Desarrollo e integración en TRENCADIS de servicios GRID para el almacenamiento, indexación y búsqueda de objetos DICOM-SR asociados a estudios de imágenes DICOM de distintos centros hospitalarios empleando componentes de gLite

Alumno: José Salavert Torres(*josator@fiv.upv.es*)
Codirector: Damià Segrelles Quilis(*dquilis@itaca.upv.es*)
Director: Ignacio Blanque Espert(*iblanque@dsic.upv.es*)

14 de diciembre de 2009

II

Índice general

Capítulo 1

Introducción

1.1. Visión general

Desde su creación, el desarrollo y los avances de Internet han estado ligados a la progresiva implantación de nuevos servicios que han facilitado el acceso y la transmisión de información entre computadores remotos.

La tecnología GRID [11] conecta hasta decenas de miles de ordenadores para que compartan no sólo capacidad de cómputo, sino también información almacenada en grandes espacios de almacenamiento distribuidos. Tal y como se menciona en [13] el GRID se concibe con la perspectiva de crear una Internet de nueva generación, un conjunto de protocolos y servicios adicionales que funcionen sobre Internet y que permitan el acceso controlado a los recursos proporcionados por una gran cantidad de ordenadores independientes pero que pueden trabajar concurrentemente dependiendo del problema que se aborde y de los permisos de la organización virtual a la que pertenezca el usuario que haya lanzado la tarea.

En la actualidad, cada vez es mayor el número de ramas de la ciencia en las que se generan grandes colecciones de datos experimentales (pongamos como ejemplo la predicción metereológica, la física de altas energías o la gestión de datos médicos entre centros hospitalarios). Debido a esto surge, en el seno de cada institución, la necesidad de compartir los datos obtenidos con el resto de miembros de la comunidad científica para realizar nuevos estudios y experimentos.

Unir distintas fuentes de información ha sido siempre un problema complicado debido a la heterogeneidad que aparece al juntar distintos repositorios, ya que los sistemas de almacenamiento empleados pueden no ser compatibles. Los desarrolladores de dichos sistemas de almacenamiento pueden emplear distintas representaciones para guardar un mismo elemento de datos, y también, las relaciones entre los datos almacenados pueden ser diferentes. Además, distintos desarrolladores pueden utilizar términos diferentes para referirse a los mismos conceptos, lo que dificulta la búsqueda de la informa-

ción.

Una forma de unir las fuentes de información es utilizar las tecnologías y servicios GRID, un enfoque que ha conducido al desarrollo de sistemas de información que se clasifican bajo el término Data GRID (GRID de datos) [4] [19].

El término Data GRID se emplea para enfatizar que se trata de una extensión y especialización del GRID, surgida para permitir el acceso a la información almacenada en repositorios distribuidos de forma que el usuario tenga la sensación de acceder a una única fuente de información. Para ello utiliza un espacio de nombres lógico compartido por todos los elementos de almacenamiento (SE) de manera que todo elemento almacenado en el Data GRID dispone de un identificador único.

Cuando hablamos de los elementos de un Data GRID conviene distinguir entre datos y metadatos. Los datos son los elementos que se almacenan y clasifican. En un Data GRID tienen cabida ficheros, catálogos o bases de datos entre otras cosas. Los metadatos son los datos que hablan sobre los datos almacenados y que nos permiten realizar búsquedas e indexar los mismos. Por ejemplo, si queremos buscar imágenes médicas referentes a un órgano determinado, necesitamos metadatos que describan el contenido de las mismas y que apunten al identificador correspondiente que las representa, para ello se podrían utilizar, como fuente de metadatos, los informes diagnósticos asociados a cada imagen.

Los Data GRID se apoyan en infraestructuras GRID existentes para funcionar, lo que les permite abordar las dificultades relacionadas con la compartición de datos entre recursos geográficamente distantes más fácilmente.

Un Data GRID debe enmascarar el mecanismo utilizado para almacenar los datos (protocolos de transferencia de ficheros, bases de datos empleadas, catálogos), separando y aislando en una capa software inferior el código relacionado con estas tareas. De esta forma será más fácil hacer compatibles con el Data GRID nuevos tipos de SE.

Los datos deben poder organizarse de formas diferentes. Por ejemplo, si damos soporte a réplicas, se debe permitir a las aplicaciones que utilicen la infraestructura del Data GRID elegir que política de replicación es más conveniente, en función de las redes de interconexión, el volumen de los datos, los accesos o la distancia entre los nodos. Una buena idea es incorporar mecanismos para la reserva de recursos, tanto de almacenamiento como de la red de conexión, con la finalidad de obtener garantías de velocidad cuando nos enfrentamos a transmisiones de datos predecibles.

Por último y muy importante, un Data GRID debe dar soporte a organizaciones virtuales (VO), de forma que si un usuario no es miembro de una VO no podrá acceder a sus recursos. Esto es vital en un entorno multidisciplinar para garantizar que la explotación de los almacenes de datos asignados a una determinada organización sólo sea realizada por personas autenticadas en la misma y con privilegios suficientes.

Una vez que un usuario está autenticado en el sistema debemos controlar qué acciones está autorizado a realizar y sobre qué datos. En este sentido podemos pensar en datos clínicos de un paciente ingresado, a los que un médico puede acceder por ser miembro del hospital (como el nombre del paciente) pero que un médico o investigador externo no puede consultar ya que se debe respetar la privacidad del paciente. Utilizar herramientas y servicios GRID que ya implementan los mecanismos de autenticación y autorización necesarios para cumplir con estos requisitos, es clave en el desarrollo de un Data GRID.

Hoy en día, el estándar DICOM (*Digital Imaging and Communications in Medicine*) [23] se utiliza mundialmente para la transmisión e intercambio de imágenes médicas y permite a un gran número de usuarios y aplicaciones compartir información. La mayoría de las instituciones médicas europeas utilizan el formato digital en sus bases de datos de imágenes. Estas bases de datos pueden estar en sitios diferentes dentro de un mismo centro, o incluso en centros diferentes, y pueden ser accedidas por distintos tipos de usuarios. Esta heterogeneidad en el origen, localización de los datos y privilegios de los usuarios, dificulta el compartir las imágenes de forma transparente. Este modelo se ajusta perfectamente a la estructura de un Data GRID.

Dentro del estándar DICOM ha surgido otra especificación que se conoce como DICOM-SR (*DICOM Structured Reporting)* [6], cuya finalidad es describir como deben almacenarse en objetos DICOM los informes asociados a estudios DICOM de imágenes médicas. Un informe estructurado DICOM tiene un esquema interno que es susceptible de ser representado en XML. Dicho esquema interno puede verse como un árbol.

El middleware TRENCADIS (***T****owards a g****R****id* ***EN****vironment to pro****C****ess and sh****A****re* ***DI****com object****S***) [21] se engloba en el contexto de los Data GRID. Este middleware ha sido desarrollado en el seno del grupo de investigación GRyCAP (*Grupo de Redes y Computación de Altas Prestaciones*) del instituto ITACA (*Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas*).

El propósito de TRENCADIS es proveer los servicios suficientes para la gestión, manejo y proceso de información en formato DICOM entre distintos centros hospitalarios, de una forma segura y organizada. Este middleware proporciona una interfaz de alto nivel que permite la creación de aplicaciones médicas que administren de forma transparente la información existente en distintos centros.

TRENCADIS sigue la filosofía de los Data GRID, empleando tecnologías GRID existentes para realizar sus operaciones. En la actualidad disponemos, entre otros, de Globus [9] y gLite [3] como plataformas para el desarrollo de aplicaciones GRID.

En concreto se han seguido las especificaciones del OGSA (*Open GRID Services Architecture*) [12] para definir servicios GRID que se han implementado utilizando la versión 4.2 del Globus Toolkit (GT4.2).

A lo largo de este documento se analizan las modificaciones de TRENCADIS motivadas por esta tesis.

1.2. Motivación

En esta sección se describe el problema que ha motivado el desarrollo de esta tesis.

En la actualidad, los centros hospitalarios obtienen gran cantidad de imágenes de sus pacientes. Además, los médicos que observan estas imágenes realizan diagnósticos en base a ellas y escriben informes. Esta información es relevante de cara a futuras consultas del paciente y ayuda a contrastar las opiniones de distintos médicos, también es útil para la realización de estudios y experimentos.

Es deseable para un médico el poder buscar imágenes de casos similares al que esté estudiando, especialmente durante su función y cuando esté realizando estudios de investigación. Sin embargo, las imágenes por sí mismas no contienen información que permita ordenarlas y clasificarlas, por lo que es necesario emplear la información contenida en los informes diagnósticos asociados a ellas para poder catalogarlas.

Las técnicas de búsqueda por contexto plantean problemas porque dos imágenes que presentan un contenido muy diferente pueden hacer referencia a la misma patología.

Así pues, necesitamos una aplicación que nos permita catalogar imágenes médicas en base a sus informes. Además, muchos estudios requieren de la colaboración de diferentes centros para conseguir una muestra variada, estadísticamente representativa y suficientemente grande, por lo que es importante poder consultar también casos de diferentes centros. En un escenario de este tipo, lo ideal es que el médico pueda realizar búsquedas distribuidas de forma transparente y segura.

Para poder acceder a los datos almacenados en diferentes centros, un médico debe autenticarse y estar autorizado. La gestión de estos permisos debe realizarse a través de organizaciones virtuales que engloben los centros implicados.

Para garantizar que los datos almacenados en todos los repositorios son homogéneos, los responsables de cada centro deberán reunirse de forma periódica para establecer plantillas que definan como deben ser los distintos tipos de informe.

Para poder definir de forma estándar la información del informe la aplicación debe trabajar con el estándar DICOM-SR, que es una ampliación del estándar DICOM de ficheros de imágenes que permite trabajar con informes estructurados.

Por ello, se ha considerado la tecnología GRID como base para el desarrollo de esta tesis, ya que tal y como se ha descrito en el punto 1.1, es

una tecnología apropiada para resolver este tipo de problemas, como lo atestiguan los numerosos proyectos GRID que trabajan en temas similares y que serán escritos en el punto 2.2. Más concretamente, se realizarán aportaciones a TRENCADIS, las cuales permitan enriquecer este *middleware* para la resolución del problema descrito de forma adecuada, solventando algunas carencias existentes a través de nuevos componentes y redefiniendo parte de la arquitectura.

1.3. Organización de los contenidos de la tesis de máster

En el capítulo 1, se introduce el tema de la tesis y se dan algunas definiciones. Los artículos referenciados en esta sección son una buena introducción al GRID y a los conceptos que se utilizan al desarrollar Data GRID. También se describe el problema que motiva el desarrollo de esta tesis.

En el capítulo 2, se describen las tecnologías existentes en la actualidad que han permitido el desarrollo de esta tesis. Posteriormente, se analizan otros proyectos en el área de los data GRID relacionados con la imagen médica.

En el capítulo 3, se centran los objetivos que deben cumplirse en el desarrollo de TRENCADIS motivado por esta tesis para resolver el problema descrito en la introducción.

En el capítulo 4, se describe la funcionalidad del *middleware* en todas sus capas tras cumplir con los objetivos definidos.

En el capítulo 5, se detallan las aportaciones concretas realizadas al cumplir con los objetivos. Esta sección tiene la finalidad de dejar claro el trabajo realizado y comentar las mejoras del *middleware* respecto a su implantación anterior.

En el capítulo 6, se referencian los artículos generados y el proyecto al que está asociado el trabajo de esta tesis.

En el capítulo 7, se proponen diferentes mejoras en los componentes del middleware que podrían realizarse en futuros desarrollos.

Finalmente, el apéndice A muestra la sintaxis de las representaciones en XML de los DICOM-SR que se han definido para poder trabajar con los contenidos de los informes estructurados.

Capítulo 2

Estado del arte

En este capítulo se analizan las tecnologías y estándares que han permitido los desarrollos realizados en el marco de esta tesis de máster.

En segundo lugar, se procede a analizar otros proyectos similares en los que se trabaja en la actualidad. Se describen los puntos en común y las ideas que son útiles y ventajosas para este proyecto. Además, se remarcan las innovaciones y ventajas que justifican la mejora y utilización de TRENCADIS y que dichos proyectos no contemplan (bien porque no abordan exactamente el mismo problema o bien porque lo hacen de una manera distinta).

2.1. Tecnologías y estándares actuales

2.1.1. Servicios GRID en GLOBUS TOOLKIT 4.2

A la hora de desarrollar los servicios necesarios en esta tesis, se ha empleado Globus Toolkit 4.2. GT4.2 es un conjunto de herramientas que pueden utilizarse para desarrollar aplicaciones GRID. Funciona completamente sobre java. Cualquiera que desee programar en GT4.2 debe consultar el manual oficial de la *Globus Alliance* [25].

Las funcionalidades de GT4.2 son las definidas en el estándar OGSA (*Open GRID Services Architecture*) [12], este estándar fue descrito por el *Open Grid Forum* y pretende definir una arquitectura común, abierta y estándar para las aplicaciones GRID. La meta de OGSA es estandarizar prácticamente todos los servicios que normalmente se pueden encontrar en una aplicación GRID.

El estándar OGSA requiere servicios con un estado asociado. Para cumplir este requisito se decidió basarse en la tecnología de servicios WEB con estado. Un servicio WEB no es más que otra herramienta para la computación distribuida (como CORBA, RMI o EJB). Sin embargo, mientras que tecnologías como CORBA o EJB están orientadas a sistemas distribuidos *fuertemente acoplados* (en los que el cliente y el servidor son dependientes el uno del otro), los servicios WEB se orientan a sistemas distribuidos *débilmente acoplados*

(en los cuales el cliente no tiene porque saber acerca del servidor hasta el momento en el que lo invoca).

Los servicios WEB son independientes de la plataforma y del lenguaje en el que se hayan escrito, ya que utilizan XML para definir sus interfaces. Esto significa que un cliente puede estar escrito en C++ y ejecutarse en Windows mientras que un servicio WEB puede estar programado en java y ejecutándose en linux.

Además, los servicios WEB emplean SOAP [15] sobre HTTP para transmitir los mensajes. Esto es una gran ventaja si se quiere construir una aplicación que se comunique a través de internet, ya que la gran mayoría de los cortafuegos de internet no dan problemas con ese protocolo (por ejemplo, si utilizamos CORBA encontramos esta clase de problemas).

Una de las principales desventajas de usar servicios WEB es el uso de XML para enviar los mensajes, ya que esto añade una sobrecarga computacional que no existe en otros formatos binarios de mensajes. Por otro lado, los servicios WEB son menos versátiles y, de momento, ofrecen menos maneras de invocar al servidor.

Cuando hablamos de un servicio WEB **con estado** estamos hablando de un servicio que conserva información entre invocación e invocación. Dicha información es relevante a la hora de funcionar y de responder a las llamadas que recibe.

Sin embargo, aunque los servicios WEB pueden tener un estado asociado, la forma de gestionar y mantener ese estado no estaba definida de forma estándar. Para ello existe la especificación WSRF(*Web Services Resource Framework*).

La especificación WSRF fue descrita por OASIS (*Advanced Open Standards for the Information Society*) y define como se puede añadir estado a los servicios WEB. El WSRF es fruto de la unión de los esfuerzos de la comunidad encargada del desarrollo de los servicios WEB y de la comunidad dedicada al GRID.

Se puede decir que WSRF proporciona los servicios con estado que OGSA necesita, es decir, OGSA es la arquitectura y WSRF la infraestructura sobre la que se construye la arquitectura. Fruto de la unión de estas especificaciones aparecieron lo que se conoce como los servicios GRID de GT4.2. En GT4.2 podemos encontrar una implementación completa de WSRF y gran parte del OGSA.

Además, en GT4.2 disponemos de una serie de servicios de alto nivel que podemos utilizar para desarrollar aplicaciones GRID. Estos servicios proporcionan muchos de los requisitos definidos en OGSA y por eso se incluyen entre las herramientas ofrecidas en GT4.2. Entre ellos podemos encontrar servicios de monitorización de recursos, de envío de trabajos o de seguridad. El más importante para los servicios desarrollados en esta tesis, es sin duda el de monitorización de recursos o MDS. El MDS permite a los servicios GRID publicar información acerca de su estado actual para que, otros servi-

cios puedan utilizar dicha información. Por ejemplo, un servicio GRID puede publicar su URI de forma que otros servidores la consulten y puedan invocar sus métodos. Una invocación típica a un servicio GRID (WEB) sería la que se observa en el diagrama 2.1.

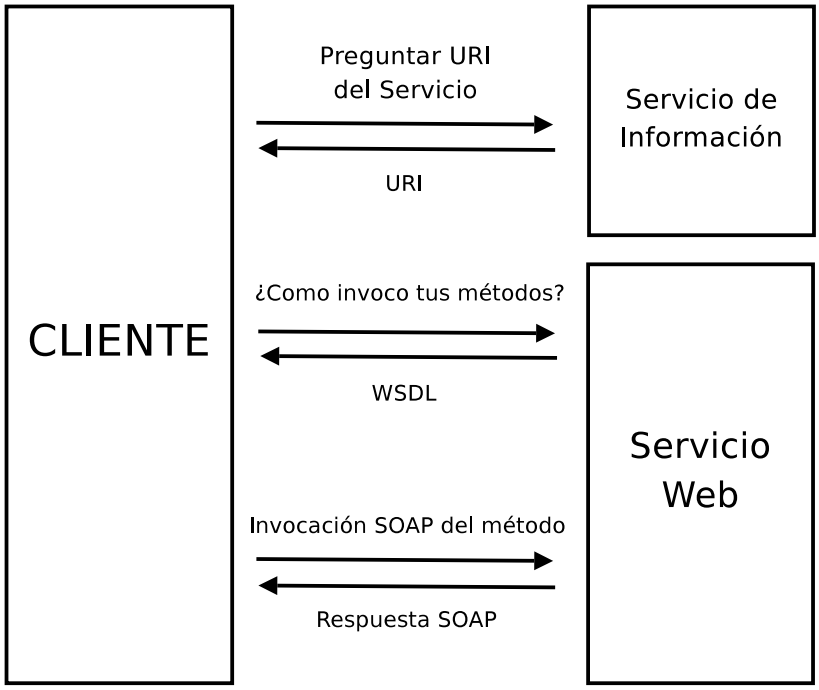


Figura 2.1: Invocación común de un servicio GRID

Los sistemas GRID, son sistemas distribuidos débilmente acoplados en los que el cliente no tiene por que saber donde está el servidor ni como se invoca. En primer lugar el cliente pregunta al servicio de información por la URI del servidor. Después le pregunta la forma de invocarlo, que viene definida en un fichero WSDL (*Web Services Description Language*). Una vez conocida la interfaz del servidor el cliente realiza una invocación mediante el uso del protocolo SOAP.

Cuando hablamos de servicios GRID, existen una serie de términos que debemos distinguir: servicio, servicio factoría, recurso del servicio y propiedades del recurso.

Cuando hablamos de **servicio** (GRID) nos referimos a la parte común de su implementación, esto incluye su interfaz y las variables, métodos y servicios remotos accedidos que sean utilizados por la interfaz de llamada y por tanto sean comunes a todos los recursos del servicio.

Un **servicio factoría** (*factory service*) es un tipo especial de servicio cuya finalidad es la creación de recursos del servicio al que está asociado. No obstante es también un servicio GRID y por tanto puede recibir invocaciones,

realizar acciones y publicar su información en el MDS.

Se llama **recurso** de un servicio a la parte individual de los servicios GRID. Un servicio puede tener más de un recurso. Los servicios GRID son servicios WEB con estado. Concretamente un servicio puede tener más de un estado asociado, uno por cada recurso del servicio que se haya creado. Para acceder a estos recursos se emplea la interfaz común que proporciona el servicio.

Todos los servicios deben tener al menos un recurso. Si un servicio sólo tiene un recurso y no dispone de servicio factoría se dice que dicho servicio es de tipo **singleton**.

Las **propiedades de recurso** (*resource properties*) son el conjunto de variables que describen el estado de un recurso. Estas propiedades deben indicarse cuando se describe el servicio. Las propiedades de recurso disponen de las funcionalidades de GT4.2 para trabajar con ellas, pudiendo publicarse en el MDS para que otros servicios accedan a las mismas.

2.1.2. Infraestructura de seguridad del GRID

2.1.2.1. Delegación de credenciales

Las características de los sistemas GRID hacen que se tenga que tratar con problemas de seguridad para los que las tecnologías de los sistemas distribuidos existentes no están preparadas.

El número de usuarios de un GRID es grande y cambia dinámicamente. Dichos usuarios pueden tener varios identificadores y pueden estar adscritos a distintas organizaciones virtuales. Los recursos disponibles del GRID también cambian constantemente, así como sus políticas de seguridad (un recurso puede cambiar de VO).

Por ejemplo, en el caso de los data GRID necesitamos acceder a la información de múltiples almacenes de datos que pueden estar en dominios administrativos diferentes. Además, la comunicación no es únicamente entre cliente y servidor, puede ocurrir que un proceso se comunique con otro y haga una consulta y este segundo proceso se conecte a los procesos de otros repositorios para elaborar una respuesta.

Para poder trabajar en estas condiciones se emplea criptografía de clave asimétrica y se delega una credencial de duración limitada que se conoce como *proxy*. Un proxy se crea a partir de las credenciales del usuario. Los proxys son de duración limitada para evitar problemas de seguridad. Cuando un usuario quiere acceder a los recursos del GRID delega un *proxy* a los recursos que quiere utilizar. A partir de ese momento, los recursos que han aceptado el proxy del usuario actúan en su nombre, pudiendo a su vez delegar la credencial a otros recursos.

En [10] se describe la infraestructura de seguridad del GRID. También se explican con rigurosidad los procesos de creación y delegación de proxys.

2.1.2.2. Extensiones VOMS para organizaciones virtuales

Una de las problemáticas del GRID es la compartición de recursos de forma coordinada en entornos multiinstitucionales. En estos casos se usa el concepto de las organizaciones virtuales (VO) [13]. El objetivo de las VO es definir una serie de reglas de acceso a los recursos del GRID. Según la VO con la que se acredite un usuario, este podrá utilizar unos recursos del GRID u otros. En la figura 2.2 se ilustra este concepto.

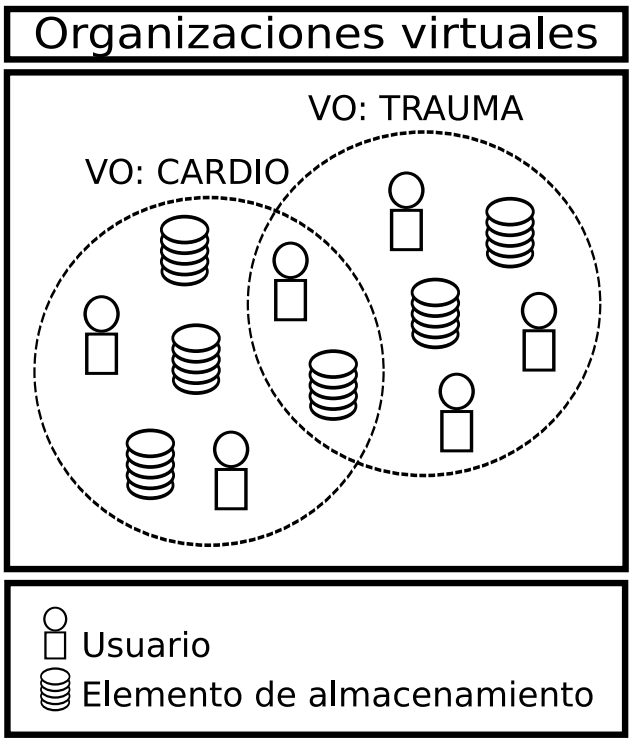


Figura 2.2: Organizaciones virtuales

Si queremos utilizar los recursos de una VO, debemos añadir a la credenciales de los usuarios la información de la VO con la que queremos que se autentiquen en el GRID. Para poder asociar a una credencial de usuario una VO, tenemos que contactar con un servidor VOMS, dichos servidores contienen la lista de usuarios dados de alta en cada VO.

La información de la VO que se añade a las credenciales de usuario se conoce como extensiones VOMS [5], estas extensiones muestran los roles del usuario y grupo dentro de la VO a la que pertenecen y las capacidades que se le otorgan al usuario dentro de la VO.

2.1.3. El middleware de gLite

El middleware gLite [3] está formado por un conjunto integrado de componentes diseñados para permitir la compartición de recursos. Es decir, este middleware permite construir la infraestructura de un sistema GRID usando sus componentes.

El middleware gLite surge a raíz del proyecto EGEE. Además del código desarrollado para el propio middleware, incluye contribuciones de otros muchos proyectos (LCG). Su modelo de distribución consiste en construir distintos servicios (cada uno de estos servicios caracteriza un tipo de nodo en el GRID) y garantizar que su instalación y configuración sea fácil en una serie de plataformas que se han elegido (Scientific Linux 4 y 5 y Debian 4 para los nodos de cómputo).

Nosotros hemos utilizado los componentes de gLite AMGA y LFC en los desarrollos de esta tesis para desplegar la infraestructura GRID.

2.1.3.1. Servidor de catálogo AMGA

El servidor de metadatos AMGA [16] proporciona una capa de abstracción sobre el modelo relacional de bases de datos. Nos referimos a esta capa mediante el término *catálogo de metadatos* (metadata catalogue).

Cuando trabajamos con un catálogo de metadatos utilizamos una serie de términos nuevos para referirnos a conceptos que ya existían en el modelo relacional de bases de datos. Estos términos tienen una nomenclatura y una representación diferente. Trabajar con un catálogo de metadatos permite simplificar el manejo y diseño de una base de datos de metadatos.

Un catálogo de metadatos AMGA permite estructurar sus contenidos en un árbol de directorios. Dicha estructura de directorios está pensada para ser equivalente a la estructura de un catálogo de datos del GRID. Por ejemplo, si en el catálogo de datos tenemos un directorio con imágenes médicas y dentro de él un fichero con el nombre 0005A10, en el catálogo de metadatos encontraremos un directorio con el mismo nombre y en su interior otro fichero 0005A10 que en vez de contener los datos de la imagen contendrá datos descriptivos como el tipo de informe, el nombre del médico o la fecha.

En un catálogo de metadatos nos referiremos a las tablas de la base de datos con el término *colecciones* (collections). Dichas colecciones tienen *atributos* (atributes) asociados (columnas de la tabla) y se representan como directorios. Las filas de la tabla se llaman *entradas* (entries) y se representan como ficheros dentro de los directorios de las colecciones.

Los directorios de las colecciones pueden contener a su vez subdirectorios además de los ficheros de las entradas, por lo que utilizando la capa de abstracción del catálogo dispondremos de una relación entre tablas que no existe en el modelo relacional. Dicha relación puede entenderse como *"Una colección está dentro del contexto de otra colección"*. Esta relación es muy

importante en el contexto de esta tesis, ya que será utilizada para organizar diferentes estructuras de informes mediante el estándar DICOM-SR dentro del catálogo AMGA. Profundizaremos en esto más adelante en la sección 4.2.1.2.

Por otra parte, AMGA implementa un mecanismo de autorización con listas de control de acceso y permisos de fichero tipo UNIX. Cada entrada del catálogo tendrá unos permisos asociados, lo que simplifica el diseño respecto al modelo relacional.

Este mecanismo proporciona el nivel de seguridad adecuado en un entorno GRID heterogéneo con usuarios de distintas organizaciones. Dichos usuarios serán autenticados por el servidor AMGA gracias a su compatibilidad con la infraestructura de seguridad GRID (soporta entre otras cosas credenciales VOMS). El servidor AMGA autorizará el acceso a los datos según los permisos establecidos por el administrador en cada colección y entrada para cada usuario y grupo.

El gestor de metadatos AMGA también tiene soporte para transacciones, lo cual es una característica a destacar, pues será utilizada en los desarrollos de esta tesis.

De hecho, el servidor de catálogo AMGA implementa muchas de las funcionalidades del estándar SQL (debe funcionar sobre PostgreSQL u otra base de datos), pero tiene su propia sintaxis para la comunicación con el servidor (cliente y servidor no se entienden mediante comandos SQL). Para saber más acerca de la instalación, configuración y funcionamiento del servidor AMGA y de la sintaxis de sus comandos puede consultarse [1].

2.1.3.2. El servidor de catálogo LFC

Los usuarios y aplicaciones del GRID necesitan una forma de localizar los ficheros de datos almacenados y sus réplicas. El LFC (*LCG File Catalogue*) cumple esta función manteniendo asociaciones entre las direcciones físicas y las direcciones lógicas de los ficheros de datos.

El LFC es de vital importancia y se dice que para que un fichero se considere un fichero del GRID debe existir físicamente en un SE y estar registrado en el catálogo de ficheros.

En los apartados 7.3 y 7.4 del manual de usuario de gLite [3] se describe la funcionalidad del LFC.

El LFC permite 4 tipos de nombres de fichero: GUID (*Grid Unique IDentifier*, LFN (*Logical File Name*), SURL (*Storage URL*) y TURL (*Transport URL*).

El GUID es el identificador único de un fichero y tiene la siguiente forma: guid:<identificador único de 36 bytes>.

El LFN es el nombre lógico del fichero, un alias elegido por el usuario. Cuando un usuario necesite un fichero normalmente utilizará el LFN en vez del GUID para localizarlo. El formato es el siguiente: lfn:<cualquier cadena>.

La SURL identifica una réplica de un fichero en un SE. Tiene la sintaxis: <sfn|srm>://<hostname del SE>/<cualquier cadena>.

La TURL es una URI válida con la información necesaria para acceder a un fichero a través de un protocolo soportado por el SE en el que reside. La sintaxis es: <protocolo>://<cualquier cadena>.

2.1.4. El estándar *DICOM-SR*

DICOM son las siglas de *Digital Imaging and Communications in Medicine*, un estándar que fue creado por la NEMA (*National Electrical Manufacturers Association*) para ayudar en la distribución y visualización de imágenes médicas, tales como radiografías o ultrasonidos.

El estándar DICOM no sólo especifica cómo deben codificarse las imágenes y otros tipos de datos relacionados con la observación de pacientes, sino que también cumple otras tareas. Por ejemplo, el estándar determina la forma en que las imágenes deben compartirse a través de la red y los dispositivos de almacenamiento. Los detalles sobre el estándar DICOM pueden consultarse en [23].

Las imágenes DICOM se agrupan en series, y estas a su vez se agrupan en estudios, en este último caso se trata de imágenes y series que se han utilizado para un mismo diagnóstico en un momento dado.

Todo objeto DICOM tiene una cabecera en la que contiene información del paciente, del estudio y de las series a las que pertenece. Esta cabecera precede a los datos codificados (por ejemplo, los píxeles de una imagen).

Los *informes estructurados DICOM* (DICOM-SR) son una parte del estándar DICOM que define la estructura y contenidos de los informes asociados a estudios de imágenes médicas. Estos informes contienen las observaciones y conclusiones relativas a los diagnósticos de los médicos que analizan las imágenes. DICOM incorpora plantillas estándar para diferentes tipos de informes, aunque permite al usuario la creación de nuevoss tipos de plantillas.

Los DICOM-SR son también objetos DICOM, por lo que tienen la misma cabecera que las imágenes DICOM, no obstante la parte con los datos codificados es diferente y contiene un informe estructurado en vez de la información de una imagen.

Los DICOM-SR están formados por diferentes elementos, como por ejemplo coordenadas espaciales (SCOORD), contenedores (CONTAINER), códigos (CODE), texto (TEXT) o nombres de persona (PNAME). Estos elementos se relacionan entre sí, formando la estructura del documento.

Todos estos elementos son esenciales a la hora de indexar las imágenes médicas, ya que contienen la información del informe estructurado que es necesaria para realizar búsquedas.

En [6] se da una descripción precisa de las reglas que deben cumplirse al crear un DICOM-SR usando los elementos que se acaban de mencionar.

En la figura 2.3 puede verse la apariencia de un DICOM-SR. En este caso los hallazgos y las conclusiones son contenedores que se relacionan con elementos texto mediante la relación "contiene". Las relaciones permiten, entre otras cosas, crear un árbol jerárquico de elementos y añadir una semántica a la ordenación de los elementos.

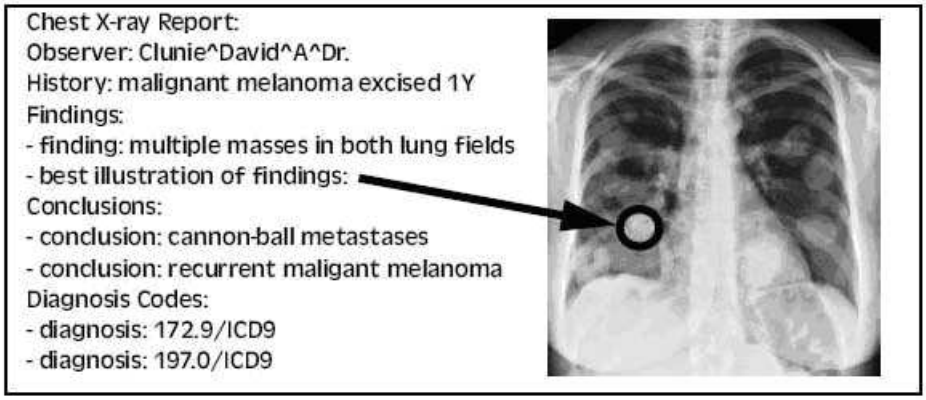


Figura 2.3: Ejemplo de DICOM-SR extraído de [6]

Trabajar con los ficheros DICOM-SR puede convertirse en una tarea compleja, ya que el estándar es muy extenso. Por esto, se ha definido una representación en XML de los DICOM-SR. De este modo, a la hora de trabajar con las representaciones disponemos de todas las herramientas y parsers del estándar XML que existen. Además, simplificamos la representación de los DICOM-SR escogiendo sólo los elementos y expresividad del estándar que necesitamos.

2.2. Análisis de otros proyectos en el mismo ámbito

2.2.1. El proyecto BIRN

Descripción del proyecto

El proyecto BIRN (*Biomedical Informatics Research Network*) [14] es un proyecto de EEUU que pretende crear una comunidad virtual de recursos compartidos en el área de la medicina. Concretamente se centra en compartir imágenes médicas que muestren la evolución de enfermedades degenerativas del cerebro.

La red está formada por centros médicos, cada uno de los cuales crea una base de datos para gestionar sus resultados experimentales y observaciones. Dichas bases de datos se agrupan en dominios y se enlazan para permitir consultas cruzadas con la información de todos los repositorios existentes.

Las fuentes de datos de cada centro médico comparten un recurso común que se utiliza para la definición de ontologías. En el contexto del proyecto

BIRN, una ontología consiste en una especificación de los términos usados para describir y representar áreas de conocimiento. Las ontologías son usadas conjuntamente por los usuarios, bases de datos y aplicaciones.

El uso de ontologías permite unificar la búsqueda de términos ya que agrupa estos en dominios. Además un término puede estar en más de un dominio, permitiendo su reutilización en búsquedas en las distintas áreas en las que el término puede tener relevancia.

Desde el punto de vista de los orígenes de datos, estos pueden utilizar las ontologías para categorizar la información que contienen. De esta forma sólo serán interrogados durante una consulta cruzada si las ontologías que describen sus contenidos son relevantes para la búsqueda en cuestión.

Para permitir las consultas cruzadas el proyecto BIRN a optado por utilizar un **mediador** en vez de un **almacén de datos**.

La estrategia de utilizar un *almacén de datos* consiste en copiar toda la información de los repositorios en un servidor central y actualizarla cada cierto tiempo. Las consultas se realizan sobre ese servidor central. No obstante, aparece un cuello de botella al realizarse la totalidad de las consultas en un único servidor.

Un *mediador* se encarga de crear la ilusión de trabajar con una única base de datos, pero agrupando y manteniendo todos los orígenes de datos existentes. Esto lo realiza utilizando lo que llaman vistas virtuales, las cuales describen como se combinan las bases de datos locales para dar la imagen de una base de datos única. La tarea del mediador es elegir las vistas virtuales según las ontologías existentes en la consulta que vaya a realizarse y las ontologías que categoricen los datos en cada repositorio. Una vez conocidas las vistas, la tarea del mediador será crear consultas para cada uno de los almacenes incluidos en la vista, recibir los resultados de cada almacén y procesarlos para devolver una respuesta equivalente a la que daría una base de datos única.

Utilizar un mediador tiene como ventajas que en cada consulta se obtiene la información actualizada de las bases de datos originales, que no hay que mantener un servidor central, que es más fácil añadir nuevos repositorios de datos (basta con que estos publiquen su localización) y que los distintos centros pueden mantener la autonomía y propiedad de sus datos. Además, dependiendo de las ontologías presentes en la consulta se interrogarán sólo los repositorios con la información.

La herramienta SRB

Muchas de las funcionalidades del proyecto BIRN han sido construidas gracias al SRB (*Storage Resource Broker*) [22]. El SRB es un middleware cliente-servidor diseñado para gestionar colecciones de ficheros en un entorno heterogéneo.

El uso del SRB permite utilizar direcciones lógicas para los archivos, creando la ilusión de trabajar con un único sistema de ficheros (la dirección física se mantiene). También gestiona metadatos con información descriptiva asociada a los ficheros. Otras de sus funcionalidades son: soporte para distintos tipos de SE de forma transparente, autenticación entre dominios administrativos y gestión de réplicas y cachés.

Comparativa

Una de las motivaciones de esta tesis se centra en la búsqueda dentro del contenido de los DICOM-SR. Por este motivo el concepto de ontología que nosotros empleamos es parecido. En este sentido, para nosotros una ontología es la estructura de un tipo determinado de informe para un área médica concreta.

BIRN se centra en obtener información relacionada con distintas áreas de la ciencia médica. Sin embargo, no permite búsquedas más exhaustivas centradas en la información contenida en los informes relativos a las imágenes, ya que no maneja informes, sino anotaciones. Vemos una carencia, por ejemplo, al buscar las imágenes en las que haya aparecido un melanoma de cierto tamaño, ya que no podemos realizar consultas sobre el contenido de informes médicos genéricos.

Como punto en común, nosotros también encontramos interesante utilizar un mediador a la hora de realizar las búsquedas en los almacenes de datos.

2.2.2. El proyecto NeuGRID

Descripción del proyecto

El proyecto NeuGRID [26] es un proyecto europeo que pretende convertirse en el Google de las imágenes médicas del cerebro. Consiste en el desarrollo de infraestructuras electrónicas que faciliten a la comunidad europea de neurociencia el acceso a la información necesaria para el estudio de las enfermedades degenerativas del cerebro.

Hablando en términos computacionales, este proyecto está formado por distintos servicios que siguen la filosofía SOA (Service Oriented Architecture). Los servicios de este proyecto son débilmente acoplados, reutilizables, escalables, fácilmente ampliables e interoperables. Además no dependen del middleware sobre el que funcionan porque implementan una capa de compatibilidad y usan estándares generales y abiertos.

Los tipos de servicios que conforman el proyecto NeuGRID se describen a continuación.

El *servicio de proceso* permite a los científicos del proyecto NeuGRID definir procesos automatizados para el tratamiento de las imágenes médicas. Estos procesos están formados por tareas atómicas predefinidas.

El *servicio de cohesión* proporciona una manera cohesionada de acceder al middleware para los distintos servicios que conforman el proyecto NeuGRID, de esta forma se potencia la reusabilidad y se facilita la creación de nuevos servicios (ya que se enmascara la problemática del middleware).

El *servicio de procedencia* permite registrar la evolución de los procesos de tratamiento de las imágenes, detectando y mostrando los errores que puedan aparecer.

El *servicio de consulta* permite enviar una consulta a los distintos orígenes de datos, dependiendo de la semántica de la misma se interrogará a unos servicios o a otros.

El *servicio del portal web* proporciona a los usuarios una forma amigable de usar el sistema y realizar las búsquedas y tareas.

El *servicio de anonimato* se encarga de garantizar el anonimato de los pacientes. Para ello borra los rostros de las imágenes y sustituye la información personal por identificadores pseudoaleatorios.

Comparativa

Esta tesis está motivada en coordinar los esfuerzos de distintos centros médicos en la fase de diagnóstico, no en la fase de obtención y procesamiento de la información previa al diagnóstico. Los centros médicos deben ponerse de acuerdo para crear plantillas de lo que será la estructura de cada tipo de informe. Usar estas plantillas proporciona homogeneidad a la forma en que se almacenan los datos en cada uno de los centros. Echamos en falta una estructuración de este tipo en NeuGRID.

Sería deseable también poder establecer permisos de acceso en los distintos campos de los informes, cosa que en este proyecto no es posible. De esta forma, los datos privados del paciente sólo serían accesibles a personas autorizadas, apareciendo para el resto de usuarios como informes anónimos.

Un punto en común es que el software que se plantea en esta tesis sigue la filosofía SOA. Esto queda patente al utilizar estándares como el OGSA o el WSRF.

El software está estructurado en distintas capas, de entre ellas cabe mencionar una capa de compatibilidad que permite el desarrollo de aplicaciones olvidando la problemática de funcionamiento de los servicios GRID inferiores. Esta capa permitiría reutilizar las aplicaciones aunque se decidiera cambiar el funcionamiento de los servicios o se decidiera portar a otra plataforma los mismos. Consideramos interesante desarrollar una capa de esta características.

2.2.3. El proyecto Health-e-child

Descripción del proyecto

El proyecto Health-e-Child pretende crear una base de datos de conocimiento con información biomédica relativa a enfermedades del campo de la pediatría, cubriendo enfermedades de tipo genético, clínico y epidemiológico relacionadas con el corazón.

En la creación de este sistema de información colaboran distintos hospitales, los cuales definen por mutuo acuerdo una serie de plantillas para lo informes diagnósticos asociados a imágenes médicas del corazón de los pacientes. Como todos los hospitales utilizan las mismas plantillas para crear los informes de un determinado tipo, se pueden realizar búsquedas globales que se centren en campos concretos de los tipos de informe que existan en cada uno de los repositorios.

Comparativa

Este proyecto se acerca más a la filosofía de lo que nosotros buscamos, sin embargo sería más conveniente utilizar el estándar DICOM-SR, que está dentro del estándar DICOM de almacenamiento de imágenes y datos médicos, para trabajar con los datos de los informes. Los DICOM-SR son objetos binarios que pueden ser utilizados por otras aplicaciones que sigan el estándar DICOM, lo que aumentaría la interoperabilidad de nuestra aplicación.

Como ya hemos mencionado antes, nosotros necesitamos poder establecer los permisos de acceso a nivel de los campos del informe, no a nivel de los informes enteros.

Claramente, el punto en común con nuestro sistema es la forma en la que se definen los informes, en nuestro despliegue trabajamos con radiólogos que son los encargados de definir las plantillas que van a utilizarse en cada tipo de observación y diagnóstico.

2.2.4. El proyecto NeuroLOG

Descripción del proyecto

El proyecto NeuroLOG [18] es un proyecto francés cuya finalidad es crear un middleware que permita compartir imágenes médicas relacionadas con enfermedades como la esclerosis múltiple, el derrame vascular o los tumores cerebrales.

Para la comunicación entre sus componentes específicos el middleware utiliza RMI, mientras que para las invocaciones remotas de los usuarios utiliza servicios web.

Para el desarrollo del middleware se han observado los procesos de tratamiento de información realizados por los usuarios, obteniendo sus requisitos típicos. Normalmente los usuarios necesitan: organizar las imágenes en conjun-

tos, controlar el acceso a los datos (utiliza VO), poder compartir y federar [19] la información, capacidad para definir y ejecutar eficientemente los procesos sobre las imágenes y capacidad para analizar los resultados obtenidos.

Los tipos de información gestionados por este middleware son: ficheros de datos (en su mayoría imágenes) conteniendo la información médica, metadatos con información procedente de distintos orígenes (información asociada a imágenes, datos generados por aplicaciones de proceso de imágenes, información administrativa...) y, por último, información semántica para las consultas.

La herramienta MDM

El middleware NeuroLOG utiliza el MDM (*Medical Data Manager*) [17] para gestionar la información. Este gestor utiliza componentes de gLite tales como el catálogo AMGA para los metadatos, el servidor gLiteIO para abrir flujos de datos (streams) y manejar los ficheros o el almacén de claves de encriptación Hydra.

Además soporta los protocolos de comunicación DICOM pudiendo acceder directamente a las fuentes de adquisición de imágenes (aparatos de ecografía, resonancia o radiografía).

El MDM gestiona el almacenamiento de los datos y a los metadatos asociados. Los datos son los ficheros con las imágenes, mientras que los metadatos son un conjunto de tablas de una base de datos relacional, a saber: Una tabla de pacientes, una de imágenes, una tabla médica y una tabla DICOM.

La tabla de pacientes contiene la información del paciente (nombre, sexo, edad). La tabla de imágenes contiene información técnica de la imagen como el tamaño, la codificación. La tabla médica tiene información adicional como la forma de adquisición, la máquina o los radiólogos involucrados. La tabla DICOM contiene los identificadores DICOM necesarios para interrogar a los repositorios de datos.

Con la información de estas tablas se realizan las búsquedas de las imágenes en el sistema.

Comparativa

De todos los proyectos, la herramienta MDM del proyecto NeuroLOG es lo más parecido al planteamiento que nosotros proponemos, ya que también usa el catálogo AMGA para gestionar los metadatos.

No obstante podemos ver claramente que aunque esta herramienta adopta el estándar DICOM a un nivel mayor (ya que tiene que tratar directamente con los dispositivos de adquisición de imágenes e incorpora los protocolos DICOM de comunicación) los metadatos asociados a las imágenes están compuestos únicamente por una serie de tablas de una base de datos relacional con algunas de sus características.

Nosotros buscamos utilizar como metadatos los contenidos de los DICOM-SR de los diagnósticos respetando su estructura. Es decir, queremos que los metadatos que utilicemos estén estandarizados en el formato DICOM-SR y podamos exportarlos a un fichero binario DICOM en cualquier momento. Gracias a esto podemos realizar las búsquedas en base a la información que aparezca en cualquier campo del informe estructurado.

Como punto en común destaca el uso de componentes de gLite. El uso de estos componentes es deseable de cara a respetar la definición de los Data GRID.

Es importante destacar que investigadores de este proyecto se han puesto en contacto para establecer posibles líneas de colaboración.

2.2.5. El proyecto CaBIG

Descripción del proyecto

El proyecto CaBIG [8] es un proyecto americano cuyo objetivo es el desarrollo de una red colaborativa que permita acelerar el descubrimiento de nuevas técnicas para la detección, diagnóstico, tratamiento y prevención del cáncer.

El proyecto busca un balance entre las decisiones de desarrollo del equipo central y las iniciativas locales, el desarrollo es abierto y se realizan votaciones para tomar las decisiones de diseño. Sigue una filosofía de código abierto, permitiendo no obstante desarrollos comerciales a partir del software del proyecto. Permite el acceso libre a los datos, siguiendo las políticas de privacidad oportunas e indicando el laboratorio de origen. Por último emplea técnicas de federación [19] ya que los servicios están geográficamente distribuidos y cada centro debe conservar cierto nivel de control de acceso, autonomía y responsabilidad local.

La infraestructura caGRID

CaGRID [24] es el nombre de la infraestructura GRID que sustenta al proyecto CaBIG. Es una infraestructura basada en servicios GRID programados con Globus Toolkit. Existen tres tipos de servicios: servicios de análisis, servicios de datos y servicios de descubrimiento.

Para garantizar la coherencia entre los distintos servicios a la hora de comunicarse se establecen unas normas de compatibilidad que deben cumplir todos los nodos. Estas normas especifican modelos de información, terminologías, ontologías y elementos comunes a toda la comunidad CaBIG. Los tipos de datos se especifican en UML y se usan representaciones en XML para su envío a través del GRID.

Comparativa

El proyecto CaBIG se orienta hacia la unificación de almacenes de datos heterogéneos. Su finalidad es la de unir a caGRID repositorios que ya existan en los centros médicos.

Cada centro es responsable de desarrollar los servicios que sean compatibles con la infraestructura de caGRID y que permitan el acceso de una forma estándar a la información de que disponen.

Sin embargo esto no es lo que nosotros buscamos, nosotros queremos crear plantillas que describan distintos tipos de informes. Estas plantillas se definirán por mutuo acuerdo entre todos los centros y todos las utilizarán a la hora de guardar la información en sus repositorios. Debido a esto los SE que nosotros vamos a manejar serán homogéneos y no necesitamos establecer reglas para estandarizar la información y la comunicación, como ocurre en el caso de CaBIG.

Además nosotros vamos a emplear componentes de gLite para almacenar los estudios de imágenes (LFC) y los metadatos (AMGA). Nuestro interés no se centra en adaptar repositorios de datos a nuestra infraestructura, sino crear nuevos repositorios homogéneos en los que nosotros elegimos los componentes.

2.2.6. Implantación CVIMO de TRENCADIS

Descripción del proyecto

El middleware TRENCADIS es una arquitectura de servicios GRID que permite la creación de repositorios distribuidos de ficheros DICOM. Sigue los estándares WSRF y OGSA. Los ficheros de imágenes DICOM son ficheros de gran tamaño.

La arquitectura de TRENCADIS, originalmente se estructuraba en cinco capas que son: (*Core middleware*, *Server services*, *Communication*, *Middleware components* y *Applications*). Estas capas se describen posteriormente en la sección 4.1.

La implantación CVIMO (*Valencian Cyberinfraestructure of Oncological Medical Images*) [2] de TRENCADIS es un proyecto subvencionado por el ministerio de empresa, universidad y ciencia de la comunidad valenciana.

El objetivo del proyecto es desplegar una infraestructura que permita compartir de forma segura estudios de imágenes DICOM entre cinco hospitales de la comunidad valenciana.

Las imágenes extraídas pertenecen al campo de la oncología y muestran la evolución del cáncer en distintas áreas del cuerpo como el pulmón, el hígado o el sistema nervioso central.

Dichas imágenes se catalogan mediante DICOM-SR. Los informes se almacenan por filas en una base de datos relacional.

Comparativa

Al almacenarse los DICOM-SR por filas de forma llana, los metadatos no contienen la información sobre la estructura en árbol del informe. Nosotros vamos a emplear la estructura de directorios del catálogo AMGA para que los metadatos almacenados tengan la misma estructura que el árbol de elementos del DICOM-SR original.

Otra de las carencias que vamos a mejorar es la falta de integración con componentes estándar de gLite como el AMGA o el LFC, lo que permitiría integrar el sistema en infraestructuras existentes como EGEE [7].

Capítulo 3

Objetivos

Una vez que hemos analizado las virtudes y carencias de los desarrollos que existen actualmente y que tenemos conocimiento de las tecnologías a nuestro alcance, podemos fijar los objetivos del desarrollo que motiva esta tesis de máster.

Los objetivos principales son:

1. Modificar la arquitectura TRENCADIS para integrar infraestructuras mediante el middleware gLite (como el AMGA y el LFC). Para ello habrá que modificar el funcionamiento de los servicios existentes.

2. Agregar y modificar los servicios para almacenar objetos DICOM-SR respetando su estructura en árbol, para lo que se deben modificar y diseñar nuevos componentes TRENCADIS en las diferentes capas de su arquitectura. También deben redefinirse los flujos de información entre los diferentes servicios GRID desplegados.

Para ello fijamos los siguientes subobjetivos, los cuales deberá cumplir el desarrollo del *middleware* de esta tesis:

1. El middleware TRENCADIS proporcionará a las aplicaciones que lo utilicen una forma sencilla de gestionar estudios DICOM de imágenes médicas que se encuentren almacenados en los repositorios de distintos centros. Para ello, se definen por mutuo acuerdo plantillas que describen los tipos de informe que estarán disponibles en todos los repositorios. Vamos a crear un servicio centralizado donde se almacenarán las plantillas, y que será consultado por todos los repositorios para averiguar la estructura de los tipos de informe. De esta forma, la información se almacenará de forma homogénea en todo el data GRID.

2. El sistema permitirá realizar consultas distribuidas de forma transparente para el usuario a través de un servicio de búsquedas que actuará como mediador. Este mediador se encargará de repartir las consultas

referentes a un tipo concreto de informe entre los repositorios y de, posteriormente, combinar los resultados antes de retornarlos. El usuario tendrá la impresión de acceder a una única base de datos.

3. Otro de los objetivos que nos fijamos es utilizar componentes de gLite. Los componentes de gLite van a sustituir a componentes incluidos en TRENCADIS y van a modificar el código de los servicios que vayan a utilizarlos. Concretamente, vamos a integrar en el middleware el catálogo AMGA, que nos permitirá gestionar los metadatos, y el LFC, con el que organizaremos los datos en los distintos SE.

4. Los metadatos se formarán a partir del contenido de los DICOM-SR asociados a los estudios de imágenes, usaremos estos metadatos para localizar los estudios de imágenes DICOM. Habrá un servicio que se encargará de almacenar y gestionar dichos metadatos en el catálogo AMGA, utilizando la organización por directorios que proporciona y manteniendo la estructura en árbol de los elementos que componen los DICOM-SR.

5. Para facilitar el desarrollo de los servicios se programará el módulo *DSR AMGA Manager*, con el cual se podrán introducir en el servidor AMGA las estructuras de directorios de cada tipo de informe y sus correspondientes entradas. Se trabajará con descripciones en XML de los DICOM-SR y sus plantillas. Este módulo utilizará parsers SAX que irán lanzando comandos al servidor AMGA conforme se recorre la estructura del documento. Todas las acciones del módulo *DSR AMGA Manager* serán transaccionales.

6. Una vez que los datos de los DICOM-SR están en el servidor de metadatos AMGA podremos lanzar consultas al catálogo que nos devuelvan el identificador del estudio asociado al informe que buscamos. También podremos obtener el contenido íntegro de la representación en XML de un DICOM-SR en una cadena de texto que después podremos convertir en un fichero codificado DICOM.

7. Se desarrollará una capa de compatibilidad que se encargará de los aspectos relacionados con la comunicación con el middleware, y que permitirá al desarrollador de aplicaciones olvidarse de los aspectos relacionados con el funcionamiento de los servicios de TRENCADIS. De esta forma, si en un futuro se deseara portar el middleware a otra plataforma que no fuera GT, sólo se tendría que modificar esta capa para conseguir que las aplicaciones funcionaran con la nueva implementación del middleware.

8. Por último, los servicios existentes en la implantación CVIMO de TRENCADIS están escritos para una versión anterior de GT, por lo que otra

de las metas es reescribirlos para que funcionen sobre la versión 4.2.1 de GT.

Capítulo 4

Descripción del desarrollo realizado

El desarrollo motivado por esta tesis ha consistido en la modificación de la arquitectura TRENCADIS y en la mejora de la funcionalidad del middleware. TRENCADIS está especializado en la gestión de estudios DICOM de imágenes médicas en un entorno data GRID, por lo que se ha buscado potenciar este enfoque (aprovechando al máximo la expresividad de los DICOM-SR y haciéndolo compatible con componentes de gLite).

En este capítulo se describe la funcionalidad global de los componentes del middleware tras aplicar las modificaciones pertinentes y cumplir con los objetivos predefinidos. Posteriormente, en el capítulo de aportaciones, se detallan las aportaciones más relevantes de esta tesis.

4.1. Arquitectura de TRENCADIS

La arquitectura de TRENCADIS originalmente se estructuraba en cinco capas, que son: *Core middleware*, *Server services*, *Communication*, *Middleware components* y *Applications*.

En esta tesis se ha extendido la arquitectura a una capa más, la capa gLite Core, la cual permite aprovecharse de las infraestructuras existentes desplegadas mediante gLite (como EGEE [7] o NGI [20]).

Podemos ver la organización de las capas en la figura 4.1.

4.2. La capa *gLite core*

Esta capa es la principal novedad que se añade al desarrollo existente de la arquitectura TRENCADIS. El propósito de la misma es permitir el uso de componentes gLite (como el AMGA o el LFC) a los servicios de la capa *Core middleware*. Los componentes gLite utilizados permiten la gestión de grandes volúmenes de datos y metadatos en entornos data GRID.

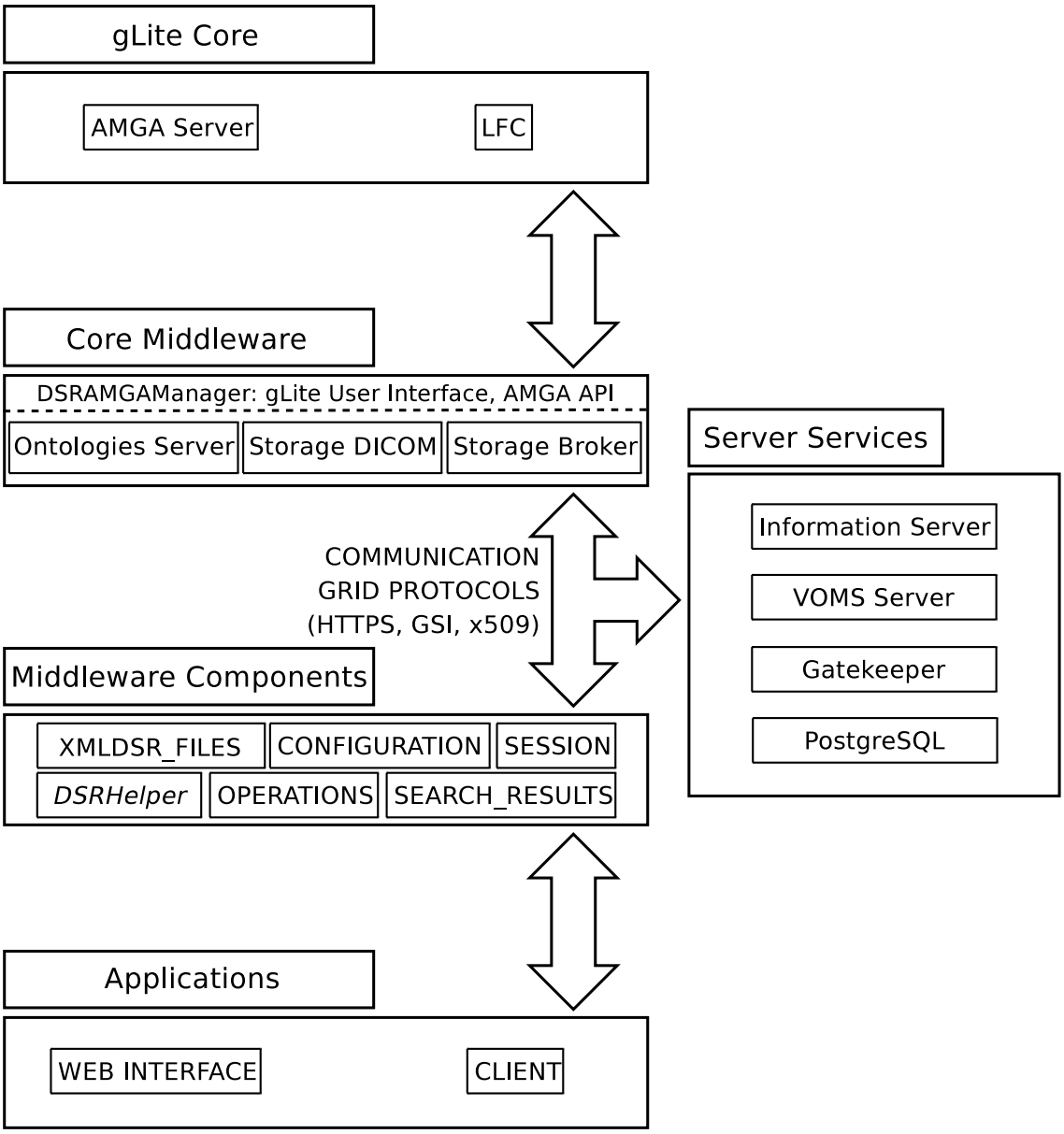


Figura 4.1: Arquitectura de TRENCADIS

4.2.1. Servidor de catálogo AMGA

En este apartado se describe como se ha utilizado el servidor de catálogo AMGA para almacenar y organizar los metadatos de los DICOM-SR. También se realiza un análisis de prestaciones en el que se estudia si la penalización que aparece al utilizar el catálogo AMGA, en vez de una base de datos relacional, es aceptable.

4.2.1.1. Equivalencia entre modelo relacional y catálogo

Este es un ejemplo que ilustra la equivalencia entre una tabla del modelo relacional y una colección del catálogo de AMGA.

La tabla de *imágenes* (cuadro 4.1) contiene la información sobre imágenes médicas de varios tipos obtenidas de distintos pacientes.

Colección	imagenes	
Atributos		
Nombres de las entradas	paciente	tipo
M12a	Juana López	Mamografía
E34b	Pedro Jaén	Ecografía
R51z	Raúl Herrero	Radiografía

Cuadro 4.1: Imágenes

La correspondencia de esta tabla en la estructura de directorios del catálogo AMGA es la siguiente:

```
[xxx@yyyy ~]$ mdclient
Connecting to localhost:8822...
ARDA Metadata Server 1.3.0
Query> listentries /imagenes/
>> /imagenes/M12a
>> /imagenes/E34b
>> /imagenes/R51z
```

Un directorio para la colección *imágenes*, con un fichero por cada entrada.

```
Query> getattr /imagenes paciente tipo
>> M12a
>> Juana López
>> Mamografía
>> E34b
>> Pedro Jaén
>> Ecografía
>> R51z
>> Raúl Herrero
>> Radiografía
```

Cada entrada contiene los atributos de la fila de la tabla original a la que corresponde y un atributo extra que se utiliza para almacenar el nombre del fichero en el sistema de directorios del catálogo. Este atributo es el *nombre de la entrada* y debe utilizarse como clave primaria de cada colección.

El motivo por el que el catálogo de metadatos se organiza siguiendo una estructura de directorios es meramente práctico. Imaginemos que en el LFC tenemos un directorio de imágenes, dichas imágenes son ficheros con un nombre determinado (pongamos los del ejemplo M12a, E34b, R51z...). El usuario de AMGA tiene la posibilidad de organizar los metadatos asociados a dichas imágenes siguiendo una estructura de directorios idéntica a la estructura del repositorio de datos (en este caso representando los metadatos como ficheros dentro de la colección de imágenes).

4.2.1.2. Creando un árbol de elementos DICOM-SR

El catálogo AMGA se desarrolló con la finalidad de almacenar los metadatos en una estructura de directorios igual a la de los ficheros de datos del GRID que describen. En esta tesis, empleamos la estructura de directorios del catálogo para introducir el árbol de contenidos de los DICOM-SR. Para hacer esto necesitamos saber más acerca de los elementos que forman los DICOM-SR [6]. Los capítulos 3, 4 y 11 de [6] explican la estructura de los DICOM-SR.

Un DICOM-SR puede verse como un árbol de elementos, estos elementos se llaman *content items*. Cada uno de estos *content items* tiene un *concept name* y un conjunto de valores.

Hay muchos tipos de *content item* (TEXT, NUM, PNAME, CODE, CONTAINER, DATE, TIME o SCOORD entre otros), todos ellos tienen un *concept name* propio. Además, todos los *content items* pueden contener otros *content items*.

El *concept name* especifica el papel en el documento del *content item* al que pertenece. Por ejemplo, un *content item* de tipo TEXT puede ser una conclusión o el tercer hallazgo del médico que escribe el informe.

En el estándar DICOM-SR, los *concept name* son pares de valores, estos valores son las cadenas de texto llamadas *code value* y *code schema* del *content item*. Además, el *concept name* puede incluir una cadena llamada *code meaning*, que nos permite poner un nombre legible al código en el *code value*.

El *code schema* es el identificador del diccionario de códigos, el *code value* es el valor que se ha escogido de ese diccionario para especificar el papel del *content item*. Este tipo de codificación es mejor para la búsqueda e indexación que el texto plano. Por ejemplo, los códigos pueden internacionalizarse, de esta forma "998M" puede significar "hallazgo 2" pero es un código, así que si trabajamos en EEUU y usamos el código "998M" la aplicación podrá mostrar "finding 2".

Este es un ejemplo de *content item* extraído del capítulo 3 de [6]. Un tipo TEXT tiene el aspecto de la figura 4.2.

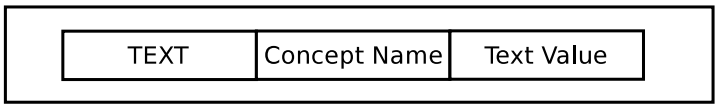


Figura 4.2: Tipo TEXT

Normalmente cuando se describe un *content item* se emplea la notación:

```
<TEXT:(209001,99PMP,"Finding")="Large, irregular mass">
```

aunque esta no es la sintaxis de los ficheros DICOM, que son binarios.

El fragmento (209001,99PMP,"Finding") corresponde a los tres elementos del *concept name* (*code value*, *code schema* y *code meaning*).

Esta es una representación de la estructura del código binario DICOM que se genera para representar el *content-item* de este ejemplo:

```
(0040,a040) Value Type "TEXT"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209001"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Finding"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a160) Text Value "Large, irregular mass"
```

El código DICOM puede resultar muy complejo, por lo que a la hora de tratar con DICOM-SR utilizamos representaciones XML. Trabajar en XML simplifica las cosas, ya que existen muchas herramientas para el tratamiento de los ficheros y siempre tenemos la opción de generar el fichero binario DICOM. El elemento en XML queda así:

```
<TEXT>
  <CONCEPT_NAME>
    <CODE_VALUE>209001</CODE_VALUE>
    <CODE_SCHEMA>99PM</CODE_SCHEMA>
    <CODE_MEANING>Finding</CODE_MEANING>
  </CONCEPT_NAME>

  <TEXT_VALUE>Large, irregular mass</TEXT_VALUE>
</TEXT>
```

Con el uso de *concept names* podemos tener en nuestro informe varios *content items* de un mismo tipo. Por ejemplo, podemos tener varios PNAME,

uno para el nombre del primer observador de la imagen, otro para el segundo y otro para la persona que tomó la imagen.

Los *concept names* definen el contexto de la información dentro de los *content item* de tipo CONTAINER. Podemos crear un contenedor para albergar hallazgos, otro para conclusiones y otro diagnósticos. Es una buena idea reflejar esto en el *concept name* del elemento CONTAINER, ya que los elementos CONTAINER no tienen información asociada a su tipo, sólo contienen otros objetos.

Otro punto importante es que, en lo que nosotros entendemos como un DICOM-SR bien definido, un *content item* no debe contener items repetidos. Esto es, si un contenedor contiene varios "hallazgos" sus posiciones en el documento serán relevantes y su *concept name* debe reflejarlo. En este caso los *concept names* de los *content items* deberán ser algo como "hallazgo 1", "hallazgo 2" y en adelante. Debe utilizarse siempre la expresividad de los *code values* en cada *content item* de nuestros informes estructurados DICOM.

No obstante, aunque esta forma en la que se representa y ordena la información de los DICOM-SR es correcta y funcional, dificulta la validación de los XML mediante el uso de XSD. El motivo es que al escribir un documento XSD de validación tendríamos que crear un elemento distinto para cada uno de los *content items* del mismo tipo pero distinto *concept name* y no podemos crear elementos de un mismo tipo en cualquier parte del documento XSD. Por otro lado, en el estándar DICOM todos los *content items* pueden contener a su vez otros *content items* de ciertos tipos determinados y en cualquier orden. Esto no es así en los documentos XML validados mediante XSD, en los cuales todos los elementos del mismo tipo deben tener el mismo contenido según unas reglas más restrictivas.

Por esto, además de utilizar ficheros XML para describir el contenido de los DICOM-SR utilizamos otros ficheros para describir la estructura de los mismos. Estos ficheros de estructura tienen la peculiaridad de ser iguales que los ficheros que describen los XML pero sólo contienen la información de los *concept names*. La información de la estructura de un DICOM-SR es lo que nosotros entendemos por **ontología**, y las utilizaremos para crear los árboles de directorios en el catálogo AMGA.

La figura 4.3 muestra un árbol de ejemplo generado por nuestra aplicación. Este árbol ha sido creado parseando un fichero en XML con la estructura de un tipo concreto de DICOM-SR. Para analizar el fichero se utiliza un parser SAX que lanza eventos conforme va detectando las etiquetas de los elementos XML. El parser se ha programado de forma que envía al AMGA los comandos necesarios para crear y establecer las colecciones y sus atributos.

Cada caja de la figura contiene tres elementos: el primero es el nombre del directorio de la colección y está compuesto por el *code value* y el *code schema* del *content item*, el segundo es el *code meaning*, utilizado para dejar más claro el rol del *content item* y el tercero es el tipo de *content item*

almacenado en el directorio.

En el apéndice A.1 puede verse un fichero de estructura que describe la ontología representada en el árbol de la figura 4.3, y en el apéndice A.2 puede verse un ejemplo de una representación en XML de un DICOM-SR que sigue dicha estructura.

Así es como creamos el árbol e indexamos los metadatos, creando colecciones en el catálogo cuyo nombre esté formado por los atributos del *concept name* del *content item* al que representa (*code value* + "_" + *code schema*). Si un *content item* se encuentra en el interior de otro *content item*, entonces el directorio de su colección estará en el interior de la colección del padre.

Cada colección tendrá diferentes atributos, dependiendo del tipo de *content item* al que corresponda.

El nombre de las entradas dentro de cada colección corresponderá con el identificador único del DICOM-SR al que pertenece la entrada.

El directorio raíz de cada estructura en árbol no es una colección, es simplemente un directorio sin atributos y cuyo nombre es el del identificador del tipo de informe. Todos los directorios raíz contienen un directorio *header* con los atributos de la cabecera de los DICOM-SR y una cadena dedicada a almacenar la representación en XML entera de cada DICOM-SR.

Almacenar los DICOM-SR completos en el catálogo AMGA como parte de los metadatos simplifica el diseño del middleware, ya que si no se hiciera de esta manera, al subir una colección de imágenes junto con su informe asociado tendríamos que: subir los metadatos del informe al AMGA, generar un binario del DICOM-SR y subirlo al LFC y, por último, subir el estudio de imágenes al LFC. Sin embargo de esta forma sólo hay que subir los metadatos al catálogo AMGA (conteniendo estos metadatos el informe completo en XML) y el estudio de imágenes al LFC.

Como hemos mencionado en los objetivos, las acciones realizadas por el middleware deben ser transaccionales, por lo que el diseño que se ha empleado facilita que ante un error sea más fácil deshacer todas las acciones (son 2 en vez de 3) y el estado del sistema se mantenga inalterado. Además, aunque no disponemos de ficheros en formato binario con los DICOM-SR en el LFC, la aproximación sigue siendo mejor, ya que el cliente recibe la representación del DICOM-SR y se encarga de convertirla a binario DICOM él mismo. Esto supone que se consumirán menos recursos de los servicios, al no tener estos que generar un fichero binario DICOM y subirlo al LFC cada vez que se introduce un nuevo informe en el sistema.

Por último cabe destacar que todos los directorios y ficheros dentro de un catálogo AMGA tienen una lista de control de acceso asociada con permisos tipo UNIX, lo que nos permite limitar el acceso a partes del informe dependiendo del usuario que realice las consultas.

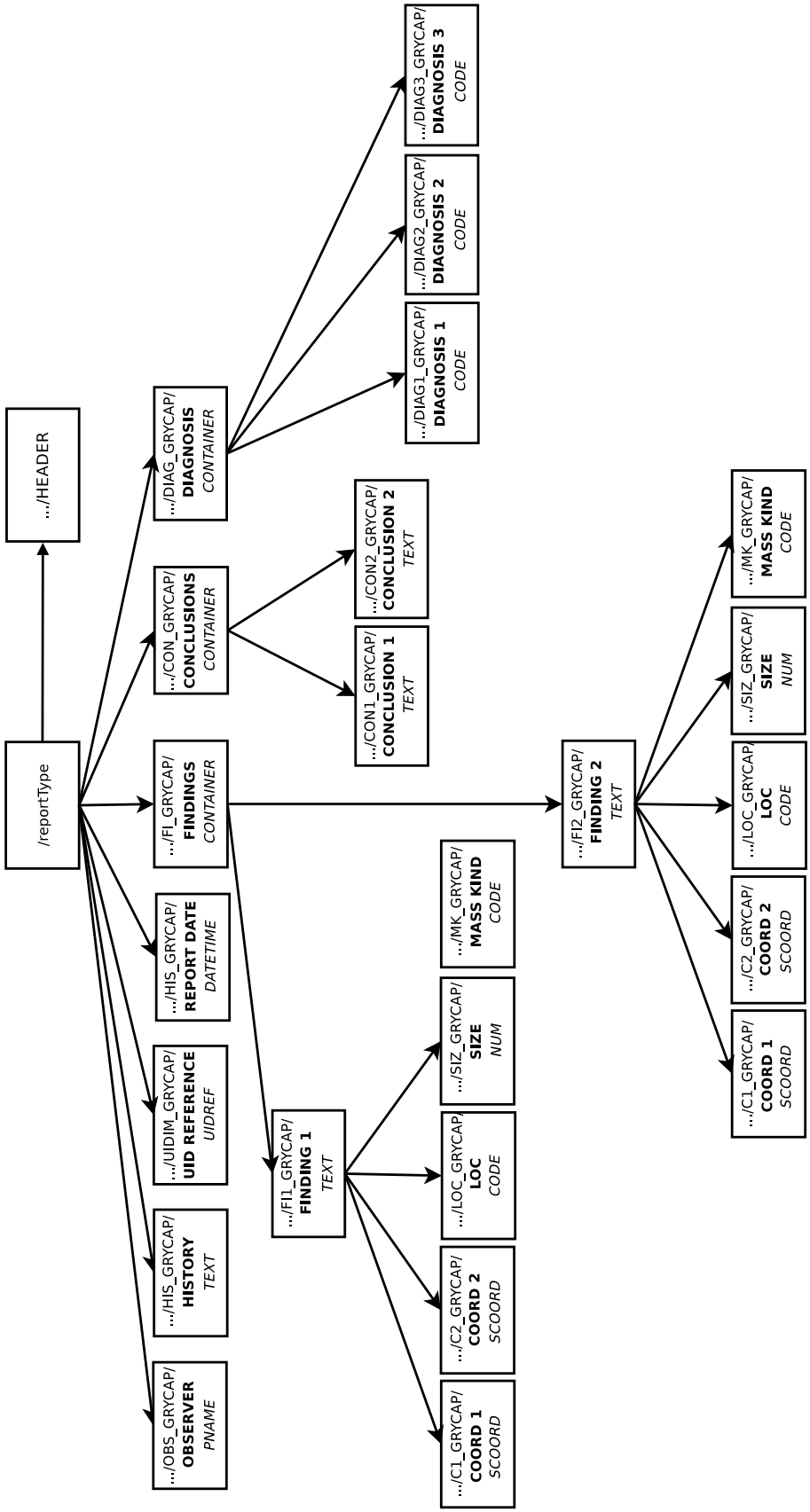


Figura 4.3: Árbol de ejemplo

4.2.1.3. Impacto del catálogo AMGA

En esta sección comparamos las prestaciones de un catálogo AMGA 1.9 sobre PostgreSQL 7.4 que utiliza nuestra aproximación frente a una base de datos relacional almacenada en un servidor PostgreSQL 7.4. Ambas configuraciones se ejecutaron sobre la misma máquina.

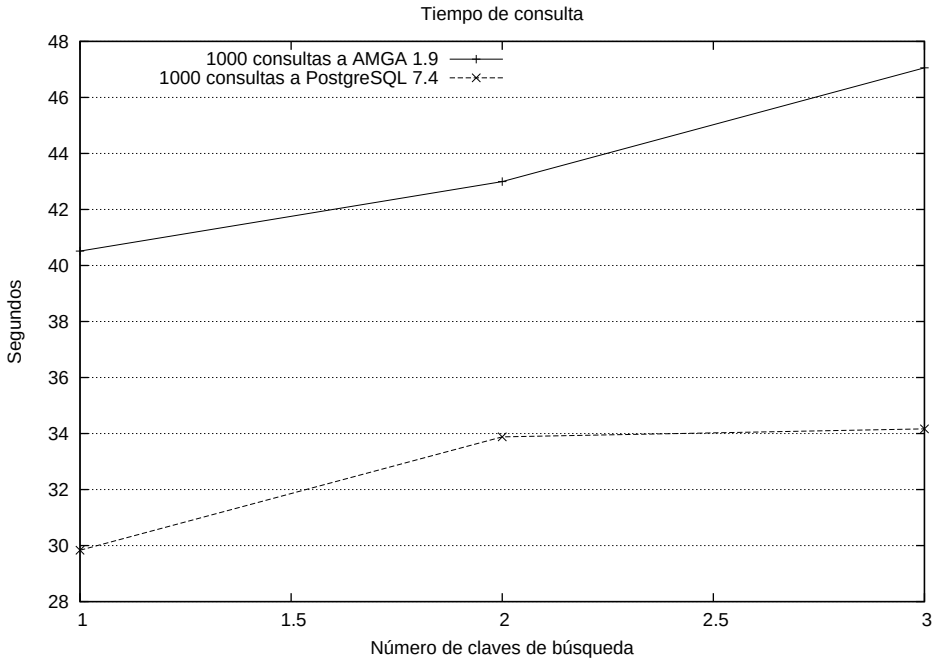
Hemos introducido datos generados por nuestra aplicación en el catálogo AMGA. Se ha creado la estructura de un tipo de DICOM-SR y se ha rellenado con la información de 1000 informes.

En el caso de la base de datos relacional sobre PostgreSQL 7.4 se han creado dos tablas asociadas mediante una clave ajena. En este caso, las filas de la tabla representan diferentes tipos de informes estructurados (1000 en total). Además, las columnas de cada tabla representan a elementos en el mismo nivel del árbol de estructura.

La información introducida en los servidores de ambas aproximaciones se ha generado aleatoriamente usando los mismos parámetros de probabilidad, de este modo sabemos que consultas equivalentes van a devolver aproximadamente el mismo número de resultados cuando se ejecuten sobre ambos conjuntos de datos.

Los clientes utilizados para realizar los test fueron las utilidades de línea de comandos psql y mdcli. Estos programas están escritos en C. Se lanzaron en la misma máquina en la que estaba su servidor, por lo que no hay latencia de red. La autenticación por certificado y el protocolo *SSL* fueron desactivados. Además los datos devueltos no son procesados. Esto se ha hecho así porque queremos aislar la sobrecarga introducida por la capa del catálogo.

En la siguiente gráfica se muestra la ejecución de 1000 consultas consecutivas. No se realizan consultas concurrentes.



En los resultados se observa que utilizar la capa del catálogo AMGA ralentiza las búsquedas entre un 20 % y un 30 % sobre el tiempo obtenido al ejecutar el test del modelo relacional que utiliza dos tablas.

Esta diferencia aumenta ligeramente al aumentar el número de términos en la búsqueda porque al seguir nuestra indexación en árbol todos los elementos de cada DICOM-SR están en colecciones diferentes, mientras que en la base de datos relacional los términos están en tan sólo dos tablas. Esto significa que las búsquedas necesitan más condiciones para relacionar tablas mediante *JOINS* y ser equivalentes a las consultas realizadas sobre las dos tablas de la configuración para el modelo relacional.

Además, la base de datos interna es más compleja cuando trabajamos con el catálogo y AMGA tiene que gestionar más tablas. En el caso de AMGA se añade la sobrecarga de tener que gestionar todo el sistema de autorización y la lógica de la capa que añade.

De todas formas, esta sobrecarga es aceptable en un escenario GRID de acceso concurrente por red al servidor AMGA (los tiempos de autenticación y comunicaciones son mucho más relevantes), ya que como se explica en [16] AMGA se comporta muy bien cuando hay múltiples conexiones concurrentes (tiene una caché interna de conexiones para conectarse al servidor *PostgreSQL*). Además, si tuviéramos que procesar los datos devueltos por las consultas, el driver del cliente AMGA ha demostrado procesar grandes cantidades de resultados más rápido que el driver *JDBC*.

Como conclusión podemos decir que esta sobrecarga es leve teniendo en cuenta el funcionamiento del resto del sistema y que es aceptable debido a

las ventajas que aporta el uso del catálogo AMGA a la hora de proveer una mayor expresividad de los datos.

4.2.2. El catálogo LFC

El catálogo LFC cumple la función de almacenar los estudios de imágenes médicas. Cada DICOM-SR está relacionado con un estudio DICOM mediante un identificador único que se incluye en la cabecera de la representación en XML del informe.

Cada hospital puede elegir el SE por defecto en donde almacenar sus estudios de imágenes DICOM. Dicho SE debe ser el más accesible y con mejor velocidad de conexión.

4.3. La capa *Server services*

La capa *Server services* está formada por servicios que dan soporte a los servicios GRID de la capa *Core middleware* y que no son parte de gLite.

4.3.1. Information Service

El IIS (*Internet Information Service*), también conocido como Globus MDS (*Monitoring and Discovery System*) permite a los servicios GRID publicar información sobre las propiedades de sus recursos.

Se trata de un servicio GRID ubicado en cada contenedor de servicios GT 4.2, en el que se publica la información de todos los servicios que contiene desplegados.

Los MDS pueden jerarquizarse, de forma que un MDS puede publicar toda su información en otro MDS padre. En nuestro caso empleamos un MDS central, en el que publican los MDS de todos los contenedores la información de los servicios GRID desplegados.

Utilizaremos este MDS central para obtener información acerca de los servicios desplegados. Por ejemplo, un recurso de un servicio puede publicar su URI en una propiedad de recurso para que otros servicios y clientes puedan invocar sus métodos.

Para visualizar los contenidos del MDS podemos utilizar utilidades como el webMDS [27], que muestra las entradas de un MDS en formato HTML.

4.3.2. VOMS Server

El servidor VOMS permite gestionar organizaciones virtuales añadiendo una extensión a las credenciales x509 estándar del usuario. En el momento en el que se crea la credencial se conecta con el servidor VOMS y se añade la extensión que indica la VO a la que pertenece el usuario.

La utilización de credenciales VOMS es necesaria para el funcionamiento del sistema de autorización que provee el *Gatekeeper*, además de que el soporte para organizaciones virtuales es esencial para gestionar los permisos de los miembros de la comunidad científica.

Todos los componentes de la capa *gLite Core* utilizan credenciales VOMS junto con el *Gatekeeper* para controlar los accesos.

4.3.3. Gatekeeper

El *Gatekeeper* [21] es el componente encargado de controlar la autenticación y la autorización de los usuarios en los servicios GRID.

Todas las invocaciones a los métodos de los servicios GRID deben preguntar primero al *Gatekeeper* si el usuario que realiza la invocación está autorizado a realizar esas acciones.

El funcionamiento del *Gatekeeper* está basado en la infraestructura de seguridad GRID y por tanto se apoya en los protocolos de la capa de comunicaciones, que se basan en SSL y certificados x509. Gracias a la criptografía de clave asimétrica se crea un canal seguro entre las dos partes en el que los datos se envían cifrados.

El primer paso lo realizará el cliente al crear un certificado x509 con extensiones VOMS que será enviado en la invocación del método remoto. Una vez que la invocación llega al servidor se extraen los atributos VOMS (grupo, roles, capacidades) del certificado recibido y se presentan al *Gatekeeper*.

El *Gatekeeper* comprueba en primer lugar que el certificado x509 que presenta el usuario haya sido emitido por una entidad certificadora (CA) de confianza. Una vez que se ha comprobado esto busca el *Distinguished Name* (DN) y el grupo (parámetro de la VO) del usuario en la base de datos y sólo permitirá que continúe la llamada al método si el usuario está autorizado.

Cada servicio GRID contiene una instancia del *Gatekeeper* que se conecta mediante un pool de conexiones (figura 4.4) a la base de datos PostgreSQL que contiene los permisos de acceso. Anteriormente no existía este pool de conexiones, es un componente nuevo que utilizamos en cualquier conexión a una base de datos por parte de los servicios.

4.4. La capa *Core middleware*

La capa *Core middleware* contiene los servicios GRID principales que nos permiten añadir nuevas ontologías, nuevos DICOM-SR y realizar búsquedas sobre la información existente en los distintos hospitales.

Los servicios de esta capa necesitan comunicarse con la capa *gLite core* y utilizar los componentes de gLite que suministra para poder funcionar. Con este propósito se ha creado una API de comunicación genérica con el AMGA y el LFC que trabaja con las representaciones en XML de los DICOM-SR y a la cual llamamos DSRAMGAManager. Además el DSRAMGAManager utiliza

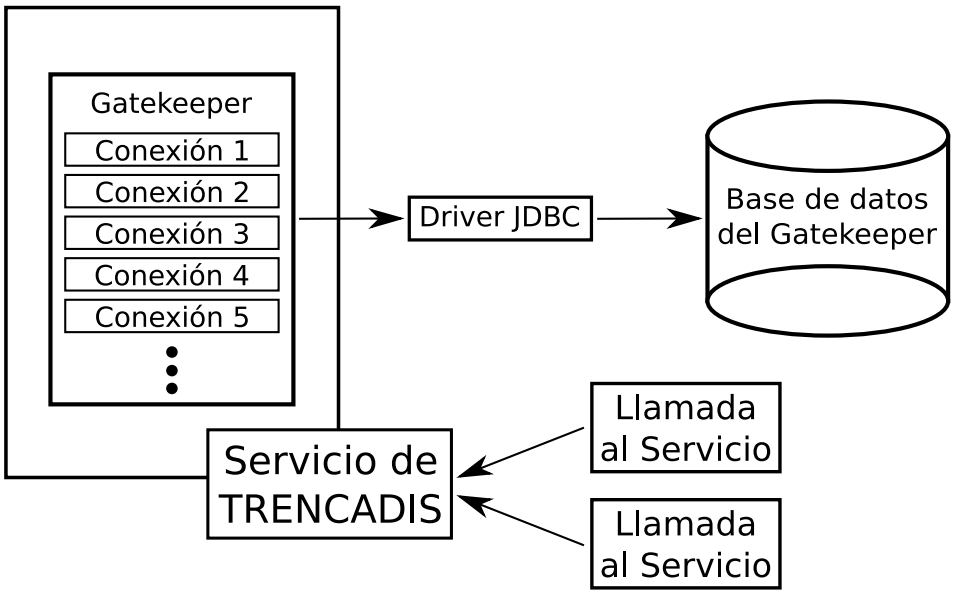


Figura 4.4: Pool de conexiones del *Gatekeeper*

otro módulo, el DSRHelper, que contiene todas las acciones realizables sobre los DICOM-SR que no involucran conexión con el servidor AMGA o el LFC.

Además, los servicios de esta capa también necesitan comunicarse con los servicios de la capa *Server services*, como el Information Service, el VOMS o el Gatekeeper que proporcionan funciones de autenticación y descubrimiento.

4.4.1. Funciones del DSRAMGAManager

El módulo DSRAMGAManager es un API desarrollada para este proyecto de tesis y que permite que cualquier elemento pueda comunicarse con los servicios AMGA y LFC de la capa *gLite core* para gestionar DICOM-SR. Engloba los parsers necesarios para analizar los DICOM-SR y las APIs necesarias para enviar y recibir los datos del AMGA y del LFC. Los servicios de esta capa necesitan este módulo para funcionar.

Para comunicarnos con el AMGA utilizamos su API de java y para comunicarnos con el LFC utilizamos un UI (*User Interface*) de gLite con los comandos que permiten acceder a él (los cuales llamamos desde java).

En esta sección se describen las tareas que realiza el módulo DSRAMGAManager y qué parámetros de entrada necesita cada una de ellas.

Como parámetros de entrada se pueden recibir ficheros XML de estructura y ficheros XML con un DICOM-SR.

Un fichero de estructura de un tipo de DICOM-SR es un fichero XML con la extensión *.str*. Este tipo de fichero utiliza etiquetas para describir las relaciones entre los *content items* que forman el árbol de estructura.

Únicamente contiene la información de los *concept names* de cada *content item*, no sus valores.

Un fichero con la representación XML de un DICOM-SR tendrá la extensión *.xml* y contendrá toda la información de sus *content items*.

En caso de que una tarea reciba como entrada un fichero de estructura o un fichero con la representación en XML de un DICOM-SR dicho fichero debe ser válido. En el caso de los ficheros DICOM-SR, estos deben además seguir correctamente el esquema definido en el fichero de estructura de su ontología. Es conveniente usar una aplicación para generar los ficheros de entrada en vez de crear los ficheros manualmente. Esta restricción es necesaria porque así se ha podido diseñar el módulo utilizando parsers SAX (los cuales lanzan eventos conforme van recorriendo los elementos del documento XML) bastante rápidos y funcionales al no tener que analizar si el documento tiene errores.

No obstante si un parser SAX intenta realizar una acción incorrecta (como introducir en el AMGA una entrada en una colección que no existe) este recogerá la excepción y cancelará todas las acciones realizadas desde el inicio del análisis del documento. El comportamiento de todas las acciones que se describen a continuación es transaccional.

Añadir estructura

Crea una estructura de directorios en el catálogo AMGA que será utilizada para almacenar una nueva ontología de DICOM-SR. Para ello el parser SAX lanza comandos al servidor AMGA conforme va analizando el fichero de estructura que recibe como argumento.

Para cada tipo distinto de DICOM-SR existirá una estructura directorios cuya carpeta raíz estará en el directorio raíz del catálogo.

Borrar estructura

Borra la estructura de directorios asociada a un tipo de DICOM-SR del catálogo. Para ello necesita únicamente el identificador de la ontología.

Esta acción borra la estructura y todas las entradas que existan en su interior.

Añadir DICOM-SR

Este comando añade a una estructura de directorios que se haya creado anteriormente un nuevo fichero XML con la representación del DICOM-SR. La función añadirá una nueva entrada en cada una de las colecciones de la estructura en árbol de su ontología para las que exista un *content item* en el DICOM-SR.

Borrar DICOM-SR

Borra un informe del catálogo AMGA, para ello necesita un fichero de estructura *.str* y el identificador del documento a borrar.

Listar DICOM-SR

Lista la información de un DICOM-SR almacenado en el catálogo AMGA, mostrando los atributos y valores de sus entradas en todas las colecciones del árbol.

Necesita de un fichero de estructura y del identificador del informe.

Añadir estudio de imágenes al LFC

Cualquier DICOM-SR que se introduzca en el sistema debe estar asociado a un estudio de imágenes DICOM que exista en el LFC. Esta función se encarga de subir dichas imágenes al catálogo de datos de nuestra infraestructura GRID.

Borrar estudio de imágenes del LFC

Borra un estudio de imágenes DICOM del LFC.

4.4.2. Módulo DSRHelper

El módulo DSRHelper contiene las acciones relacionadas con los ficheros DICOM-SR que no involucran conexión. El módulo DSRAMGAManager y algunos objetos de la capa *Middleware components* (TRENCADIS_XMLDSRSTR_FILE y TRENCADIS_XMLDSR_FILE) lo utilizan.

Listar estructura

Lista una ontología leyendo un fichero *.str*.

Muestra un listado de los nodos del árbol y sus atributos. Este método es útil si queremos realizar consultas manuales en la consola del servidor AMGA, ya que muestra la ruta a todas y cada una de las colecciones.

Generar DICOM-SR

Esta función convierte una descripción en XML de un DICOM-SR en un fichero binario acorde al estándar DICOM. Los DICOM-SR introducidos en el catálogo AMGA pueden ser convertidos a binario DICOM con éste método.

Imprimir los códigos DICOM-SR en ascii

Esta función genera, a partir de una representación XML de un DICOM-SR, un fichero *.adcm* con la representación en ascii de su código binario DICOM. Puede ser útil en tareas de depuración.

4.4.3. Descripción de los servicios

Con la finalidad de cumplir los objetivos descritos en la sección 3, ha sido necesario modificar algunos servicios de la capa *Core Middleware*.

En esta sección se describe el propósito y la interfaz de los servicios GRID que se han modificado, haciendo hincapié en las nuevas funciones.

Uno de los motivos para su modificación ha sido hacerlos compatibles con la última versión de GT, la 4.2.1, programándose su código desde cero para garantizar la compatibilidad. Los detalles de cada servicio se describen a continuación. Las propiedades de recurso son la información que cada servicio publica en el IIS y deja al alcance del resto (permiten la interoperabilidad de los componentes).

Los métodos de cada servicio, así como sus interacciones e inicialización, se describen en las secciones 4.4.4, 4.4.5, 4.4.6 y 4.4.7.

Servicio *Storage DICOM*

La finalidad de este servicio es la de gestionar los DICOM-SR en el repositorio AMGA asociado al servicio. En despliegues anteriores de TRENCADIS el servicio *Storage DICOM* no se comunicaba con el catálogo AMGA y almacenaba los metadatos en una única tabla, sin conservar el árbol de estructura del documento.

Además, el servicio *Storage DICOM* antes era *singleton*, pero ahora soporta múltiples recursos, creando uno por cada ontología en el sistema. Estos recursos del servicio son los que se utilizan posteriormente para la gestión de cada tipo de DICOM-SR.

Métodos Añadir informe, borrar informe, bajar informe.

Propiedades del recurso URI del recurso, identificador de ontología, identificador del centro, dirección y puerto del AMGA asociado al recurso.

Servicio *Storage Broker*

El servicio *Storage Broker* crea recursos que actúan de mediadores de cada tipo de ontología, permitiendo búsquedas simultáneas en todos los repositorios AMGA del sistema.

Para ello, utiliza la información de las propiedades de recurso que publican los recursos *Storage DICOM* y que aparecen en el IIS central. De esta

forma, puede conectarse a cada repositorio AMGA en el que se almacene un tipo de ontología y realizar las consultas simultáneas.

Métodos Consultar informes.

Propiedades del recurso URI del recurso, identificador de ontología.

Servicio *Ontologies Server*

El *Ontologies Server* se encarga de almacenar las ontologías de los informes estructurados. Las ontologías son las descripciones en XML de la estructura de los distintos tipos de DICOM-SR. Estas ontologías se definen por mutuo acuerdo entre los centros médicos, para garantizar que todos utilizan el mismo formato a la hora de escribir y almacenar los informes de cada tipo de diagnóstico (tendrán un esqueleto común).

La misión del *Ontologies Server* es, por tanto, garantizar la homogeneidad de los datos en el sistema y centralizar la gestión de las ontologías.

Para ello, el *Ontologies Server* envía regularmente a los servicios *Storage DICOM Factory* y *Storage Broker Factory* la lista de ontologías, estos analizan dicha lista y actualizan sus recursos para mantener su estado actualizado. El *Ontologies Server* también envía la lista de ontologías cuando se realiza algún cambio sobre la misma o cuando el usuario desea forzar la actualización del estado de todo el sistema.

El *Ontologies Server* es de tipo *singleton* por lo que sólo tiene un recurso.

Métodos Añadir ontología, borrar ontología, obtener ontología, obtener todas las ontologías, actualizar servicios *Storage DICOM* y *Storage Broker*.

Propiedades del recurso URI del servicio

Interacción entre los servicios

En la figura 4.5, puede verse la interacción global entre los servicios.

El *Ontologies Server* envía la lista de ontologías a los servicios factoría, y estos crean recursos para mantener su estado actualizado con el sistema.

El *Storage Broker* utiliza la información publicada por los recursos *Storage DICOM* en el IIS para realizar las consultas distribuidas.

4.4.4. Inicialización de los servicios

Los flujos de llamadas descritos en esta sección están relacionados con los procesos de inicialización de los distintos servicios GRID desplegados. Estos diagramas nos ayudan a entender el estado inicial del sistema.

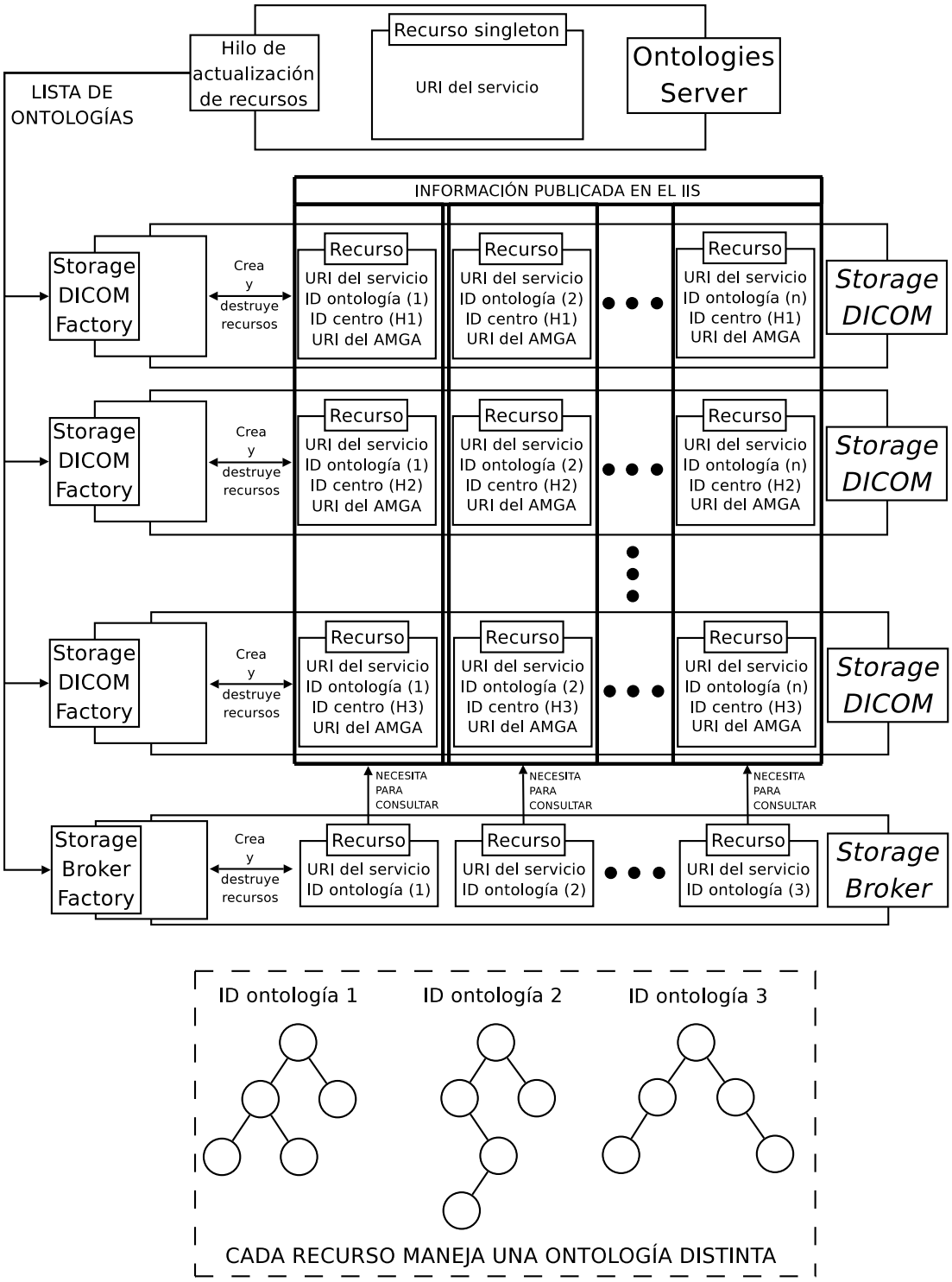


Figura 4.5: Interacción entre los servicios *Storage DICOM*

Los servicios interactúan internamente con varias bases de datos, componentes gLite y servicios (como por ejemplo la base de datos del *Ontologies Server* o el IIS).

Al iniciarse, cada servicio lee su fichero de configuración y usa esta información para inicializar una cache de conexiones por cada una de las bases de datos a las que se conecta.

Una caché de conexiones (*connection pool*) acelera la velocidad de comunicación manteniendo las conexiones abiertas para ser reutilizadas. Por ejemplo, todos los recursos de un mismo servicio pueden compartir la misma caché de conexiones con el Gatekeeper del servicio.

Inicialización del Ontologies Server

La inicialización del *Ontologies Server* puede verse en la figura 4.6.

Al iniciarse, el *Ontologies Server* registra su *URI* en el *IIS* (4), de esta forma otros servicios podrán contactar con él. El *Ontologies Server* debe ser el primer servicio de la capa *Core middleware* en arrancar, ya que es necesario para el funcionamiento del *Storage DICOM* y del *Storage Broker*.

El último paso (6) inicia el hilo que se encarga de enviar a los servicios *Storage DICOM* y *Storage Broker* la lista de ontologías en el sistema, lo que les permite sincronizarse y tener un recurso por cada ontología.

El *Ontologies Server* es de tipo *singleton* y sólo tiene un recurso asociado que se encarga de manejar todas las ontologías.

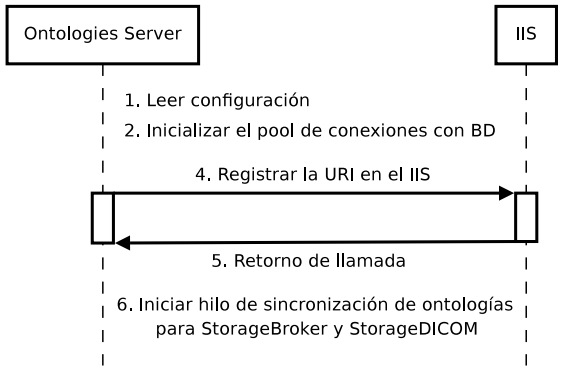


Figura 4.6: Inicialización del Ontologies Server

Inicialización del Storage DICOM

La inicialización del *Storage DICOM* puede verse en la figura 4.7.

Al iniciarse, el servicio *Storage DICOM* lee su configuración e inicializa sus cachés de conexiones.

Cada recurso del *Storage DICOM* se encarga de almacenar en el catálogo AMGA los DICOM-SR de una determinada tipología. Pueden coexistir difer-

entes recursos *Storage DICOM* para una misma ontología de DICOM-SR, pero tienen que residir en distintos hospitales.

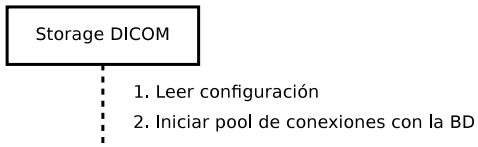


Figura 4.7: Inicialización del Storage DICOM

Inicialización del Storage Broker

La inicialización del *Storage Broker* puede verse en la figura 4.8.

La inicialización del *Storage Broker* es similar a la del *Storage DICOM*.

Cada recurso del *Storage Broker* puede realizar búsquedas en todos los catálogos AMGA gestionados por recursos del *Storage DICOM* de una determinada ontología.

Esto nos permite ejecutar consultas en un recurso del *Storage Broker* que serán automáticamente distribuidas entre los distintos catálogos AMGA.

Aunque el diseño del servicio nos permitiría ejecutar múltiples servicios *Storage Broker* en contenedores diferentes, la planificación de este despliegue consiste en ejecutar un sólo servicio *Storage Broker* que se encargará de distribuir todas las consultas.

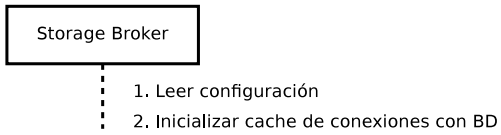


Figura 4.8: Inicialización del Storage Broker

Inicialización del Storage DICOM Factory

La inicialización del *Storage DICOM Factory* puede verse en la figura 4.9.

El constructor del *Storage DICOM Factory* realiza numerosas acciones al iniciarse. Cuando todas estas tareas han terminado, el servicio *Storage DICOM* dispone de un recurso por cada ontología dada de alta en el *Ontologies Server*.

La primera tarea específica que se realiza es obtener la lista de ontologías almacenada en el *Ontologies Server* (6) para ello necesita la URI de este servicio, que se encuentra en el IIS (4). Si el constructor del *Storage DICOM Factory* no es capaz de obtenerla el servicio no se iniciará correctamente.

Cabe resaltar que creamos un certificado *VOMS* con las credenciales de la máquina en la que se ejecuta el contenedor de este servicio, esto es necesario

para autenticarse con el *Ontologies Server* al arrancar. Las credenciales de la máquina que ejecuta el contenedor del servicio *Storage DICOM* deben aparecer en la base de datos del *Gatekeeper* del *Ontologies Server*.

El servicio analiza la lista de ontologías obtenida y crea un recurso por cada ontología llamando a su propio método *createResource()* (8). Los pasos que realiza *createResource()* se describen más adelante.

En el diagrama remarcamos que cuando un recurso es creado sus propiedades se publican en el IIS (9).

Tras haberse creado todos los recursos, el *Storage DICOM Factory* publica su URI en el IIS (10). Desde el momento en el que se publica empezará a recibir difusiones del *Ontologies Server* con la lista actualizada de ontologías, lo que le permitirá crear o destruir recursos y mantenerse al día.

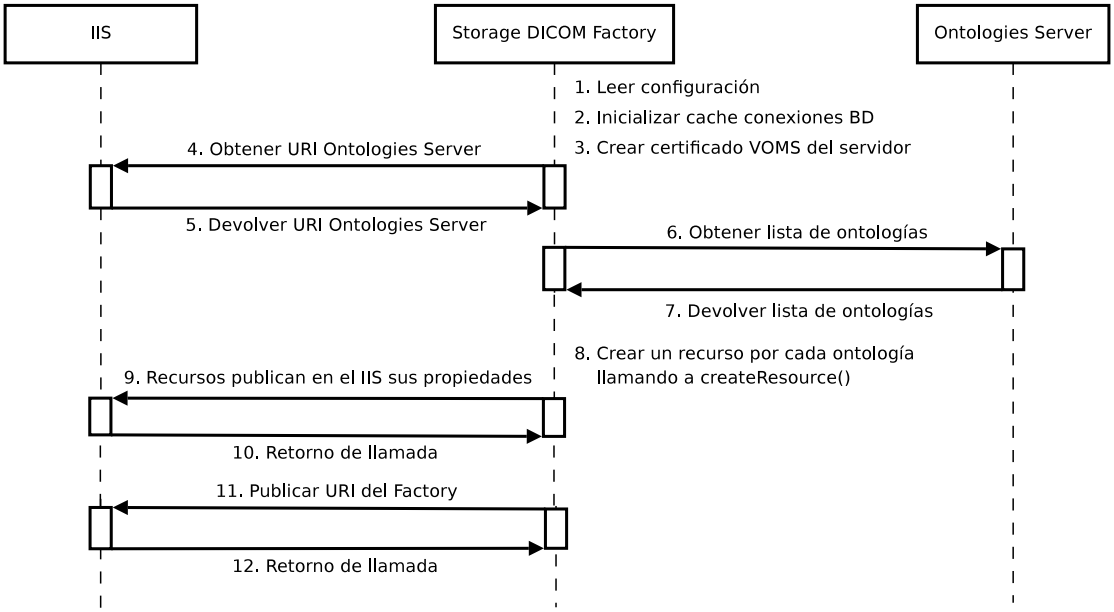


Figura 4.9: Inicialización del Storage DICOM Factory

Inicialización del Storage Broker Factory

La inicialización del *Storage Broker Factory* puede verse en la figura 4.10.

El constructor del servicio *Storage Broker Factory* es muy similar al constructor del *Storage DICOM Factory*, sólo difiere en el tipo de recursos (*Storage Broker Resources*) que se crean. El proceso de creación de recursos del *Storage Broker* se describe posteriormente.

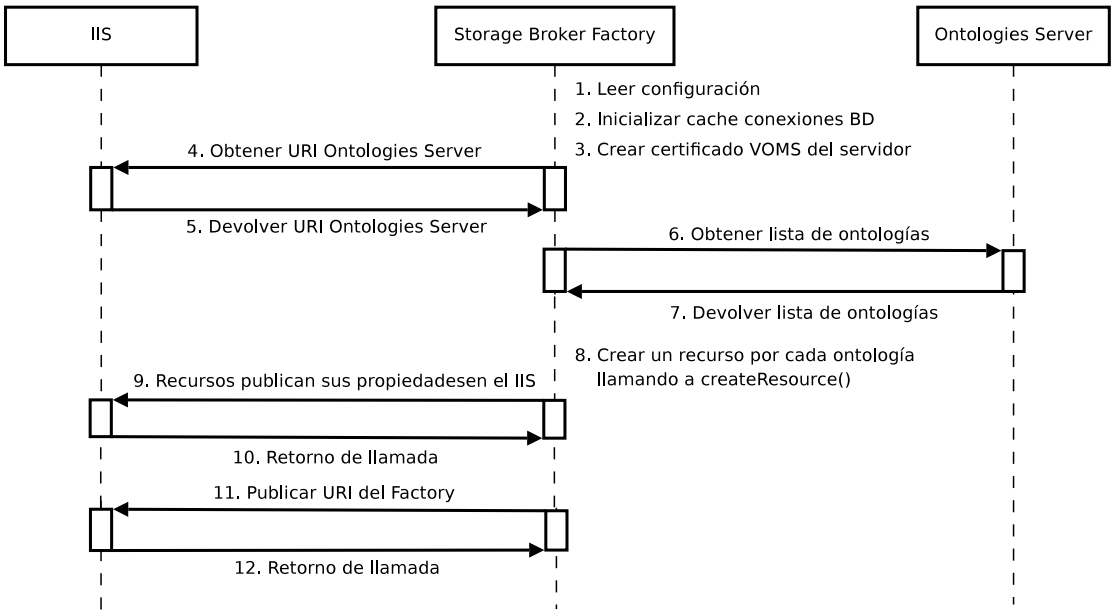


Figura 4.10: Inicialización del Storage Broker Factory

4.4.5. Creación de recursos

Los servicios GRID *Storage DICOM* y *Storage Broker* pueden soportar múltiples recursos, cada uno con valores diferentes en sus propiedades. En estos casos se utiliza el método *createResource()* de su servicio *factory* para crear nuevos recursos.

Cada recurso tiene un método *initialize()* que se ejecuta en el momento de su creación.

En esta sección hablamos acerca de cómo se crean los recursos y de qué acciones se realizan en el proceso de creación.

Creación de recursos del Ontologies Server

El *Ontologies Server* es de tipo *singleton*. No tiene servicio *Factory*. Esto significa que al iniciarse crea un único recurso con las propiedades de recurso que necesita para trabajar.

Creación de recursos del Storage DICOM

La creación de recursos del *Storage DICOM* puede verse en la figura 4.11.

El método *CreateResource()* del servicio *Factory* es el encargado de crear nuevos recursos. Puede recibir parámetros definidos por el usuario.

En primer lugar (1), el método lee los parámetros de entrada y los utiliza para inicializar el nuevo recurso, asignando el valor inicial de sus propiedades

de recurso. Cada recurso se encarga de un tipo de ontología y esto queda reflejado en sus propiedades de recurso.

El nuevo recurso se inicializa llamando a su método *initialize()* (paso 3). Esto se hace después de fijar el valor de inicio de las propiedades del recurso, ya que el método *initialize()* inicializa dichas propiedades pero es una función utilizada por el contenedor y no puede recibir parámetros.

En el método *initialize()*, el recurso carga su configuración (4), publica sus propiedades del recurso (5) y crea una colección en el servidor AMGA (6) que refleja la estructura de la ontología que tiene que gestionar (sólo si dicha estructura no existe aún).

Finalmente, el método *CreateResource()* publica las propiedades del nuevo recurso en el IIS.

Normalmente será el propio servicio factoría de cada servicio el que llame a su método *CreateResource()*, según la información que le llegue del *Ontologies Server*. De esta forma el diseño es más sencillo y rápido, ya que el *Ontologies Server* no invoca remotamente a *CreateResource()* por cada nueva ontología, sólo invoca una vez a *UpdateResources()* y este método es el que llamará las veces que sea necesario a *CreateResource()*.

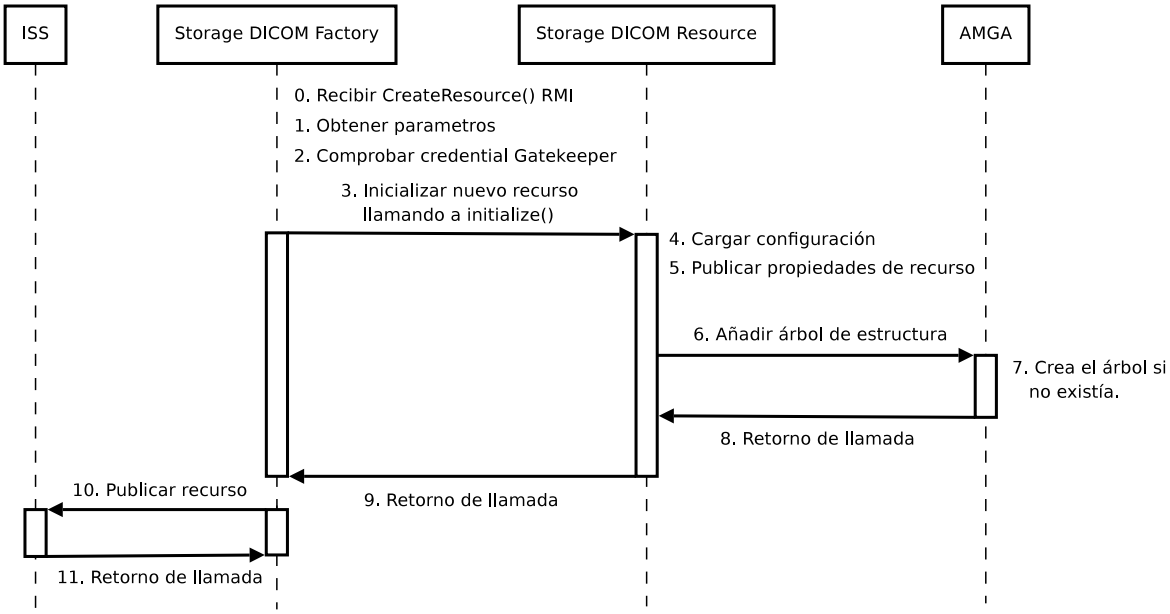


Figura 4.11: Creación de recursos del Storage DICOM

Creación de recursos del Storage Broker

La creación de recursos del *Storage Broker* puede verse en la figura 4.12.

El proceso de creación de recursos del *Storage Broker* es similar al proceso

de creación de recursos del *Storage DICOM*, únicamente difiere en los pasos realizados por el método *initialize()* del recurso que se va a crear.

Cada recurso de este servicio controla las consultas relativas a un determinado tipo de ontología.

En este caso contactamos con el IIS para preguntarle acerca de los servidores de catálogo AMGA gestionados por los servicios *StorageDICOM* correspondientes al tipo de ontología del recurso (6).

Cada vez que el recurso contacta con el IIS se almacena el instante de tiempo en una variable (8). Si después se realizan búsquedas en el *Storage Broker*, el recurso correspondiente a la ontología de la consulta comprobará esta variable temporal. Sólo preguntará de nuevo por los catálogos AMGA al IIS cuando el tiempo desde la última consulta sea mayor que un periodo definido por el usuario. La primera consulta al IIS se realizará siempre al crear el recurso.

El objetivo de comprobar que se haya sobrepasado el intervalo de tiempo es que, aunque se produzcan en el *Storage Broker* muchas consultas simultáneas, sólo se interrogue al IIS en la primera, ya que es poco probable que se añadan nuevos recursos del *Storage DICOM* en tan poco tiempo.

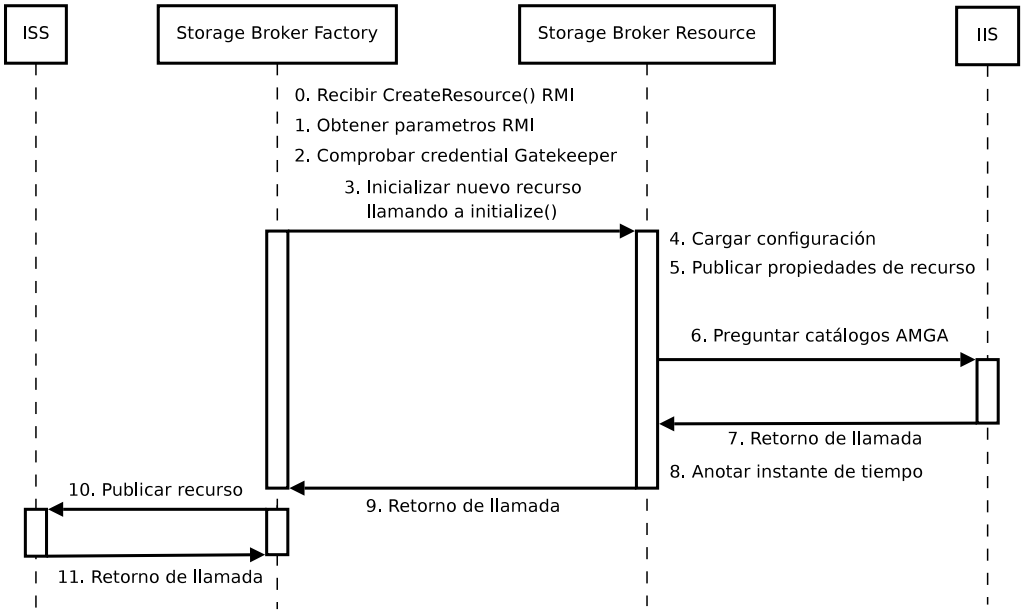


Figura 4.12: Creación de recursos del Storage Broker

4.4.6. Actualización de recursos

Cada servicio *Storage DICOM* y *Storage Broker* debe poseer un recurso por cada ontología registrada en el *Ontologies Server*. Si la lista de ontologías

cambia, los servicios deberán crear o destruir recursos para mantener su estado acorde a esta lista. En esta sección se describe cómo se mantiene el estado del sistema actualizado.

Hilo de actualización de recursos del *Ontologies Server*

El comportamiento del hilo de actualización de recursos puede verse en la figura 4.13.

Cada cierto tiempo el *Ontologies Server* envía a los servicios *Storage DICOM Factory* y *Storage Broker Factory* la lista actualizada de ontologías existentes en el sistema. Para ello, al final del proceso de inicialización crea un hilo de notificación. Dicho hilo llama concurrentemente al método UpdateResources() de todos los servicios *Storage DICOM Factory* (4) y al *Storage Broker Factory* (8) publicados en el IIS, enviando la lista de ontologías. Al ser las llamadas concurrentes las latencias de red se solapan.

Para autenticarse el *Ontologies Server* emplea el certificado de la máquina en la que se ejecuta, por lo que dicho certificado debe estar en la base de datos del *Gatekeeper* de los servicios factoría a los que se conecta. Además, el *Ontologies Server* a través de este hilo es el único con permisos para ejecutar el método UpdateResources(), lo que garantiza que no se realicen llamadas concurrentes del mismo, simplificando el diseño del sistema.

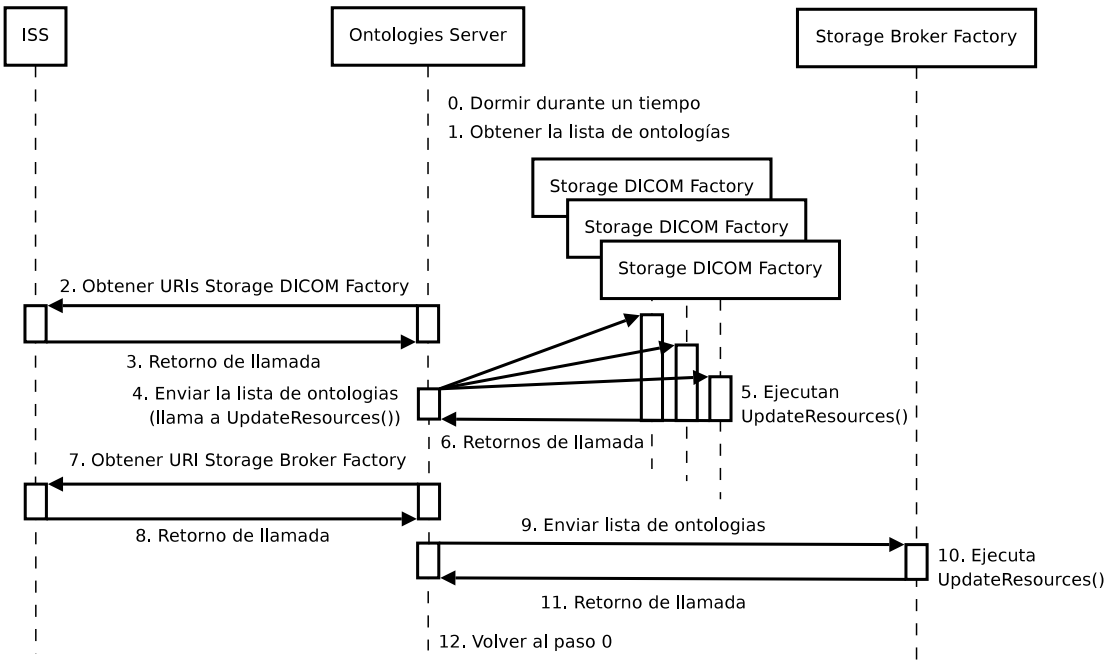


Figura 4.13: Hilo de actualización de recursos del Ontologies Server

Actualización de recursos del *Storage DICOM Factory*

La actualización de recursos del *Storage DICOM Factory* se ilustra en la figura 4.14.

Este es el comportamiento del método UpdateResources() del *Storage DICOM Factory*, que es llamado por el *Ontologies Server* cuando le envía al *Storage DICOM Factory* la lista actualizada de ontologías.

Una vez arrancado el *Storage DICOM Factory* la creación o destrucción de recursos se realiza únicamente a través de este método. Es el *Ontologies Server* el que toma la iniciativa en este proceso enviando periódicamente la información a los servicios factoría (funciona como un sistema de notificaciones basado en subscripción).

El método compara la lista de ontologías existente en el servicio con la nueva que le envía el *Ontologies Server* (4). El servicio factoría creará entonces nuevos recursos para las nuevas ontologías (6) y eliminará los recursos para los que ya no hayan ontologías (5).

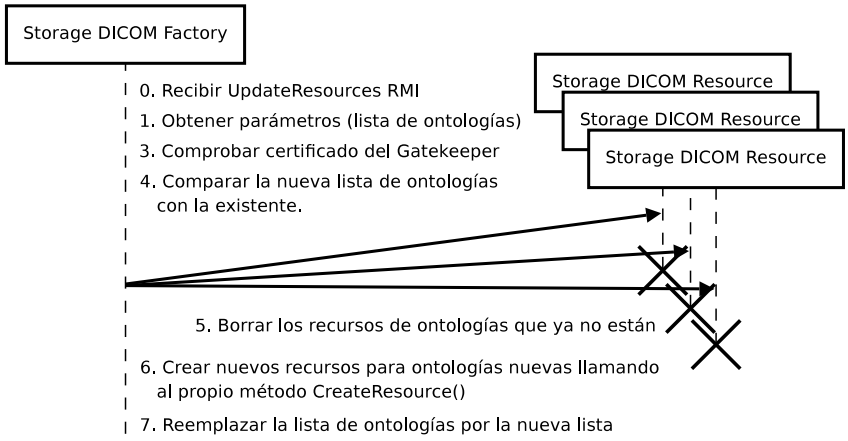


Figura 4.14: Actualización de recursos del Storage DICOM Factory

Actualización de recursos del *Storage Broker Factory*

La actualización de recursos del *Storage Broker Factory* se ilustra en la figura 4.15.

El método es idéntico al del *Storage DICOM Factory*, pero crea los recursos del *Storage Broker*.

4.4.7. Interfaz de los servicios

Después de la inicialización, los servicios GRID aceptan invocaciones remotas de sus métodos (RMI). Muchos de estos procedimientos involucran intercomunicación con otros servicios.

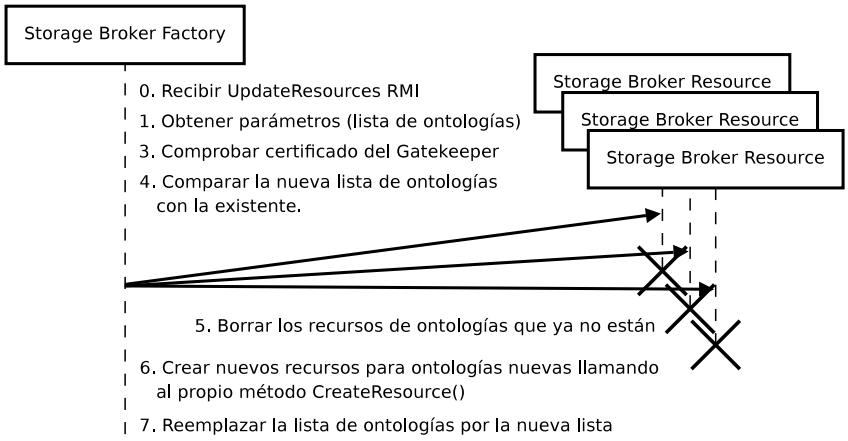


Figura 4.15: Actualización de recursos del Storage Broker Factory

4.4.7.1. Interfaz del *Ontologies Server*

Crear ontología La creación de ontologías del *Ontologies Server* se describe en la figura 4.16.

Si deseamos añadir un nuevo tipo de ontología a nuestro sistema, debemos utilizar este método del *Ontologies Server*.

En primer lugar, el *Ontologies Server* obtiene los parámetros de llamada y el certificado del usuario que realiza la RMI (1). Entonces el servicio comprueba en el *Gatekeeper* si el usuario tiene los permisos suficientes para ejecutar la operación correspondiente (2). En el paso 3 la ontología se añade a la base de datos del *Ontologies Server*. Este es el paso más importante, si falla la RMI no acabará correctamente.

Una vez que la ontología está en la base de datos tenemos que notificar a los servicios *Storage DICOM* y al servicio *Storage Broker* de la existencia de la misma, y también debemos crear los recursos de dichos servicios que se encargarán de gestionarla. Esto se hace despertando al hilo de notificación de ontologías (5), el cual enviará la lista actualizada de ontologías a los servicios *Storage DICOM Factory* y *Storage Broker Factory*.

Si estamos arrancando un servicio *Storage DICOM Factory* o *Storage Broker Factory* y ocurre que después de que el servicio solicite la lista de ontologías y antes de que se publique su URI en el IIS se introduce una nueva ontología, la lista de ontologías actualizada no llegará a este servicio. La lista de ontologías tampoco llegará si se produce un fallo de comunicación o si el IIS central cae y es consultado cuando está reiniciando de nuevo y no ha registrado de nuevo los recursos disponibles en el sistema.

Sin embargo, como el *Ontologies Server* envía la lista de ontologías cada cierto tiempo, los servicios factoría acabarán recibiendo la información y el estado quedará actualizado de forma adecuada.

Si un servicio *Storage DICOM Factory* o *Storage Broker Factory* no se encuentra en ejecución cuando se añade una nueva ontología no ocurre nada, ya que sincronizará sus recursos cuando pida la lista de ontologías al *Ontologies Server* al inicializarse.

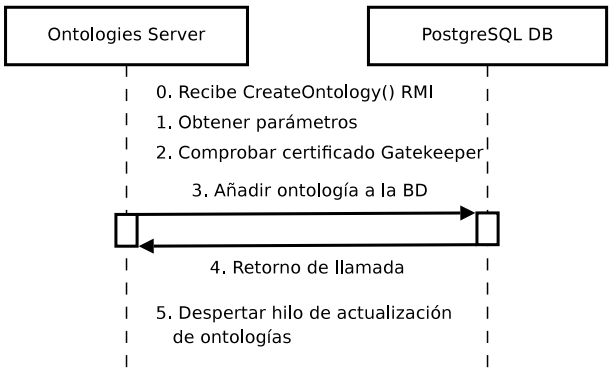


Figura 4.16: Creación de ontologías en el *Ontologies Server*

Borrar ontología El borrado de ontologías del *Ontologies Server* se describe en la figura 4.17.

El método *removeOntology()* es similar al método *createOntology()*, pero en vez de añadir una ontología la elimina.

El paso 3, en el que se borra la ontología de la base de datos del *Ontologies Server*, es crítico, al igual que añadir nuevas ontologías.

Para notificar de los cambios a los otros servicios se despierta al hilo que envía la lista de ontologías a los servicios *Storage DICOM Factory* y *Storage Broker Factory* (5), estos servicios al ver que falta una ontología eliminarán los recursos asociados. Eliminar un recurso del *Storage DICOM* no elimina la estructura de directorios ni los DICOM-SR del catálogo AMGA gestionados por dicho recurso.

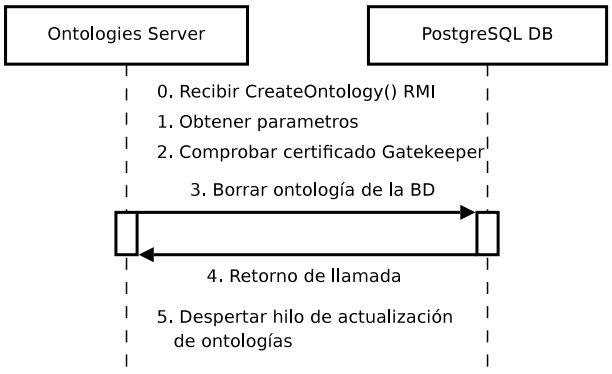


Figura 4.17: Borrado de ontologías en el *Ontologies Server*

Obtener ontología Devuelve una ontología concreta a través de un fichero XML.

Obtener todas las ontologías Devuelve una lista con todas las ontologías.

Actualizar servicios Esta función simplemente despierta el hilo de notificación de ontologías. Su finalidad es que el usuario pueda ordenar manualmente la actualización de los recursos de los servicios *StorageDICOM* y *Storage Broker* del sistema.

4.4.7.2. Interfaz del Storage DICOM

Añadir DICOM-SR El método añadir informes del *Storage DICOM* se describe en la figura 4.18.

Una vez que un recurso *Storage DICOM* ha sido creado un cliente puede acceder a él para añadir nuevos *DICOM-SR* al catálogo AMGA asociado. El hospital, la URI y el tipo de ontología de un recurso del servicio *Storage DICOM* se publican en el IIS.

En el paso 3, comprueba si el tipo de informe es el mismo que el tipo de ontología gestionado por el recurso *Storage DICOM* al que estamos accediendo.

Un *DICOM-SR* debe estar asociado a un estudio *DICOM* de imágenes, estas imágenes se almacenan en el LFC, así que antes de añadir un nuevo informe debemos comprobar que el estudio al que se asocia esté presente en el LFC (4).

Si las condiciones anteriores se satisfacen, el nuevo informe se añadirá al catálogo AMGA (6).

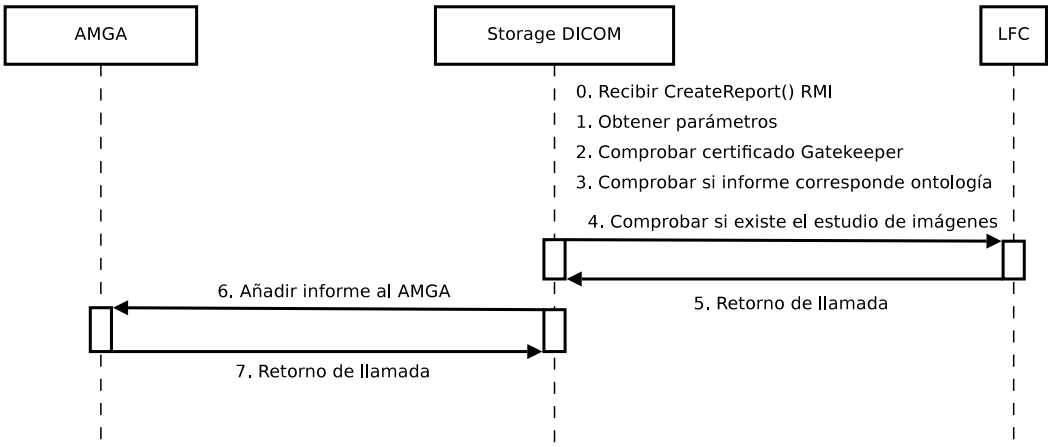


Figura 4.18: Añadir DICOM-SR del servicio *Storage DICOM*

Borrar DICOM-SR El método borrar informes del *Storage DICOM* se describe en la figura 4.19.

Si queremos borrar un informe del catálogo AMGA comprobaremos los permisos del usuario que solicita esta acción en el *Gatekeeper* y entonces borraremos el informe.

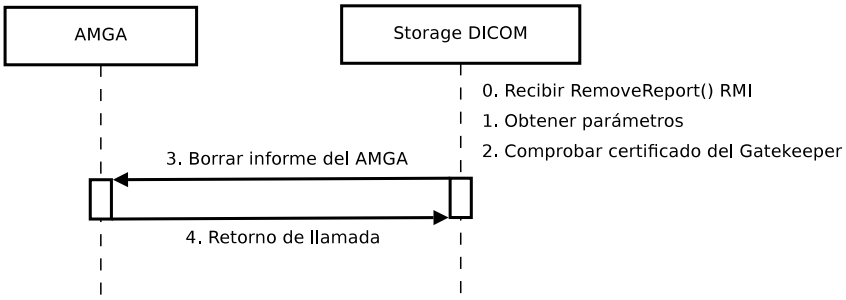


Figura 4.19: Eliminar DICOM-SR del servicio *Storage DICOM*

Descargar DICOM-SR El método descargar informes del *Storage DICOM* se describe en la figura 4.20.

Los *DICOM-SR* se almacenan en el catálogo AMGA. Por lo tanto, si queremos descargar una representación en XML de un *DICOM-SR* tendremos que acceder a la colección dentro de la estructura en árbol del catálogo que contiene las cadenas con los informes completos.

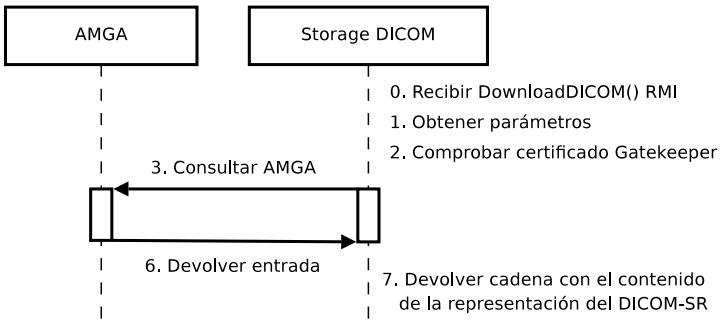


Figura 4.20: Descargar DICOM-SR del servicio *Storage DICOM*

4.4.7.3. Interfaz del Storage Broker

Consultar informes El método consultar informes del *Storage Broker* se describe en la figura 4.21.

Cada vez que una consulta llega al servicio *Storage Broker*, este debe actuar de mediador generando una consulta por cada recurso *Storage DICOM* de un tipo particular de ontología. Para ello debe consultar al IIS los recursos

Storage DICOM en el sistema que gestionen la misma ontología que el propio servicio.

Sin embargo hacer esta consulta al IIS cada vez que se buscan informes es costoso e innecesario, ya que si las consultas son muy próximas en el tiempo es poco probable que los recursos disponibles hayan cambiado y no es necesario consultar el IIS.

Por ello, el *Storage Broker* comprueba cuanto tiempo ha pasado desde la última consulta y contacta con el IIS sólo si ha vencido un plazo de tiempo determinado.

Una vez obtenida la lista, esta se almacena en el recurso del *Storage Broker* correspondiente al tipo de ontología al que pertenece la consulta y se pregunta directamente a los servidores AMGA. Las consultas se realizan concurrentemente (espera en 8), permitiendo el solapamiento de los tiempos de comunicación.

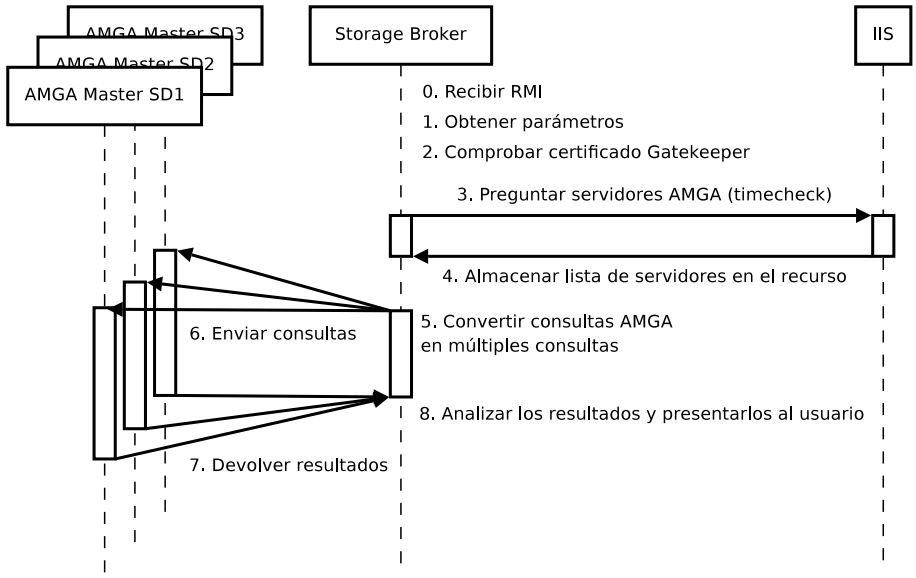


Figura 4.21: Consultar informes del servicio *Storage Broker*

4.5. La capa de comunicaciones

Para comunicarse los servicios utilizan el protocolo HTTP o el protocolo HTTPS para las conexiones seguras. Todas las llamadas a los servicios del *middleware* contienen un certificado X509 con extensiones VOMS, dicho certificado es necesario para autorizar las acciones de dicho usuario y para utilizarlo cuando los servicios necesitan contactar con componentes como el AMGA o el LFC. Dichos componentes de gLite utilizan conexiones SSL con certificado.

4.6. La capa *Middleware components*

La capa *Middleware components* se encarga de proveer un interfaz orientado a objetos al programador, facilitando los desarrollos de programas y abstrayendo de los aspectos de comunicación con los recursos a los programadores. Ha sido parte del desarrollo de esta tesis.

Los programadores pueden utilizar los componentes desarrollados por esta capa, para utilizar los recursos desplegados en la capa *Core Middleware* y *Server Services*, de esta forma no tendrá que pensar en si los servicios GRID funcionan sobre GT4.2 o sobre otro middleware, todos los detalles de comunicación y funcionamiento entre cliente y servidor son gestionados por la capa *Middleware components*.

La capa *Middleware components* necesita comunicarse con la capa *Core middleware* y también con la capa *Server services* para poder acceder al IIS y así obtener la URI de los servicios.

Si más adelante se decide cambiar la arquitectura del contenedor de los servicios del *middleware* las aplicaciones no tendrán que cambiarse, ya que los cambios necesarios sólo afectarán a la capa *Middleware components*.

A continuación se describen los componentes de la capa *Middleware components* desarrollados en el marco de esta tesis.

Componentes de inicio de sesión

Los componentes de inicio de sesión gestionan la configuración necesaria para las conexiones que se puedan dar con los servicios GRID desplegados.

TRENCADIS_CONFIGURATION Contiene información básica para utilizar el middleware (la dirección del IIS central), las variables de entorno con la ruta a los certificados de las CA y de los servidores VOMS y el directorio temporal que usará la parte cliente para las gestiones internas del middleware.

Si un componente necesita de uno de estos parámetros para funcionar necesitará recibir este componente en su constructor.

TRENCADIS_SESSION El componente de sesión añade a las variables que proporciona el componente *TRENCADIS_CONFIGURATION* la ruta a las credenciales del usuario y una VO, esto permite crear un certificado VOMS que se almacena dentro del propio objeto y que se utilizará para conectarse con todos los servicios del GRID.

Se utilizará un componente de este tipo distinto para cada sesión que se necesite.

Componentes para la gestión de ficheros

Estos componentes permiten almacenar las representaciones en XML que utilizamos para modelar los DICOM-SR y además contienen las operaciones básicas que podrían realizarse sobre dichas representaciones.

TRENCADIS_XMLDSRSTR_FILE Está diseñado para contener un fichero XML de estructura con una ontología. Dicho fichero está almacenado en una cadena de caracteres.

El componente además contiene las operaciones básicas sobre este tipo de fichero que provee el DSRHelper, como listar su estructura y obtener algunos de sus atributos básicos como el tipo de informe.

TRENCADIS_XMLDSR_FILE Puede contener una descripción en XML de un fichero DICOM-SR. Entre las acciones del DSRHelper que implementa destaca la conversión a fichero binario DICOM. También pueden obtenerse sus atributos básicos de tipo de informe, identificador de informe, estudio asociado y nombre del paciente.

Componentes para la realización de operaciones

Estos componentes realizan las operaciones sobre los servicios desplegados, por tanto necesitan de un objeto sesión válido para poder llevarse a término. Para realizar las operaciones realizan llamadas a los métodos de los servicios que se indican.

TRENCADIS_XMLDSRSTR_CREATE_ONTOLOGY Añade una ontología al sistema. Esto lo realiza contactando con el *Ontologies Server*.

TRENCADIS_XMLDSRSTR_GET_ONTOLOGY Descarga una ontología del *Ontologies Server*.

TRENCADIS_XMLDSRSTR_REMOVE_ONTOLOGY Borra una ontología del *Ontologies Server*.

TRENCADIS_UPDATE_STORAGE_SERVICES Fuerza al *Ontologies Server* a enviar la lista actualizada de ontologías a los servicios de tipo *Storage* para que actualicen sus recursos.

TRENCADIS_XMLDSR_DOWNLOAD Descarga un DICOM-SR del servidor *Storage DICOM* que se especifique.

TRENCADIS_XMLDSR_UPLOAD Introduce un DICOM-SR en el servidor *Storage DICOM* que se especifique.

TRENCADIS_XMLDSR_REMOVE Borra un DICOM-SR del servidor *Storage DICOM* que se especifique.

TRENCADIS_XMLDSR_SEARCH Cuando recibe una consulta, se conecta al servidor *Storage Broker* para que la distribuya entre los repositorios AMGA de los servicios *Storage DICOM* que haya disponibles disponibles.

El resultado de la búsqueda se almacena en un componente *TRENCADIS_SEARCH_RESULTS*.

TRENCADIS_SEARCH_RESULTS Almacena los resultados de una búsqueda organizados por repositorios. Cada repositorio contiene una lista con los elementos que concuerdan con los criterios de búsqueda. A petición del usuario puede descargarse el DICOM-SR asociado a cualquiera de los elementos mencionados.

4.7. La capa *Applications*

Las aplicaciones utilizan los componentes de la capa *Middleware components* que se han descrito anteriormente. Esto permite al programador olvidarse de la problemática de las capas inferiores.

El trabajo realizado en esta tesis de máster y en la capa del *Middleware*, ha permitido la creación de un interfaz WEB, dicho interfaz web utiliza las funciones implementadas por los servicios desarrollados.

Capítulo 5

Conclusiones y aportaciones

Las aportaciones se han centrado en tres vertientes. Esta tesis de máster se centra en la modificación de la arquitectura del middleware TRENCADIS. En primer lugar, se ha modificado el middleware para que permita la federación de informes médicos estructurados (manteniendo el árbol de elementos de los informes).

En segundo lugar, se ha adaptado el middleware para que utilice como *backend* los servicios del middleware gLite (AMGA, LFC), aumentando sus capacidades de despliegue. Finalmente, se han realizado importantes cambios en los componentes del middleware que han aumentado su eficiencia y fiabilidad. A continuación se describen en detalle estas aportaciones.

Se han cumplido los objetivos descritos en el apartado 3, modificándose la arquitectura de TRENCADIS y mejorando el *middleware*, que ahora permite a los investigadores compartir estudios de imágenes y diagnósticos médicos de una manera más eficiente.

Los responsables de los distintos centros pueden reunirse y definir ontologías que se almacenan en un servidor centralizado, el *Ontologies Server*. Esto garantiza que los datos almacenados en todos los centros sean homogéneos, aunque se permite cierta flexibilidad.

El servicio *Storage DICOM* gestiona los repositorios de metadatos de cada centro y el servicio *Storage Broker* permite realizar consultas distribuidas con la información de todos los repositorios (este servicio actúa de mediador conectándose a todos los centros).

Estos tres servicios se han modificado, reescribiéndose desde cero para hacerlos compatibles con la versión 4.2.1 de *Globus Toolkit*. Los servicios desarrollados antes eran de tipo *singleton*, ahora permiten gestionar múltiples recursos del servicio, uno por ontología registrada en el sistema, aprovechando mejor la expresividad de los servicios GRID. También se han implementado mecanismos de sincronización entre los servicios que antes no existían en el sistema, como los procesos de inicialización de los servicios factoría o el sistema de notificaciones del *Ontologies Server*, esto se ha hecho con la final-

idad de mantener sincronizados los recursos de los servicios con el número de ontologías registradas en el *Ontologies Server*.

Los servicios ahora utilizan componentes de gLite, como el AMGA o el LFC, para almacenar los metadatos y los estudios de imágenes. Esto permite aprovechar las características nuevas que se añadan en futuras versiones de gLite y hace más acorde al *middleware* con la definición de data GRID. Además, permite aprovecharse de infraestructuras existentes que utilicen gLite.

Los metadatos se forman a partir del contenido de los DICOM-SR. Para almacenar los ficheros DICOM-SR, se ha creado una representación en XML de los mismos. Dicha representación es la que se utiliza al almacenar la información, antes no se utilizaba. Además, al almacenar los ficheros DICOM-SR en el AMGA se respeta la estructura del árbol del documento utilizando la estructura de directorios del catálogo AMGA, antes los DICOM-SR se almacenaban en una tabla de una base de datos de forma plana. Esta representación en árbol ofrece mucha más expresividad en las búsquedas y no implica una sobrecarga considerable, teniendo en cuenta el aumento de la funcionalidad (estructura de directorios y autorización del AMGA).

Para que los servicios puedan comunicarse con el servidor de metadatos AMGA y con el LFC, se ha programado el módulo *DSR AMGA Manager*, que utiliza parsers SAX para analizar el contenido de las representaciones en XML de los DICOM-SR e introducir la información en el AMGA. También puede trabajar con los datos existentes en el LFC.

También se ha desarrollado la capa *Middleware components*, que se encarga de proporcionar a las aplicaciones la funcionalidad del middleware mediante orientación a objetos, ocupándose de los aspectos de comunicación con los servicios propios de *Globus Toolkit*. A la hora de desarrollar una aplicación, utilizar esta capa simplificará la tarea del desarrollador.

Otra de las aportaciones ha sido la implementación de un componente de los servicios que permite crear *pools* de conexiones con las bases de datos. Así, cuando un componente como el *Gatekeeper* necesita conectarse a la base de datos, no tiene que crear conexiones nuevas, ya que reutiliza las conexiones en la caché.

El desarrollo motivado por esta tesis de máster ha creado un middleware funcional, sobre el que ha podido desarrollarse una aplicación WEB con la que los radiólogos pueden acceder a las imágenes. En este momento se está trabajando en colaboración con miembros del hospital universitario *Doctor Peset* de Valencia para implantar este sistema en el ámbito de la mamografía.

Capítulo 6

Artículos y proyectos asociados

6.1. Publicaciones

El desarrollo motivado por esta tesis ha generado hasta el momento las siguientes publicaciones:

- Un abstract en el congreso *4th EGEE User forum* titulado "Using AMGA to Store and Organise DICOM Structured Reports", cuyos autores son Ignacio Blanquer, Vicente Hernández, José Salavert y Damián Segrelles. ISBN: 978-92-9083-326-0. Página: 92.

- Un artículo en el congreso *HealthGrid 2009* titulado "Using Grid-Enabled Distributed Metadata Database to Index DICOM-SR", cuyos autores son Ignacio Blanquer, Vicente Hernández, José Salavert y Damián Segrelles. ISBN: 978-1-60750-027-8. Volumen: 147. Página: 117-126.

- Un artículo en la revista *Journal of Grid Computing* que está pendiente de revisión y se titula "Integrating TRENCADIS Components in gLite to Share DICOM Medical Images and Structured Reports", cuyos autores son Ignacio Blanquer, Vicente Hernández, José Salavert y Damián Segrelles. ISSN: 1570-7873 y 1572-9184 (versión electrónica)

6.2. Proyectos asociados

El trabajo de esta tesis de máster se enmarca dentro del ámbito del proyecto EGEE. El proyecto europeo EGEE une a expertos de más de 50 paises con el objetivo común de construir una infraestructura GRID que esté disponible en todo momento para los científicos.

El proyecto pretende proporcionar a los investigadores, académicos y a la industria el acceso a los grandes recursos computacionales que proporciona el GRID, independientemente de su situación geográfica. Además el proyecto

pretende proporcionar servicio a todas las áreas de la ciencia que necesiten capacidades altas de cómputo.

El proyecto se centra en dos objetivos principales:

- Construir un GRID robusto, consistente y seguro que atraiga a nuevas instituciones con recursos computacionales y usuarios. El uso del GRID debe abarcar múltiples disciplinas.
- Mejorar continuamente el *middleware* con la finalidad de ofrecer servicios fiables y preparar la migración del actual sistema GRID Europeo basado en proyectos globales a un modelo sostenible basado en la federación de infraestructuras según las iniciativas GRID nacionales de cada país.

Actualmente la infraestructura de EGEE es la mayor de Europa, con más de 150.000 CPU disponibles de media en todo momento para los 14.000 usuarios registrados. A diario se lanzan 330.000 trabajos en la infraestructura GRID de EGEE.

Capítulo 7

Trabajos futuros

7.1. Anonimización

Uno de los problemas al tratar con información médica es que hay determinados datos del paciente que deben permanecer confidenciales para según quién accede a la información. En este sentido, el catálogo de metadatos AMGA nos permite asociar permisos a nivel de los campos del informe con facilidad. Podemos asignar permisos de acceso a cada uno de los campos de los DICOM-SR, modificando los permisos de cada directorio representante de una colección en el AMGA. De esta forma el nombre del paciente puede estar disponible para el personal del hospital a cargo del paciente, mientras que puede no estarlo para un usuario no relacionado con la asistencia a dicho paciente, de una forma sencilla y compatible con el modelo de seguridad del GRID. Dicho de otro modo, podemos asignar distintos permisos de acceso a cada una de las partes de los DICOM-SR.

Actualmente, los servicios no emplean esta funcionalidad del AMGA, aunque en un futuro se puede mejorar el sistema para que al añadir un fichero de estructura se especifiquen los permisos, o para que los ficheros de estructura contengan la lista de control de acceso de cada *content item* de los DICOM-SR de ese tipo de ontología.

7.2. Federación

Otro punto importante es la federación, tal y como se menciona en [19], los datos de un dominio administrativo pueden hacerse accesibles a otros, con determinadas restricciones.

El catálogo AMGA permite federar datos siguiendo el esquema conocido como *Master-Slave Zones* en [19]. Toda la información almacenada en el servidor maestro se copia a los servidores esclavo. Los datos copiados se representan en un punto de montaje en un directorio del servidor esclavo y no pueden modificarse.

Actualmente, la federación se consigue desde el lado del cliente, invocando múltiples búsquedas y combinando los resultados. Si bien la lógica necesaria para la federación de plantillas ha sido desarrollada en esta tesis, AMGA no soporta federación desde el servidor.

El planteamiento en un futuro consiste en modificar el catálogo AMGA para que siga representando los datos federados montando los contenidos de los servidores maestros en la estructura de directorios de los servidores esclavos, pero sin copiar los datos del servidor maestro en el esclavo. Para ello el servidor esclavo deberá redireccionar los comandos que reciba que afecten a los datos montados a los servidores maestros.

En el desarrollo realizado para esta tesis, el servicio *Storage Broker* actúa como un mediador, lanzando las consultas a los distintos repositorios y uniendo el resultado.

Si se sigue este planteamiento, el servicio *Storage Broker* ya no tendrá que actuar de mediador, sólo tendrá que conectarse a un catálogo AMGA que actúe de esclavo y monte el contenido de los servidores maestros de los repositorios de todos los centros. El *Storage Broker* enviará las consultas solamente a este servicio y se encargará de gestionar los puntos de montaje, simplificando el diseño del servicio.

7.3. Procesos asociados a las imágenes

En la actualidad existen numerosos algoritmos que se aplican a las imágenes médicas para mejorar su visualización, eliminar ruido, ajustar el contraste o borrar caras y mantener al paciente en el anonimato.

Esta propuesta consiste en desarrollar servicios GRID que puedan recibir instrucciones sobre que algoritmos deben aplicarse para procesar un conjunto de estudios de imágenes.

TRENCADIS ya da soporte al proceso individual de las imágenes, si bien necesitaría complementarse con una gestión de *pipelines* de proceso.

Apéndice A

Representación en XML de los DICOM-SR

A.1. Fichero de estructura de una ontología

Este es el ejemplo de un fichero de estructura u ontología cuya representación aparece en el árbol de la figura 4.3:

```
<DICOM_SR reportType="001">

  <PNAME>
    <CONCEPT_NAME>
      <CODE_VALUE>OBS</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Observer</CODE_MEANING>
    </CONCEPT_NAME>
  </PNAME>

  <TEXT>
    <CONCEPT_NAME>
      <CODE_VALUE>HIS</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>History</CODE_MEANING>
    </CONCEPT_NAME>
  </TEXT>

  <UIDREF>
    <CONCEPT_NAME>
      <CODE_VALUE>UID</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>UID Reference</CODE_MEANING>
    </CONCEPT_NAME>
```

```
</UIDREF>

<DATETIME>
  <CONCEPT_NAME>
    <CODE_VALUE>DTR</CODE_VALUE>
    <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
    <CODE_MEANING>Report Date</CODE_MEANING>
  </CONCEPT_NAME>
</DATETIME>

<CONTAINER>
  <CONCEPT_NAME>
    <CODE_VALUE>FI</CODE_VALUE>
    <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
    <CODE_MEANING>Findings</CODE_MEANING>
  </CONCEPT_NAME>
  <CHILDS>
    <TEXT>
      <CONCEPT_NAME>
        <CODE_VALUE>FI1</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Finding 1</CODE_MEANING>
      </CONCEPT_NAME>
      <CHILDS>
        <SCOORD>
          <CONCEPT_NAME>
            <CODE_VALUE>C1</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Coordinate 1</CODE_MEANING>
          </CONCEPT_NAME>
        </SCOORD>
        <SCOORD>
          <CONCEPT_NAME>
            <CODE_VALUE>C2</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Coordinate 2</CODE_MEANING>
          </CONCEPT_NAME>
        </SCOORD>
        <CODE>
          <CONCEPT_NAME>
            <CODE_VALUE>LOC</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Localization</CODE_MEANING>
          </CONCEPT_NAME>
```

```
    </CODE>
    <NUM>
      <CONCEPT_NAME>
        <CODE_VALUE>SIZ</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Size</CODE_MEANING>
      </CONCEPT_NAME>
    </NUM>
    <CODE>
      <CONCEPT_NAME>
        <CODE_VALUE>MK</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Mass kind</CODE_MEANING>
      </CONCEPT_NAME>
    </CODE>
  </CHILDS>
</TEXT>

<TEXT>
  <CONCEPT_NAME>
    <CODE_VALUE>FI2</CODE_VALUE>
    <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
    <CODE_MEANING>Finding 2</CODE_MEANING>
  </CONCEPT_NAME>
  <CHILDS>
    <SCOORD>
      <CONCEPT_NAME>
        <CODE_VALUE>C1</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Coordinate 1</CODE_MEANING>
      </CONCEPT_NAME>
    </SCOORD>
    <SCOORD>
      <CONCEPT_NAME>
        <CODE_VALUE>C2</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Coordinate 2</CODE_MEANING>
      </CONCEPT_NAME>
    </SCOORD>
    <CODE>
      <CONCEPT_NAME>
        <CODE_VALUE>LOC</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Localization</CODE_MEANING>
```

```
        </CONCEPT_NAME>
      </CODE>
      <NUM>
        <CONCEPT_NAME>
          <CODE_VALUE>SIZ</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Size</CODE_MEANING>
        </CONCEPT_NAME>
      </NUM>
      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>MK</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Mass kind</CODE_MEANING>
        </CONCEPT_NAME>
      </CODE>
    </CHILDS>
  </TEXT>

  <TEXT>
    <CONCEPT_NAME>
      <CODE_VALUE>FI3</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Finding 3</CODE_MEANING>
    </CONCEPT_NAME>
    <CHILDS>
      <SCOORD>
        <CONCEPT_NAME>
          <CODE_VALUE>C1</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Coordinate 1</CODE_MEANING>
        </CONCEPT_NAME>
      </SCOORD>
      <SCOORD>
        <CONCEPT_NAME>
          <CODE_VALUE>C2</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Coordinate 2</CODE_MEANING>
        </CONCEPT_NAME>
      </SCOORD>
      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>LOC</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
```

```
            <CODE_MEANING>Localization</CODE_MEANING>
          </CONCEPT_NAME>
        </CODE>
        <NUM>
          <CONCEPT_NAME>
            <CODE_VALUE>SIZ</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Size</CODE_MEANING>
          </CONCEPT_NAME>
        </NUM>
        <CODE>
          <CONCEPT_NAME>
            <CODE_VALUE>MK</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Mass kind</CODE_MEANING>
          </CONCEPT_NAME>
        </CODE>
      </CHILDS>
    </TEXT>

  </CHILDS>

</CONTAINER>

<CONTAINER>
  <CONCEPT_NAME>
    <CODE_VALUE>CON</CODE_VALUE>
    <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
    <CODE_MEANING>Conclusions</CODE_MEANING>
  </CONCEPT_NAME>
  <CHILDS>
    <TEXT>
      <CONCEPT_NAME>
        <CODE_VALUE>CON1</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Conclusion 1</CODE_MEANING>
      </CONCEPT_NAME>
    </TEXT>

    <TEXT>
      <CONCEPT_NAME>
        <CODE_VALUE>CON2</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Conclusion 2</CODE_MEANING>
```

```
        </CONCEPT_NAME>
      </TEXT>
    </CHILDS>

  </CONTAINER>

  <CONTAINER>
    <CONCEPT_NAME>
      <CODE_VALUE>DIAG</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Diagnosis</CODE_MEANING>
    </CONCEPT_NAME>
    <CHILDS>
      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>DIAG1</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Diagnosis 1</CODE_MEANING>
        </CONCEPT_NAME>
      </CODE>

      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>DIAG2</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Diagnosis 2</CODE_MEANING>
        </CONCEPT_NAME>
      </CODE>

      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>DIAG3</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Diagnosis 3</CODE_MEANING>
        </CONCEPT_NAME>
      </CODE>
    </CHILDS>
  </CONTAINER>

</DICOM_SR>
```

Según lo explicado en el apartado 4.2.1.2, dentro de **Findings** podríamos añadir un segundo elemento **Finding 2** sin ningún problema. Nótese que sólo aparece la información de los *concept names* y no aparecen los campos que

contienen la información de los elementos (como el campo TEXT_VALUE de los tipo TEXT).

A.2. Fichero de representación en XML de los DICOM-SR

El siguiente fichero es una representación en XML de un DICOM-SR cuya estructura se corresponde con la del árbol de la figura 4.3.

```
<DICOM_SR reportType="001" reportID="a" studyUID="001" patientsName="HDL">

  <PNAME>
    <CONCEPT_NAME>
      <CODE_VALUE>OBS</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Observer</CODE_MEANING>
    </CONCEPT_NAME>

    <PERSON_NAME>Clunie^David^A^Dr</PERSON_NAME>
  </PNAME>

  <TEXT>
    <CONCEPT_NAME>
      <CODE_VALUE>HIS</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>History</CODE_MEANING>
    </CONCEPT_NAME>

    <TEXT_VALUE>Malignant melanoma excised 1Y</TEXT_VALUE>
  </TEXT>

  <UIDREF>
    <CONCEPT_NAME>
      <CODE_VALUE>UID</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>UID Reference</CODE_MEANING>
    </CONCEPT_NAME>

    <UID>1.2.3.4.5.12345</UID>
  </UIDREF>

  <DATETIME>
    <CONCEPT_NAME>
```

```
      <CODE_VALUE>DTR</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Report Date</CODE_MEANING>
    </CONCEPT_NAME>

    <DATETIME_VALUE>20000712102512.582+0200</DATETIME_VALUE>
  </DATETIME>

  <CONTAINER>
    <CONCEPT_NAME>
      <CODE_VALUE>FI</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Findings</CODE_MEANING>
    </CONCEPT_NAME>

    <CHILDS>
      <TEXT>
        <CONCEPT_NAME>
          <CODE_VALUE>FI1</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Finding 1</CODE_MEANING>
        </CONCEPT_NAME>

        <TEXT_VALUE>First Melanoma</TEXT_VALUE>

        <CHILDS>

          <SCOORD>
            <CONCEPT_NAME>
              <CODE_VALUE>C1</CODE_VALUE>
              <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
              <CODE_MEANING>Coordinate 1</CODE_MEANING>
            </CONCEPT_NAME>

            <GRAPHIC_DATA>105.6,100.0,108.3,100.0</GRAPHIC_DATA>
            <GRAPHIC_TYPE>POLYLINE</GRAPHIC_TYPE>
          </SCOORD>

          <SCOORD>
            <CONCEPT_NAME>
              <CODE_VALUE>C2</CODE_VALUE>
              <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
              <CODE_MEANING>Coordinate 2</CODE_MEANING>
            </CONCEPT_NAME>
```

```
    <GRAPHIC_DATA>104.6,102.0</GRAPHIC_DATA>
    <GRAPHIC_TYPE>POINT</GRAPHIC_TYPE>
  </SCOORD>

  <CODE>
    <CONCEPT_NAME>
      <CODE_VALUE>LOC</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Localization</CODE_MEANING>
    </CONCEPT_NAME>

    <CODED_ENTRY>
      <ENTRY_VALUE>RL</ENTRY_VALUE>
      <ENTRY_SCHEMA>GRYCAP</ENTRY_SCHEMA>
      <ENTRY_MEANING>Right Lung</ENTRY_MEANING>
    </CODED_ENTRY>
  </CODE>

  <NUM>
    <CONCEPT_NAME>
      <CODE_VALUE>SIZ</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Size</CODE_MEANING>
    </CONCEPT_NAME>

    <UNITS>
      <UNITS_VALUE>CM</UNITS_VALUE>
      <UNITS_SCHEMA>GRYCAP</UNITS_SCHEMA>
      <UNITS_MEANING>Centimeters</UNITS_MEANING>
    </UNITS>

    <NUMERIC_VALUE>2</NUMERIC_VALUE>
  </NUM>

  <CODE>
    <CONCEPT_NAME>
      <CODE_VALUE>MK</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Mass kind</CODE_MEANING>
    </CONCEPT_NAME>

    <CODED_ENTRY>
      <ENTRY_VALUE>SPH</ENTRY_VALUE>
```

```
          <ENTRY_SCHEMA>GRYCAP</ENTRY_SCHEMA>
          <ENTRY_MEANING>Spherical</ENTRY_MEANING>
        </CODED_ENTRY>
      </CODE>

    </CHILDS>
  </TEXT>

  <TEXT>
    <CONCEPT_NAME>
      <CODE_VALUE>FI2</CODE_VALUE>
      <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
      <CODE_MEANING>Finding 2</CODE_MEANING>
    </CONCEPT_NAME>

    <TEXT_VALUE>Right lung malformation</TEXT_VALUE>

    <CHILDS>
      <SCOORD>
        <CONCEPT_NAME>
          <CODE_VALUE>C1</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Coordinate 1</CODE_MEANING>
        </CONCEPT_NAME>

        <GRAPHIC_DATA>128.5,128.5</GRAPHIC_DATA>
        <GRAPHIC_TYPE>POINT</GRAPHIC_TYPE>
      </SCOORD>

      <SCOORD>
        <CONCEPT_NAME>
          <CODE_VALUE>C2</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Coordinate 2</CODE_MEANING>
        </CONCEPT_NAME>

        <GRAPHIC_DATA>129.6,128.1</GRAPHIC_DATA>
        <GRAPHIC_TYPE>POINT</GRAPHIC_TYPE>
      </SCOORD>

      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>LOC</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
```

```
            <CODE_MEANING>Localization</CODE_MEANING>
          </CONCEPT_NAME>

          <CODED_ENTRY>
            <ENTRY_VALUE>RL</ENTRY_VALUE>
            <ENTRY_SCHEMA>GRYCAP</ENTRY_SCHEMA>
            <ENTRY_MEANING>Right Lung</ENTRY_MEANING>
          </CODED_ENTRY>
        </CODE>

        <NUM>
          <CONCEPT_NAME>
            <CODE_VALUE>SIZ</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Size</CODE_MEANING>
          </CONCEPT_NAME>

          <UNITS>
            <UNITS_VALUE>CM</UNITS_VALUE>
            <UNITS_SCHEMA>GRYCAP</UNITS_SCHEMA>
            <UNITS_MEANING>Centimeters</UNITS_MEANING>
          </UNITS>

          <NUMERIC_VALUE>1</NUMERIC_VALUE>
        </NUM>

        <CODE>
          <CONCEPT_NAME>
            <CODE_VALUE>MK</CODE_VALUE>
            <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
            <CODE_MEANING>Mass kind</CODE_MEANING>
          </CONCEPT_NAME>

          <CODED_ENTRY>
            <ENTRY_VALUE>IRR</ENTRY_VALUE>
            <ENTRY_SCHEMA>GRYCAP</ENTRY_SCHEMA>
            <ENTRY_MEANING>Irregular</ENTRY_MEANING>
          </CODED_ENTRY>
        </CODE>

      </CHILDS>
    </TEXT>
  </CHILDS>
```

```
</CONTAINER>

<CONTAINER>
  <CONCEPT_NAME>
    <CODE_VALUE>CON</CODE_VALUE>
    <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
    <CODE_MEANING>Conclusions</CODE_MEANING>
  </CONCEPT_NAME>

  <CONTINUITY>CONTINUOUS</CONTINUITY>

  <CHILDS>
    <TEXT>
      <CONCEPT_NAME>
        <CODE_VALUE>CON1</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Conclusion 1</CODE_MEANING>
      </CONCEPT_NAME>

      <TEXT_VALUE>Cannon-ball metastases</TEXT_VALUE>
    </TEXT>

    <TEXT>
      <CONCEPT_NAME>
        <CODE_VALUE>CON2</CODE_VALUE>
        <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
        <CODE_MEANING>Conclusion 2</CODE_MEANING>
      </CONCEPT_NAME>

      <TEXT_VALUE>A rigth lung malformation exists but has no relation with the
    </TEXT>
  </CHILDS>

</CONTAINER>

<CONTAINER>
  <CONCEPT_NAME>
    <CODE_VALUE>DIAG</CODE_VALUE>
    <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
    <CODE_MEANING>Diagnosis</CODE_MEANING>
  </CONCEPT_NAME>
  <CHILDS>
    <CODE>
      <CONCEPT_NAME>
```

```
          <CODE_VALUE>DIAG1</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Diagnosis 1</CODE_MEANING>
        </CONCEPT_NAME>

        <CODED_ENTRY>
          <ENTRY_VALUE>1</ENTRY_VALUE>
          <ENTRY_SCHEMA>MEDDIS</ENTRY_SCHEMA>
          <ENTRY_MEANING>Malignant melanoma</ENTRY_MEANING>
        </CODED_ENTRY>
      </CODE>

      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>DIAG2</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Diagnosis 2</CODE_MEANING>
        </CONCEPT_NAME>

        <CODED_ENTRY>
          <ENTRY_VALUE>2</ENTRY_VALUE>
          <ENTRY_SCHEMA>MEDDIS</ENTRY_SCHEMA>
          <ENTRY_MEANING>Multiple melanoma</ENTRY_MEANING>
        </CODED_ENTRY>

      </CODE>

      <CODE>
        <CONCEPT_NAME>
          <CODE_VALUE>DIAG3</CODE_VALUE>
          <CODE_SCHEMA>GRYCAP</CODE_SCHEMA>
          <CODE_MEANING>Diagnosis 3</CODE_MEANING>
        </CONCEPT_NAME>

        <CODED_ENTRY>
          <ENTRY_VALUE>3</ENTRY_VALUE>
          <ENTRY_SCHEMA>MEDDIS</ENTRY_SCHEMA>
          <ENTRY_MEANING>Patological malformation</ENTRY_MEANING>
        </CODED_ENTRY>
      </CODE>
    </CHILDS>
  </CONTAINER>

</DICOM_SR>
```

Bibliografía

[1] N. Santos B. Koblitz. *AMGA user's and administrator's manual*, October 2008. Version 1.9 of AMGA.

[2] Ignacio Blanquer, Vicente Hernández, Javier Meseguer, and Damià Segrelles. Deployment of a cyber-infrastructure on top of trencadis architecture to share and create dicom studies and structured reports. *Ibergrid conference proceedings*, 2007.

[3] Stephen Burke, Simone Campana, Patricia Mendez Lorenzo, Christopher Nater, Roberto Santinelli, and Andrea Sciaba. *GLite 3.1 User Guide*, April 2009. Version 1.2.

[4] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, March 02 1999.

[5] Vincenzo Ciaschini, Valerio Venturi, and Andrea Ceccanti. The voms attribute certificate format. *Open Grid Forum Technical Report*, 2006.

[6] Dr. David A. Clunie. *DICOM Structured Reporting*. PixelMed Publishing, 2000.

[7] EGEE. Egee project home page. `http://www.eu-egee.org/`.

[8] Steven Decker et Al. Cabig overview. *Technical Report of the National Center for Research Resources*, 2006.

[9] Luis Ferreira, Viktors Berstis, Jonathan Armstrong, Mike Kendzierski, Andreas Neukoetter, Masanobu Takagi, Richard Bing-Wo, Adeeb Amir, Ryo Murakawa, Olegario Hernández, James Magowan, and Norbert Bieberstein. *Introduction to GRID computing with Globus*. IBM RedbooksPixelMed Publishing, 2003.

[10] I. Foster, C. Kesselman, G. Tsudik, and S.Tuecke. A security architecture for computational grids. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 83–92, San Francisco, California, November 1998. ACM Press.

[11] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 1999.

[12] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, June 28 2002.

[13] Ian T. Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *CoRR*, cs.AR/0103025, 2001. informal publication.

[14] Jeffrey S. Grethe, Chaitain Baru, Amarnath Gupta, Mark James, Bertram Ludaescher, Maryann E. Martone, Philip M. Papadopoulos, Steven T. Peltier, Arcot Rajasekar, Simone Santini, IlyaÑ. Zaslavsky, and Mark H. Ellisman. Biomedical informatics research network: Building a national collaboratory to hasten the derivation of new understanding and treatment of disease. *Studies in health technology and informatics, Vol. 112, pp. 100-109*, 2005.

[15] Joshy Joseph. Handling attachments in soap. *IBM Software Group Technical Report*, 2002.

[16] B. Koblitz, N. Santos, and V. Pose. The AMGA metadata service. 2008.

[17] Johan Montagnat, Ákos Frohner, Daniel Jouvenot, Christophe Pera, Peter Kunszt, Birger Koblitz, Nuno Santos, Charles Loomis, Romain Texier, Diane Lingrand, Patrick Guio, Ricardo Brito Da Rocha, Antonio Sobreira de Almeida, and Zoltán Farkas. A secure Grid medical data manager interfaced to the gLite middleware. *Journal of Grid Computing*, 6(1):45–59, March 2008.

[18] Johan Montagnat, Alban Gaignard, Diane Lingrand, Javier Rojas Balderrama, Philippe Collet, and Philippe Lahire. Neurolog: A community-driven middleware design. *HealthGrid*, 2008.

[19] Reagan Moore, Arun Jagatheesan, Arcot Rajasekar, Michael Wan, and Wayne Schroeder. Data grid management systems. In Ben Kobler and P. C. Hariharan, editors, *21st IEEE Conference on Mass Storage Systems and Technologies / 12th NASA Goddard Conference on Mass Storage Systems and Technologies, Greenbelt, Maryland, USA, April 13-16, 2004*, pages 1–15. IEEE, 2004.

[20] NGI. Iniciativa de grid nacional. `http://http://www.e-ciencia.es/red.jsp`.

[21] J. Damián Segrelles Quilis. *Tesis doctoral: 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. Universidad Politécnica de Valencia, 2008.

[22] Arcot Rajasekar, Michael Wan, Reagan Moore, Wayne Schroeder, George Kremenek, Arun Jagatheesan, Charles Cowart, Bing Zhu, Sheau-Yen Chen, and Roman Olschanowsky. Storage resource broker: Managing distributed data in a grid. *Computer Society of India Journal, special issue on SAN, Vol. 33, No. 4, pp. 42-54*, 2003.

[23] Bas Revet. *DICOM Cookbook.* Philips Medical Systems Nederland, 1997.

[24] Joel H. Saltz, Scott Oster, Shannon Hastings, Stephen Langella, Tahsin M. Kurç, William Sanchez, Manav Kher, Arumani Manisundaram, Krishnakant Shanbhag, and Peter A. Covitz. cagrid: Design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics*, 22(15):1910–1916, 2006.

[25] Borja Sotomayor. *The Globus Toolkit 4 Programmer's Tutorial*, 2005. Not updated for version 4.2.

[26] Bristol UK University of the West of England. Neugrid deliverable: D6.1b distributed medical services provision (design strategy). *Technical Report*, 2009.

[27] webMDS. Página de documentación del webmds. `http://www-unix.globus.org/toolkit/docs/4.0/info/webmds/WSMDSWebMDSFacts.html`.