UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MÁSTER UNIVERSITARIO EN COMPUTACIÓN PARALELA Y DISTRIBUIDA

# Bases de datos orientadas a grafos aplicadas al estudio de informes radiológicos: utilizando entornos de computación en la nube para abordar estudios de gran dimensión

Lorena Calabuig Monerris

**Director:**
Dr. J. Damià Segrelles Quilis

**Director experimental:**
Dr. Erik Torres Serrano

València, Septiembre de 2015

# Agradecimientos

*M'agradaria agrair al GRyCAP l'oportunitat que m'ha brindat per a treballar amb ells i realitzar esta tesi de màster. Especialment agrair als meus directors tot el suport i els consells que m'han donat al llarg d'aquest període. També a la resta de companys del grup, dels que m'emporte una grata experiència.*

*Agrair també als meus pares, Tere i Juan, i a Mauro pel seu suport incondicional durant aquest últim any, animant-me a aspirar a més i traure el millor de mi. Tampoc m'oblide de tots aquells que ja no estan.*

*A tots ells, gràcies.*

# Resumen

Los centros médicos generan cientos de miles de registros al año con la información diagnóstico y tratamiento de los pacientes. Estos registros carecen de vínculos que puedan ser aprovechados para realizar estudios traslacionales, con lo que se podrían mejorar los procedimientos clínicos, diseñar pruebas de diagnóstico y tratamientos más efectivos, así como reducir el gasto en sanidad pública. Para mejorar la representación de los informes médicos, en este trabajo se desarrolla e implementa una arquitectura software para almacenar y estudiar informes médicos utilizando bases de datos orientadas a grafos. Dicha arquitectura está basada en entornos de computación en la nube para abordar estudios de gran dimensión, y se centra en el estudio de informes de mamografía, ecografía y resonancia magnética para el diagnóstico de cáncer de mama que están almacenados en formato DICOM-SR en TRENCADIS, una infraestructura Grid para compartir informes médicos.

**Palabras clave:** informes radiológicos, DICOM-SR, Cloud, Grid, NoSQL, grafos

# Abstract

Medical centers generate a thousands of registers per year with diagnostic and treatment information of patients. These records have no connection which could be exploited to make translational studies, which could improve clinical procedures, designing diagnostic tests and more effective treatments, and reduce spending on public health. This work gathers to improve the representation of medical reports, developing and implementing a software architecture to store and to study medical reports using a graph oriented data base. This architecture is based in cloud computing environment to deal with great dimension studies, and it is focused on the study of reports of mammography, ultrasound and magnetic resonance imaging for the diagnosis of breast cancer, stored in DICOM SR format in TRENCADIS, a Grid infrastructure for sharing medical reports.

**Keywords:** radiological reports, DICOM SR, Cloud, Grid, NoSQL, graphs

# Índice general

# Índice de figuras

# Índice de tablas

# CAPÍTULO 1

# Introducción

Los centros médicos generan grandes cantidades de registros al año que contienen información acerca del diagnóstico y tratamiento de los pacientes. Dado el carácter sensible de estos datos, hace que estén bajo un estricto control de los hospitales y de las entidades públicas. La reciente modernización del marco legal europeo para potenciar las iniciativas de *open-data* [1] facilita el acceso a grandes bases de datos anónimas para generar nuevos conocimientos entorno al área clínica.

La sanidad pública española proporciona una excelente oportunidad para el análisis masivo de datos. Los centros médicos de los que se compone generan un gran volumen de registros al año que carecen de vínculos que puedan ser aprovechados para realizar estudios traslacionales, con los que se podrían mejorar los procedimientos clínicos y diseñar pruebas de diagnóstico y tratamientos más efectivos.

Las infraestructuras de computación europeas como EGI[1], PRACE[2] y EUDAT[3] y la creciente popularidad de los sistemas *cloud* hacen posible abordar el estudio masivo de estos datos. Estas infraestructuras pueden almacenar registros en formato DICOM-SR [2], formato estándar para los ficheros clínicos que facilita la interoperabilidad entre diferentes centros médicos.

## 1.1. Motivación

Actualmente se dispone de pocas herramientas para explotar las bases de datos médicas debido a su volumen y a la reciente anomización de los registros. Es por esto por lo que surge la necesidad de crear aplicaciones que permitan crear vínculos entre los regis-

[1] EGI – European Grid Infrastructure: http://www.egi.eu
[2] PRACE – Partnership for Advanced Computing in Europe: http://www.prace-ri.eu
[3] EUDAT – European Collaborative Data Infrastructure: http://eudat.eu

tros de los pacientes para ayudar a los radiólogos en el diagnóstico y el tratamiento del cáncer y, en concreto, del cáncer de mama.

La aplicación del modelo de grafos puede ayudar a mejorar la representación de los informes clínicos, contribuyendo de esta forma a desarrollar nuevos procedimientos para el análisis y el estudio de los mismos.

Gpf4Med (*Graph Processing Framework for Medical Information*)[4] es un *framework* analítico que permite abordar estudios masivos de informes médicos. Aunque inicialmente fue diseñado para ser ejecutado en entornos de *cloud computing*, las crecientes necesidades de análisis, coindiciendo con el aumento de los datos disponibles, hacen necesaria una actualización de Gpf4Med que permita aprovechar los avances recientes en arquitectura software y modelos de programación para el *cloud*, con el objetivo de mejorar las prestaciones de los análisis realizados con este *framework*.

## 1.2. Objetivos

En el presente trabajo se presente añadir un nuevo almacén de datos a Gpf4Med, mediante un módulo opcional, que permita recuperar informes almacenados en TRENCADIS [3], una infraestructura Grid para el almacenamiento distribuido de informes médicos que almacena gran cantidad de registros radiológicos de determinados hospitales de la Generalitat Valenciana.

Un segundo objetivo de este trabajo consiste en adaptar Gpf4Med al modelo de programación reactiva, implementando un primer caso de uso que permita recuperar información almacenada remotamente, con una eficiencia muy superior a la conseguida hasta la fecha. Esta adaptación permitirá mejorar las prestaciones de Gpf4Med y sentar las bases para futuras mejoras.

Además de las mejoras técnicas, se persigue abordar otras tareas que permitan ampliar el soporte de Gpf4Med para el análisis de informes médicos. En particular, se ampliarán el número de modelos de grafo disponibles para analizar dos nuevas modalidades de informes de cáncer de mama.

## 1.3. Estructura del documento

El resto del documento está estructurado como sigue:

[4] Gpf4Med – Graph Processing Framework for Medical Information: http://github.com/grycap/gpf4med

**Capítulo 2:** en este capítulo se presenta el estado del arte de de las tecnologías, estándares e infraestructuras de las que deriva y utiliza esta tesis de máster.

**Capítulo 3:** este capítulo se dedica a explicar el desarrollo de un módulo de almacenamiento externo de informes médicos y su integración en el *framework* Gpf4Med.

**Capítulo 4:** tras integrar el módulo de almacenamiento, se abordará cómo utilizar el modelo de programación reactivo para mejorar la eficiencia en la descarga de informes.

**Capítulo 5:** este capítulo presenta la incorporación de nuevas modalidades de imágenes médicas para poder realizar el análisis de los grafos construidos a partir de los informes descargados del módulo de almacenamiento desarrollado.

**Capítulo 6:** finalmente, en este capítulo se presentan las conclusiones obtenidas y se mencionan las líneas de trabajos futuros que derivan de este trabajo.

# CAPÍTULO 2

# Estado del arte

En este capítulo se analizan las diferentes tecnologías, infraestructuras, estándares y *frameworks* utilizados para el desarrollo de este trabajo.

## 2.1. Estándares para el tratamiento de datos clínicos

Los centros médicos generan grandes cantidades de datos asociados a la información de los pacientes. Dicha información puede obtenerse de diferentes departamentos de un hospital o de otros centros hospitalarios. En el caso específico del cáncer de maama, se puede obtener información a partir de diversas pruebas clínicas como el examen clínico, las técnicas de imagen clínica (mamogarfía, ultrasonidos y resonancia magnética) y el análisis de tejido obtenido por biopsias.

Debido al problema de la diversidad de pruebas clínicas y de la organización que las realice, se necesita de un estándar para la representación de la información obtenida y conseguir así interoperabilidad entre diferentes sistemas software que traten dichos datos. *Digital Imaging and Communications in Medicine* (DICOM) [4] es el estándar más utilizado para la representación de imágenes médicas y de los informes médicos asociados (*DICOM Structured Reporting* (DICOM-SR)) y presenta las siguientes características:

- La mayoría de los sistemas de adquisición de imagen soportan el estándar.
- Las imágenes DICOM soportan una profundidad de color de 16 bits de gris, lo que proporciona una elevada fiabilidad en la adquisición.
- Además de almacenar la imagen adquirida, también se registran características relativas al momento de la adquisición (como la posición del paciente, dimensiones de los objetos de la imagen, grosor del corte, etc.).
- Los ficheros DICOM utilizan atributos estandarizados para facilitar un diagnóstico más preciso y reflejar todos los aspectos de la imagen obtenida.

## 2.2. Nuevas arquitecturas software y modelos de programación para *cloud computing*

En la actualidad, las comunidades de desarrolladores y los mercados están experimentando con nuevas arquitecturas de software y nuevos modelos de programación con el objetivo de aprovechar las oportunidades que brinda el *cloud computing* para el desarrollo de aplicaciones. Como resultado de este movimiento, ha emergido un nuevo paradigma para el desarrollo de aplicaciones distribuidas y, en particular, aplicaciones *cloud*-nativas. Este nuevo paradigma está impulsado en gran medida por la adopción del *cloud* como modelo dominante para la distribución de tecnologías innovadoras. Además, el desarrollo de las tecnologías *cloud* ha permitido grandes avances en otros campos de la informática, como Big Data e IoT (Internet of Things), que están cambiando por completo el panorama de la industria del software.

Las aplicaciones *cloud*-nativas se caracterizan por sus propiedades en la facilidad de instalación y configuración, en la capacidad para recuperarse de posibles fallos, en la escalabilidad y, en general, en la flexibilidad para adaptarse a entornos de ejecución y condiciones de carga de trabajo que pueden ser muy diversos. Como consecuencia, el nuevo paradigma consiste en combinar dinámicamente diferentes componentes software con el objetivo de facilitar el desarrollo y mantenimiento de aplicaciones que cumplan con las propiedades enunciadas anteriormente.

### 2.2.1. Microservicios

Los estilos de arquitectura de software que funcionaban antes de la popularización del *cloud* han sido reformados para su uso en las aplicaciones *cloud*-nativas. Este es el caso de los microservicios [5], una tendencia reciente en arquitectura de software que se basan en principios de diseño muy bien establecidos, recuperando la modularidad funcional y adoptando el principio Unix “dotadiw” (*do one thing and do it well*) como las bases para el diseño aplicaciones.

Las aplicaciones desarrolladas siguiendo la arquitectura de microservicios deben evitar los mecanismos centralizados y complejos para coordinar las diferentes partes del sistema, así como para mantener la coherencia entre las partes replicadas, en un intento por facilitar el desarrollo de sistemas de gran escala y sistemas de gran complejidad.

### 2.2.2. Programación reactiva

Otras tendencias actuales de desarrollo de software son más novedosas lo que hace que, a diferencia los microservicios, requieran un esfuerzo mayor de implementación. Tal es el caso del modelo de programación reactiva [6], y en general, del estilo de programación asíncrona. En general, estos modelos consisten en programar utilizando flujos de datos asíncronos, a los que se suscriben observadores que reaccionan a los eventos producidos en el sistema, ejecutando operaciones colaterales.

La programación reactiva consiste en generalizar esta idea al resto de las operaciones del sistema, convirtiendo en flujos muchos de los componentes básicos de cualquier aplicación, como las variables, las estructuras de datos, las cachés, etc., para permitir que los observadores reaccionen a otros eventos como pueden ser cambiar el valor asignado a una variable. Como los flujos tienen poco coste computacional y son totalmente ubicuos, este enfoque ha demostrado una sorprendente capacidad para superar las prestaciones de los sistemas tradicionales (basados en operaciones síncronas), a la vez que reducen la latencia y el sobrecoste de las operaciones [7]. Sin embargo, el estilo de programación asíncrona requiere un esfuerzo adicional, principalmente porque añade complejidad a las pruebas unitarias y de integración.

### 2.2.3. Tecnologías para el desarrollo de aplicaciones *cloud*-nativas

El enfoque más extendido para desarrollar aplicaciones *cloud*-nativas consiste en utilizar APIs REST. REST es un estilo de arquitectura basado en HTTP, un protocolo de nivel de aplicación que es intrínsecamente síncrono, y que requiere que ambas partes que participan en una comunicación sigan un patrón de solicitud-respuesta donde una de las partes (cliente) inicia la comunicación al enviar una solicitud a la otra parte (servidor). Como consecuencia, el iniciador debe conocer de antemano la dirección IP del servidor.

Es por ello que las aplicaciones construidas sobre APIs REST deben tener acceso a catálogos de servicios que les permitan registrar y luego descubrir las direcciones IP de los servidores que proporcionan funcionalidades específicas de un sistema. El servicio de DNS es la forma más simple de catálogo de servicios, pero en muchos casos no es suficiente. Los servidores HTTP como nginx[5] ofrecen capacidades de balanceo de carga sobre varios servidores, que también pueden ser utilizadas para mantener un catálogo de servicios. Algunos proyectos como Consul[6] ofrecen catálogos adaptados para su uso en entornos *cloud*.

[5] nginx: http://nginx.org/en/
[6] Consul: https://www.consul.io

En cualquier caso, es necesario tener en cuenta el esfuerzo adicional necesario para mantener un catálogo de servicios, añadiendo y eliminando servicios a medida que estén disponibles o no en el sistema. También es necesario configurar los nuevos servidores, antes de que estos puedan ser utilizados por el sistema, para que puedan interactuar correctamente con el catálogo de servicios. Esto es especialmente relevante para las aplicaciones desplegadas en un entorno *cloud*, ya que generalmente las direcciones IP de las nuevas máquinas virtuales son asignadas automáticamente durante el arranque. Un último aspecto a tener en cuenta es que algunos proveedores cloud ofrecen sus propios catálogos de servicios, pero estos podrían causar dependencia con un proveedor o tecnología (*vendor lock-in*), reduciendo las opciones y limitando la interoperabilidad de una aplicación.

Otra consecuencia del protocolo HTTP es que, a no ser que la aplicación implemente un envoltorio asíncrono, las comunicaciones serán síncronas, bloqueando el hilo de ejecución hasta que se reciba la respuesta del servidor. Por tanto, es responsabilidad del desarrollador garantizar que la aplicación no se bloquea con cada solicitud HTTP que hace, en cuyo caso la escalabilidad de la aplicación se vería seriamente reducida, como se explicó anteriormente.

Por último, las APIs REST a menudo requieren de una representación intermedia de los datos basada en texto plano, siendo JSON el formato más común. Las aplicaciones deberán convertir sus datos a JSON antes de enviarlos por la red, y el texto JSON será comprimido en el cliente HTTP y descomprimido al ser recibido en el servidor HTTP antes de que los datos puedan ser convertidos en objetos de la aplicación y finalmente ser consumidos en el servidor. Aunque este proceso está muy optimizado, comunidades como Big Data han optado por desarrollar sus propios formatos binarios (como Apache Thrift[7], Apache Avro[8] y Google's Protocol Buffer[9]) para reducir el sobre coste asociado a la transformación y transmisión de los datos. Estos nuevos formatos binarios superan a JSON tanto en reducir el tamaño de mensaje como en mejorar las prestaciones de los procesos de codificación/decodificación de los datos. Otras comunidades que desarrollan aplicaciones móviles, como IoT, se están planteando alternativas a JSON para extender la duración de la batería de los dispositivos involucrados en una comunicación [8].

Aun con las limitaciones mencionadas, las APIs REST basadas en HTTP y JSON han demostrado ser efectivas en la mayoría de los casos y muestra de ello es que son el modelo de aplicación dominante en la actualidad.

[7] Apache Thrift: https://thrift.apache.org
[8] Apache Avro: https://avro.apache.org
[9] Google's Protocol Buffer: https://developers.google.com/protocol-buffers

## 2.3. Gpf4Med: *Graph Processing Framework for Medical Information*

Gpf4Med es un *framework* modular para la integración y el estudio de información médica que está distribuido bajo la EUPL v1.1[10]. Este *framework* soporta el formato estándar para imágenes médicas DICOM-SR y ha sido desarrollado con la colaboración de un grupo de radiólogos de la Comunitat Valenciana, que han elaborado un *dataset* de gran calidad que ha sido utilizado para validar e identificar las funcionalidades del sistema.

Los datos se representan mediante un grafo en el que se plasman los términos médicos y las anotaciones de las ontologías para interconectar los objetos representados (paciente, lesión, exploración, etc.). Solo se permite la representación estática del grafo generado con la herramienta de visualización *open-source* Graphviz[11].

### 2.3.1. Arquitectura de Gpf4Med

La arquitectura de Gpf4Med (figura 2.1) ha sido diseñada tomando como base los principios descritos en 2.2. Aunque no llega a ser microservicios, Gpf4Med comparte algunos principios de diseño con esta arquitectura. Los componentes de Gpf4Med están diseñados como un conjunto de módulos funcionales que pueden ser combinados dinámicamente para analizar un conjunto de datos. La forma en que se combinan dependerá de los requerimientos del análisis, así como de los recursos computacionales disponibles.

En general, todos los módulos de Gpf4Med implementan una API REST basada en HTTP y JSON que les permite comunicarse entre ellos a través de Internet. Sin embargo, el *Graph Store* (base de datos especializada que soporta el almacenamiento, indexación y procesamiento de peticiones sobre los grafos almacenados) está diseñado para operar en redes de área local sobre las que se transmiten los datos y las operaciones que se ejecutan sobre el grafo. Como consecuencia, Gpf4Med necesita un clúster desplegado en estas condiciones para poder abordar estudios de gran escala, donde el grafo no puede ser almacenado en una única instancia, principalmente por las limitaciones de memoria RAM.

[10] EUPL – European Union Public Licence: https://joinup.ec.europa.eu/software/page/eupl/licence-eupl
[11] Graphviz: http://www.graphviz.org

#### 2.3.1.1. *Coordination layer*

Los componentes de la capa de coordinación son los responsables de recibir las peticiones de los usuarios y transformarlas en operaciones que se enviarán a la capa de procesamiento. Puesto que las operaciones pueden requerir un procesamiento específico de los datos, el componente *Infrastructure Manager* se encarga de evaluar los requisitos de dichas operaciones y proporcionar los recursos necesarios para su ejecución.

El *Resource Aggregation Service*, es un componente opcional que permite interactuar con varios sistemas de gestión y proveedores *cloud* (entre los que se encuentran, por ejemplo, Amazon EC2 y OpenStack) para negociar la puesta en marcha y eliminación de máquinas virtuales, con el objetivo de escalar Gpf4Med horizontalmente. Este componente obtiene las credenciales y otras propiedades de un fichero configuración.

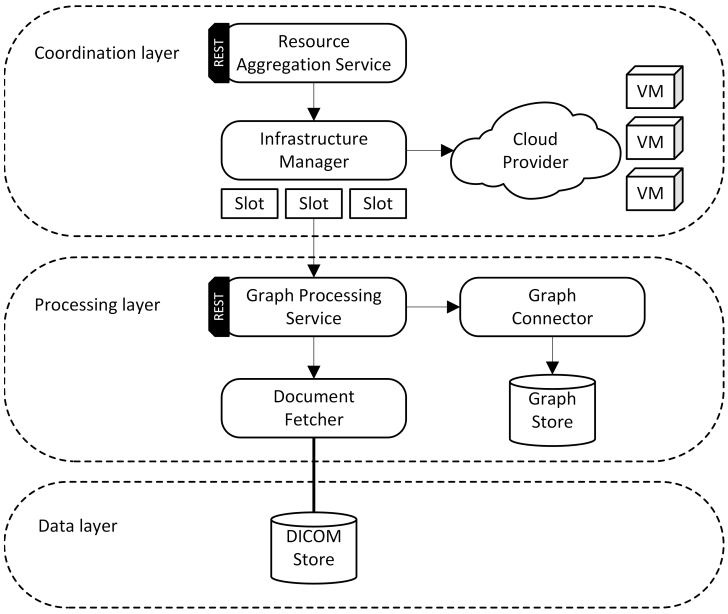


Figura 2.1: Arquitectura de Gpf4Med

#### 2.3.1.2. *Processing layer*

Una vez la *Coordination layer* ha procesado las peticiones, la capa de procesamiento es la encargada de realizar las operaciones necesarias. Estas peticiones serán atendidas por el *Graph Processing Service*, componente principal de esta capa, que las que encolará para no sobrecargar el sistema. Asimismo, este servicio accederá al *Graph Store*.

El *Document Fetcher* es el componente encargado de proporcionar una herramienta para la descarga de informes médicos. Esta herramienta permitirá que los servidores de la capa de procesamiento realicen la descarga, de forma coordinada, de los registros de los almacenes DICOM. Este componente también se encarga de analizar y validar los informes descargados.

Para cumplir las exigencias de seguridad para tratar los informes clínicos, Gpf4Med ofrece la posibilidad de cifrar aquellos documentos que sean adquiridos de un almacén de datos externo al sistema. Esto es necesario sobre todo cuando se utiliza un entorno *cloud*, ya que en este caso es el proveedor el que impone las restricciones de seguridad.

#### 2.3.1.3. *Data layer*

Esta capa es la encargada de proporcionar el acceso a almacenes de datos externos donde se almacenan los documentos clínicos; está compuesta por un conjunto de conectores que permiten la descarga de los informes mediante distintos protocolos (HTTP, FTP) y soporta diferentes formatos de registros (texto llano, XML, JSON).

### 2.3.2. Despliegue e instalación de Gpf4Med

El despliegue e instalación de Gpf4Med, así como la distribución de nuevas versiones del *framework* también están influenciadas por las tendencias actuales de DevOps (Development Operations), que es un nuevo método que requiere mayores niveles de comunicación, colaboración, integración y automatización de todas las tareas y actores implicados en el proceso de desarrollo de software.

Será necesario disponer de suficientes instancias (ya sean servidores físicos o máquinas virtuales) para soportar el *Graph Store*. Los requerimientos del grafo varían según el tipo de análisis y el número de informes médicos que serán analizados. Gpf4Med facilita el despliegue y configuración de nuevas instancias mediante un script BASH que permite realizar todas las tareas necesarias de descarga, instalación y configuración de manera automática. Una vez instalado, las funcionalidades de cada instancia se pueden extender y actualizar mediante nuevos módulos que se cargan de Internet a partir de binarios. Cada módulo va anotado con un número de versión que permite añadir nuevas funcionalidades de forma gradual.

## 2.4. TRENCADIS: infraestructura Grid

TRENCADIS es una infraestructura Grid [9] que proporciona almacenamiento y procesamiento seguro de una gran cantidad de imágenes médicas y de los objetos DICOM asociados; también almacena las ontologías [10] para dichos objetos. Este sistema permite a los radiólogos guardar, ordenar y buscar imágenes e informes estructurados de una forma sencilla, mejorando así la manipulación de los datos clínicos de los pacientes.

Esta infraestructura está desarrollada con las herramientas que proporciona el proyecto *open-source* Globus Toolkit[12] 4.2, o GT4.2, que ofrece un conjunto de servicios que permiten gestionar los recursos computacionales y de almacenamiento.

### 2.4.1. Arquitectura de TRENCADIS

La infraestructura de TRENCADIS está compuesta por una serie de servicios, compuestos por módulos desarrollados en Java y basados en la tecnología Grid, que están integrados en una organización virtual o VO. Está compuesta por dos bloques principales de servicios desarrollados en Java: los ***Core Services*** y ***Server Services***.

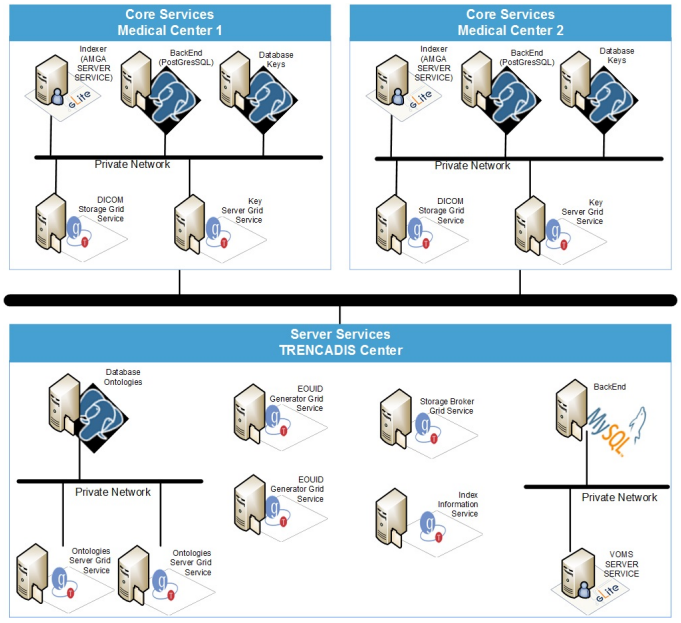


Figura 2.2: Modelo de despliegue de la infraestructura TRENCADIS

[12] Globus Toolkit: http://toolkit.globus.org

La figura 2.2 muestra el modelo de despliegue de los servicios de la infraestructura TRENCADIS y se observan los dos servicios mencionados anteriormente. Los *Core Services* se despliegan en cada centro médico involucrado en la VO y los *Server Services* corresponden a un centro específico denominado *TRENCADIS Center*.

#### 2.4.1.1. Core Services

Los *Core Services* están formados por los dos servicios que se explican a continuación; un centro médico debe desplegar, como mínimo, uno de cada tipo:

- **DICOM Storage Services:** permite la creación de almacenes de datos de imágenes DICOM y de informes DICOM-SR. Las conexiones de acceso a los datos se realizan mediante una interfaz homogénea basada en Web Services Resources Framework (WSRF). Este servicio necesita los siguientes componentes software:
  - *Base Toolkits*: proporciona los mecanismos necesarios para la implementación y despliegue de servicios Grid.
  - *DICOM Storage Grid Service*: este componente se despliega en un contenedor Grid, permitiendo el acceso a los datos mediante una interfaz basada en WSRF (utilizando los protocolos HTTP y HTTPS y los puertos 80 y 443). Este componente accede utiliza las APIs del *Indexer* y del *BackEnd* para la indexación y el almacenamiento de los datos respectivamente.
  - *Indexer*: sistema de indexación de tipo AMGA [11] que permite indexar los datos contenidos en cada informe estructurado DICOM-SR.
  - *BackEnd*: almacena, de forma cifrada, las imágenes DICOM y los informes estructurados asociados. El *backend* puede ser un sistema de ficheros, un servidor GridFTP [12] (*backend* actual) o una base de datos.
- **Key Server Services:** es necesario para implementar el modelo de seguridad ya que todos los informes almacenados en un DICOM Storage necesitan de una clave para ser cifrados.
  - *Key Server Grid Service*: este componente se despliega en un contenedor de servicios Grid y accede a los componentes *SQL Keys Database* y *BackEnd* utilizando sus APIs.
  - *BackEnd*: almacena las claves necesarias para cifrar y descifrar los datos del DICOM Storage. Este *backend* solo soporta BDs relacionales de tipo PostgreSQL.
  - *SQL Keys Database*: este componente es la base de datos en sí que utiliza el *BackEnd*. Esta BD tiene una estructura predefinida y necesita ser creada y configurada cuando se despliega el *BackEnd*.

#### 2.4.1.2. Server Services

Este tipo de servicios están desplegados en un centro externo a los hospitales y lo conforman estos cinco servicios:

- **EOUID Generator Service:** este servicio es necesario para implementar el modelo de seguridad definido por la infraestructura y se utiliza para generar un identificador único universal (*Encrypted Object Unique Identifier* (EOUID)) para cada informe que se desee cargar en el almacén. Utiliza una interfaz *Web Services Resources Framework* (WRSF) para que los clientes puedan generarlo.
- **Storage Broker Service:** este servicio se encarga de canalizar las consultas que se quieran realizar sobre los datos almacenados en la infraestructura: las distribuye en los diferentes DICOM Storage Service, recuperando y unificando los resultados obtenidos. Utiliza también una interfaz basada en WSRF.
- **Ontologies Server Service:** se encarga de guardar la estructura de la base de datos que almacena la indexación de datos en los DICOM Storage Services.
  - *Ontologies Server Grid Service*: este componente está desplegado en un contenedor Grid y es necesario para integrarse en la VO. Proporciona una interfaz basada en WSRF.
  - *BackEnd*: almacena las ontologías utilizadas para organizar y estructurar los informes DICOM.
  - *SQL Ontologies Database*: este componente es la base de datos instalada en el *BackEnd* y tiene una estructura predefinida.
- **Information Server Service (IIS):** se encarga de la monitorización y del descubrimiento de recursos, almacenando la información necesaria sobre todos los servicios de la infraestructura. Esta información es accesible a todos los usuarios que la consulten. Está basada en el *Monitoring and Discovery System* (MDS) [13] de Globus.
- **VOMS Service:** este componente es el responsable de gestionar los usuarios, los grupos de usuarios y roles de la VO a través de credenciales de tipo proxy.

### 2.4.2. TRENCADIS Middleware

TRENCADIS ofrece también un middleware para construir aplicaciones cliente y permite realizar operaciones sobre los elementos de la infraestructura, tanto de los Core como los Server Services.

Para poder utilizar este middleware se necesita crear un proxy con unas credenciales de usuario válidas. Una vez creado, el usuario podrá almacenar, borrar y descargar tanto

informes como ontologías de los diferentes almacenes de datos. Estas operaciones abstraen de la arquitectura subyacente y de cómo y dónde están almacenados los archivos.

A continuación se explican con detalle las operaciones que se pueden realizar:

- **Operaciones sobre informes:** se pueden cargar y borrar informes de forma individual y descargar informes con un EOUID concreto o aquellos que correspondan a un hospital y/o ontología específicos. También se pueden recuperar los EOUIDs de los informes y la información relativa al *backend* donde se almacenan, concretando si se quieren obtener todos o solo aquellos que pertenezcan a un hospital y/o ontología determinados.
- **Operaciones sobre ontologías:** se pueden cargar, borrar y descargar ontologías de forma individual, u obtener todas las ontologías disponibles en el sistema.

## 2.5. Modelo de actores para programación concurrente

La adopción de un modelo de programación reactivo aporta tolerancia a fallos, concurrencia transparente, escalabilidad y elasticidad a cualquier aplicación que se desarrolle. Akka[13] es una herramienta *open-source* de alto nivel basada en el modelo de actores que gestiona, a bajo nivel, todas las características reactivas mencionadas sin que el desarrollador tenga que preocuparse. El uso de esta herramienta será de gran ayuda para distribuir y reducir el tiempo de descarga de los informes de los diferentes centros médicos. Akka basa el intercambio de información entre los actores o procesos en mensajes persistentes.

### 2.5.1. Sistema de actores

El sistema de actores, o *Actor System*, es una estructura de alto nivel que contiene cierta cantidad de hilos (actores). Cada sistema de actores es independiente, por lo que pueden estar ejecutándose varios sistemas al mismo tiempo en una máquina sin que entren en conflicto entre ellos.

Un actor (observador) es un objeto que forma parte de un sistema de actores y que realiza un conjunto de acciones de forma atómica y tiene estado propio. Las acciones que realiza vienen dadas por los mensajes que recibe.

Los actores pueden o realizar una acción, o un conjunto de acciones, y terminar su ejecución, o bien crear, de forma asíncrona, nuevos actores (denominados subordinados o hijos) para dividir su trabajo en subtareas, creando así una estructura jerárquica. El

[13] Akka: http://akka.io

actor que genera nuevos actores pasa a ser el supervisor de los mismos. El concepto de supervisión se explica en el apartado 2.5.2.

El hecho de que cada actor tenga un estado propio y se ejecute de forma independiente permite que, en caso de fallo, no afecte al resto de actores en ejecución y pueda reiniciarse. Cuando un actor termina, ya sea porque ha fallado, porque ha terminado su ejecución o porque ha terminado su supervisor, libera los recursos que estaba utilizando.

### 2.5.2. Supervisión y monitorización

La supervisión describe una relación de dependencia jerárquica entre actores; cuando un actor supervisor crea subordinados, éste se hace responsable de sus fallos y se encarga de reiniciarlos o de terminarlos, o incluso de terminar su propia ejecución, según la estrategia de supervisión definida (2.5.2.1). En caso que un supervisor sufra un fallo, se encargará de comunicarlo a todos sus subordinados, en caso de tenerlos, y a su supervisor.

#### 2.5.2.1. Estrategias de supervisión

Existen dos estrategias de supervisión cuando el supervisor es notificado de los fallos de sus subordinados; dependiendo del fallo, los actores fallidos podrán seguir, reiniciar o terminar su ejecución:

- **Estrategia *uno por uno* u *One-for-One*:** se aplica solo al subordinado que ha fallado.
- **Estrategia *todos por uno* o *All-for-One*:** se aplica a todos los actores subordinados, haciendo que la caída de uno provoque la finalización del resto.

## 2.6. Bases de datos NoSQL

Como se ha visto en el apartado 2.1, los datos clínicos obtenidos de los pacientes contienen gran cantidad de parámetros que están relacionados entre ellos. Estas relaciones tienen por lo general una complejidad muy alta, por lo que se necesitan de mecanismos para modelarlas y estudiarlas.

El modelo relacional clásico resulta útil y eficiente en cuanto al almacenamiento y búsqueda de informes médicos, pero el estudio de las relaciones mencionadas no puede ser abordado mediante este modelo. Es por esto por lo que se necesita de un modelo no relacional, es decir, NoSQL, para solventar este problema.

El término NoSQL comenzó a implantarse aproximadamente sobre el año 2009 y presenta claras ventajas frente al modelo clásico, como el modelado de datos, que facilitan la escalabilidad y las altas prestaciones que requieren muchas aplicaciones actuales.

### 2.6.1. Clasificación de las bases de datos NoSQL

En [14] se hace una clasificación general de las bases de datos NoSQL de las que se dispone hasta la fecha. Según esta clasificación, los modelos de datos de estas BD pueden estar orientados a:

- **Clave-valor:** es un sistema de almacenamiento muy sencillo basado en el que se utiliza una clave para acceder a un valor. Se utiliza para sistemas en los que se indexan objetos de gran tamaño y en los que prima la escalabilidad. Ejemplos de SGBD que implementan este modelo son DynamoDB, MemcacheDB y Oracle Berkeley DB.
- **Columnas:** la información se almacena en una matriz dispersa que se accede mediante una clave formada por el número de fila y columna de la matriz. Los utilizan los sistemas en los que es posible relajar el cumplimiento de las propiedades de consistencia en contextos Big Data. Ejemplos de este tipo son Apache HBase, Cassandra e Hypertable.
- **Grafos:** este tipo de almacenamiento está pensado para dar soporte a los problemas que hacen uso de intensivo de las relaciones, por lo que se suele utilizar en casos en los que los datos están fuertemente interrelacionados, como es el caso de las redes sociales. Neo4j, OrientDB o AllegroGraph son ejemplos de implementaciones de este modelo.
- **Documentos:** en este tipo de almacenamiento se almacenan estructuras de datos jerárquicas directamente en una base de datos; el formato de depósito es el mismo en el que se encuentran los datos para facilitar la búsqueda de documentos. Ejemplos de SGDB que implementan este modelo son MongoDB, CouchDB y Oracle Berkeley DB XML.

Según esta clasificación, las bases de datos NoSQL orientadas a grafos sobresalen en cuanto a flexibilidad y en cuanto a la capacidad para modelar relaciones de una elevada complejidad [15]. Además, este tipo de bases de datos pueden ayudar a los investigadores a realizar análisis rápidos acerca de los vínculos que se establecen entre los diferentes estudios clínicos e identificar patrones que puedan ser relevantes para el diagnóstico, tratamiento y seguimiento de los pacientes.

#### 2.6.1.1. Bases de datos orientadas a grafo

Este tipo de bases de datos están diseñadas para almacenar grafos dirigidos que están compuestos por nodos y relaciones. Las relaciones pueden tener propiedades, que se definen por pares clave-valor, y añaden información adicional para poder aplicar algoritmos (como el camino más corto, grado de conectividad de los nodos o comparar las vecindades), añadir relaciones semánticas y realizar consultas.

Además de la flexibilidad en el modelado y en la inserción de datos, este modelo ofrece otras ventajas con respecto al modelo relacional: un mayor rendimiento cuando se trata con datos interrelacionados; y una mayor agilidad a la hora de realizar consultas a la base de datos al no tener un esquema definido.

### 2.6.2. Sistemas de gestión de bases de datos orientadas a grafo

En este apartado se van a estudiar los diferentes sistemas gestores de bases de datos que siguen el modelo orientado a grafos que existen en el mercado; no se van a estudiar aquellos que no tengan licencias *open-source* y los que requieran de un hardware específico.

| Nombre | Modelo de BD | Descripción |
|---|---|---|
| ArangoDB[a] | Orientado a documentos con soporte a modelos orientados a clave-valor y a grafo. | Implementa un lenguaje de consultas propio (AQL) de sintaxis parecida a SQL. |
| Neo4j[b] | Orientada a grafo. | Implementa un lenguaje de consultas propio (Cypher) de sintaxis similar a SQL. Tiene la mayor comunidad de usuarios de las consideradas según [16] |
| OrientDB[c] | Orientado a documento con extensión para grafos. | Tiene capacidad de recuperación ante fallos y utiliza una extensión del lenguaje SQL. |
| Titan[d] | Orientado a grafo. | Escala utilizando varios *backends* de almacenamiento orientados a columnas. Se integra con herramientas de consulta georreferenciadas y de texto completo. |

Tabla 2.1: Bases de datos orientadas a grafos

[a] ArangoDB: https://www.arangodb.com
[b] Neo4j: http://neo4j.com
[c] OrientDB: http://orientdb.com
[d] Titan: http://thinkaurelius.github.io/titan

La tabla 2.1 muestra los diferentes SGBD que se han considerado para el estudio de bases de datos orientadas a grafos. Tras el estudio de las características que tiene cada una de ellas se ha concluido que todas soportan transacciones ACID, acceso concurrente y persistencia de los datos, y soportan todos los tipos de datos comunes e índices secundarios.

Neo4j y Titan son las únicas que son puramente orientadas a grafos, implementando el resto el soporte para este modelo. Excepto Titan, el resto operan sin esquema, es decir, el esquema de la BD está codificado en la aplicación y la base de datos no necesita conocerlo.

Los sistemas multiplataforma son Neo4j, OrientDB y Titan al estar desarrollados en Java. Todas las soluciones proporcionan APIs para varios lenguajes de programación, siendo Java y Python los presentes en todas, y proporcionan acceso web (excepto Titan) mediante un API RESTful con soporte a JSON.

Excepto OrientDB, que se basa en el modelo máster-máster, el resto se basan en el modelo máster-esclavo para proporcionar escalabilidad. Además, OrientDB proporciona la capacidad de recuperación ante fallos que el resto no posee. En lo referente al particionado de datos mediante *sharding*, solo lo soportan ArangoDB y OrientDB y, en cambio, Titan delega esta tarea al *backend* de almacenamiento.

A pesar que OrientDB proporciona ciertas ventajas frente a Neo4j, se ha realizado un estudio en el que se comparan ambos en cuanto a prestaciones y se ha concluido que el primero es mejor en condiciones que simulan la próxima generación de supercomputadores [17] pero en las máquinas actuales tiene unas mejores prestaciones Neo4j [18].

Por las razones comentadas anteriormente y por tratarse del sistema más expandido y mejor valorado por la comunidad de usuarios, se ha decidido utilizar Neo4j para el desarrollo de este trabajo.

#### 2.6.2.1. Lenguajes de consulta para Neo4j

Los lenguajes de consulta que soporta el sistema gestor de bases de datos elegido son Cypher, Gremlin y el propio lenguaje nativo de Neo4j.

En [19] se hace una comparativa de estos tres lenguajes y se analizan el rendimiento y la eficiencia, la curva de aprendizaje, la legibilidad y el mantenimiento del código.

**Acceso nativo**

Neo4j implementa un lenguaje de consulta de forma nativa utilizando Java y en el que se necesita instanciar la BD para poder realizar las consultas que se consideren oportunas.

De las consultas resultantes se obtienen objetos de los que se podrán conocer sus atributos. A continuación se muestra un ejemplo de consulta para obtener los identificadores los usuarios de una red de amigos:

```
GraphDatabaseService database = new GraphDatabaseFactory()
   .new EmbeddedDatabase("/path/db/");
Index<Node> peopleNodes = database.index().forNodes("people");
IndexHits<Node> matching = peopleNodes.get("id-key", "user-id");
```

Permite aplicar ciertos algoritmos como el camino más corto o el de Dijkstra.

### Cypher

Cypher es el lenguaje propio de consultas de Neo4j, que está inspirado en el lenguaje SQL, y que se utiliza para describir patrones en grafos. Permite las operaciones de selección, inserción, actualización y borrado sobre una BD, de forma similar al lenguaje de consulta de una base de datos relacional. Cypher además proporciona un API REST para realizar consultas remotas sobre la BD. El siguiente código es un ejemplo para obtener, en una red social, las sugerencias de amigos que pueda conocer una persona:

```
START person=node:people(id = {id})
MATCH person-[:FRIEND_OF]->friend-[:FRIEND_OF]->friend_of_friend
WHERE not (friend_of_friend<-[:FRIEND_OF]-person)
RETURN friend_of_friend, COUNT(*)
ORDER BY COUNT(*) DESC
```

Neo4j proporciona una interfaz web (Neo4j Server) en la que se pueden realizar consultas a la BD utilizando este lenguaje y visualizar los datos obtenidos, tanto en formato de grafo como en formato de tabla.

### Gremlin

Gremlin es un lenguaje de bajo nivel para recorrer los grafos que utiliza una sintaxis compacta, lo que dificulta la lectura de las consultas. Gremlin es el lenguaje de consultas para grafos de TinkerPop[14], un proyecto *open-source* que desarrolla estándares para el almacenamiento, gestión y consulta de bases de datos a orientadas a grafo. Además, este lenguaje es compatible con todos los SGDB comentados en la sección 2.6.2. El siguiente

[14] TinkerPop: http://tinkerpop.incubator.apache.org

fragmento de código muestra cómo obtener el identificador y el nombre de los amigos que pueda conocer una persona en una red social:

```
t = new Table();
x = [];
g.idx('persons')[[id:id_param]].out('FRIEND_OF').fill(x);
g.idx('persons')[[id:id_param]].out('FRIEND_OF').out('FRIEND_OF')
   .dedup().except(x).id.as('ID').back(1).displayName.as('name')
   .table(t,['ID','name']){it}{it}.iterate();
t
```

A pesar de que Gremlin es una opción a tener en cuenta por ser un estándar de consulta sobre bases de datos orientadas a grafo, Cypher ofrece ciertas ventajas al ser un lenguaje de sintaxis parecida a SQL y, por lo tanto, la curva de aprendizaje es menor.

| Nombre | Licencia | Características |
|---|---|---|
| Neo4j Server | Dual: Core GPL v3 y módulos AGPL v3; también comercial. | - Servidor para la visualización y consulta de grafos mediante una interfaz web.<br>- Visualización de consultas en Cypher en forma de grafo y de tabla<br>- Permite guardar consultas personalizadas.<br>- API REST para consultas remotas.<br>- Permite crear plugins personalizados. |
| Gephi[a] | Dual: CDDL 1.0 y GNU GPL v3. | - Herramienta para explorar y entender grafos.<br>- Interacción y manipulación de estructuras, formas y colores para revelar propiedades ocultas de los grafos.<br>- Facilidad en la creación de hipótesis a partir de estadísticas y en el aislamiento de patrones y estructuras.<br>- Agrupación y filtrado dinámico de nodos. |
| KeyLines[b] | Comercial. | - Toolkit que permite personalizar la visualización de los grafos.<br>- Soporte para todos los navegadores y dispositivos.<br>- Construcción de layouts que permiten una visualización clara de la estructura del grafo.<br>- Agrupación y filtrado dinámico de nodos.<br>- Integración de datos en mapas geográficos. |

Tabla 2.2: Aplicaciones para la visualización de bases de datos orientadas a grafos

[a] Gephi: http://gephi.github.io
[b] KeyLines: http://keylines.com

### 2.6.3. Visualización de bases de datos orientadas a grafo

La visualización de los grafos que se obtienen de las consultas tiene especial relevancia cuando se realiza un análisis de, por ejemplo, casos similares en los informes de cáncer de mama. Para ello, se ha realizado un estudio (tabla 2.2) con diversas aplicaciones para la visualización de grafos existentes en el mercado.

KeyLines, a pesar de ofrecer funcionalidades interesantes, se descarta al estar bajo licencia comercial y no proporcionar ninguna licencia académica. Gephi puede ser interesante cuando se necesite obtener de forma rápida ciertos datos de un grafo como, por ejemplo, el grado de conectividad de los nodos, la densidad del grafo o la longitud del camino promedio [20].

Dada la flexibilidad que ofrece la interfaz web de Neo4j Server y la visualización de los datos a partir de consultas Cypher, se va a utilizar esta aplicación.

Una vez estudiado el marco tecnológico de este trabajo, en los siguientes capítulos se procederá a desarrollar cada uno de los objetivos de este trabajo: agregar un nuevo módulo para el almacenamiento de datos remotos, descarga masiva de los datos de almacenes externos y análisis de los informes médicos en sus distintas modalidades mediante la creación de grafos.

CAPÍTULO 3

# Integración de almacenes de datos externos en Gpf4Med

Este capítulo se centra en la creación de módulo para Gpf4Med que permita obtener los informes médicos y las ontologías de TRENCADIS, un sistema de almacenamiento distribuido que contiene miles de registros que serán utilizados posteriormente para crear los grafos.

## 3.1. Especificaciones

Gpf4Med es un *framework* modular que permite incorporar nuevas funcionalidades en un módulo sin tener que modificar el resto. Para poder añadir TRENCADIS como almacén de datos, se necesitará crear un punto de conexión en el módulo *core* de Gpf4Med, que se ubica en el *Data layer* (apartado 2.3.1.3).

El *framework* se basa en ficheros de configuración para especificar los parámetros de configuración de las máquinas virtuales que va a utilizar, la encriptación de los ficheros, las fuentes de las que va a obtener los informes médicos y dónde se van a almacenar dichos informes. Para introducir una nueve fuente de datos, se va a tener que añadir un nuevo fichero de configuración indicando los datos para la conexión con TRENCADIS y el certificado y contraseña del usuario para poder crear un proxy.

TRENCADIS proporciona un middleware (apartado 2.4.2) para acceder a la infraestructura y realizar operaciones concretas sobre los datos que almacena. Será necesario pues añadir la librería del middleware y sus dependencias para poder acceder al sistema de almacenamiento y poder obtener los informes.

Una vez se hayan configurado los parámetros para la conexión y se hayan importado las librerías necesarias, se tendrá que modificar la clase *ConfigurationManager*, que lee de

los ficheros de configuración, y las clases *DocumentManager* y *TemplateManager*, que son los gestores de descarga para obtener los informes y las ontologías, respectivamente, de las fuentes que se hayan indicado en los ficheros de configuración.

Tanto los informes médicos como las ontologías están en formato XML y actualmente Gpf4Med los valida utilizando las clases *DocumentLoader* y *TemplateLoader*, que recorren secuencialmente cada fichero y comprueban que contienen todos los campos necesarios para crear los objetos asociados a cada uno de ellos. Esto supone un inconveniente tanto en eficiencia como a la hora de reestructurar dichas clases si los ficheros cambian de formato. Por este motivo se va a utilizar JAXB[15], una herramienta que permite serializar los objetos Java a XML y deserializar XML en objetos Java; es interesante, en el caso que se aborda, la deserealización de los ficheros en objetos para poder tratarlos posteriormente. Esto se realiza mediante la creación de esquemas XSD[16].

## 3.2. Detalles de la implementación

El fichero de configuración que se ha creado para especificar los parámetros de configuración de TRENCADIS se denomina `gpf4med-trencadis.xml`. En él, se establecen: el directorio que contiene el fichero `trencadis.config` con los parámetros del middleware (ubicación del *usercert* y del *userkey*, del certificado de la CA y del directorio temporal), los parámetros del VOMS (host, puerto, identidad (DN), organización virtual (VO) y certificado digital) y las direcciones de los IIS disponibles en la infraestructura; y la contraseña del usuario para la creación del proxy. La clase *ConfigurationManager* se ha modificado para interpretar el nuevo fichero de configuración y sus parámetros.

La librería del middleware y sus dependencias se han integrado en Gpf4Med para poder utilizar las operaciones de las que dispone.

Para la descarga de informes médicos y de ontologías desde el nuevo *backend* se han modificado las clases *DocumentManager* y *TemplateManager*, respectivamente. La clase *DocumentManager* contenía un método que permitía descargar los informes, según los parámetros que haya introducido el usuario, de diferentes fuentes de almacenamiento (sistema de ficheros o servidor web). Se ha reutilizado este método para obtener los datos de TRENCADIS aprovechando las operaciones que proporciona su middleware. Concretamente, se ha hecho uso de la operación que recupera los identificadores (EOUIDs) de los informes y el *backend* (host, puerto, tipo de *backend* y directorio raíz) en el que se en-

[15] JAXB – Java Architecture for XML Binding: https://jaxb.java.net
[16] XSD – XML Schema Definition: http://www.w3schools.com/schema

cuentran para poder acceder directamente a los DICOM Storage Elements (DICOM-SE) de los hospitales, mejorando así el tiempo en la adquisición.

Del mismo modo, se ha modificado la clase *TemplateManager* para añadir la conexión al almacén de ontologías de TRENCADIS. Las ontologías se encuentran centralizadas en el TRENCADIS Center, por lo que en este caso no se obtiene el *backend* y se adquieren mediante conexiones WSRF.

En caso que se quisiera añadir un nuevo almacén de datos, habría que seguir los mismos pasos que se han explicado en los anteriores párrafos.

Tomando como base un informe de mamografía y su ontología, se han creado dos esquemas XML (XSD) para poder deserealizar los ficheros en objetos Java. Esto permite crear objetos de tipo *Document* (informe médico) y de tipo *Template* (ontología) para validar los ficheros descargados, guardar (si es necesario) los objetos en ficheros en formato XML y crear, de forma más sencilla y rápida, los nodos del grafo de cada informe descargado.

## 3.3. Resultados obtenidos

El objetivo de la implementación de un plugin para integrarlo con Gpf4Med era la recuperación de informes médicos y ontologías de un almacén distribuido para poder desplegar el *framework* en un entorno *cloud* sin necesidad de tener un repositorio local embebido.

Como resultado, en la capa de datos de la arquitectura de Gpf4Med se ha añadido un plugin para la conexión con TRENCADIS. En la figura 3.1 se puede observar cómo se ha introducido dicho plugin y cuáles son los principales servicios de los que se compone (ver 2.4.1 para más información).

El módulo implementado permite descargar los informes médicos y las ontologías si existe el fichero de configuración con los parámetros para acceder a TRENCADIS y si se dispone de un proxy válido (con la identidad del usuario autorizada en la infraestructura). Los ficheros descargados se almacenan en una carpeta del directorio temporal local al tratarse de datos sensibles.

El uso de esquemas XML facilita la creación de las clases de los informes DICOM y de las ontologías para validar los ficheros y construir los objetos que se utilizarán para la creación de los grafos.

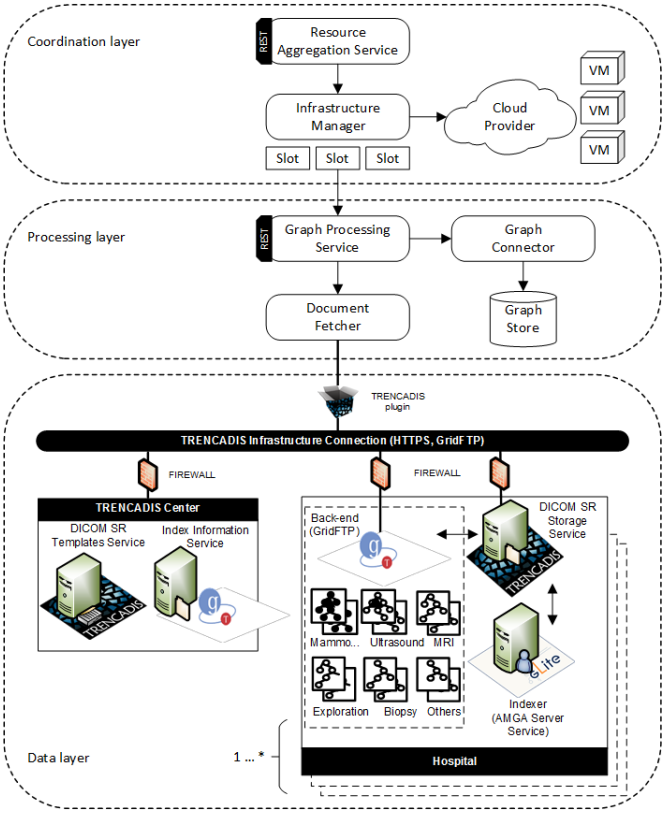


Figura 3.1: Arquitectura de Gpf4Med con el módulo de almacenamiento TRENCADIS

Desarrollado el módulo de almacenamiento, en el siguiente capítulo se va a adaptar Gpf4Med al modelo de programación reactiva para poder recuperar los informes médicos del almacén externo y mejorar así la eficiencia.

CAPÍTULO 4

# Mejora de prestaciones en la descarga de informes

Este capítulo presenta un primer caso de uso para la adaptación de Gpf4Med a un modelo de programación reactivo para distribuir la descarga de informes médicos sobre un conjunto de agentes.

## 4.1. Especificaciones

El *graph store* de Gpf4Med está diseñado para operar en redes de área local sobre las que se transmiten los datos y las operaciones que se ejecutan sobre el grafo. Esto, junto con la imposibilidad de almacenar el grafo en una única instancia debido a la limitación de la memoria RAM, hace necesario que el *framework* se despliegue sobre un clúster para poder abordar estudios de gran escala.

Este requisito de Gpf4Med hace que se planteen nuevos modelos de programación para el diseño de nuevos componentes del *framework* que permitan aprovechar al máximo las prestaciones de las máquinas.

En el gestor de descargas de informes se va a utilizar el modelo de programación reactiva para distribuir sus operaciones sobre un conjunto de agentes que aprovecharán los hilos de ejecución y servidores para abrir varias conexiones concurrentes a los *data stores*, de donde se descargarán los informes médicos. Estos agentes se comunican a través de un bus que incluye la red local donde se despliega el clúster que soporta el *graph store*.

La herramienta que se va a utilizar y que cumple con todos los requisitos mencionados anteriormente es Akka (apartado 2.5), que permite desarrollar aplicaciones reactivas para la JVM siguiendo el modelo de actores.

## 4.2. Detalles de la implementación

Gpf4Med ha sido adaptado en este trabajo para utilizar Akka, un framework que permite desarrollar aplicaciones reactivas para la JVM. El primer caso de uso de este modelo de programación consiste en descargar informes médicos almacenados remotamente.

Esta descarga ha sido implementada como un conjunto de operaciones asíncronas, distribuidas sobre un conjunto de actores, que se ejecutan en las instancias del *Graph Processing Service* disponibles en el despliegue. De esta forma, Gpf4Med aprovecha todas las instancias para abrir múltiples conexiones concurrentes a los almacenes de datos, mejorando considerablemente las prestaciones de la descarga. Para ello, se ha utilizado como *data store* el módulo de almacenamiento externo desarrollado en el capítulo 3.

En la clase *DocumentManager* se obtienen el conjunto de identificadores y de su *backend* asociado que el usuario haya especificado, mediante las operaciones disponibles del middleware de TRENCADIS. Tras recuperar los identificadores de cada centro médico e inicializar el sistema de actores de Akka, la carga de trabajo se distribuye como sigue: 1) se crea un actor que maneje los identificadores obtenidos de *n* hospitales; 2) se crea un actor por cada centro médico y se dividen los identificadores en *m* particiones; 3) se crean *m* agentes (actores) por cada hospital. El número de particiones (*m*) depende del número de identificadores que se hayan recuperado, siendo el tamaño máximo de cada partición de 50 identificadores para evitar sobrecargar la red. Este tamaño máximo se ha determinado tras realizar numerosas configuraciones modificando el número de conexiones concurrentes a un mismo servidor.

En caso de que no se hayan obtenido los informes correctamente debido, por ejemplo, a un fallo en la red o a la sobrecarga del sistema de almacenamiento del *backend* de cada centro médico, se ha diseñado una política de tolerancia a fallos que permita realizar hasta un máximo de 10 peticiones, tras el minuto posterior al momento del fallo, al almacén correspondiente. Se ha utilizado la estrategia *One-for-One* (2.5.2.1) para gestionar los fallos ocurridos.

## 4.3. Resultados obtenidos

Tras realizar la adaptación del gestor de descargas a un modelo reactivo, se ha realizado un estudio con los conjuntos de datos disponibles en los almacenes DICOM de dos hospitales de la Comunitat Valenciana. En la tabla 4.1 se puede observar el *dataset* utilizado, que se compone por 5.253 informes de las diferentes ontologías disponibles.

La tabla 4.2 muestra los tiempos que se han obtenido al descargar el conjunto de datos disponibles en los *backends* virtuales en TRENCADIS que simulan los hospitales universitarios Doctor Peset y La Fe. Se puede observar en ella que se obtiene un mayor rendimiento en la descarga utilizando un modelo de actores (que proporciona concurrencia en las conexiones) frente a un modelo secuencial. Como consecuencia, se reduce hasta en un 89.04% el tiempo de descarga por lo que, el siguiente paso del análisis, que consiste en importar los informes al grafo, se acelera considerablemente y, en general, el tiempo total que se necesita para construir el grafo también se ve reducido.

| Hospital | Mamografía (ontología 5) | Ecografía (ontología 6) | IRM (ontología 7) | Total |
|---|---|---|---|---|
| Hospital Doctor Peset (1) | 787 | 964 | 1.111 | 2.862 |
| Hospital La Fe (2) | 615 | 809 | 967 | 2.391 |

Tabla 4.1: Conjunto de datos de la prueba de carga

| Hospitales | Ontologías | Informes | Concurrente (ms) | Secuencial (ms) |
|---|---|---|---|---|
| 1 | 5, 6 y 7 | 2.862 | 104.973 | 919.548 |
| | 5 | 787 | 31.961 | 253.909 |
| 2 | 5, 6 y 7 | 2.391 | 94.742 | 813.283 |
| | 5 | 615 | 29.673 | 203.074 |
| 1 y 2 | 5, 6 y 7 | 5.253 | 187.783 | 1.712.747 |
| | 5 | 1.402 | 53.597 | 477.284 |

Tabla 4.2: Tiempos de descarga de los informes médicos

En el siguiente capítulo se va a construir el grafo a partir de los informes médicos obtenidos mediante el modelo de actores, desarrollado en el presente capítulo, y se van a realizar consultas sobre el mismo que puedan responder a las cuestiones de los radiólogos.

CAPÍTULO 5

# Análisis de informes médicos

En este capítulo se va a añadir soporte a los diferentes tipos de informes utilizados en el diagnóstico de cáncer de mama y que actualmente no se encuentran en Gpf4Med. Asimismo, se van a intentar resolver las preguntas planteadas por los radiólogos mediante las consultas a los grafos que se van a generar tras obtener los informes médicos.

## 5.1. Escenario de referencia

En el escenario de referencia propuesto por los radiólogos se pretende realizar estudios traslacionales de cáncer de mama. Los estudios traslacionales son aquellos que facilitan la transición de la investigación médica en aplicaciones clínicas que redunden en beneficio de la salud. En este escenario, se persigue descubrir nuevos conocimientos mediante el análisis masivo de informes médicos para mejorar las prácticas actuales y encontrar tratamientos más efectivos para esta enfermedad.

Los procesos clínicos involucrados en el escenario se describen en la Oncoguía de mama elaborada por la Conselleria de Sanitat de la Comunitat Valenciana [21], en donde se definen las directrices a seguir en los hospitales de esta región. Aunque el cumplimiento de estas directrices es obligatorio, no se impone ningún formato de datos por lo que cada departamento de radiología decide el diseño de los informes según sus necesidades, lo que produce que sean heterogéneos.

Según la Oncoguía, hay tres procesos clínicos que, combinándolos con técnicas de imagen médica (mamografía, ecografía y resonancia magnética) y otros tipos de estudios (exploración clínica y biopsia), conforman el procedimiento a seguir ante un caso de cáncer de mama: diagnóstico, seguimiento y respuesta al tratamiento.

El proceso de diagnóstico es en el que se centra este escenario. Este proceso se realiza inicialmente a través del análisis de imágenes médicas que, en un número considerable

de casos, conduce a resultados poco concluyentes (clasificados como BI-RADS 4) que requieren de la confirmación mediante análisis patológicos del tejido obtenido mediante biopsia. Las conclusiones obtenidas tras realizar el diagnóstico determinan el tratamiento del paciente, que por lo general consiste en varios ciclos separados durante semanas o meses en los que se combinan cirugía, radioterapia y terapia sistémica.

Los radiólogos formularon una serie de preguntas abiertas que, en su mayoría, pueden ser abordadas mediante el uso de las herramientas tradicionales. Sin embargo, se han encontrado cuatro preguntas que requieren del uso de la teoría de grafos para modelar las complejas relaciones entre los diferentes campos de información obtenidos mediante las técnicas de imagen. Cualquier avance en la resolución de estas cuatro preguntas supondría un impacto positivo casi inmediato en los procedimientos clínicos de esta enfermedad. A continuación se detallan cada una de las preguntas requieren de la teoría de grafos para su resolución:

**P1.** ¿Es posible recomendar la asignación de una subcategoría de BI-RADS 4 a un paciente mediante la comparación de casos similares?

Todos los hallazgos clasificados como BI-RADS de categoría 4 serán, en un principio, sujetos a biopsia. Sin embargo, un hallazgo puede ser clasificado correctamente con una baja sospecha de ser cáncer de tipo 4A, por lo que el paciente puede ser evaluado mediante imagen médica sin tener que recurrir a biopsias. Estudios previos han demostrado que el valor predictivo positivo (*positive predictive value* (PPV)) de BI-RADS 4 mejora con la combinación de la clasificación obtenida mediante ultrasonidos y la examinación clínica (PPV $= 54\,\%$ en lesiones palpables y $16.8\,\%$ en lesiones no palpables [22]). La precisión del PPV puede incrementarse si se descubren nuevos factores.

**P2.** ¿Cuál es el tratamiento más corto (en días) que, dadas ciertas condiciones de una lesión, reducen la lesión a un tamaño específico?

Se han hallado algunas variables que tienen cierto impacto sobre el resultado del pronóstico. Sin embargo, no se han obtenido estudios retrospectivos de informes médicos creados en diferentes hospitales para encontrado estudios que se puedan correlacionar las condiciones del tratamiento (como la dosis de radiación) con la reducción de la masa de un tumor. Mejorar la comprensión de los factores que conducen a la reducción de una lesión en determinadas circunstancias podría ayudar a diseñar tratamientos personalizados para reducir el tamaño de un tumor hasta que sea adecuado para una intervención quirúrgica y, evitar así, la mutilación de los senos [23] [24].

**P3.** Dada una lesión con un diagnóstico desconocido y un conjunto de datos de casos anteriores con un resultado conocido, encontrar los casos que compartan más similitudes con dicha lesión.

Tener casos de referencia que compartan unos antecedentes similares con una lesión puede ayudar a los radiólogos a conseguir un mayor entendimiento de los hallazgos en las exploraciones radiológicas. A pesar de que las herramientas tradicionales se han utilizado ya para identificar casos de referencia, la teoría de grafos abre una nueva perspectiva en la que se pueden relacionar casos aparentemente distintos que se hayan descartado al no compartir relaciones comunes con el alcance de la lesión. Por ejemplo, los estudios realizados mediante diferentes técnicas de imagen poseen campos de información diferentes que dificultan cualquier intento de compararlos utilizando las técnicas tradicionales. En cambio, la teoría de grafos facilita este tipo de estudios basados en los nodos vecinos de una lesión. Hay numerosas evidencias que apoyan que mejorar la precisión de los métodos no invasivos para distinguir tumores malignos de los benignos, puede evitar innecesarias biopsias [25] [26], lo que ayudaría a reducir los costes sanitarios y la mortalidad producidos por el cáncer de mama.

**P4.** Encontrar información relevante en el *dataset* para realizar un estudio retrospectivo de los pacientes con un determinado subtipo de cáncer de mama.

El cáncer de mama es una enfermedad multifacética compuesta por distintos subtipos biológicos con diferentes pronósticos e implicaciones terapéuticas [27]. La clasificación inmunohistoquímica (*immunohistochemistry classification* (IHC)) proporciona tanto información terapéutica como de pronóstico sobre esta enfermedad. IHC clasifica el cáncer de mama por grupos basados la expresión positiva o negativa del receptor del estrógeno (*strogen receptor* (ER)), del receptor de la progesterona (*progesterone receptor* (PR)) y del receptor 2 del factor de crecimiento epidérmico humano (*human epidermal growth factor receptor* (Her2)). Para llevar a cabo estudios retrospectivos, es necesario encontrar casos que proporcionen información detallada acerca de un subtipo específico; estos estudios han demostrado ser especialmente útiles en el tratamiento de subtipos con baja incidencia.

## 5.2. Especificaciones

El escenario de referencia mencionado en la sección anterior requiere del desarrollo de nuevos modelos de grafo para cubrir los informes obtenidos por las diferentes técnicas de imagen médica. Gpf4Med soporta solo los informes relacionados con mamografías,

por lo que habría que incluir los correspondientes a ecografía y resonancia magnética. Asimismo, y puesto que se han introducido cambios en la estructura de los documentos, se tiene que adaptar el modelo existente.

Un aspecto fundamental a tener en cuenta a la hora de diseñar los modelos es estructurarlos conforme a las consultas que se quieran realizar. De esta forma, se podrán responder a las cuatro preguntas planteadas por los radiólogos de una manera más sencilla, permitiendo, además, adaptar el modelo a un contexto Big Data en el que se puedan relacionar informes de distintas modalidades de grandes repositorios remotos.

El personal clínico necesita visualizar los grafos generados a partir de las consultas que haya realizado. Por este motivo se va a utilizar Neo4j Server, la herramienta de visualización que proporciona el propio Neo4j. Esta herramienta permite realizar consultas sobre los grafos y visualizar tanto el grafo generado como los datos relativos a cada nodo de dicho grafo (en formato de tabla). Puntualmente, también se pueden utilizar programas de visualización específicos para grafos, como Gephi, si se desean obtener estadísticas como el camino más corto de un nodo a otro, la modularidad de los nodos o la densidad del grafo, entre otros.

## 5.3. Modelado de los grafos

El modelado de los grafos se ha realizado de forma que todos los tipos de informes médicos tengan la misma estructura de forma que, si se desean encontrar características comunes entre, por ejemplo, un informe de mamografía y otro de resonancia magnética, sea más sencillo realizar la consulta y luego ver las relaciones generadas en el grafo. La figura 5.1 refleja la estructura que se ha utilizado para definir los modelos de todas las modalidades de informes.

Puesto que Gpf4Med solo contemplaba el modelo para tipo mamografía, se han añadido los correspondientes a ecografía y resonancia magnética. Para ello, se han creado dos clases nuevas, denominadas *EcographyCreator* y *MagneticResonanceCreator*, en las que se crea el modelo conforme a la estructura de los informes. También se ha modificado el grafo generado para la mamografía para adaptarlo a este modelo genérico. Como se ha explicado en 2.6.2, se ha utilizado el API que proporciona Neo4j para el desarrollo de la base de datos que contendrá el grafo.

Los nodos que corresponden a *Radiological Study*, *Patient*, *Finding*, *Lesion Property*, *Tumour Size*, *Tumour Location* y *BI-RADS* van a ser únicos en la base de datos para en-

contrar nodos comunes entre diferentes tipos de lesiones, modalidades de informes y/o pacientes.

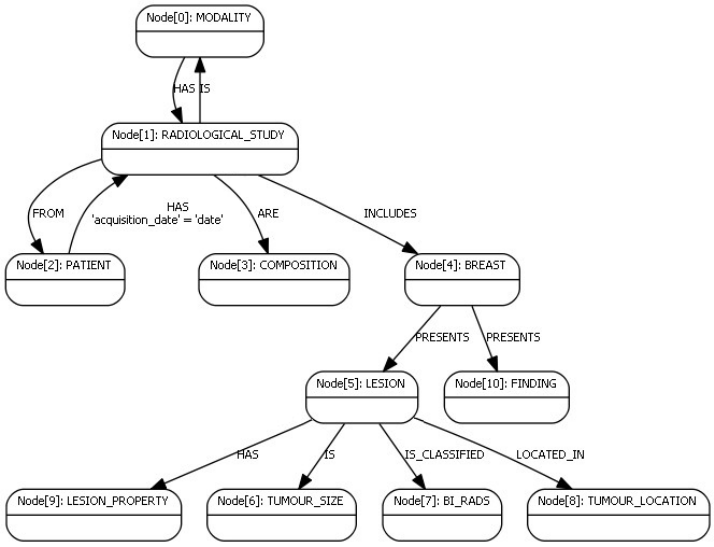


Figura 5.1: Modelo genérico de los grafos

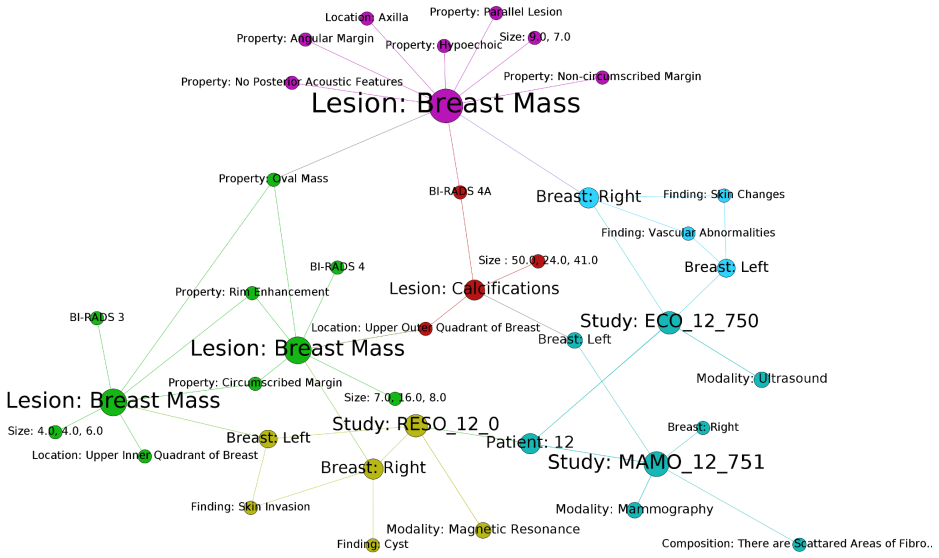


Figura 5.2: Visualización del grafo con los informes de un paciente utilizando Gephi

Las figuras 5.2 y 5.3 muestran la visualización del grafo generado a partir de los informes de mamografía, ecografía y resonancia magnética de un determinado paciente utilizando el programa Gephi y la interfaz que proporciona Neo4j Server.

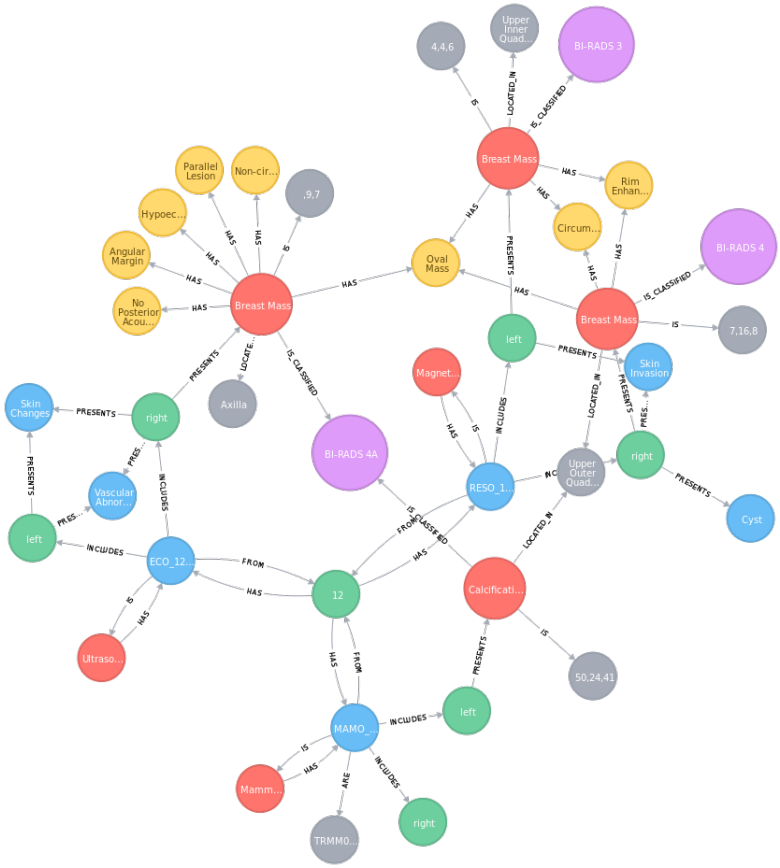


Figura 5.3: Visualización del grafo con los informes de un paciente utilizando Neo4j Server

Como se puede observar en las figuras, la creación de nodos únicos por cada campo de información de los informes, evitando así repeticiones, hace que se puedan observar de forma inmediata las características que comparten las lesiones observadas en las imágenes médicas de un paciente, así como el tipo de BI-RADS y de la localización de las lesiones.

## 5.4. Validación del sistema

En los capítulos 3 y 4 se han desarrollado unas herramientas que permiten descargar de forma eficiente un *dataset* con los informes radiológicos de cáncer de mama que van a servir para generar los grafos. Estos grafos pueden ser de gran ayuda para los radiólogos para esclarecer las cuatro preguntas descritas en el escenario de referencia (apartado 5.1) que no se pueden resolver mediante el uso de herramientas tradicionales.

A pesar de haber diseñado un modelo de grafo que cumpla los requisitos para poder formular las cuatro preguntas propuestas por los radiólogos, sólo se va a poder abordar la pregunta P3, en la que se quería conocer las vecindades de los nodos de las lesiones con un BI-RADS asociado. La P3 servirá como referencia para realizar la P1 ya que también necesita conocer las vecindades para asignar un tipo de BI-RADS a una lesión pero, en este caso, teniendo en cuenta si dicha lesión ha derivado en biopsia o no. Actualmente no se disponen de informes que reflejen si en un determinado diagnóstico ha sido necesaria la realización de biopsia, por lo que no se puede resolver esta pregunta con la información de la que se dispone.

Tampoco se disponen de informes que reflejen la evolución de un paciente con el tratamiento que le han asignado para reducir el tamaño de una lesión. Esto hace que la pregunta P2 sea inabordable de momento, aunque en el modelo se tiene en cuenta la fecha de adquisición del informe, lo que ayudará a construir un nuevo grafo que permita al *framework* encontrar el camino más corto entre el hallazgo inicial y la respuesta al tratamiento, donde el tamaño de la lesión será menor que el inicial.

La P4 no se puede responder hasta que no se disponga información acerca de los factores ER, PR y Her2, que forman parte de la clasificación IHC. Se construirá un grafo del mismo tipo del construido para responder las preguntas P1 y P3, en el que se reflejarán los factores de la IHC y se puedan obtener las vecindades de los nodos lesión y seleccionar aquellos con los valores más altos. Puesto que un grafo con un grado de conectividad mayor puede contribuir con más evidencias a un caso de estudio, en este tipo de grafo se tendrá en cuenta el grado de conectividad medio entre los nodos.

### 5.4.1. Consultas sobre los grafos

Tras diseñar la estructura de los grafos y adaptar la disposición de la información de los informes de imagen médica al modelo diseñado, se han realizado una serie de consultas de una base de datos, que contiene solo los informes con la modalidad de mamografía para simplicar los resultados que se van a mostrar, utilizando la interfaz web que proporciona

Neo4j en la que se pueden realizar consultas en Cypher y visualizar los grafos derivados de ellas, junto con los datos asociados en forma de tabla.

Las tres consultas que se muestran a continuación sirven para responder a la pregunta P3 planteada por los radiólogos.

**Consulta 1: Obtener las propiedades de aquellas lesiones que tengan un BI-RADS de tipo 4A**

En esta consulta se van a obtener las propiedades asociadas las lesiones que tengan asociado un BI-RADS de tipo 4C, que es el que indica una anomalía sospechosa y que tiene una probabilidad alta de ser maligna.

```
MATCH (bi_rads:BI_RADS)<--(lesion:LESION)-->(location:TUMOUR_LOCATION),
    lesion-->(lesionProperty:LESION_PROPERTY)
WHERE bi_rads.class =~ "BI-RADS 4C"
RETURN lesion, bi_rads, lesionProperty;
```

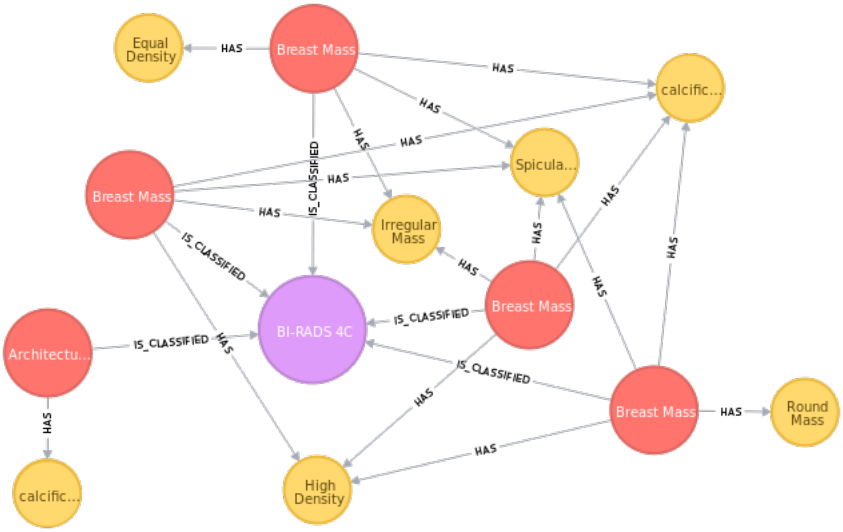


Figura 5.4: Grafo generado de la consulta 1

**Consulta 2: Obtener la localización y las propiedades de aquellas lesiones que sean de un determinado tipo y tengan un BI-RADS asociado de tipo 4**

El resultado de esta consulta va a ser un grafo con las propiedades y la localización de las lesiones que tengan asociado cualquier subtipo de BI-RADS 4 y que sean de tipo asimetría (“Asymmetries”).

```
MATCH
    (bi_rads:BI_RADS)<--(lesion:LESION)-->(lesionProperty:LESION_PROPERTY),
lesion-->(location:TUMOUR_LOCATION)
WHERE bi_rads.class =~ "BI-RADS 4.*" AND lesion.description =~
    "Asymmetries"
RETURN lesion, bi_rads, lesionProperty, location;
```

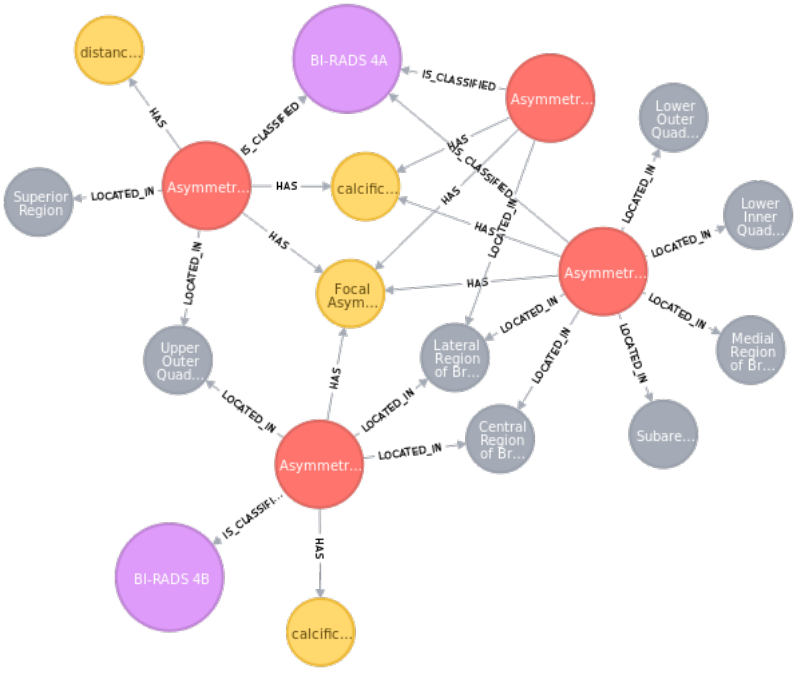


Figura 5.5: Grafo generado de la consulta 2

**Consulta 3: Obtener las lesiones, y su BI-RADS asociado, que cumplan con dos determinadas propiedades**

Dadas dos propiedades concretas (margen espiculado y masa irregular) se van a obtener las lesiones que cumplan con, al menos, una de ellas y el tipo de BI-RADS al que está ligado. Además, se van a obtener los informes médicos que cumplen dichas condiciones por si el radiólogo considera necesario ver con más detalle otros datos del informe.

```
WITH ['Spiculated Margin','Irregular Mass'] AS props
MATCH (lesionProperty:LESION_PROPERTY)<--(lesion:LESION)
    -->(bi_rads:BI_RADS)
WHERE lesionProperty.description IN props
WITH lesion AS Lesion, collect(CASE WHEN lesionProperty.description IS
    NULL THEN lesionProperty.value ELSE lesionProperty.description END)
    AS Properties, bi_rads AS BIRADS
MATCH Lesion<--(breast:BREAST)<--(study:RADIOLOGICAL_STUDY)
RETURN Lesion.description AS LesionDescription, BIRADS.class AS
    TypeOfBIRADS, Properties, count(Properties) AS Occurrences,
    collect(study.id_report) AS Studies
ORDER BY length(Properties) DESC, Occurrences DESC, BIRADS.class,
    LesionDescription;
```

| LesionDescription | TypeOfBI-RADS | Properties | Occurrences | Studies |
|---|---|---|---|---|
| Breast Mass | BI-RADS 4C | [Spiculated Margin, Irregular Mass] | 3 | [MAMO_10_751, MAMO_30_752, MAMO_36_752] |
| Breast Mass | BI-RADS 4B | [Spiculated Margin, Irregular Mass] | 2 | [MAMO_5_500, MAMO_20_752] |
| Breast Mass | BI-RADS 4B | [Spiculated Margin] | 3 | [MAMO_38_752, MAMO_41_752, MAMO_26_752] |
| Breast Mass | BI-RADS 4A | [Spiculated Margin] | 1 | [MAMO_6_750] |
| Breast Mass | BI-RADS 4A | [Irregular Mass] | 1 | [MAMO_11_751] |
| Breast Mass | BI-RADS 4C | [Spiculated Margin] | 1 | [MAMO_32_752] |
| Breast Mass | BI-RADS 4C | [Irregular Mass] | 1 | [MAMO_38_752] |

Tabla 5.1: Tabla generada como resultado de la consulta 3

### 5.4.2. Resultados obtenidos

Como resultado de la combinación de varias consultas sobre un grafo generado con los informes de mamografías, se han obtenido las tablas 5.2, 5.3, 5.4 y 5.5 que reflejan los diferentes tipos de BI-RADS que contienen (BI-RADS 3, que indica hallazgos probablemente benignos, y BI-RADS 4, que indica anomalías sospechosas que sean probablemente malignas), así como la el porcentaje de las ocurrencias de cada tipo de lesión y de sus características asociadas. De esta forma se puede observar qué características son las más comunes en un determinado tipo de lesión según su grado de anomalía.

| BI-RADS | Tipo de lesión | % | Propiedad | % |
|---|---|---|---|---|
| BI-RADS 3 | Nódulo | 44.44 % | Calcificación: no | 100 % |
| | | | Margen circunscrito | 100 % |
| | | | Masa ovalada | 75 % |
| | | | Masa redondeada | 25 % |
| | | | Igual densidad | 100 % |
| | Asimetrías | 33.33 % | Calcificación: no | 33.33 % |
| | | | Calcificación: sí | 33.33 % |
| | | | Asimetría focal | 33.33 % |
| | | | Asimetría | 33.33 % |
| | Calcificaciones | 22.22 % | Calcificación redondeada | 100 % |
| | | | Lechada de cal | 50 % |

Tabla 5.2: Lesiones y propiedades clasificadas como BI-RADS 3

| BI-RADS | Tipo de lesión | % | Propiedad | % |
|---|---|---|---|---|
| BI-RADS 4A | Nódulo | 33.33 % | Calcificación: no | 100 % |
| | | | Margen indistinto | 50 % |
| | | | Margen espiculado | 25 % |
| | | | Margen oscurecido | 25 % |
| | | | Masa ovalada | 50 % |
| | | | Masa irregular | 25 % |
| | | | Masa redondeada | 25 % |
| | | | Alta densidad | 50 % |
| | | | Igual densidad | 50 % |
| | Asimetrías | 25 % | Calcificación: no | 100 % |
| | | | Asimetría focal | 100 % |
| | | | Distancia al pezón: 12 mm | 33.33 % |
| | Calcificaciones | 25 % | – | – |
| | Alteración estructural | 16.67 % | Calcificación: no | 100 % |

Tabla 5.3: Lesiones y propiedades clasificadas como BI-RADS 4A

| BI-RADS | Tipo de lesión | % | Propiedad | % |
|---|---|---|---|---|
| BI-RADS 4B | Nódulo | 42.86 % | Calcificación: no | 66.67 % |
| | | | Calcificación: sí | 33.33 % |
| | | | Margen espiculado | 55.56 % |
| | | | Margen indistinto | 22.22 % |
| | | | Margen circunscrito | 22.22 % |
| | | | Masa redondeada | 44.44 % |
| | | | Masa ovalada | 33.33 % |
| | | | Masa irregular | 22.22 % |
| | | | Alta densidad | 55.56 % |
| | | | Igual densidad | 33.33 % |
| | | | Baja densidad | 11.11 % |
| | | | Distancia al pezón: 8 mm | 11.11 % |
| | Calcificaciones | 38.10 % | – | – |
| | Alteración estructural | 14.29 % | Calcificación: no | 66.67 % |
| | | | Calcificación: sí | 33.33 % |
| | Asimetrías | 4.76 % | Calcificación: sí | 100 % |
| | | | Asimetría focal | 100 % |

Tabla 5.4: Lesiones y propiedades clasificadas como BI-RADS 4B

| BI-RADS | Tipo de lesión | % | Propiedad | % |
|---|---|---|---|---|
| BI-RADS 4C | Nódulo | 83.33 % | Calcificación: no | 100 % |
| | | | Margen espiculado | 80 % |
| | | | Margen indistinto | 20 % |
| | | | Masa ovalada | 75 % |
| | | | Masa irregular | 80 % |
| | | | Masa redondeada | 20 % |
| | | | Alta densidad | 60 % |
| | | | Igual densidad | 20 % |
| | | | Baja densidad | 20 % |
| | Alteración estructural | 16.67 % | Calcificación: sí | 100 % |

Tabla 5.5: Lesiones y propiedades clasificadas como BI-RADS 4C

En los datos obtenidos se puede observar que los nódulos son el tipo de lesión más frecuente en todos los tipos de BI-RADS y, en la mayoría de casos, no va a contener calcificaciones. Según las tablas, se podría aventurar a decir que la probabilidad de que las asimetrías sean el tipo de lesión presente va a disminuir conforme aumente la sospecha de que una lesión sea maligna; también se podría afirmar que hay una mayor probabilidad de que los nódulos tengan una alta densidad cuando sean de cualquier subtipo de BI-RADS 4.

CAPÍTULO 6

# Conclusiones y trabajos futuros

En esta tesis de máster se ha ampliado Gpf4Med, una herramienta que permite abordar el análisis a gran escala de estudios radiológicos de cáncer de mama. Para ello se han aplicado los avances más recientes en bases de datos orientadas a grafos, que nos han permitido profundizar en el análisis comparativo de lesiones para determinar las características que son relevantes para el diagnóstico del cáncer de mama.

Asimismo, se han empleado técnicas de Big Data y Cloud Computing para mejorar la utilización de los recursos computacionales en Gpf4Med. En particular, se ha adaptado esta herramienta al modelo de actores, lo que ha permitido reducir el tiempo de acceso a los datos, disminuyendo considerablemente el tiempo total empleado en el análisis de los mismos.

Otro resultado de este trabajo ha permitido mejorar la interoperabilidad de TRENCADIS, un almacén distribuido de informes médicos, haciéndolo compatible con la infraestructura de seguridad de EGI. De esta forma, hemos conseguido ampliar el alcance de TRENCADIS, facilitando que se puedan integrar datos de cualquier hospital europeo que utilice este modelo de seguridad.

De este trabajo se derivan dos líneas principales de trabajos futuros. Por una parte, es necesario continuar con el desarrollo de Gpf4Med, principalmente en la interfaz de usuario que requiere mejoras para facilitar el acceso de los radiólogos a los datos, tanto para la visualización de los mismos como para realizar consultas a la base de datos.

La segunda línea de trabajos futuros consiste en seguir aumentando la capacidad analítica de Gpf4Med mediante nuevos tipos de grafos que incluyan información disponible en los informes médicos que ahora mismo no está siendo aprovechada, tales como detalles de la evolución de una lesión sometida a tratamiento y la información de otras técnicas de diagnóstico no basadas en imagen médica, como las biopsias pre y post-operatorias. Todo ello podría contribuir a mejorar nuestros conocimientos actuales acerca de los factores que influyen en el desarrollo del cáncer, y con ello mejorar la gestión de esta enfermedad.

APÉNDICE A

# Proyectos y publicaciones

El trabajo realizado en la presente tesis de máster está enmarcado en el proyecto "Evolución de un modelo de datos para el análisis de informes estructurados a un contexto BigData" de la Generalitat Valenciana, con referencia GV/2014/036, y ha dado lugar a la siguiente publicación:

- Lorena Calabuig, Erik Torres, Damià Segrelles and Ignacio Blanquer: *Gpf4Med: A large-scale graph processing system applied to the study of breast cancer*, Computational Science & Engineering, IEEE (2015).

# APÉNDICE B

# Bibliografía


[1] V. Reding, "The European data protection framework for the twenty-first century," *International Data Privacy Law*, vol. 2, no. 3, pp. 119–129, 2012.

[2] R. Hussein, U. Engelmann, A. Schroeter, and H.-P. Meinzer, "DICOM structured reporting: Part 1. Overview and characteristics," *Radiographics: a review publication of the Radiological Society of North America, Inc*, vol. 24, no. 3, pp. 891–896, 2004.

[3] J. D. Segrelles, M. Caballer, E. Torres, G. Moltó, and I. Blanquer, "Platform to Ease the Deployment and Improve the Availability of TRENCADIS Infrastructure," in *7th Iberian Grid Infrastructure Conference (IberGrid)*, pp. 133–145, 2013.

[4] O. S. Pianykh, *Digital imaging and communications in medicine (DICOM): a practical introduction and survival guide*. Springer Science & Business Media, 2009.

[5] "Microservices." http://martinfowler.com/articles/microservices.html. (Consultado el 30 de agosto de 2015).

[6] "The Reactive Manifesto." http://www.reactivemanifesto.org. (Consultado el 30 de agosto de 2015).

[7] "Web Framework Benchmarks." https://www.techempower.com/benchmarks. (Consultado el 31 de agosto de 2015).

[8] "REST Without JSON: The Future of IoT Protocols." https://dzone.com/articles/json-http-and-the-future-of-iot-protocols. (Consultado el 31 de agosto de 2015).

[9] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International journal of high performance computing applications*, vol. 15, no. 3, pp. 200–222, 2001.

[10] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The semantic web: The roles of XML and RDF," *Internet Computing, IEEE*, vol. 4, no. 5, pp. 63–73, 2000.

[11] B. Koblitz, N. Santos, and V. Pose, "The AMGA metadata service," *Journal of Grid Computing*, vol. 6, no. 1, pp. 61–76, 2008.

[12] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus striped GridFTP framework and server," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, p. 54, IEEE Computer Society, 2005.

[13] J. M. Schopf, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. D'Arcy, and A. Chervenak, "Monitoring the grid with the globus toolkit mds4," in *Journal of Physics: Conference Series*, vol. 46, p. 521, IOP Publishing, 2006.

[14] A. Moniruzzaman and S. A. Hossain, "NoSQL database: New era of databases for Big Data analytics-classification, characteristics and comparison," *International Journal of Database Theory and Application*, vol. 6, no. 4, pp. 1–13, 2013.

[15] D. McCreary and A. Kelly, "Making sense of NoSQL," *Manning Publications*, 2013.

[16] "DB-Engines: Knowledge Base of Relational and NoSQL Database Management Systems." http://db-engines.com. (Consultado el 22 de julio de 2015).

[17] M. Dayarathna and T. Suzumura, "Graph database benchmarking on cloud environments with XGDBench," *Automated Software Engineering*, vol. 21, no. 4, pp. 509–533, 2014.

[18] S. Jouili and V. Vansteenberghe, "An empirical comparison of graph databases," in *2013 International Conference on Social Computing (SocialCom)*, pp. 708–715, IEEE, 2013.

[19] F. Holzschuher and R. Peinl, "Performance of graph query languages: comparison of Cypher, Gremlin and native access in Neo4j," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pp. 195–204, ACM, 2013.

[20] K. Cherven, *Network graph analysis and visualization with Gephi*. Packt Publishing Ltd, 2013.

[21] Generalitat Valenciana. Conselleria de Sanitat, "Oncoguía de Cáncer de Mama Comunidad Valenciana." http://publicaciones.san.gva.es/publicaciones/documentos/V.2478-2006.pdf.

[22] E.-K. Kim, K. H. Ko, K. K. Oh, J. Y. Kwak, J. K. You, M. J. Kim, and B.-W. Park, "Clinical application of the BI-RADS final assessment to breast sonography in conjunction with mammography," *American journal of roentgenology*, vol. 190, no. 5, pp. 1209–1215, 2008.

[23] G. Bonadonna, U. Veronesi, C. Brambilla, L. Ferrari, A. Luini, M. Greco, C. Bartoli, G. C. de Yoldi, R. Zucali, F. Rilke, *et al.*, "Primary chemotherapy to avoid mastectomy in tumors with diameters of three centimeters or more," *Journal of the National Cancer Institute*, vol. 82, no. 19, pp. 1539–1545, 1990.

[24] S. V. Liu, L. Melstrom, K. Yao, C. A. Russell, and S. F. Sener, "Neoadjuvant therapy for breast cancer," *Journal of surgical oncology*, vol. 101, no. 4, pp. 283–291, 2010.

[25] H.-c. Cho, L. Hadjiiski, B. Sahiner, H.-P. Chan, M. Helvie, C. Paramagul, and A. V. Nees, "A similarity study of content-based image retrieval system for breast cancer using decision tree," *Medical physics*, vol. 40, no. 1, p. 012901, 2013.

[26] O. Steichen, C. Daniel-Le Bozec, M. Thieu, E. Zapletal, and M.-C. Jaulent, "Computation of semantic similarity within an ontology of breast pathology to assist inter-observer consensus," *Computers in Biology and Medicine*, vol. 36, no. 7, pp. 768–788, 2006.

[27] A. A. Onitilo, J. M. Engel, R. T. Greenlee, and B. N. Mukesh, "Breast cancer subtypes based on ER/PR and Her2 expression: comparison of clinicopathologic features and survival," *Clinical medicine & research*, vol. 7, no. 1-2, pp. 4–13, 2009.