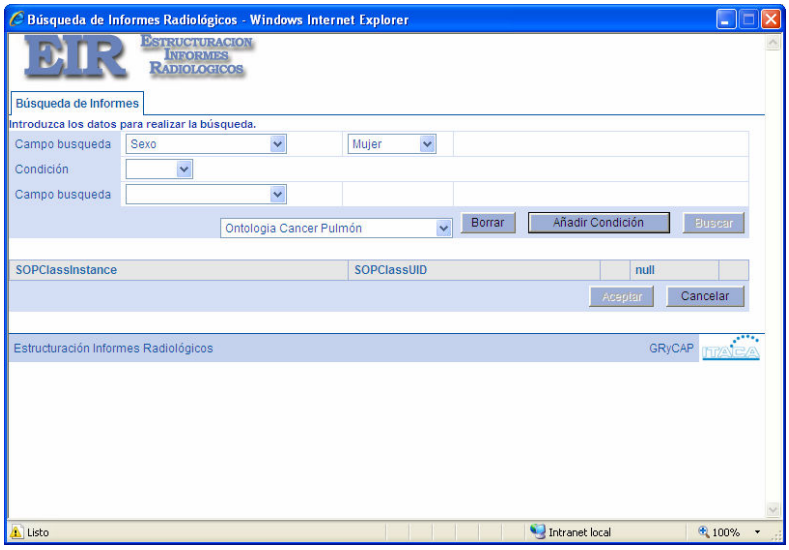

figura 72. Pantalla de Búsqueda de Informes Estructurados DICOM-SR en función de una Ontología.

5.3.5. Resultados experimentales

En el caso de esta aplicación, los resultados experimentales se relacionan más con la implantación de la infraestructura y la manera de colaborar entre las diferentes entidades implicadas, pues los experimentos presentados en las aplicaciones anteriores son extrapolables a esta aplicación, al utilizar las mismas componentes. La aplicación en sí, es un experimento aplicado dentro del del proyecto CVIMO [120] que será descrito en la sección 8.3.

CAPÍTULO VI.
Conclusiones

Los resultados obtenidos en esta tesis han sido desarrollados dentro del marco de tres proyectos de investigación (IDEAS in e-Health, GRID-IT y CVIMO). Las áreas en las que se desarrollan estos proyectos y los resultados obtenidos están relacionadas con el diseño de arquitecturas orientadas a servicios (SOA), tecnologías GRID y compartición de imágenes médicas DICOM.

Durante el desarrollo de la tesis, se ha realizado un estudio de todas aquellas áreas relacionadas con los objetivos planteados en el capítulo 2, es decir, la definición e implementación de una arquitectura genérica SOA que, de forma segura, provea de servicios suficientes para la gestión, manejo y proceso de información en formato DICOM (imágenes, señales, videos, documentos estructurados, etc.) y organizada semánticamente a partir de ontologías. Los aspectos que han sido estudiados en el primer capítulo, referente al estado del arte, son los metalenguajes, el estándar DICOM, los sistemas de información médica y sus protocolos, las tecnologías Grid y, finalmente, una breve descripción de los proyectos actuales relacionados con alguno de los objetivos planteados. A lo largo del estudio realizado, se han identificado las necesidades y carencias existentes a la hora de abordar dichos objetivos. Estas necesidades son las siguientes:

- La necesidad de una arquitectura genérica estándar, que sea capaz de integrar todos los recursos de almacenamiento y proceso DICOM distribuidos entre diferentes centros. Esta arquitectura debe tener las siguientes características:

    - **Manejo de información DICOM de forma homogénea.** Las sucesivas ampliaciones del estándar DICOM ha propiciado que la información almacenada por los diferentes almacenes sea diversa (imágenes, señales, videos e informes estructurados), por lo que ha surgido la necesidad de manejar toda esta información de forma homogénea.
    - **Autonomía de los administradores sobre los recursos que administran.** La integración de diversos recursos de almacén y proceso DICOM, de diferentes centros, requiere de un modelo de seguridad dentro de la arquitectura que permita a los administradores de los recursos no perder su control, además de poder administrar de una manera global los permisos de acceso y usuarios de los diferentes centros y recursos, es decir, se necesita integrar diferentes dominios administrativos en organizaciones virtuales.
    - **Autenticidad y confidencialidad de los datos.** La necesidad de incluir en el modelo de seguridad un mecanismo que impida el acceso a la información a

aquellos usuarios no autorizados, incluso si disponen de permisos de administración sobre los recursos locales, garantizando la privacidad y autenticidad de los datos.

- La conveniencia de manejar la información compartida por los diferentes centros de una manera organizada, y en función del área de trabajo o experimento, pues la integración de diversos recursos da lugar a una gran cantidad de información, que puede ser intratable por su gran volumen. Por ello se necesita determinar la manera de describir y especificar ontologías de manera sencilla sobre la información DICOM, permitiendo una indexación eficiente de los contenidos a la hora de ser consultados.
- La necesidad de aumentar la productividad de las aplicaciones que usen los recursos compartidos, pues éstas pueden compartir componentes independientemente de las áreas de trabajo en las que estén integradas.
- Necesidad de crear una infraestructura básica que permita una rápida integración de nuevos recursos de almacén y proceso, de manera que propicie una escalabilidad elevada.

Con respecto a todas estas necesidades, los desarrollos realizados en la tesis combinan la aplicación de tecnologías y sistemas existentes con la integración en éstos de nuevos componentes que mejoren dichos sistemas y que permita solucionar y cubrir las necesidades y carencias descritas anteriormente. Las aportaciones más importantes a destacar son las siguientes:

- Diseño de una arquitectura Grid de propósito general (TRENCADIS) para manejar el almacenamiento distribuido de datos en formato DICOM y su proceso, basada en tecnologías estándares (OGSA/WSRF, XML), frente a otras arquitecturas existentes no basadas en estándares.
- La arquitectura TRENCADIS posee una especificación completa, tanto de las capas que forman la arquitectura como de los componentes que conforman los diferentes elementos (servicios funcionales, servicios lógicos, componentes middleware y aplicaciones), de manera que, para la inclusión de cualquier elemento dentro de la arquitectura, se define la capa donde se ubicará según su función, las tecnologías estándar a utilizar para su composición y la manera de interactuar con el resto de los elementos de la arquitectura. Esto permite que el sistema sea fácilmente escalable, pues la inclusión de nuevos elementos que amplíen la funcionalidad resulta sencilla. Además de la especificación, otra aportación importante es la inclusión, en las capas de la arquitectura, de los componentes middleware de alto nivel, cuyas implementaciones ofrecen a los programadores una mayor productividad, además de una mejor reutilización de los componentes.
- La arquitectura TRENCADIS incluye un modelo para la indexación y estructuración de los datos DICOM basado en ontologías médicas. La principal aportación en este campo es la inclusión de los informes radiológicos DICOM-SR, los cuales aportan la evaluación diagnóstica de los estudios DICOM frente a los modelos convencionales basados solo en la información incluida en las propias imágenes, señales y videos DICOM, tales como información sobre el paciente, dimensiones de las imágenes etc...
- Dentro de las especificaciones de TRENCADIS, la arquitectura también aporta un lenguaje para la especificación de ontologías basado en XML, el cual permite referenciar los campos

del metadata de imágenes DICOM a través de nombres lógicos, además de poder referenciar los campos de los informes estructurados DICOM-SR.

- La arquitectura TRENCADIS define un modelo de seguridad completo, cubriendo las necesidades de autenticación y privacidad en las comunicaciones, gestión de políticas de seguridad de OV y privacidad de los datos en dominios administrativos ajenos. Las aportaciones que ofrece el modelo de seguridad TRENCADIS respecto a los existentes son las siguientes:
    - La gestión de políticas de seguridad de las OV es un sistema de autorización basado en credenciales proveídas por servidores VOMS. La aportación en este campo ha sido el desarrollo de un nuevo "Gatekeeper" integrado en los servicios Grid basados en OGSA/WSRF. Además, el sistema de autorización de TRENCADIS estructura los permisos de los miembros de las organizaciones virtuales a partir de ontologías médicas.
    - La privacidad de los datos en dominios administrativos esta inspirado en el modelo de compartición de claves basado en esquemas de Shamir. Las aportaciones respecto a este punto son, por una parte, la modificación del modelo original, permitiendo la distribución de las claves de cifrado de los datos en dominios administrativos diferentes de una misma OV. El modelo original, sólo permitía distribuir las claves en diferentes OV. Además, otra aportación significativa, es que la gestión de las claves de cifrado y descifrado tienen asociada una ontología, de manera que se deniega la operación de descifrado por los usuarios que no estén autorizados para el manejo de la ontología de la clave.
- La arquitectura TRENCADIS define el sistema de autorización e información MDS4-Extended, el cual es una ampliación del MDS4. La principal aportación de este sistema, con respecto al MDS4, es que lo hace más escalable y permite la consulta de datos masivos de manera más eficiente y mediante un lenguaje más expresivo y eficiente (SQL).
- Respecto a la transferencia de datos, la aportación principal es la inclusión de los protocolos de transferencia de imagen médica DICOM que utilizan el formato JPEG2000 para la transferencia progresiva, permitiendo la visualización de las imágenes a medida que se van recibiendo los datos.
- Otra de las aportaciones ha sido el despliegue de una infraestructura real basada en la arquitectura TRENCADIS, de manera que se ha implementado una plataforma y objetos de alto nivel, junto con diversas aplicaciones para la asistencia en la investigación en diagnóstico por imagen.

Dadas las aportaciones y los estudios previos realizados en las áreas de la tesis, se puede concluir que la hipótesis de trabajo queda demostrada y que la arquitectura TRENCADIS es idónea para compartir recursos que manejan cualquier tipo de información DICOM (imágenes, documentos estructurados, videos, señales etc…) de una manera segura, garantizando la autenticidad y la integridad de la información en todo momento y permitiendo la definición de diferentes ontologías que estructuren dicha información, permitiendo la creación de diferentes vistas sobre los mismos conjuntos de objetos DICOM.

El trabajo actual desarrollado pretende ser continuado de forma inmediata en las siguientes direcciones:

- Desarrollar los componentes de alto nivel en otros lenguajes (C++, C#, etc…) además de las implementaciones actuales en JAVA, para aportar a los programadores diferentes alternativas para sus desarrollos. JAVA es una manera sencilla de hacer que las componentes puedan ser utilizadas en diferentes plataformas, pero la programación en lenguajes como C++ aportan una mayor eficiencia temporal en sus operaciones, aunque añadiendo una mayor complejidad en los desarrollos. En el caso de C# también mejora la eficiencia de las operaciones de una manera mas sencilla que C++, pero limita las programaciones a plataformas Microsoft.
- Incrementar el soporte para diferentes almacenes de recursos DICOM. En la actualidad se soportan almacenes que utilizan bases de datos relacionales y manejadores de discos duros.
- Implementación de servicios dentro de la arquitectura para el proceso de imagen médica DICOM, tales como el proceso para el cálculo de imágenes paramétricas de captación a partir de resonancia magnética nuclear en T1 y T2, y proceso 3D mediante renderizado de volúmenes.
- Inclusión en la arquitectura TRENCADIS de un modelo para el manejo de Workflows.

Finalmente, hay que resaltar que se ha realizado una importante labor de diseminación de los resultados a través de publicaciones en congresos, revistas y prensa. Estos resultados son descritos de forma detallada en el siguiente capítulo. También hay que destacar que la transferencia de los resultados obtenidos ha sido efectiva dentro del proyecto CVIMO [120], en el cual han participado cinco hospitales (uno privado y cuatro públicos) y una empresa.

CAPÍTULO VII.
Diseminación de Resultados

El trabajo desarrollado en esta tesis ha dado lugar, hasta el momento, a 19 publicaciones en congresos nacionales e internacionales, capítulos de libros, revistas científicas, eventos, exposiciones y prensa.

Las revistas internacionales en las que se han publicado resultados relacionados con este trabajo son:

- **"A Web-Based Application Service Provision Architecture for Enabling High-Performance Image Processing."** C. De Alfonso, Ignacio Blanquer, Vicente Hernández, Damià Segrelles. Publisher: Springer. Lecture Notes in Computer Science (LNCS). ISBN 3-540-25424-2. ISSN: 0302-9743. Issue: Volume 3402, 2005. Pages 260-273.
- **"An OGSA Middleware for Managing Medical Images using Ontologies"**. Ignacio Blanquer, Vicente Hernández, Damià Segrelles. Journal of Clinical Monitoring and Computing. Publisher: Springer Science+Bussines Media B.V., Formerly Kluwer Academic Publishers B.V. ISSN: 1387-1307 (paper) 1573-2614 (Online). DOI: 10.1007/s10877-005-0675-0. Issue: Volume 19, Number 4-5, 2005, Pages 295-305.
- **"TRENCADIS - A Grid Architecture for Creating Virtual Repositories of DICOM Objects in an OGSA-based Ontological Framework".** Ignacio Blanquer, Vicente Hernandez and Damià Segrelles. Lecture Notes in Computer Science (LNCS), Subseries Lecture Notes in Bioinformatics (LNBI). Biological and Medical Data Analysis. ISSN: 0302-9743, ISBN-10: 3-540-68063-2, ISBN-13: 978-3-540-68063-5. Issue: Volume 4345, 2006. Pages 183-194.

Los congresos internacionales en los que se han publicado resultados relacionados con este trabajo son:

- **"A Web-based Application Service Provision Architecture for Enabling High-Performance Image Processing."**. C. De Alfonso, Ignacio Blanquer, Vicente Hernández, Damià Segrelles. 6th International Meeting Higth Performance Computing for Computational Science (VECPAR'04), Valencia (Spain), 2004). Proceedings of VECPAR'04. Volume II. Pages 499-512.
- **"A Middleware Grid for Storing, Retrieving and Processing DICOM Medical Image"**. Ignacio Blanquer, Vicente Hernández, Ferran Mas, Damià Segrelles. Worshop DIDAMIC-2004 (Distributed Databases and Processing in Medical Image Computing), Rennes (France), 2004. Proceedings DIDAMIC-2004.

- **"Creating Virtual Storages and Searching DICOM Medical Images Through a Grid Middleware Based in OGSA"**. Ignacio Blanquer, Vicente Hernández, Damià Segrelles. International Symposium on Cluster Computing and Grid (CCGRID05), Cardiff (United Kingdom), 2005. ISBN 0-7803-9075-X.
- **"TRENCADIS.-A WSRF Grid Middleware for Managing DICOM Structured Reporting Objects"**. Ignacio Blanquer, Vicente Hernández, Damià Segrelles. Proceedings of HealthGrid 2006 (Challenges and Opportunities of HealthGrids), Valencia (Spain), 2006. Publisher: IOS Press. ISBN 1-58603-617-3. Pages 381-396.
- **"TRENCADIS – Secure Architecture to Share and Manage DICOM Objects in an Ontological Framework Based on OGSA".** Ignacio Blanquer, Vicente Hernández, Damià Segrelles, Erik Torres. Proceedings of Healthgrid 2007 (From Genes to Personalized HealthCare: Grid Solutions for the Life Sciences), Geneva (Switzerland), 2007. Publisher IOS Press. ISBN: 978-I-58603-738-3. Pages 115-126
- **"CVIMO – Deployment of a Cyberinfrastructuree on Top of TRENCADIS Architecture to Share and Create DICOM Studies and Structured Reports".** Ignacio Blanquer, Vicente Hernández, Javier Meseguer, Damià Segrelles. IBERGRID: 1st Iberian Grid Infrastructure Conference, Santiago de Compostela (Spain), 2007. Proceedings. ISBN: 978-84-611-6634-3.
- **"Long-term Storage and Management of Encrypted Biomedical Data for Real Users in Real Organizations".** Ignacio Blanquer, Vicente Hernández, Damià Segrelles, Erik Torres. International Conference on Emerging Security Information, Systems and Technologies SECUREWARE 2007. Valencia (Spain), 2007.

Los congresos nacionales en los que se han publicado resultados relacionados con esta tesis son:

- **"Diagnóstico por Imagen de Altas Prestaciones sobre Servicios ASP".** Ignacio Blanquer, Vicente Hernandez, Damià Segrelles. Congreso XIV Jornadas del Paralelismo. Publicación ISBN 84-89315-34-5. Leganés (Madrid, España). Año 2003.
- **"Servicios ASP para el Proceso y Transmisión de Imágenes Médicas Tomográficas".** Ignacio Blanquer, Vicente Hernández, Damià Segrelles. VII Congreso Nacional de Informática de la Salud (Inforsalud). Publicación ISBN 84-930487-8-X. Madrid (España), 2004.

También se han realizado algunas exposiciones y demostraciones en los siguientes eventos:

- **Exposición y Demostración en la II Jornada de Nuevos Retos en Tecnologías de la Salud.** Título de la exposición "Middleware Grid para el Tratamiento, Proceso y Compartición de Imágenes Médicas Radiológicas". Valencia (España). Año 2005.
- **Exposición y Demostración en la III Jornada de Nuevos Retos en Tecnologías de la Salud.** Título de la exposición "Entorno Basado en Tecnologías Grid y Servicios Web para la Registración de Imágenes Médicas". Valencia (España). Año 2006.
- **Exposición y Demostración en la III Jornada de Nuevos Retos en Tecnologías de la Salud.** Título de la exposición "TRENCADIS – Middleware Grid para la Creación de

Repositorios Virtuales de Objetos DICOM en un Marco Ontológico". Valencia (España). Año 2006.

- **Exposición y Demostración en el congreso HealthGRID 2006.** Título de la exposición "TRENCADIS Project. A Middleware Grid for CreatingVirtual Repositories of DICOM Objects in an Ontological Framework". Valencia (España). Año 2006.
- **Jornada de presentación del proyecto CVIMO y del despliegue de la infraestructura.** Lugar de celebración: Conselleria de Empresa, Universidad y Ciencia de la Generalitat Valenciana. Fecha 23 de Marzo de 2007. http://www.grycap.upv.es/cvimo/jornada/index.html

Las notas de prensa relacionadas con los resultados del trabajo son los siguientes:

- **Periódico "ABC".** Fecha de publicación: 23 de Marzo de 2007. "Los hospitales podrán compartir casos de cáncer relevantes para la investigación". Pag 64.
- **Periódico "Las Provincias".** Fecha de publicación: 23 de Marzo de 2007. "Nuevo Programa Informático para Oncología". Pag.10.

CAPÍTULO VIII. Proyectos de Investigación Asociados

Los resultados de esta tesis se han desarrollado en el marco de los siguientes proyectos de investigación:

- "An Integrated Distributed Environment for Application Services in e-Health. (IDEAS in e-Health)". IST-2001-34614. Financiado por el V Programa Marco de la Comisión Europea y la Dirección General de la Sociedad de la Información. La duración fue de 18 meses, desde el 1/4/2002 al 31/12/2003.
- "Investigación y Desarrollo de Servicios Grid: Aplicación a Modelos Cliente-Servidor, Colaborativos y de Alta Productividad (GRID-IT)". TIC2003-01318. Financiado por el Ministerio de Educación y Ciencia de España. La duración fue de 36 meses, desde el 1/12/2003 al 30/11/2006.
- "Ciberinfraestructura Valenciana de Imágenes Médicas Oncológicas (CVIMO)" GVEMP06/004. Financiado por la Conselleria de Empresa, Universidad y Ciencia de la Generalidad Valenciana. La duración fue de 12 meses, desde el 1/1/2006 al 31/12/2006.

En los siguientes apartados de este capítulo se describe con más detalle cada uno de los proyectos, además de resaltar los resultados que están más directamente relacionados con los desarrollos de la tesis.

8.1. An Integrated Distributed Environment for Application Services in e-Health (IDEAS in e-Health)

El proyecto "An Integrated Distributed Environment for Application Services in e-Health (IDEAS in e-Health IST-2001-34614)" fue financiado por la Dirección General de la Sociedad de la Información de la Comisión Europea en el V programa Marco. La duración fue de 18 meses, desde el 1/4/2002 al 31/12/2003. Los participantes en el proyecto fueron la Universidad Politécnica de Valencia (UPV)-Grupo de Redes y Computación de Altas Prestaciones (GRyCAP)-Grupo de Bioingeniería, Electrónica y Biomedicina (BET), la empresa APLITEC-Telehealth, el Hospital Universitario la Fe y el Hospital 9 de Octubre. El Investigador Principal del proyecto fue el Profesor Vicente Hernández.

El objetivo principal de este trabajo fue el desarrollo de una plataforma de servicios de teleradiología, basándose en un sistema de diagnóstico por imagen de altas prestaciones sobre una arquitectura de servicios ASP, aprovechando así las ventajas que proporciona Internet para las comunicaciones (bajo coste y fácil conectividad) y proporcionando soluciones a aquellos problemas que supone la utilización de una red no dedicada (ancho de banda limitado, seguridad, etc…). El interfaz para la interacción con los servicios de la plataforma fue elaborado mediante web.

figura 73. Esquema resumen de las interacciones entre los entornos que interactuaron en el proyecto IDEAS-EHEALTH.

Los resultados obtenidos relacionados con los trabajos de ésta tesis son los siguientes:

- La arquitectura de la plataforma de teleradiología, desarrollada en este proyecto, sirvió como base para el desarrollo de la arquitectura TRENCADIS, al ser ésta una arquitectura genérica, basada en estándares y orientada a servicios.
- Uno de los resultados más importantes del proyecto fue el estudio y desarrollo de algoritmos de transferencia progresiva de imágenes médicas y la aplicación del formato JPEG2000 para dichas transferencias.
- Estudio de tecnologías para la autenticación y privacidad de los datos en las comunicaciones, siendo Internet la red de interconexión utilizada.

8.2. Investigación y Desarrollo de Servicios Grid. Aplicación a Modelos Cliente-Servidor, Colaborativos y de Alta Productividad (GRID-IT)

El proyecto "Investigación y Desarrollo de Servicios Grid. Aplicación a Modelos Cliente-Servidor, Colaborativos y de Alta Productividad (GRID-IT TIC2003-01318)" fue financiado por el Ministerio de Educación y Ciencia de España. La duración fue de 36 meses, desde el 1/12/2003 al 30/11/2006. El Investigador Principal del proyecto fue el Profesor Vicente Hernandez, responsable del Grupo de Investigación en Redes y Computación de Altas Prestaciones (GRyCAP).

El objetivo principal del proyecto GRID-IT consistió en el desarrollo de una serie de servicios y componentes Grid generales, a partir de la migración y desarrollo de un conjunto de aplicaciones piloto con diferentes paradigmas de interacción (cliente-servidor, peer to peer y alta productividad) en un entorno Grid. De este modo se creó un marco que facilitase el desarrollo posterior de otras aplicaciones, en diferentes áreas científicas y tecnológicas, basadas en tecnología Grid.

Los resultados relacionados con los trabajos de esta tesis son los siguientes:

- Desarrollo de una arquitectura Grid en la que se incluyen servicios y componentes para el almacenamiento, proceso y manejo distribuido de objetos DICOM (imágenes médicas, documentos estructurados, señales, etc...), y la creación de aplicaciones para la asistencia en el diagnóstico, la formación y la investigación en diferentes áreas médicas y clínicas. Se podría considerar que el inicio de la arquitectura TRENCADIS desarrollada en la tesis nace en este proyecto.
- La arquitectura permite compartir objetos DICOM mediante almacenes virtuales distribuidos, pudiendo especificar la información perteneciente a cada almacén mediante ontologías. La arquitectura proporciona herramientas para la creación de las ontologías.
- Transferencias de los objetos DICOM, tanto al guardarlos como al descargarlos, de manera segura. La arquitectura permite el cifrado de la información de forma que se facilita el acceso únicamente a aquellos usuarios que se considere necesario. Además, los objetos se codifican mediante formatos que permiten la transmisión progresiva y tolerante a fallos.
- Servicios de proceso de los objetos DICOM en batch sobre los recursos del Grid, atendiendo a algoritmos de post-proceso básicos (corregistración, lVolume 3D u otros).
- La arquitectura es fácilmente extensible a otros problemas, incluso fuera del ámbito clínico, en los que el objetivo sea crear un repositorio virtual de datos voluminosos y con información no estructurada (imágenes, señales, vídeos, registros de mediciones, etc.) a la que se le puede asociar una metainformación estructurada que se utilizará para definir las ontologías y los criterios de agrupación, indexación y búsqueda. Por ejemplo el GRyCAP-UPV, en colaboración con el Grupo de Teledetección y Sistemas de Información Geográfica del Instituto de Desarrollo Regional de la Universidad de Castilla la Mancha, está desarrollando una herramienta que, basándose en la arquitectura TRENCADIS, permitirá el uso de tecnologías Grid para la indexación y búsqueda de Información Geográfica de forma distribuida.

8.3. Ciberinfraestructura Valenciana de Imagen Médica Oncológica (CVIMO)

El proyecto "Ciberinfraestructura Valenciana de Imágenes Médicas Oncológicas (CVIMO GVEMP06/004)" fue financiado por la "Conselleria D'empresa, Universitat i Ciencia de la Generalitat Valenciana". La duración fue de 12 meses, desde el 1/1/2006 al 31/12/2006. Los participantes en el proyecto fueron cinco hospitales valencianos (Clínica Quirón, Hospital Dr. Peset, Hospital de la Ribera, Fundación Instituto Valenciano de Oncología y la Fundación para la Investigación del Hospital la Fe) y la empresa British Telecom. El Investigador Principal del proyecto fue el Profesor Vicente Hernández.

El objetivo de CVIMO fue poner en marcha una infraestructura informática, entre varios hospitales, para organizar y compartir de forma segura estudios radiológicos especialmente relevantes, utilizando el informe radiológico estructurado para su organización. CVIMO no compite con los sistemas PACS y RIS, que se orientan a la asistencia en la práctica clínica diaria, sino que complementa estos sistemas permitiendo a los expertos médicos archivar y compartir casos que durante la práctica consideren relevantes para investigación o formación. CVIMO utiliza las tecnologías Grid como base de desarrollo, utilizando protocolos y entornos estándares y soportados tanto por la comunidad científica como la industria (DICOM, DICOM-SR, XML, https, WS, WSRF, GridFTP, X.509, etc…).

El despliegue de CVIMO se ha orientado principalmente al área oncológica, y específicamente al cáncer de hígado, pulmón y sistema nervioso central, aunque la tecnología es válida y general para soportar la organización de estudios de otras patologías o áreas, una vez definidos los esquemas de los informes.

Los resultados fundamentales en el desarrollo de CVIMO relacionados con la tesis son los siguientes:

- Proporcionar un entorno en el que los estudios puedan ser clasificados de acuerdo a la información estructurada y estandarizada de los informes, permitiendo búsquedas tan complejas como "Consultar los estudios de varones menores de 20 años con una lesión principal de más de 20 mm, con márgenes irregulares, contenido graso y con ausencia de metástasis".
- Proporcionar herramientas para la codificación de informes estructurados de estadificación y seguimiento, en base a plantillas definidas por los expertos radiológicos y oncológicos presentes en el proyecto, incluyendo la codificación automática del estadio y los valores TNM.
- Proporcionar herramientas para la organización de los estudios de forma que únicamente los relevantes a un grupo de usuarios sean accesibles y visibles, reduciendo los problemas que supone la publicación de resultados.
- Proporcionar un entorno seguro en el que se garantice la identidad, autorización y privacidad de usuarios y datos.

- Permitir el despliegue de servicios de altas prestaciones, como la corregistración o postproceso de imagen funcional sobre los datos existentes en el sistema.
- Utilizar protocolos estándar y una arquitectura de forma que el despliegue del sistema interfiera lo mínimo posible con los sistemas existentes, aislando el sistema de práctica clínica de CVIMO y dejando a la elección del experto los casos sensibles, evitando problemas de cortafuegos, interoperabilidad y multiplataforma para asegurar que no se resiente ni la administración de sistemas ni la seguridad.

Referencias Bibliográficas

[1] H. K. Huang . "PACS and Imaging Informatics: Basic Principles and Applications". ISBN**:** 0471251232. Editorial Wiley-Liss

[2] Jason Oakley. "Digital Imaging: A Primer for Radiographers, Radiologists and Health Care Professionals". ISBN 1841101214.

[3] H U Prokosch, J. Dudeck. "Hospital Information Systems: Design and Development Characteristics, Impact & Future Architecture." ISBN. 0444821295

[4] Digital Imaging and Communications in Medicine (DICOM) : http://medical.nema.org

[5] Clunie DA. DICOM Structured Reporting, Bangor, PA, PixelMed Publishing, 2000.

[6] DICOM part 3 – Information object definitions. Available at: http://medical.nema.org/dicom/2001/01_03PU.pdf

[7] DICOM part 6 – Data dictionary. Available at: http://medical.nema.org/dicom/2001/01_06PU.pdf

[8] Health Level Seven: http://www.hl7.org

[9] Integrating the Healthcare Enterprise: http://www.rsna.org/IHE

[10] Council Of Europe – Committee of Ministers, Recommendation No. R(97)5 of The Committee Of Ministers to Member States on the Protection Of Medical Data, Council of Europe Publishing, Strasbourg, 12 February 1997

[11] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, OJ L, 23 Nov. 1995, http://europa.eu.int/com-m/internal_market/privacy/law_en.htm

[12] Ley Orgánica de Protección de Datos de Carácter Personal (LOPD). http://www.aeat.es/normlegi/otros/lortad2000.htm

[13] Colby, Martin; Jackson, David S. Special Edition. Using SGML. QUE Special Edition Series. Indianapolis, IN: QUE Corporation, Macmillan Publishing. ISBN: 0-7897-0414-5. Extent: 600+ pages, with CDROM

[14] Extensible Markup Language (XML) 1.0 W3C Recommendation 10 February 1998. Available at: http://www.w3c.org/XML

[15] An Introduction to the Resource Description Framework by Eric Miller, D-Lib Magazine, May 1998.

[16] OWL Web Ontology Language Guide, Michael K. Smith, Chris Welty, and Deborah L. McGuinness, Editors, W3C Recommendation, 10 February 2004, 12 http://www.w3.org/TR/2004/REC-owl-guide-20040210/ . Latest version available at http://www.w3.org/TR/owlguide

[17] International Organization for Standardization. http://www.iso.org

[18] "HyperText Markup Language (HTML) Home Page". http://www.w3.org/MarkUp

[19] "World Wide Web Consortium". www.w3.org

[20] "Comparison of SGML and XML. World Wide Web Consortium Note 15-December-1997" . http://www.w3.org/TR/NOTE-sgml-xml.html

[21] "Unicode Home Page". http://www.unicode.org

[22] "The Extensible Stylesheet Language Family (XSL)". http://www.w3.org/Style/XSL

[23] "Cascading Style Sheets, Level 2 CSS2 Specification. W3C Recommendation 12-May-1998".http://www.w3.org/TR/REC-CSS2

[24] XML Schema: http://www.w3c.org/XML/Schema

[25] XSL Transformations (XSLT) http://www.w3.org/TR/xslt

[26] "XML Linking Languaje (XLink). Version 1.0. W3C Recommendation 27 June 2001". http://www.w3.org/TR/xlink

[27] "XML Pointer Language (XPointer). Version 1.0. W3C Last Call Working Draft 8 January 2001". http://www.w3.org/TR/WD-xptr

[28] XML Spy. Available at: http://www.altova.com. Accessed Nov 9, 2004.

[29] K.P.Lee, PhD, Jingkun Hu, Ms. XML Schema Representation of DICOM Structured Reporting. Journal of the America Medical Informatics Associations. Volume 10, Number 2, March/April 2003.

[30] Luyin Zhao, MS, Kwork Pun Lee, PhD, Jingkun Hu, DPS. Generating XML Schemas for DICOM Structured Reporting Templates. Journal of the America Medical Informatics Associations. Volume 12, Pag.72-83, August 2005.

[31] Mertz D. Comparing W3C XML Schemes and Document Type Definitions (DTD). Available at: http://www-106.ibm.com/developerworks/library/x-matters7.html

[32] U. S. Department of Health and Human Services, Public Health Service Health Care Financing Administration. "ICD-9-CM: International Classification of Diseases", 9th Revision, Clinical Modification, Fourth Edition, Volumes 1,2, and 3: Official Authorized Addendum, Effective October 1, 1992. "Coding Clinic for ICD-9-CM 9" , no. Special Edition (1992): 1-63.

[33] "SNOMED International". http://www.snomed.org

[34] McDonald, C. J., Huff, S. M., Suico, J. G., Hill, G., Leavelle, D., Aller, R., Forrey, A., Mercer, K., DeMoor, G., Hook, J., Williams, W., Case, J., & Maloney, P. (2003). "LOINC, a Universal Standard for Identifying Laboratory Observations: A 5-year Update". Clin Chem, 49(4), 624-633.LOINC.

[35] Robert H. Dolin, Liora Alschuler, et al. "HL7 Clinical Document Architecture, Release 2.0.". http://www.hl7.org/v3ballot/html/infrastructure/cda/cda.htm

[36] "Towards Open Grid services Architecture". http://www.globus.org/ogsa/

[37] "The Web Services Resource Framework". http://www.globus.org/wsrf/

[38] S. Tuecke, K. Czajkowski, I. foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling,P. Vanderbilt. "Open Grid Service Architecture OGSI".

[39] I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Intl. J. Supercomputing Applications, 2001.

[40] Globus Toolkit, http://www.globus.org/toolkit/

[41] Global Grid Forum, www.ggf.org

[42] "Web Services Description Language (WSDL)". http://www.w3.org/TR/wsdl

[43] Organization for the Advancement of Structured Information Standards. http://www.oasis-open.org.

[44] B. Tierney, R. Aydt, D. Gunter, W. Smith,V. Taylor, R. Wolski, and M. Swany, Tech. Rep.GWD-Perf-16-1, GGF (2001).

[45] "Overview of the Grid Security Infraestructure" http://www.globus.org/security/overview.html

[46] "gLite. LightWeight Middleware for Grid Computing". http://glite.web.cern.ch/glite/

[47] "LCG Middleware". http://lcg.web.cern.ch/LCG

[48] "Globus 4 toolkit". http://www.globus.org/toolkit

[49] "gCitizen, Grid Technology for eGovernment Systems Integration". Carlos de Alfonso, Miguel Caballer y Vicente Hernández. Proceedings of IADIS International Conference e-Commerce 2005. ISBN: 972-8924-06-2 .Pags 321 - 324.

[50] Carlos de Alfonso, Miguel Caballer, Vicente Hernandez. "DIDA, a Distributed Discovery Architecture for Grid Environments". Proceedings Cracow 05 Grid Workshop. November 20-23, 2005. Cracow, Poland.

[51] "Federated Advanced Directory architecture". http://fada.sourceforge.net/

[52] Debbra Wetteroth. OSI Reference Model for Telecommunications. ISBN: 0071380418.

[53] Web security and commerce / Simson Garfinkel. - Cambridge : O'Reilly, 1997. - 483 p.; 23 cm. ISBN 1565922697

[54] PKI. Infraestructura de Claves Públicas. Nash (Editorial McGraw-Hill). ISBN: 9584102834. 1ª edición (11-APR-02).

[55] Public-Key Infrastructure (X.509) (pkix), http://www.ietf.org/html.charters/pkix-charter.html

[56] Grid Security Infrastructure: http://www.globus.org/security/

[57] R. Housley, T. Polk,W. Ford and D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC3280 (2002)

[58] D. Kelsey, L. Cornwall, Security Requirements and Testbed-1Security Implementation, DataGrid-07-D7.5–0111-4–0, 2002.

[59] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the GRID", The International Journal of High Performance Computing Applicacions, 14/3/2001.

[60] A. Frohner, V. Ciaschini, VOMS Credential Format, EDG Draft, 2004.

[61] S. Farrel, R. Housley, An Internet Attribute Certificate Profile for Authorization, RFC 3281, 2002.

[62] The DataGRID Project. http://www.edg.org/

[63] S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman, C. Kesselman, Internet X.509 Public Key Infrastructure Proxy Certificate Profile, draft-ggf-gsi-proxy-04, 2002.

[64] S. Farrel, R. Housley, An Internet Attribute Certificate Profile for Authorization, RFC 3281, 2002.

[65] Architectural design and evaluation criteria: WP4 Fabric Management, DataGrid-04-D4.2–0119-2–1, 2001.

[66] M. Draoli, G. Mascari, R. Piccinelli, Project Presentation, DataGrid-11-NOT-0103- 1.

[67] http://www. datatag. org/.

[68] "MySQL" http://www.mysql.com/

[69] S. Tuecke. D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman, C. Kesselman. Internet x509 Public key Infraestructure Proxy Certificate Profile, draft-ggf-gsi-proxy-04, 2002

[70] D. Box, D.Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.Nielsen, S. Thatte, D. Winer, Simple Object Access protocol (SOAP) 1.1, Note 20000508, 2000

[71] D. W. Chadwick, O.Otenko, The PERMIS x509 role based privilege management infrastructure, Fut. Grn. Comput. Syst. 19 (2), Elsevier Science Publishers B.V., 2003, pp. 277-289.

[72] L. Pearlman, V.Welch, I. Foster, K. Kesselman, S. Tuecke, A Community authorization service for group collaboration, in: IEEE Workshop on Policies for Distributed Systems and Networks, 2002.

[73] M. Thompson, A. Essiari, S. Mudumbai, Certificate-based authorization policy in a PKI environment, ACM Trnas. Inform. Syst. Secur. 6 (4) (November 2003) 566-588.

[74] Stephen T. C. Wong. A cryptologic based trust center for medical images. J. Am. Med. Inform. Assoc. 1996 Nov–Dec; 3(6): 410–421

[75] Council Of Europe – Committee of Ministers, Recommendation No. R(97)5 of The Committee Of Ministers to Member States on the Protection Of Medical Data, Council of Europe Publishing, Strasbourg, 12 February 1997

[76] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, OJ L, 23 Nov. 1995, http://europa.eu.int/com-m/internal_market/privacy/law_en.htm

[77] Carlos de Alfonso, Ignacio Blanquer, Vicente Hernández. "Large Medical Datasets on the GRID". Methods of Information in Medicine. 2005;44(2):172-6

[78] Claerhout B, De Moor G. From Grid to HealthGrid: introducing Privacy Protection. Proceedings of the First European HealthGrid Conference, p. 226-233. (Jan 16th-17th, 2003). Document from Information Society Technologies, European Commission

[79] Zhang N, Rector A, Buchan I, Shi Q, Kalra D, Rogers J, Goble C, Walker S, Ingram D, Singleton P. A Linkable Identity Privacy Algorithm for HealthGrid. Stud Health Technol Inform. 2005;112:234-45

[80] Kalra D, Singleton P, Ingram D, Milan J, MacKay J, Detmer D, Rector A. Security and confidentiality approach for the Clinical E-Science Framework (CLEF). CLEF Industrial Forum. Sept. 2003, Sheffield Hallam University

[81] Seitz L, Pierson JM, Brunie L. Encrypted storage of medical data on a grid. Methods of Information in Medicine 2005; 44(2):198-201

[82] Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH protocol architecture. Technical report, The Internet Engineering Task Force IETF, 2002. http://www.ietf.org/internet-drafts/draft-ietf-secsharchitecture-13.txt

[83] T. Dierks and C. Allen. The TLS protocol version 1.0. Technical report, The Internet Engineering Task Force IETF, 1999. http://www.ietf.org/rfc/rfc2246.txt

[84] L. Seitz, JM. Pierson, and L. Brunie. Key management for encrypted data storage in distributed systems. Second International IEEE Security in Storage Workshop (SISW), Washington DC, USA, October 2003

[85] Ian Foster, Carl Kesselman, Laura Pearlman, Steven Tuecke, and Von Welch. The Community Authorization Service: Status and Future. In Proceedings of Computing in High Energy Physics 03 (CHEP '03), 2003

[86] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á . Frohner, A. Gianoli, K. Lörentey, and F. Spataro. VOMS, an authorization system for virtual organizations. In Proceedings of the 1st

European Across Grids Conference, 2003

[87] L. Seitz, J. Pierson, and L. Brunie. Semantic access control for medical applications in grid environments. In Euro-Par 2003 Parallel Processing, volume LNCS 2790, pages 374–383. Springer, 2003

[88] P. Zimmermann. The Official PGP User's Guide. MIT Press, 1995

[89] M. Blaze. A cryptographic file system for UNIX. In ACM Conference on Computer and Communications Security, pages 9–16, 1993

[90] J. Hughes, C. Feist, S. Hawkinson, J. Perrault, M. O'Keefe, and D. Corcoran. A universal access, smart-card-based, secure file system. In Proceedings of the 3rd annual Atlanta Linux Showcase, 1999

[91] Seitz L, Pierson JM, Brunie L. Encrypted storage of medical data on a grid. Methods of Information in Medicine 2005; 44(2):198-201

[92] Torres E, De Alfonso C., Blanque I., Hernandez V. "Privacy Ptotection in HealthGrid: Distributing Encryption Management over the VO". Proceeding of HealthGrid 2006. ISBN. Technology and Informatics. Ed. IOPress. ISBN:1-58603-617-3

[93] Adi Shamir, How to share a secret. In Communications of the ACM, volume 22, pages 612–613, 1979

[94] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone. Handbook of Applied Cryptography. August 2001.

[95] Neal R. Wagner. The Laws of Cryptography with Java Code, 2003. http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf

[96] M. Burns, A.L. Rowland, D. Rueckert, J.V. Hajnal, D.L.G. Hill. CD-Rom for the Proceedings of the UK e-Science All Hands Meeting 2004. A Grid Infrastucture for Image Registration and Segmentation.

[97] M. Burns, A.L. Rowland, D. Rueckert, J.V. Hajnal, Kelvin Leung, D.L.G. Hill, J. Vickers. CD-Rom for the Proceedings of the UK e-Science All Hands Meeting 2004. Information Extraction from Images (IXI) Grid Services for Medical Imaging.

[98] A.L. Rowland, M. Burns, T. Hartkens, J.V. Hajnal, D. Rueckert, D.L.G. Hill, J. Vickers. Proceeding of Distributed Databases and processing in Medical Image Computing (DIDAMIC) 2004. Information Extraction from Images (IXI): Image Processing Workflows Using a Grid Enabled Image Database.

[99] ACI project MEDIGRID: medical data storage and processing on the GRID http://www.creatis.insa-lyon.fr/MEDIGRID

[100] J. Montagnat, H. Duque, J.M Pierson, V. Beton, L. Brunie, I.E. Magnim. Proceedings Healthgrid'03. Medical Image Content-Based Queries using the GRID. Lyon, France, January 2003.

[101] Tom OINN, Mark GREENWOOD, Matthew ADDIS, M. Nedim ALPDEMIR, Justin FERRIS, Kevin GLOVER, Carole GOBLE, Antoon GODERIS, Duncan HULL, Darren MARVIN, Peter LI, Philip LORD, Matthew R. POCOCK, Martin SENGER, Robert STEVENS, Anil WIPAT and Chris WROE. Concurrency and Computation: Practice and Experience. Taverna: Lessons in creating a workflow environment for the life sciences.

[102] myGRID proyect. http://www.mygrid.org.uk/

[103] Enabing Grids for E-Science (EGEE). http://www.eu-egee.org

[104] GRID5000 Project. http://www.grid5000.fr/

[105] An Information System for Managing Distributed Knowledge and Data Bases in Neuroimaging (NeuroBase). www.irisa.fr/visages/neurobase

[106] C. Barillot, R. Valabregue, J-P. Matsumot, F. Aubry, H. Benali, Y. Cointepas et al."NeuroBase : Management of Distributed and Heterogeneous Information Sources in Neuroimaging.". DiDaMIC-2004, a satellite workshop of Miccai-2004

[107] "BIRN. Biomedical Informatics Research Network". http://www.nbirn.net

[108] "MammoGrid Project". http://www.mammogrid.com

[109] "eDiaMoND Project". http://www.ediamond.ox.ac.uk/objectives.html

[110] "GPCALMA Project". http://gpcalma.to.infn.it

[111] "CaBIG. Cancer Biomedical Informatics Grid". https://cabig.nci.nih.gov

[112] "The OGSA/DAI Project". http://www.ogsadai.org.uk

[113] "Grid User Management System". http://grid.racf.bnl.gov/GUMS

[114] Aden T, Eichelberg M, Thoben W, A fault-tolerant cryptographic protocol for patient record requests, Proceedings of EuroPACS-MIR 2004

[115] Dogac, A., Laleci, G., Kirbas, S., Kabak, Y., Sinir, S., Yildiz, A., Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain, Information Systems Journal (Elsevier) special issue on Semantic Web and Web Services, accepted for publication in 2005

[116] I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[117] Bruce Schneier " Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. ISBN: 0-471-12845-7.

[118] Adams, Carlisle Lloyd, Steve. Understanding PKI: Concepts, Standards, and Deployment Considerations. ISBN: 0672323915

[119] "The OGSA/DAI Project". http://www.ogsadai.org.uk

[120] "Ciberinfraestructura Valenciana de Imagen Médica Oncológica (CVIMO)" http://www.grycap.upv.es/cvimo/

ANEXO I. Documentos XML sobre Ontologías

Documento XML de una ontología sobre una imagen Médica DICOM

```
<ONTOLOGY>
        <IDONTOLOGY>1</IDONTOLOGY>
        <CERTIFICATE>1</CERTIFICATE>
        <TYPEONTOLOGY>1</TYPEONTOLOGY>
        <DESCRIPTION>Ontologia de Test para Estudios de Imagenes en General</DESCRIPTION>
        <RESTRICTIVE>
        </RESTRICTIVE>
        <CREATION>
                <FIELD>
                        <CODE1>0x0008</CODE1>
                        <CODE2>0x0060</CODE2>
                        <ALIAS>MODALITY</ALIAS>
                        <DESCRIPTION>Modality Type of Image</DESCRIPTION>
                        <TYPE>STRING</TYPE>
                </FIELD>
        </CREATION>
        <FILTER>
                <FIELD>
                        <CODE1>0x0008</CODE1>
                        <CODE2>0x0070</CODE2>
                        <ALIAS>MANUFACTURER</ALIAS>
                        <DESCRIPTION>Manufacturer</DESCRIPTION>
                        <TYPE>STRING</TYPE>
                </FIELD>
                …
        </FILTER>
        <FIXED>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0000</CODE2>
                        <ALIAS>STUDY_UID</ALIAS>
                        <DESCRIPTION>Study UID</DESCRIPTION>
                        <TYPE>STRING</TYPE>
                </FIELD>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0000</CODE2>
                        <ALIAS>SERIE_UID</ALIAS>
                        <DESCRIPTION>Serie UID</DESCRIPTION>
                        <TYPE>STRING</TYPE>
                </FIELD>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0000</CODE2>
                        <ALIAS>IMAGENUMBER</ALIAS>
                        <DESCRIPTION>Image Number</DESCRIPTION>
                        <TYPE>INTEGER</TYPE>
                </FIELD>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0000</CODE2>
                        <ALIAS>IMAGE_WIDTH</ALIAS>
                        <DESCRIPTION>Image Width</DESCRIPTION>
                        <TYPE>INTEGER</TYPE>
                </FIELD>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0000</CODE2>
                        <ALIAS>IMAGE_HEIGTH</ALIAS>
                        <DESCRIPTION>Image Heigth</DESCRIPTION>
                        <TYPE>INTEGER</TYPE>
                </FIELD>
                <FIELD>
```

```
                        <CODE1>0x0028</CODE1>
                        <CODE2>0x0100</CODE2>
                        <ALIAS>BIT_ALOCATOR</ALIAS>
                        <DESCRIPTION>Bit allocator</DESCRIPTION>
                        <TYPE>INTEGER</TYPE>
                </FIELD>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0100</CODE2>
                        <ALIAS>SOP_CLASS_UID</ALIAS>
                        <DESCRIPTION>SOP CLASS UID</DESCRIPTION>
                        <TYPE>STRING</TYPE>
                </FIELD>
                <FIELD>
                        <CODE1>0x0000</CODE1>
                        <CODE2>0x0000</CODE2>
                        <ALIAS>SOP_CLASS_INSTANCE_UID</ALIAS>
                        <DESCRIPTION>SOP CLASS INSTANCE UID</DESCRIPTION>
                        <TYPE>STRING</TYPE>
                </FIELD>
        </FIXED>
</ONTOLOGY>
```

Documento XML de una ontología sobre un documento estructurado DICOM-SR

```
<ONTOLOGY>
        <IDONTOLOGY>4</IDONTOLOGY>
        <TYPEONTOLOGY>2</TYPEONTOLOGY>
        <DESCRIPTION>Ontologia C?ncer P?lm?n</DESCRIPTION>
        <RESTRICTIVE>
                <CONCEPT_NAME>
                        <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                        <CODE_VALUE>51</CODE_VALUE>
                        <CODE_MEANING>Informe Estructurado C?ncer Pulm?n</CODE_MEANING>
                        <TYPE>STRING</TYPE>
                        <CRITERIA>*</CRITERIA>
                        <VALUE>*</VALUE>
                </CONCEPT_NAME>
        </RESTRICTIVE>
        <CREATION>
                <CONCEPT_NAME>
                        <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                        <CODE_VALUE>0098</CODE_VALUE>
                        <CODE_MEANING>Tipo Informe</CODE_MEANING>
                        <TYPE>CODE</TYPE>
                </CONCEPT_NAME>
        </CREATION>
        <FILTER>
                <CONCEPT_NAME>
                        <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                        <CODE_VALUE>090203</CODE_VALUE>
                        <CODE_MEANING>Fumador</CODE_MEANING>
                        <TYPE>BOOLEAN</TYPE>
                </CONCEPT_NAME>
                <CONCEPT_NAME>
                        <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                        <CODE_VALUE>000401</CODE_VALUE>
                        <CODE_MEANING>Identificador Del Paciente</CODE_MEANING>
                        <TYPE>STRING</TYPE>
                </CONCEPT_NAME>
                <CONCEPT_NAME>
                        <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                        <CODE_VALUE>000402</CODE_VALUE>
                        <CODE_MEANING>Edad</CODE_MEANING>
                        <TYPE>INTEGER</TYPE>
                </CONCEPT_NAME>
                <CONCEPT_NAME>
                        <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                        <CODE_VALUE>000403</CODE_VALUE>
```

```
                    <CODE_MEANING>Sexo</CODE_MEANING>
                    <TYPE>STRING</TYPE>
                </CONCEPT_NAME>
                <CONCEPT_NAME>
                    <CODE_SCHEME>CVIMO_CONCEPT</CODE_SCHEME>
                    <CODE_VALUE>0002</CODE_VALUE>
                    <CODE_MEANING>Fecha Informe</CODE_MEANING>
                    <TYPE>DATE</TYPE>
                </CONCEPT_NAME>
                …..
        </FILTER>
        <FIXED>
                <FIELD>
                    <CODE1>0x0000</CODE1>
                     <CODE2>0x0000</CODE2>
                    <ALIAS>SOP_CLASS_UID</ALIAS>
                    <DESCRIPTION>SOP CLASS UID</DESCRIPTION>
                    <TYPE>STRING</TYPE>
                </FIELD>
                <FIELD>
                    <CODE1>0x0000</CODE1>
                    <CODE2>0x0000</CODE2>
                    <DESCRIPTION>SOP CLASS INSTANCE UID</DESCRIPTION>
                    <ALIAS>SOP_CLASS_INSTANCE_UID</ALIAS>
                    <TYPE>STRING</TYPE>
                 </FIELD>
        </FIXED>
</ONTOLOGY>
```

ANEXO II. Glosario de Términos

A

ACR	American College of Radiology
AES	Avanced Encrypted Standard

B

BD	Base de Datos
BDII	Berkely Database Information Index
BIRN	Biomedical Informatics Research Network

C

CA	Certificate Authority
CAMS	Cagrid Attribute Management System
CAS	Community Authorization Service
CaBIG	Cancer Biomedical Informatics Grid
CDA	Clinical Data Architecture
CFB	Cipher FeedBack
CIE9	Codigo Internacional Europeo de Enfermedades
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets

D

DAG	Directed Acyclic Graph
DES	Delivered Ex Ship
DICOM	Digital Imaging and Communication in Medicine
DICOM-SR	Digital Imaging and Communication in Medicine Structure Report
DIDA	Distributed Discovery Architecture for Grid Environments
DSA	Digital Signature Algorithm
DTD	Document Type Definition

E

EBI	European Bioinformatics Institute
ECDSA	Elliptic Curve Digital Signature Algorithm (ECDSA)
EDG	European Data Grid
EGEE	Enabling Grids for E-sciencE
EOUID	Encrypted Object Unique Identifier
EPSRC	Engineering and Physical Sciences Research Council

F

FADA	Federated Advanced Directory Architecture
FQAN	Fully Qualified Attribute Name
FTP	File Transfer Protocol

G

GGF	Global Grid Forum
GIIS	Grid Index Structure Service
GMA	Grid Monitoring Architecture
GRIS	Grid Resource Information Service
GridFTP	Grid File Transfer Protocol
GRyCAP	Grupo de gRid y Computación de Altas Prestaciones
GSI	Grid Security Infraestructure
GT	Globos Toolkit
GUMS	Grid User Management System

H

HIMSS	Healthcare Information Systems and Management Society
HIS	Hospital Information System
HL7	High Level 7
HTML	HyperText Markup Language

I

ICD9	Internacional Classification of Deseases
IDEA	International Data Encryption Algorithm
IDL	Interface Definition Language
IDXSRV	Index Service
IETF	Internet Engineering Task Force
IHE	Integrating Healthcare Enterprise
IOD	Information Object Definition
IP	Information Provider
ISO	International Organization for Standardization
IS-OSI	Systems Interconnection Reference Model
IXI	Information eXtraction from Images

J

JPEG	Joint Photographic Experts Group

L

LCAS	Local Credential Authorization Service
LCG	LHC Computing Grid
LDAP	Lightweight Directory Access Protocol
LDIF	Lightweight Directory Interchange Format
LOINC	Logical Observation Identifiers Names and Codes

M

MAC	Message Authentication Code
MDM	Medical Data Manager
MDS	Monitoring and Dicovery System
MICL	Medical Imaging Component Language
MITM	Man-In-The-Middle
MOTEUR	hoMe-made OpTimisEd scUfl enactoR

N

NEMA	Nacional Electrical Manufactures Association
NeSC	National e-Science
NeuroBase	An Information System for Managing Distributed Knowledge and Data Bases in Neuroimaging

O

OGF	Open Grif Forum
OGSA	Open Grid Architecture
OGSA/DAI	Open Grid Architecture / Data Acces Infraestructura
OGSI	Open Grid Service Infraestructure
OSI	Open System Interconnection
OV	Organización Virtual
OWL	Web Ontology Language

P

P2P	Peer to Peer
PACS	Picture Archives Computer System
PERMIS	Privilege and Role Management Infraestructure Standards Validation
PET	Positron emission tomography
PGP	Pretty Good Privacy
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infraestructure

R

RC4	Ron's Code 4
RDF	Resource Description Framework
RFCGR	Rosalind Franklin Centre for Genomic Research
R-GMA	Relational Grid Monitoring Architecture
RIS	Radiology Information System
RM	Resonancia Magnética
RMI	Java Remote Method Invocation
RP	Resource Provider
RPC	Remote Procedure Call
RSNA	Radiological Society of North America

S

SGML	Standard Generalizad Markup Language
SNOMED	Systematized Nomenclature of Medicine
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRB	Storage Resource Broker
SSL	Secure Socket Layer

T

TAC	Tomografia Axial Computerizada
TCP	Transmisión Control Protocol
TLS	Transport Layer Security
TRENCADIS	Towards a gRid ENvironment to proCess and shAre DIcom objectS

U

UPV	Universidad Politécnica de Valencia

V

VOMS	Virtual Organization Membership Service

W

W3F	World Wide Web Consortium
WMS	Workload Management System
WS	Web Service
WSDL	Web Services Description Language
WSRF	Web Service Resource Framework

X

XML	Extensible Markup Language
XPATH	XML Path Language
XSL	eXtensible StyleSheet Language
XSLT	eXtensible StyleSheet Language Transformations