Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Interfaz visual para la navegación de Big Semantic Data

Autor:
Dña. Silvia González Gómez

Tutores:
Miguel Ángel Martínez Prieto
Javier David Fernández García

"*Aprendemos errando*"

(Pietro Metastasio)

A Sikmar

RESUMEN

El actual “diluvio de datos” está inundando Internet con grandes volúmenes de información, dando lugar a la Web de Datos. Esta es una simplificación de la Web Semántica, que proponía inicialmente superar las limitaciones de la red actual introduciendo descripciones explicitas del significado. Este tipo de datos semánticos se describen habitualmente mediante el lenguaje RDF, y el formato RDF/HDT permite reducir la redundancia del lenguaje para favorecer el procesamiento por parte de la máquina y la gestión de los datos.

Este proyecto presenta una primera aproximación hacia una interfaz gráfica que permita realizar consultas sobre un archivo HDT, mostrando los resultados en forma de grafo navegable.

ABSTRACT

The amount of information available at the Internet is growing day by day, almost in an exponential way, in what some sources call a “Data Flooding”, and this massive amount of accessible data is changing the current Internet into a brand new monster: the Web of Data. This new step towards the future simplifies some semantic web concepts, although it keeps in mind their ultimate goal: to give the web some level of self-stored meaning, making it “aware”, to some extent, of the kind of information it contains.

The RDF language is one of the most extended ways of defining semantic data, and the RDF/HDT format allows the user to reduce this language’s level of redundancy while speeding up computer processing and handling of data. This project establishes a first approach to a graphic interface capable of searching within a HDT file, and showing results as an interactive graph.

1. ÍNDICES

1.1 ÍNDICE DE CONTENIDOS

1.2 ÍNDICE DE FIGURAS

2. INTRODUCCIÓN

2.1 DESCRIPCIÓN DEL PROYECTO

El siguiente documento presenta el sistema *"Implementación de una Interfaz Visual para la navegación de Big Semantic Data"*, realizado por Silvia González como parte del Trabajo Fin de Grado del Curso Puente de acceso al Grado en Informática.

2.2 ALCANCE DEL PROYECTO

Big Data es una de las *buzzwords* (palabra de moda) en el escenario tecnológico actual, y viene a describir todos aquellos conjuntos de datos que, por su volumen, variedad o necesidades especiales de procesamiento, no pueden ser tratados con técnicas estándares de bases de datos.

El concepto de *Big Semantic Data* es una especialización del anterior y tiene que ver con la representación y manejo de grandes colecciones de datos semánticos, es decir, entendibles por procesos automáticos.

Este tipo de datos semánticos se describen, habitualmente, mediante RDF (*Resource Description Framework*).

Este modelo lógico de datos se basa en describir ternas (sujeto, propiedad, objeto), y puede interpretarse como un grafo etiquetado donde el nodo sujeto es la entidad que se describe, la etiqueta o propiedad es lo que se dice de dicho sujeto y el nodo destino es el valor para esa propiedad.

2.3 OBJETIVO

El objetivo de este proyecto es implementar una metáfora de navegación visual para la consulta de Big Semantic Data. Es decir, se desarrollará una aplicación que permita cargar un subconjunto de datos y navegar por ellos desplazándose por las distintas entidades y propiedades, utilizando las diferentes aristas que relacionan la información.

Se partirá de una librería de programación, ya implementada, que indexa este tipo de colecciones en espacio comprimido. El trabajo se centrará en diseñar e implementar un modelo de interfaz que realice las llamadas apropiadas a la librería para ir obteniendo la información a medida que se navega por el grafo.

2.4 METODOLOGÍA A UTILIZAR

Para la gestión de proyecto se seguirán las buenas prácticas definidas dentro del *PMbok v4* del Project Management Institute (PMI).

Según esta metodología el ciclo de vida del proyecto se basa cinco grandes grupos de gestión de proyectos:

- Inicio
- Planificación
- Ejecución
- Seguimiento y control
- Cierre

2.5 CONTENIDO DE LA MEMORIA

La presente memoria está organizada en diferentes capítulos, subdivididos a su vez en diferentes bloques:

1. **Índices**

   Guías de referencia para los contenidos y las figuras utilizadas

2. **Introducción**

   Presentación del proyecto y objetivos a alcanzar. Incluye el resumen de la memoria y el contenido del CD adjunto.

3. ¿Qué es la Web semántica?

Introducción a los principales conceptos de la Web Semántica, diferencias respecto a la Web actual y formatos relacionados.

4. Proyecto RDF/HDT

Introducción al proyecto RDF/HDT en el que se engloba el presente Trabajo Fin de Grado. Principales objetivos y herramientas de las que consta.

5. Metodología de Gestión de Proyecto

Introducción a las buenas prácticas del PMI que se utilizarán como base en la gestión del proyecto. Descripción de las principales fases y entregables del proyecto en cada una.

6. Ejecución del proyecto

Descripción detallada de la fase de ejecución, que engloba principalmente las etapas de análisis, diseño y desarrollo del *software* del proyecto.

Se recogerá también el detalle de los casos de prueba y sus resultados.

Se incluirán los manuales de instalación y funcionalidad del sistema resultante.

7. Conclusiones

Descripción de los objetivos alcanzados y la posible evolución y mejoras del sistema.

8. Bibliografía

Fuentes bibliográficas y referencias Web consultadas durante la ejecución del proyecto

2.6 CONTENIDO DEL CD

Resumen del contenido del CD adjunto

1. **Memoria:** documento *pdf* correspondiente a la presente memoria en un archivo único con el nombre "*Memoria*".
2. **Software:** directorio con el código fuente desarrollado, incorporado dentro del proyecto HDT it!
3. **GraphViz**: las librerías y documentación que se han utilizado para la integración con esta herramienta externa.
4. **Manuales:** breve manual de uso de la aplicación.

3. ¿QUÉ ES LA WEB SEMÁNTICA?

3.1 INTRODUCCIÓN

El avance de Internet en los últimos años ha sido exponencial, llegando a ser un instrumento cotidiano fundamental en nuestra sociedad y superando incluso a otros medios tan vitales y ampliamente establecidos entre el público general como la televisión o el teléfono.

De forma paralela a este crecimiento, las tecnologías que soportan la Red de Redes han experimentado una rápida evolución que ha permitido la obtención de una Web mejor, más flexible y más fácil de mantener.

Entre las últimas tendencias ha surgido la visión de la *Web semántica* promovida por el propio Tim Bernes Lee [1] (inventor de la Web y presidente de W3C) que se centra en conseguir que los ordenadores puedan entender, y por lo tanto utilizar, la información contenida en la Web.

Esta nueva Web se basaría en la representación de forma comprensible y estandarizada de los diversos recursos que la componen, ofreciendo a los diferentes agentes una clasificación más eficiente que permitiría devolver resultados más precisos ante una búsqueda de información.

La Web semántica propone superar las limitaciones de la red actual introduciendo descripciones explicitas del significado, la estructura interna y la global de los contenidos y servicios disponibles. Según la definición del W3C:

La Web semántica proporciona un marco común que permite que los datos sean compartidos y reutilizados a través de aplicaciones, empresas y fronteras comunitarias. Es un esfuerzo colaborativo liderado por el W3C con la participación de un gran número de investigadores y socios industriales. Está basado en Resource Description Framework (RDF) e integra una variedad de aplicaciones utilizando XML para la sintaxis y URI para las denominaciones (www.w3.org/2001/sw/)

3.2 WEB SINTÁCTICA VS WEB SEMÁNTICA

Si tomamos como ejemplo la representación a través de un grafo, la diferencia fundamental entre la Web actual y la propuesta por la Web semántica [2], mostrada en la siguiente imagen, se centra en que en el caso de la Web actual los nodos siempre estarían compuestos por páginas HTML con arcos de relación basados en hiperenlaces. Esto supone que no existiría ninguna diferencia entre un *blog* personal o una página de reserva de hoteles.

El caso de la Web semántica es diferente, porque cada nodo tendrá un dato, que será la mínima unidad de información, y cada arco representará la relación existente entre cada uno de estos nodos. Esto hace que la Web actual se considere una solución orientada al documento y la Web semántica orientada al dato.

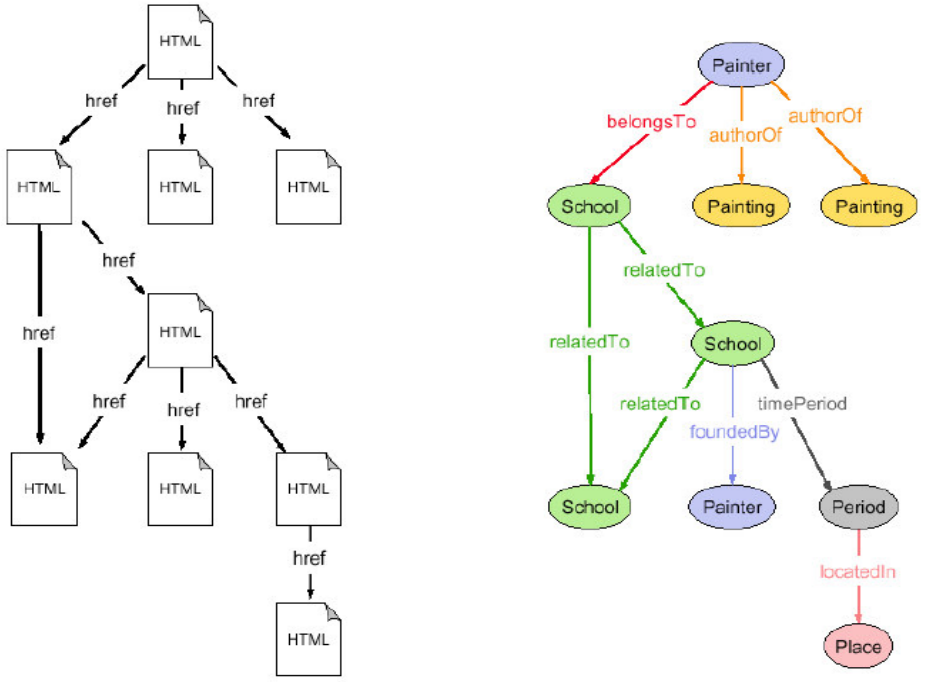


Figura 1 – Diferencia entre la Web tradicional y la Web semántica (extraído de [2])

Las páginas HTML actuales disponen de diferentes etiquetas que aportan valor estructural o que permiten incorporar diferentes elementos gráficos. Estas marcas, además de no aportar ningún valor semántico, pueden ser utilizadas de forma incorrecta. Esto provoca que las páginas sean comprensibles únicamente para seres humanos.

Las búsquedas que se realizan a través de los diferentes buscadores, especialmente Google, se basan en la comparación de la palabra a localizar con los términos que tiene almacenados en su índice. Esto quiere decir que si buscamos, por ejemplo, “depresión”, no existe ninguna forma de saber si nos estamos refiriendo a un término médico o a uno económico.

La Web semántica abandona la idea de buscar a través de la simple comparación de caracteres para avanzar hacia la búsqueda de conceptos. Esta aproximación se asemeja bastante al objetivo de la Inteligencia Artificial.

Realmente se trata de una estructura compleja compuesta de diferentes niveles y componentes:

- **Alfabeto Unicode:** es una codificación del texto que permite expresar información en cualquier idioma.
- **Referencias URI:** es el acrónimo de "Uniform Resource Identifier", identificador único que permite la localización de un recurso que puede ser accedido vía Internet.
- **RDF:** es un modelo lógico de datos simple mediante el cual definimos sentencias en el formato de ternas (sujeto: el recurso al que nos referimos; predicado: el recurso que indica qué es lo que estamos definiendo; y objeto: puede ser el recurso o un literal que podría considerarse el valor de lo que acabamos de definir). Por su parte *RDF Schema* provee un vocabulario definido sobre RDF que permite el modelo de objetos con una semántica claramente definida.
- **Formato XML:** ofrece un formato común para intercambio de documentos, y *XML Schema* proporciona una plantilla para elaborar documentos estándar. Ambas tecnologías que hacen posible que los agentes puedan entenderse entre ellos.
- **Vocabularios Ontologías:** ofrecen un criterio para describir y clasificar la información.
- **Reglas de inferencia Lógica:** permiten a la computadora manipular los términos de modo más eficiente, beneficiando la inteligibilidad humana.
- **Pruebas:** intercambio de "pruebas" escritas en el lenguaje unificador, a través del uso de reglas de inferencia.

- **Confianza:** comprobación exhaustiva de las fuentes de información.
- **Firma digital:** bloque encriptado de datos para verificar que la información adjunta ha sido ofrecida por una fuente específica y confiable.

Gráficamente estos niveles y componentes serían:

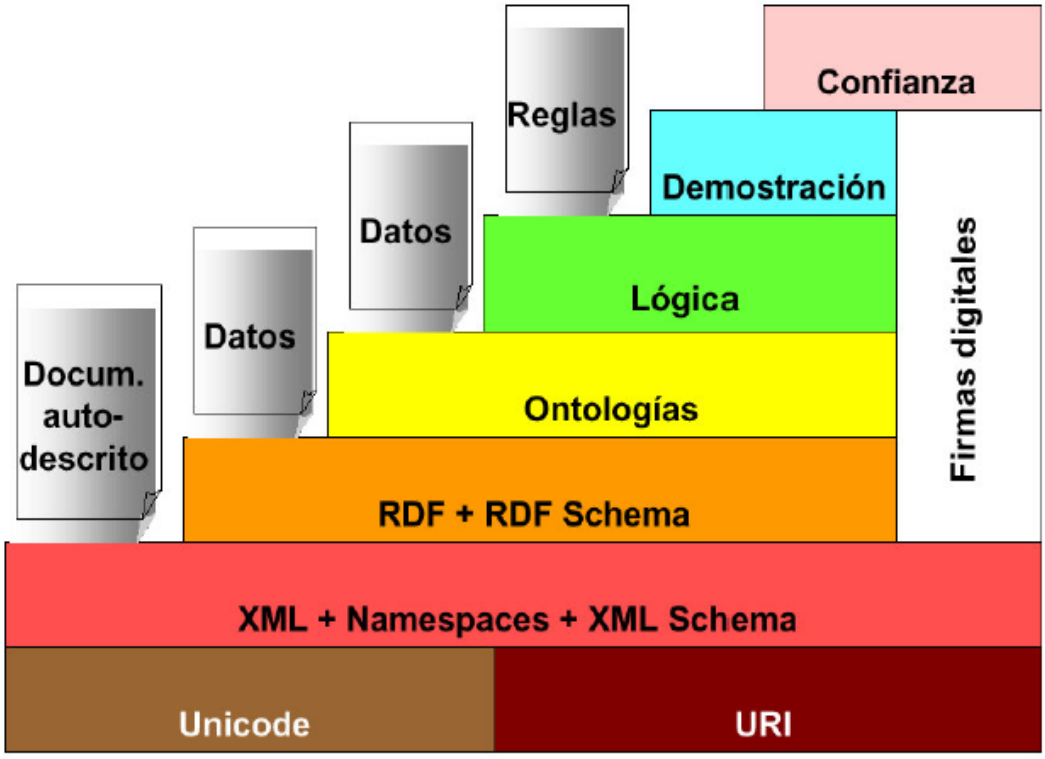


Figura 2-Representación de la Web semántica (extraído de [2])

3.3 BIG DATA

La expresión *"Big Data"* [3] aparece cada vez con mayor frecuencia dentro del escenario tecnológico, e incluso en otros contextos más generales como el científico o el económico. En general nos referimos con ella a los nuevos problemas que surgen al abordar el tratamiento y análisis de repositorios de información de tamaño tan enorme que resulta imposible procesarlos con los sistemas de bases de datos convencionales.

Tradicionalmente, este volumen de datos y esta necesidad de tratamiento se asociaban a procesamientos científicos de gran complejidad pero con una repercusión muy limitada, como la predicción meteorológica, la prevención de fenómenos sísmicos o la secuenciación de ADN, pero en los últimos años han aparecido escenarios de *Big Data* que resultan mucho más cercanos para el gran público: proliferación de páginas *web*, análisis masivo de imágenes y secuencias de vídeo, control de las redes sociales, multiplicación de los dispositivos móviles y sus "*apps"*, sensores con respuesta en tiempo real, Internet de las cosas...

Estos nuevos escenarios generan información a una escala hasta ahora inimaginable, y según algunas estimaciones el 90% de los datos del mundo han sido creados durante los últimos dos años.

En cualquier caso, aunque el volumen de la información sea la dificultad más evidente con la que hay que lidiar, los escenarios de Big Data suelen caracterizarse por la convergencia de "las tres V's":

- El **Volumen** es la dimensión más obvia del problema, especialmente cuando se considera el aspecto de la escalabilidad. El método de almacenamiento escogido influye mucho en la capacidad de recuperación de datos, y es difícil mantener la tasa de respuesta que requiere por ejemplo un sistema en tiempo real con un volumen de datos que no sólo es muy grande cuando se plantea el sistema por primera vez, sino que además puede crecer por encima de cualquier previsión.

- La **Velocidad** es un segundo frente de batalla, ya que lo datos deben fluir en grandes cantidades y en sistemas cada vez más distribuidos. Cualquier de aplicación de *streaming*, como por ejemplo un servidor multimedia alojado en la nube, puede estar proporcionando audio y video a numerosos usuarios en distintas plataformas, en un modelo de negocio donde la respuesta en tiempo real es clave.

- Finalmente, la **Variedad** sería el tercer vértice en el conjunto de desafíos que presenta un escenario Big Data. La información que tratamos puede proceder de orígenes muy dispares, cada uno con su propia semántica y sin un modelo estructural común presente en trabajos científicos más concretos como la secuenciación del ADN. El objetivo aquí sería conseguir mecanismos para enlazar de forma efectiva tipos de información que no comparten una estructura interna común.

Volumen y velocidad tienen en común que son problemas básicamente físicos, donde la mejora de la tecnología permitirá alcanzar nuevas metas. La variedad, en cambio, es un problema de carácter lógico, pues buscamos una modelo que integre todos nuestros datos, y un mejor modelo permitirá exprimir más la información y generar un conocimiento más valioso. Por eso este tercer vértice [5] es el reto al que se enfrenta el Big Data en primer lugar, y el enfoque líder para resolverlo usa como base la **información semántica** descrita en el punto anterior.

3.4 RDF

El **Marco de Descripción de Recursos** (***Resource Description Framework***, **RDF**) es un *framework* para metadatos en la World Wide Web (WWW), desarrollado por el World Wide Web Consortium (W3C).

Es un lenguaje de objetivo general para representar la información sobre recursos en la web. Este modelo se basa en la idea en definir las declaraciones de los recursos en expresiones sujeto-predicado-objeto. La combinación de RDF con otras herramientas como RDF Schema y OWL permite añadir significado a las páginas, y es una de las tecnologías esenciales de la Web semántica [4].

RDF se basa en los estándares de URIs y Unicode además de que se puede presentar en XML (por lo que se le considera como una de sus aplicaciones).

La esencia de RDF es, por tanto, la definición de un modelo formal para la representación de las propiedades y los valores de esas propiedades:

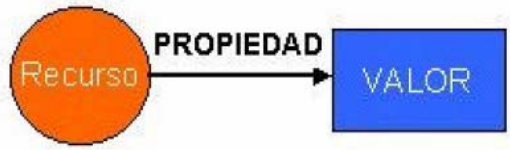


Figura 3 – RDF: Recurso, Propiedad y valor (extraído de [4])

Según el modelo de datos que propone RDF consiste en tres tipos de objetos:

- **Recursos**: cualquier objeto de la Web identificable a través de una URI (Universal Resource Identifier). Un recurso puede ser un documento HTML, una parte de una página Web (como por ejemplo un elemento XML dentro de un documento fuente) o, generalizando la definición, cualquier objeto de información.
- **Propiedades**: aspectos específicos, atributos o relaciones utilizadas para describir recursos. Cada tipo de propiedad tiene sus valores específicos y define los valores permitidos, los tipos de recursos que puede describir y las relaciones que existen entre las distintas propiedades.

- **Descripciones**: Son el conjunto formado por un recurso, un nombre de propiedad y el valor de esa propiedad. Así pues, una declaración o sentencia está compuesta por 3 partes individuales:
    - sujeto: recurso
    - predicado: propiedad
    - objeto: valor de la propiedad, que puede ser otro recurso, especificado por un URI, o un literal (una cadena simple de caracteres u otros tipos de datos primitivos definidos por XML). El contenido de un literal no es interpretado por RDF en sí mismo y puede contener marcado XML adicional. Los literales se distinguen de los recursos en que el modelo RDF no permite que los literales sean sujeto de una declaración.

Este conjunto de tres elementos se conoce como **terna** y es la unidad básica de información en RDF.

La sintaxis básica es la de *XML 1.0,* y se pueden distinguir dos tipos de construcciones sintácticas para codificar RDF. Por un lado la serializada, que expresa de una forma muy regular todas las capacidades de un modelo de datos RDF. Por otro lado está la sintaxis abreviada, que incluye construcciones adicionales.

A pesar de todo, el modelo y la sintaxis no facilitan los mecanismos para definir las propiedades ni las relaciones entre esos predicados y otros recursos o sujetos: por ello se ha definido también una especificación para definir los esquemas. Un esquema RDF es un conjunto de informaciones relativas a las clases de recursos, que sirve para explicitar las relaciones jerárquicas que se establecen entre ellos, o bien para matizar el carácter obligatorio u opcional de las propiedades y otras restricciones como el número de ocurrencias, etc.

Para aclarar conceptos, pongamos un ejemplo[4] basado en la especificación *Primer RDF* (http://www.w3.org/TR/rdfprimer/) donde se muestran una serie de declaraciones: "hay una persona identificada por http://www.w3.org/People/EM/contact#me, cuyo nombre es Eric Miller, cuya dirección de correo electrónico es em@w3.org, y cuyo título es "Dr." que podría representarse como el grafo RDF de la siguiente figura:

Figura 4 – Ejemplo de RDF (extraído de [4])

Con identificación basada en *URIs:*

- Individuos, como Eric Miller, que está identificado por http://www.w3.org/People/EM/contact#me
- Clases de cosas, como Persona, identificada por http://www.w3.org/2000/10/swap/pim/contact#Person
- Propiedades de estos conceptos, como *mailbox*, identificada por: http://www.w3.org/2000/10/swap/pim/contact#mailbox
- Valores de estas propiedades, mailto:em@w3.org como el valor de la propiedad *mailbox.*

En sintaxis XML el grafo correspondiente a de la ilustración anterior sería:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdfsyntax-ns#"
xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
<contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
<contact:fullName>Eric Miller</contact:fullName>
<contact:mailbox rdf:resource="mailto:em@w3.org"/>
<contact:personalTitle>Dr.</contact:personalTitle>
</contact:Person>
</rdf:RDF>
```

3.5 HDT

La actual "Web de Datos", como evolución del concepto de Web semántica, está produciendo conjuntos de datos RDF de tamaño cada vez mayor, y la necesidad de intercambiar esta información masiva ha desvelado los inconvenientes tradicionales de la representación RDF: alto nivel de redundancia y escasa capacidad de procesamiento automático. Los teléfonos inteligentes de nueva generación y otros dispositivos portables se enfrentan a problemas similares, incluso trabajando con conjuntos de datos de tamaño más reducido, por lo que la escena tecnológica actual reclama un formato más eficiente para la publicación, intercambio y consumo de RDF.

RDF/HDT[10] es un formato centrado en los datos que reduce la redundancia del lenguaje para favorecer el procesamiento por parte de la máquina y la gestión de los datos. La especificación define un formato compacto de serialización binaria, optimizado para el almacenamiento y la transmisión a través de una red.

Un conjunto de datos codificado con HDT[11] está formado por tres secciones lógicas, como se muestra en la imagen siguiente, la cabecera, el diccionario y las ternas, diseñadas específicamente para lidiar con las peculiaridades de RDF.

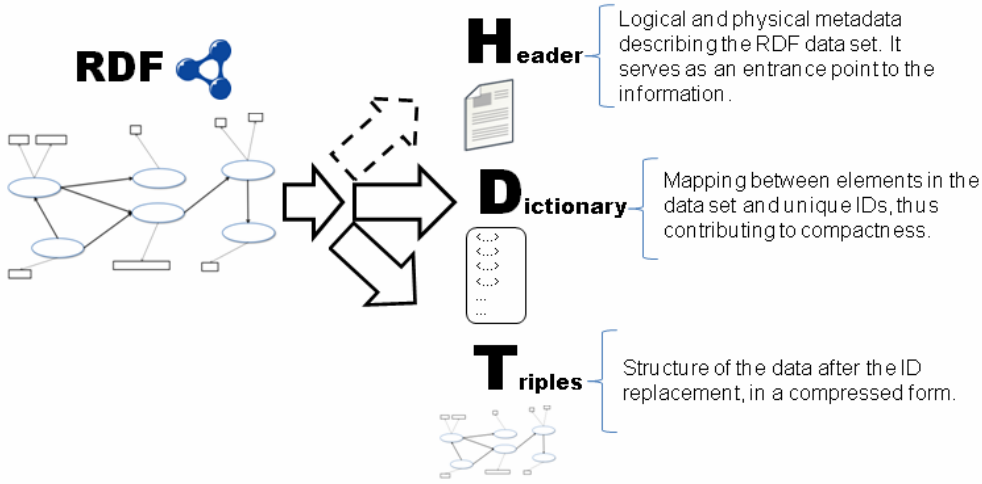


Figura 5 – HDT: Cabecera, Diccionario y Ternas (extraído de [10])

- **Cabecera**: La cabecera contiene los metadatos que describen el conjunto de datos semánticos del HDT, usando el RDF estándar. Actúa como un punto de entrada para el consumidor, que puede así hacerse una idea básica de las propiedades clave del contenido incluso antes de recuperar el conjunto de datos completo.

- **Diccionario**: El diccionario es un catálogo que comprende todos los términos utilizados en el conjunto de datos (URI's, literales y nodos vacíos). Cada término recibe un identificador único (ID), que permite la representación de una terna de tres identificadores, y cada identificador referencia dentro del diccionario a su respectivo término sujeto/predicado/objeto. Este es un primer paso de cara a lograr una buena compresión, ya que evita que los términos largos se repitan una y otra vez. Además, las cadenas de caracteres similares se guardan de forma consecutiva dentro del diccionario, circunstancia que puede aprovecharse para mejorar aún más la compresión.

- **Ternas**: Como se ha explicado en el punto anterior, las ternas RDF pueden verse ahora como tuplas de tres identificadores, por lo que esta última sección del fichero mapea el grafo de relaciones entre los términos del conjunto de datos, tal y como se muestra en la figura 6. Comprendiendo las propiedades típicas de un grafo RDF podemos crear formas más eficientes de representar esa información, tanto para reducir su tamaño global como para ofrecer operaciones de búsqueda más eficientes.

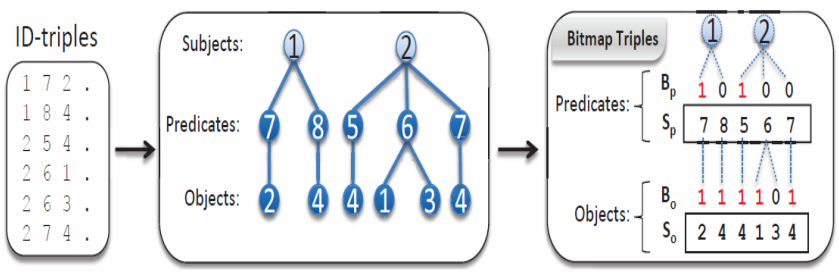


Figura 6 – HDT: Ternas (extraído de [6])

Los puntos fuertes del formato HDT son tres:

- **Compactación**. Se usa menos espacio para codificar la misma información, con el consiguiente ahorro en dispositivos de almacenamiento, en ancho de banda y en tiempos de transmisión. Como el diseño de la compresión tiene en cuenta las peculiaridades del

RDF, mejora las cifras de compresores de ámbito generalista como GZIP o BZIP2.

- **Intercambio y publicación**. Se puede acceder fácilmente a los metadatos, al mantenerse estos en la sección de cabecera, junto con las estadísticas sobre su procedencia. El contenido separa los términos (diccionario) de la estructura de relaciones entre ellos (ternas)

- **Acceso indexado bajo demanda**. Permite operaciones de búsqueda rápidas sobre porciones del fichero, sin necesidad de descomprimirlo en su totalidad.

Finalmente, un fichero binario HDT[12] puede cargarse directamente en la memoria RAM de un sistema informático, con lo que accederíamos a él como a una estructura de datos, evitando costosas operaciones de indexación. Además, gracias a las técnicas de compresión, aumenta la cantidad de información que queda disponible en los niveles superiores de la jerarquía de memoria, y almacenar los datos en la memoria más rápida siempre redunda en operaciones de consulta más rápidas.

4. EL PROYECTO RDF/HDT

4.1 DESCRIPCIÓN DEL PROYECTO

Actualmente los datos RDF se tratan mediante un formato serializado basado en texto que hace que sea poco eficiente su envío y las búsquedas realizadas sobre él.

HDT (Header, Dictionary, Triples) es un estructura de datos que se basa en serialización binaria para RDF, que permite ahorran gran cantidad de espacio, además de optimizar las operaciones de consulta. Esto lo convierte en un formato ideal para almacenar y compartir conjuntos de datos RDF en la Web.

En la siguiente imagen se muestra una comparativa de HDT frente a las técnicas tradicionales, tomando como referencia la descarga y la consulta de un conjunto de datos.

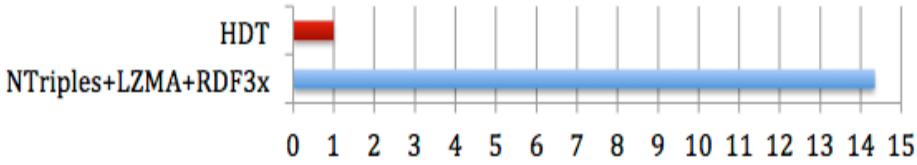


Figura 7 – HDT: comparativa descarga y consulta (extraído de [6])

Datos de relevancia de este formato:

- El **tamaño de los archivos es menor** que otros formatos de serialización RDF.
- El archivo de **HDT ya está indexado**. Mediante el uso de HDT, se descarga el archivo y se puede iniciar su consulta en minutos, en lugar de perder el tiempo con herramientas de análisis que son difíciles de configurar.
- **Alto rendimiento de consulta**: las técnicas de compresión de HDT permiten que la mayor parte de los datos se puede mantener en la memoria principal.
- Altamente **concurrente**: HDT es un formato de de sólo lectura, por lo que puede realizar varias consultas por segundo utilizando varios subprocesos.
- El **formato es abierto y en proceso de estandarización.** Esto asegura que cualquier usuario de Internet puede generar y consumir los archivos, o incluso escribir su propia implementación.

- Las **bibliotecas son de código abierto (LGPL).** Puede adaptar las bibliotecas a sus necesidades, y la comunidad puede detectar y corregir problemas.

Incluimos en este punto algunos usos relevantes para este formato:

- Compartir datos RDF en la Web.
- Análisis y Visualización de Datos: extracción de estadísticas y minería de datos.
- *Smartphones* y dispositivos integrados: la utilización de formatos de datos optimizados permite a este tipo de dispositivos utilizar los recursos de manera eficiente en el desarrollo de aplicaciones semánticas.

Dentro del proyecto RDF/HDT[6] se incluyen las siguientes herramientas:

- C ++ y bibliotecas Java / herramientas de línea de comandos para crear y buscar archivos de HDT.
- **HDT it! GUI** para abrir y navegar por los archivos HDT. Incluye una visualización 3D de la matriz del gráfico RDF.
- Jena Integración que permite el uso de Jena ARQ para realizar consultas SPARQL y hace posible la creación de un punto final SPARQL Fuseki especificando uno o más archivos de HDT como gráficos con nombre.
- Servicio Web *online* para convertir archivos RDF a HDT.

4.2 HDT IT!

HDT it! Herramienta GUI es el punto central de este Trabajo Fin de Grado y está disponible para las siguientes plataformas:

- Windows de 64 bits
- Windows de 32 bits
- Mac OS X
- I386 Linux
- Linux x86_64

Esta interfaz permite la gestión de ficheros en formato HDT y ofrece a través de su pantalla principal las principales funcionalidades:

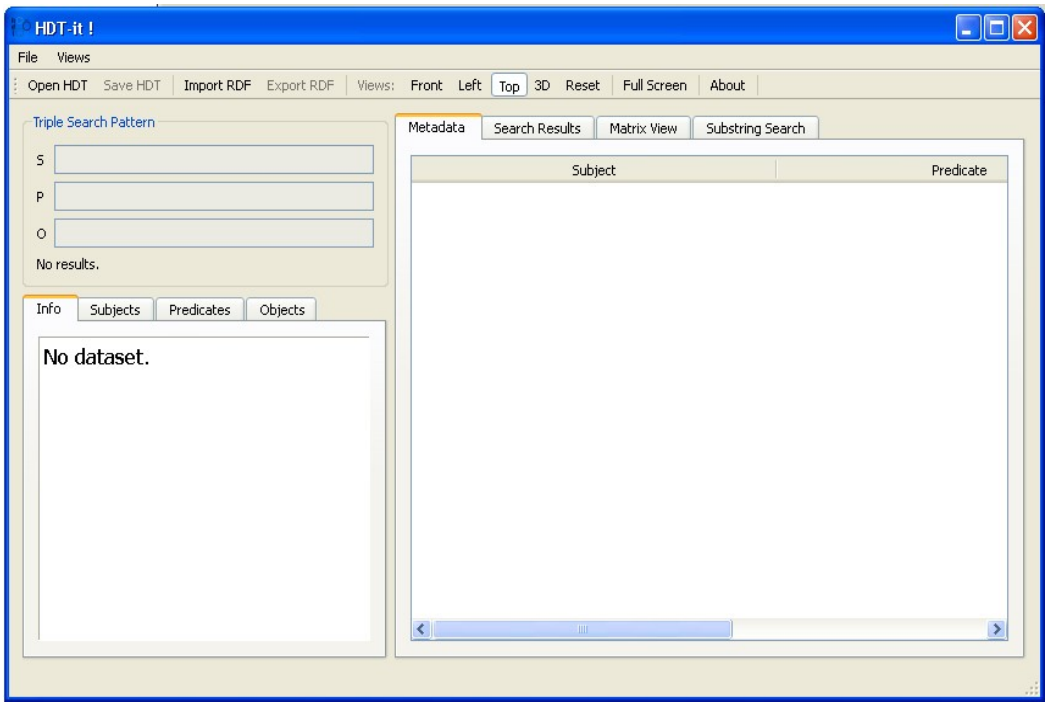


Figura 8 – Pantalla principal de HDT it!

- Apertura de ficheros HDT
- Consulta de metainformación: DataSet, Diccionario y Ternas
- Visualización de la información de las ternas: Sujeto, Predicado, Objeto
- Realización de consultas sobre Sujetos, Predicados y/o Objetos y exportación a RDF (Figura 9).
- Visualización de los datos a través de matriz 3D (Figura 10).

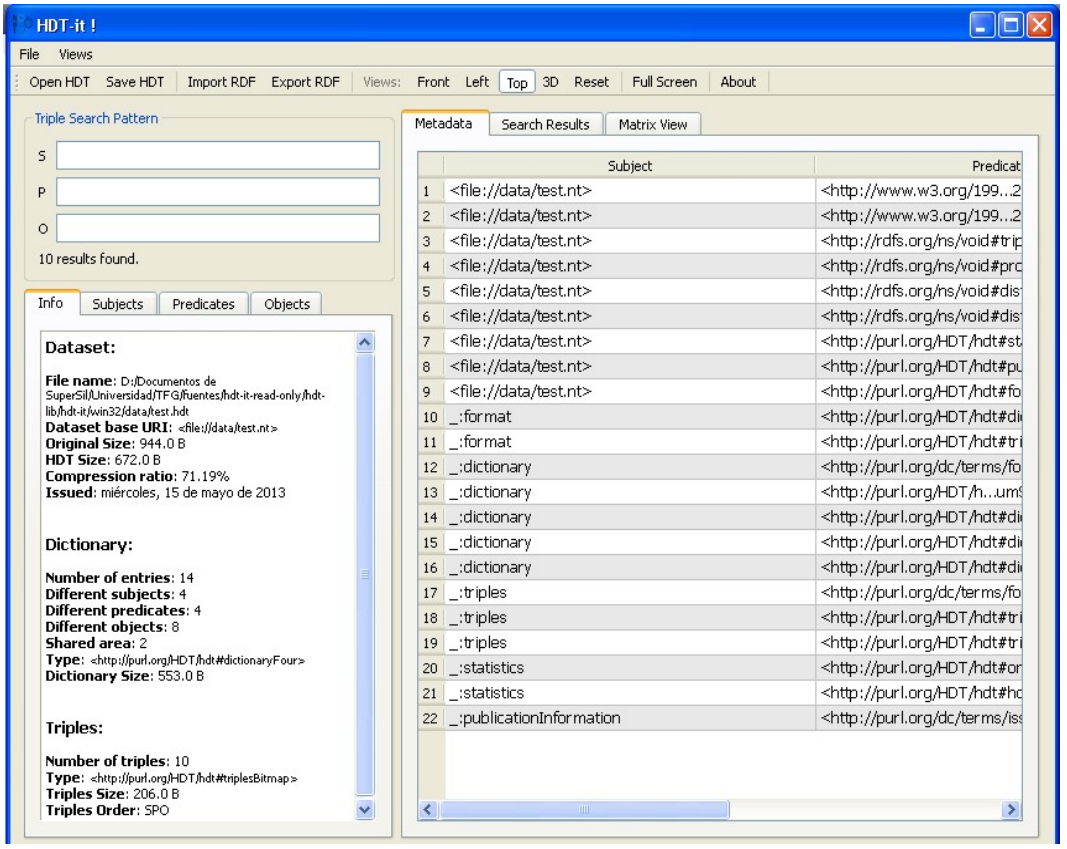


Figura 9 – Pantalla principal de HDT it! tras la carga de un HDT

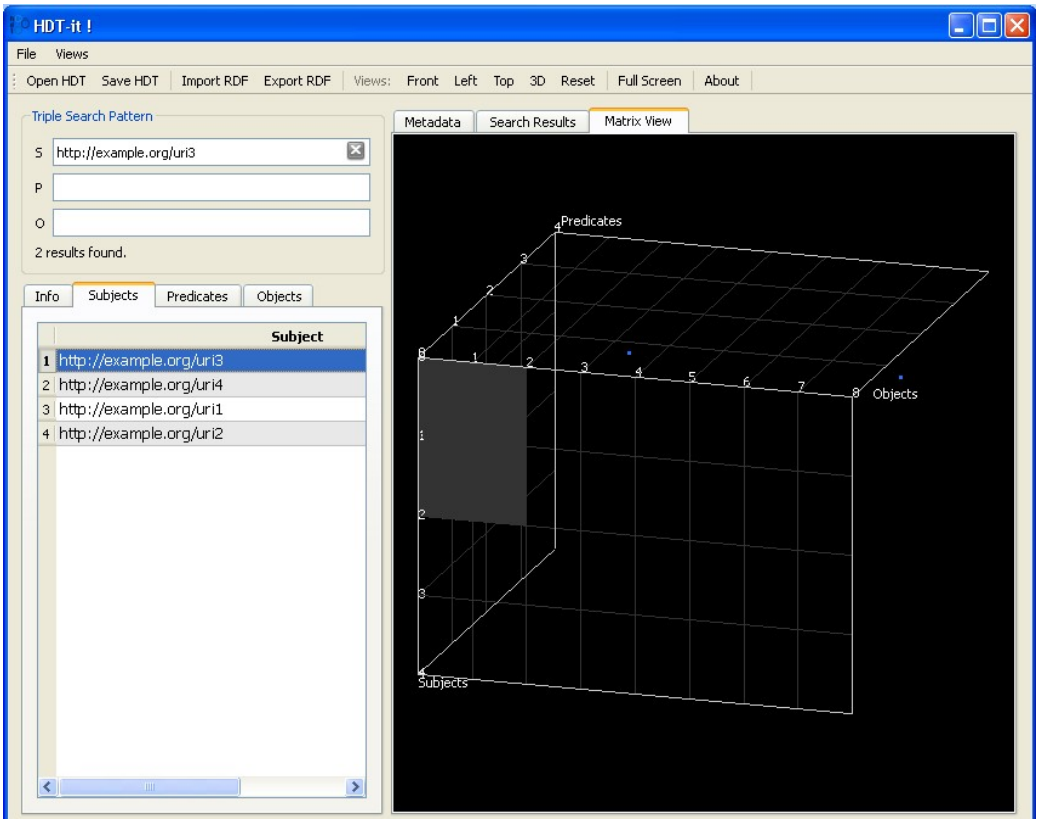


Figura 10 – Pantalla principal de HDT it! – Pestaña Matriz 3D

4.3 INTERFAZ GRÁFICA PARA GRAFOS HDT

4.3.1 OBJETIVO

El objetivo fundamental de este trabajo es añadir una funcionalidad más a la herramienta gráfica HDT it!: la posibilidad de visualizar y realizar consultas a través de una vista gráfica que muestre el formato HDT en forma de grafo.

Se tomará de partida el HDT ya cargado, para mostrar el grafo resultante en una nueva pestaña de la aplicación, filtrando los datos representados por los arcos y nodos del grafo según los criterios de búsqueda introducidos en el Sujeto, Predicado y/o Objeto.

En cada nodo del grafo se mostrarán las URI's correspondientes al Sujeto y al Objeto, y en el arco, el valor del Predicado.

Los nodos podrán ser pulsados por el usuario para provocar una nueva búsqueda, a partir del valor del nodo seleccionado.

Obtenida la información a mostrar, el principal **problema** se centra en el **cálculo de posiciones de los nodos.** Se requiere analizar en qué posición se ubicará cada uno de los nodos resultantes para que la información a mostrar se vea de una forma adecuada en el espacio disponible en la pantalla.

Desde el inicio del proyecto se plantea la posibilidad de estudiar si existen librerías gráficas especializadas en grafos, y compatibles con la arquitectura de la aplicación, que puedan facilitar todos estos cálculos.

¿Qué deberían cumplir estas herramientas?

- Las librerías deben ser integrables dentro de la herramienta HDI-it!, desarrollada con Qt.
- Deben permitir el cálculo de posiciones de los nodos
- Deben facilitar la incorporación de información textual dentro de cada nodo y arco.
- Finalmente, deben ofrecer un formato de salida compatible con la gestión de señales/eventos realizada por Qt.

Tras una primera búsqueda de posibilidades el listado de herramientas para dibujar gráficos se centraba en herramientas como Isaviz, Graphviz, yEd y Microsoft Automatic Graph Layout, entre otras.

Analizando en detalle cada una de ellas se descubre que en su mayoría ofrecen soluciones independientes que permiten visualizar grafos en diferentes formatos, con resultados más o menos decorativos, pero sin la posibilidad de integrarse con una herramienta ya existente.

Descartando todas las que no ofrecían una librería integrable, el estudio se centró en analizar las posibilidades ofrecidas por **Graphviz**, que dispone de una interfaz gráfica propia pero también de la opción de integración dentro de proyectos existentes, además de ofrecer diferentes formatos de salida.

El principal problema detectado para esta integración es la imposibilidad de gestionar la interacción de los usuarios con los formatos ofrecidos como salida, por lo que este punto se desarrollará a través de las herramientas de Qt.

Esta herramienta tiene a su favor una larga trayectoria, pues está disponible desde 1988, hay versiones actualizadas y una gran comunidad de usuarios y herramientas que la utilizan como núcleo central de sus soluciones. Facilita mucha documentación y soporte para la resolución de problemas.

Toda la información oficial sobre Graphviz puede consultarse en: http://www.graphviz.org/

4.3.2 INTRODUCCIÓN A GRAPHVIZ

Graphviz[8] es una aplicación de visualización de grafos de código abierto que incluye un gran número de programas de trazado de gráfico; además cuenta con interfaces interactivas y vía web, así como con herramientas auxiliares y bibliotecas de funciones, existiendo versiones tanto para Windows como para Linux. Fue creado por los laboratorios AT&T por John Ellson en los Estados Unidos.

Los programas de diseño de **Graphviz** parten de descripciones de gráficos en texto plano, lo que les permite ser editados por los usuarios y no necesitar ningún programa adicional para ello. Los diagramas son realizados en varios formatos: imágenes (jpg ó png), SVG (Scalable Vector Graphics, gráficos vectoriales en dos dimensiones) para páginas web, Postscript para ser incluido

en PDF's u otros documentos, o también pueden ser representados en un navegador interactivo, donde el usuario pueda editarlo (Graphviz también soporta GXL, Graph eXchange Language). En la Figura 11 se muestra un ejemplo de grafo obtenido con esta herramienta:

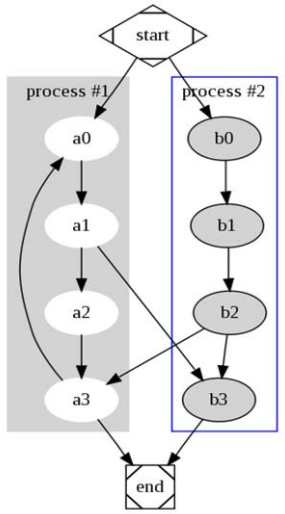


Figura 11 – GraphViz (extraído de [8])

Graphviz cuenta con muchas características para personalizar los diagramas tales como opciones para etiquetas, colores, fuentes, diseños en forma de tabla, estilos de línea, enlaces y formas. En la práctica, los gráficos suelen ser generados partiendo de fuentes externas de datos, pero también puede hacerse manualmente, bien editando un fichero de texto plano en lenguaje DOT o bien mediante un editor gráfico.

Como punto central de esta herramienta se encuentran los diferentes **algoritmos** de generación de gráficos, entre los que destacamos:

- **dot**: realiza dibujos "jerárquicos" o por capas de gráficos directos. El algoritmo de representación trata de colocar todos los enlaces en la misma dirección (de arriba abajo o de izquierda a derecha) y después trata de evitar los cruces entre enlaces, y por último, reducir sus longitudes.
- **neato**: crea "trazados elásticos". Emplea el algoritmo Kamada-Kawai, que es equivalente a un escalado estadístico multidimensional.

- **fdp:** crea “trazados elásticos”. Implementa el heurístico Fruchterman-Reingold que incluye un solucionador por cuadrículas para manejar los gráficos más grandes y los grupos dentro de los gráficos indirectos.
- **twopi:** diagrama radial. Se sitúan los nodos en círculos concéntricos dependiendo de su distancia a un nodo raíz dado.

De los algoritmos comentados el que más se ajusta a las necesidades de nuestro trabajo es el “dot”, por lo que será el que se utilice en el mismo.

El **lenguaje dot** es la herramienta principal utilizada por Graphviz para dibujar las diferentes estructuras jerárquicas.

Dot es un lenguaje estructurado que permite de manera sencilla e intuitiva la representación de los tres objetos básicos utilizados dentro de los grafos: nodos, vértices y subgrafos.

Este formato representa los nodos a través de la palabra *“node”* de la siguiente manera:

node [shape=box];

Los arcos se representan uniéndoos dos nodos a través del operador “->”, como se muestra en la siguiente línea:

n*odo1->nodos2;*

A ambos elementos se les podrá agregar diferentes características como color, estilo de letra, fondo, etc. para darle tanto representación visual, como aportación informativa.

La siguiente imagen muestra la representación de un grafo a través de lenguaje dot:

```
digraph a_log0_f_1_1 {
graph [fontsize=12, compound=true, labelloc=top,
       labeljust=r, label="a_log0_f Iter: 1 Substructure: 1
            Graph(2v,1e) Instancias: 12654"];
node [label="\N", fontsize=12];
edge [fontsize=12];
graph [lp="160,133",
      bb="0,0,320,144"];
1 [label=" ACCESO", pos="133,104", width="1.06", height="0.50"];
2 [label=" FROM", pos="133,18", width="0.86", height="0.50"];
      1 -> 2 [label=" SITIO_REMOTO", pos="e,133,36 133,86 133,74
      133,59 133,46", lp="179,61"];
}
```

Como se puede apreciar en el ejemplo anterior, para poder visualizar los elementos de una forma agradable se deben incluir las posiciones de cada elemento. Esta tarea es complicada y el escollo principal a solucionar para utilizar Graphviz en este proyecto.

Como se ha comentado, **Graphviz** ofrece **diferentes formatos de salida** que tras su análisis **se descartaron** como viables para integrar en este Trabajo Fin de Grado, ya que **no permitían la gestión adecuada de la interacción con el usuario.** Esta responsabilidad se trasladará a las **herramientas gráficas de las que dispone Qt: QGraphicsScene**. Se trata de una clase incluida en la distribución de este entorno de desarrollo con la que se pueden dibujar diversos objetos geométricos (líneas, rectángulos, elipses y polígonos) y también cargar imágenes y texto.

4.3.3 USO DE GRAPHVIZ EN EL PROYECTO

La utilización de Graphviz como librería [9] dentro del proyecto se basa fundamentalmente en **optimizar y agilizar el cálculo de posiciones.** Se ha utilizado la versión de Graphviz-2.30.1.

Para poder utilizar esta funcionalidad es necesario incluir las cabeceras de las siguientes clases, que forman parte de las librerías de Graphviz:

- gvc.h
- types.h
- gvplugin.h

- geom.h
- gvcext.h
- pathgeom.h
- textpara.h
- cgraph.h
- cdt.h
- usershape.h

Además del uso de las librerías gvc y cgraph, que son el núcleo de los cálculos a realizar.

El proceso a seguir se centra en los siguientes pasos:

1) Abrir e iniciar el *layout*.

2) Realizar la búsqueda de los nodos del grafo a dibujar a través de las operaciones disponibles en las librerías de HDT it!

3) Crear el grafo (nodos y arcos) según la información obtenida y añadirla al *layout* de Graphviz.

4) Realizar el cálculo de las posiciones de los nodos a través de las librerías de Graphviz utilizando el algoritmo *"dot".* Se indicara en la llamada al *Render* correspondiente que no se va a utilizar ningún archivo de salida tras la llamada, para que almacene en las variables adecuadas las posiciones asignadas.

5) Recuperar las posiciones calculadas y realizar el tratamiento adecuado para que los datos obtenidos de Graphviz se ajusten a las coordenadas utilizadas como punto central en las herramientas gráficas de Qt: *QGraphicsScene.*

6) Almacenar la información en estructuras de datos creadas a medida para que *QGraphicsScene* pueda luego dibujarlas en pantalla.

7) Cerrar el *layout* de Graphviz utilizado.

8) Controlar los eventos dentro de la *QGraphicsScene* para volver a lanzar una búsqueda o alejar/acercar el área del gráfico seleccionado.

Todo este proceso se explicará desde un punto de vista más técnico en las secciones correspondientes al Diseño y Desarrollo de la aplicación.

5. METODOLOGÍA DE GESTIÓN DEL PROYECTO

5.1 INTRODUCCIÓN AL PMBOK

La Guía del PMBOK es un estándar en la gestión de proyectos desarrollado por el **Project Management Institute (PMI)**[13]. La primera versión de PMBOK fue publicada en 1987.

La segunda versión fue publicada en 2000[14], basado en los comentarios recibidos de parte de los miembros. PMBOK fue reconocido como estándar por el American Nacional Standards Institute (ANSI) en 1998, y más adelante por el Instituto de los Ingenieros Electrónicos y Eléctricos (IEEE). Esta versión constaba de 39 procesos.

La tercera versión fue publicada en 2004, con mejoras importantes en la estructura del documento, adiciones a los procesos, términos y dominios del programa y de portafolios. Esta versión constaba de 44 procesos.

La cuarta versión fue publicada en 2008. No introdujo cambios mayores, pero sí se organizó de manera más precisa, clara y fácil de entender. Esta versión consta de 42 procesos y, aunque no es la versión vigente en la actualidad (existe una quinta versión), será la que se tome como referencia en el presente Trabajo Fin de Grado.

PMI pretende definir, mantener y difundir un cuerpo de conocimiento con dos objetivos principales:

- Mejorar el desarrollo de proyectos en diferentes industrias mediante el uso de buenas prácticas.
- Definir procesos de gestión de proyectos estándares y homogéneos para todo tipo de proyectos

Organiza los procesos de la dirección de proyectos según diferentes áreas de conocimiento [15]:

Integración del Proyecto

Describe los procesos y actividades que sirven para integrar los diversos elementos de la dirección de proyectos. Esta compuesto por las siguientes tareas:

- Desarrollar el Acta de Constitución del Proyecto
- Desarrollar el Plan para la Dirección del Proyecto
- Dirigir y Gestionar la Ejecución del Proyecto
- Monitorizar y Controlar el Trabajo del Proyecto

- Realizar Control Integrado de Cambios
- Cerrar el Proyecto o la Fase

Gestión del Alcance

Incluye los procesos involucrados en garantizar que el proyecto incluya todo (y únicamente) el trabajo requerido para completarlo exitosamente, lo que incluye:

- Recopilar los Requisitos
- Definir el Alcance
- Crear la Estructura de Desglose del Trabajo (EDT)
- Verificar el Alcance
- Controlar el Alcance

Gestión del Tiempo

Centrado en los procesos utilizados a garantizar la conclusión a tiempo del proyecto. Compuesto por:

- Definir las Actividades
- Secuenciar las Actividades
- Estimar los Recursos para las Actividades
- Estimar la Duración de las Actividades
- Desarrollar el Cronograma
- Controlar el Cronograma

Gestión de Costes

Describe los procesos involucrados en planificar, estimar, presupuestar y controlar los costes de modo que se complete el proyecto dentro del presupuesto aprobado. Para ello, es necesaria la realización de las siguientes actividades:

- Estimar los Costos
- Determinar el Presupuesto
- Controlar los Costos

Gestión de la Calidad

Incluye los procesos involucrados en planificar, dar seguimiento, controlar y garantizar que se cumpla con los requisitos de calidad del proyecto, lo que incluye:

- Planificar la Calidad

- Realizar el Aseguramiento de Calidad
- Realizar el Control de Calidad.

Gestión de los Recursos Humanos

Procesos involucrados en la planificación, adquisición, desarrollo y gestión del equipo del proyecto. Para ello es necesario:

- Desarrollar el Plan de Recursos Humanos
- Adquirir el Equipo del Proyecto
- Desarrollar el Equipo del Proyecto
- Gestionar el Equipo del Proyecto.

Gestión de la Comunicación

Identifica los procesos involucrados en garantizar que la generación, recopilación y distribución de la información del proyecto entre los diferentes involucrados sea adecuada. Esto incluye:

- Identificar a los Interesados
- Planificar las Comunicaciones
- Distribuir la Información
- Gestionar las Expectativas de los Interesados
- Informar el Desempeño

Gestión del Riesgo del Proyecto

Describe los procesos involucrados en la identificación, análisis y control de los riesgos del el proyecto:

- Planificar la Gestión de Riesgos
- Identificar los Riesgos
- Realizar Análisis Cualitativo de Riesgos
- Realizar Análisis Cuantitativo de Riesgos
- Planificar la Respuesta a los Riesgos
- Dar seguimiento y Controlar los Riesgos.

Gestión de Adquisiciones

Incluye los procesos involucrados en la compra o adquisición de productos, servicios o resultados para el proyecto.

- Planificar las Adquisiciones
- Efectuar las Adquisiciones
- Administrar las Adquisiciones

- Cerrar las Adquisiciones

No es imprescindible ejecutar todos los procesos en los proyectos, ya que el PMBOK es un conjunto de buenas prácticas que deben adaptarse a cada proyecto en función de la naturaleza de este, de la cultura de la organización y otras variables. En nuestro caso nos centraremos fundamentalmente en:

- Integración
- Alcance
- Tiempo
- Riesgos

En la siguiente figura [11] se muestran las dos dimensiones de un proyecto consideradas dentro del PMBok, la correspondiente a las áreas de conocimiento y la temporal, que será la que utilicemos en los siguientes apartados:

Figura 12- Dimensión temporal PMBok (extraído de [14])

5.2 PROCESOS DE DIRECCIÓN DE PROYECTOS

5.2.1 INICIO

En este grupo de procesos se incluyen las actividades de inicio formal de un proyecto. La iniciación del proyecto implica autorizar formalmente el proyecto y proveer de toda la información necesaria al gestor para iniciar el proyecto.

Recogemos en este punto los puntos relevantes considerados en el Acta de Constitución de proyecto:

Titulo y descripción del proyecto: "Implementación de una Interfaz Visual para la navegación de Big Semantic Data". Creación de un interfaz gráfico para representar y navegar a través de grafos HDT.

Caso de Negocio: este trabajo fin de grado se incluye dentro del proyecto RDF/HDT llevado a cabo por el grupo de investigación DataWeb Research, dentro del Departamento. de Informática de la UVA.

Requisitos iniciales:

1. Dibujar los elementos del formato HDT en modo grafo.
2. Permitir navegar al usuario dentro del grafo dibujado, actualizando el mismo con los hijos del nodo pulsado.
3. Integrar la interfaz visual dentro de la herramienta HDT it!

Entregables:

1. Ejecutable y código de esta nueva interfaz integrada dentro de la aplicación HDT it!
2. Documentación técnica y funcional
3. Memoria

Objetivos medibles:

1. Mejorar la interrelación de los usuarios con la herramienta, al permitir visualizar de una forma gráfica las búsquedas realizadas sobre los datos.

2. Tiempos: fecha de entrega Septiembre 2013.

Riesgos a alto nivel:

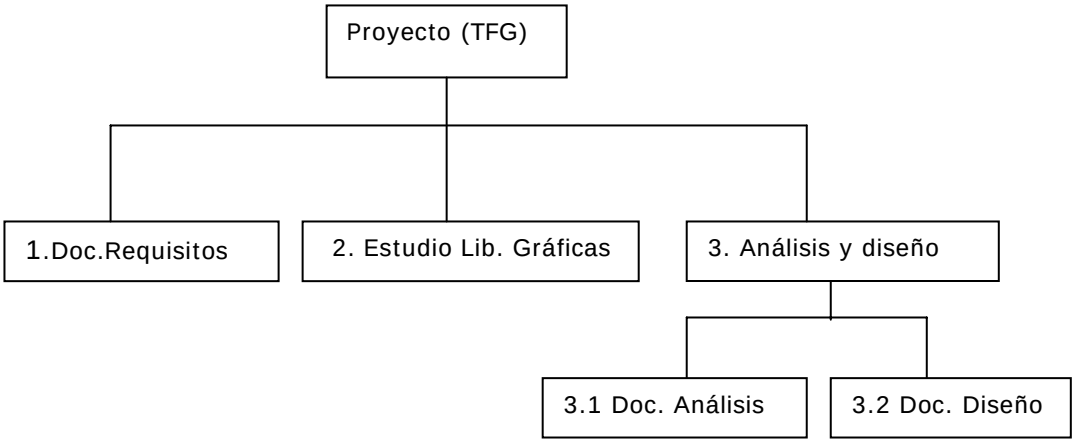
1. Dificultad en el cálculo de posiciones de los elementos de grafo a dibujar para que su diseño fuera agradable a los usuarios.

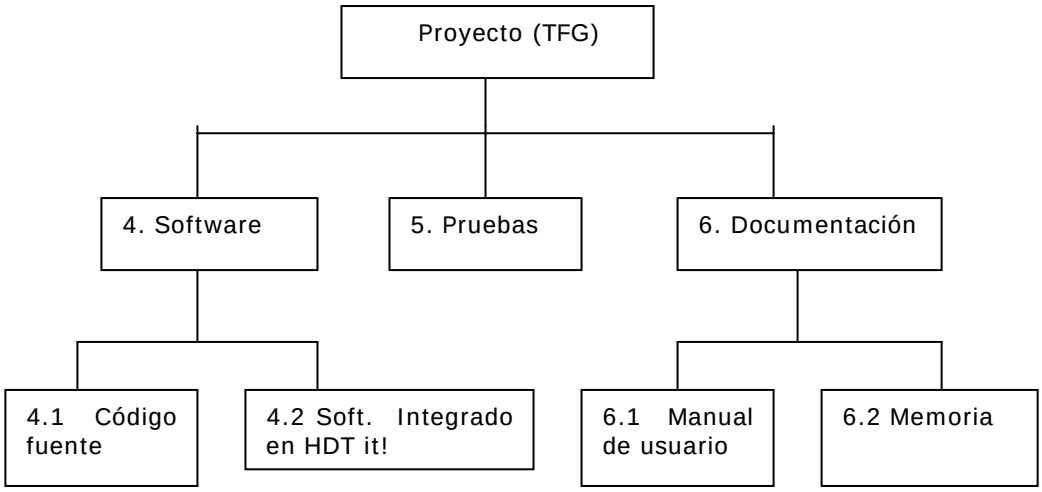
2. Integración en un proyecto ya en ejecución.

5.2.2 PLANIFICACIÓN

Durante la planificación se realizará un análisis detallado para saber si los objetivos marcados en el acta de constitución del proyecto se pueden lograr.

En esta fase se elaboró el Plan para la Dirección del Proyecto, que en nuestro caso se inicia con la definición preliminar del alcance, para lo que se definió una **EDT** (Estructura de Desglose de Tareas):





Adjuntamos detalle del Diccionario de la EDT:

1. Documento de Requisitos	
Descripción:	Lista de requisitos funcionales y no funcionales a implementar
Criterios de aceptación	Deberá recoger todos los requisitos y ser validado por los tutores de proyecto
Riesgos	Ninguno especialmente relevante
Duración aproximada	15 días
Hitos del cronograma	- Entrega de documento - Validación y aceptación del mismo
Fecha límite	- 15 de Marzo de 2013.
Notas	El detalle de este entregable se puede encontrar en el punto *"Descripción de requisitos"*

2. Estudio de librerías gráficas	
Descripción:	Resumen de las posibilidades de las librerías gráficas existentes.
Criterios de aceptación	Ajuste a los requisitos recogidos en el Documento de requisitos
Riesgos	Ninguno especialmente relevante
Duración aproximada	1 mes
Hitos del cronograma	- Entrega de documento de conclusiones - Validación de conclusiones
Fecha límite	- 15 de Abril de 2013
Notas	El detalle de este entregable se puede encontrar en el punto *"Interfaz gráfica para grafos HDT"*

3.1 Documento de análisis	
Descripción:	Detalle del análisis del proyecto siguiendo el Proceso Unificado de Modelado (RUP)
Criterios de aceptación	Deberá recoger todos el análisis realizado para el proyecto y ser validado por los tutores de proyecto
Riesgos	Ninguno especialmente relevante
Duración aproximada	7 días
Hitos del cronograma	- Entrega de documento de análisis - Validación

Fecha límite	- 30 de Abril de 2013
Notas	El detalle de este entregable se puede encontrar en el punto "*Análisis*".

3.2. Documento de diseño	
Descripción:	Detalle del diseño técnico del proyecto siguiendo el Proceso Unificado de Modelado (RUP)
Criterios de aceptación	Deberá recoger todos los puntos del diseño técnico y ser validado por los tutores de proyecto.
Riesgos	Ninguno especialmente relevante
Duración aproximada	15 días
Hitos del cronograma	- Entrega de documento de diseño - Validación
Fecha límite	- 15 de Mayo de 2013
Notas	El detalle de este entregable se puede encontrar en el punto "*Diseño*"

4. Código fuente (4.1) y software integrado en HDT it! (4.2)	
Descripción:	Código fuente (clases) y ejecutable
Criterios de aceptación	Verificar que el ejecutable correspondiente cumple los objetivos marcados en el proyecto
Riesgos	Riesgo 1: Uso de un entorno de desarrollo (Qt) desconocido y con poco soporte y documentación en la comunidad. Riesgo 2: Problemas derivados de la necesidad de

	integración de este proyecto dentro de la herramienta HDT it! ya desarrollada.
Duración aproximada	3 meses
Hitos del cronograma	- Entrega I: dibujar grafos sin integración con HDT – it! - Entrega II: dibujar grafos integrados con HDT – it! - Entrega final - Validación
Fecha límite	- 15 de Agosto de 2013
Notas	La obtención de estos entregables de detalla en el punto *"Ejecución del proyecto"*

5. Documento de resultado de las pruebas	
Descripción:	Documento de resultado de las pruebas funcionales finales
Criterios de aceptación	Deberá validar el cumplimiento de todos los requisitos especificados en el documento correspondientes
Riesgos	Ninguno especialmente relevante
Duración aproximada	Durante el desarrollo + 7 días para las pruebas y validación final
Hitos del cronograma	- Entrega de documento resumen de las pruebas - Validación
Fecha límite	- 22 de Agosto de 2013
Notas	El detalle de este entregable se puede encontrar en el punto "*Pruebas*"

6. 1 Manual de usuario	
Descripción:	Documento que resumen cómo se debe usar la nueva funcionalidad desarrollada para que cualquier usuario pueda utilizarla
Criterios de aceptación	Especificación clara y sencilla de la funcionalidad
Riesgos	Ninguno especialmente relevante
Duración aproximada	1 día
Hitos del cronograma	- Entrega de documento - Validación
Fecha límite	- 23 de Agosto de 2013
Notas	El detalle de este entregable se puede encontrar en el punto *"Manual de usuario"*

6.2 Memoria resumen	
Descripción:	Memoria de proyecto que resuma el trabajo realizado a lo largo del Trabajo Fin de Grado
Criterios de aceptación	Deberá resumir todos los pasos realizados y el trabajo que se ha llevado a cabo
Riesgos	Ninguno especialmente relevante
Duración aproximada	
Hitos del cronograma	- Entrega de Memoria
Fecha límite	- 30 de Agosto de 2013
Notas	

Pasaremos en los siguientes puntos a describir las **actividades identificadas** para cada uno de los entregables detectados:

1. Documento de requisitos
    a. Entrevistas con involucrados
    b. Elaboración del listado de requisitos
    c. Entrega (Hito)
    d. Validación (Hito)
2. Estudio de librerías gráficas
    a. Análisis inicial: recopilación de librerías existentes
    b. Estudio de ajuste a los requisitos
    c. Pruebas de dibujo de grafos
    d. Elaboración de conclusiones
    e. Entrega de conclusiones (hito)
    f. Validación (Hito)
3. Análisis (3.1) y Diseño (3.2)
    a. Elaboración de Modelo de Casos de uso
    b. Elaboración del Modelo de Dominio
    c. Elaboración de casos de prueba
    d. Entrega de análisis (hito)
    e. Validación de análisis (hito)
    f. Arquitectura del sistema
    g. Diagrama de secuencia

   h. Diagramas de clases

   i. Entrega de diseño (hito)

   j. Validación de diseño (hito)

4. Código fuente y ejecutable integrado en HDT it!

   a. Integración de Graphviz con librerías gráficas de Qt

   b. Desarrollo de interfaz de dibujo de grafos

   c. Entrega I (hito)

   d. Lectura del formato HDT, almacenamiento de información en estructuras de datos internas y representación en modo grafo

   e. Integración en HDT it!

   f. Entrega II (hito)

   g. Búsqueda según parámetros y mejoras gráficas

   h. Obtención de ejecutable

   i. Entrega final (hito)

5. Pruebas

   a. Ejecución de pruebas funcionales

   b. Documentar resultado de las pruebas.

6. Documentación

   a. Elaboración de manual de usuario

   b. Elaboración de memoria

Posteriormente se realizó un trabajo para **identificar los riesgos existentes y realizar un análisis cualitativo sobre los mismos.**

A la hora de valorar los riesgos, se tomará en cuenta la siguiente escala que analiza tanto la probabilidad de ocurrencia, como el impacto que supone el riesgo en el proyecto:

Impacto	Probabilidad		
	Alta	Media	Baja
Alto	Alto	Alto	Medio
Medio	Alto	Medio	Medio
Bajo	Medio	Bajo	Bajo

Según esta clasificación, las acciones a llevar a cabo serían:

Gravedad	Acción Preventiva	Acción Correctiva
Alto	X	X
Medio	X	
Bajo		

Riesgo 1:	Uso de un entorno de desarrollo (Qt) desconocido y con poco soporte y documentación en la comunidad			
Probabilidad	Alta	**Impacto**	Medio	
Acción preventiva	Búsqueda información de soporte			**Fecha:** Marzo 2013
Estado	Ocurrido			
Conclusiones sobre las acciones tomadas	Existe poca documentación actualizada, ya que parece que es un entorno no muy extendido. La búsqueda de información antes y durante fue más costosa que con cualquier otro entorno de desarrollo.			

Riesgo 2:	Problemas derivados de la necesidad de integración de este proyecto dentro de la herramienta HDT it! ya desarrollada.			
Probabilidad	Alta	**Impacto**	Alto	Variable afectada: tiempo
Acción preventiva	1. Realizar un estudio detallado previo de la herramienta **2.** Planificar con holgura por el posible impacto en tiempo de este riesgo			**Fecha:** Marzo 2013
Acción correctiva	1. Soporte de equipo de proyecto HDT it!			
Estado	Ocurrido	Impacto en planificación. El retraso no fue relevante sobre la planificación inicial, ya se planificó adecuadamente para tener en cuenta situación similares.		
Conclusiones sobre las acciones tomadas	Los problemas de la integración se debieron a que aunque el proyecto HDT it! estaba preparado para funcionar correctamente con Linux, Windows y Mac, la compilación de las librerías a usar no estaban realizas para utilizarse en modo Windows, lo que supuso un esfuerzo inicialmente no contemplado.			

Riesgo 3:	Problemas técnicos para poder dibujar los nodos y arcos del grafo de una forma agradable a la vista de un usuario.			
Probabilidad	Alta	**Impacto**	Alto	Variable afectada: calidad del proyecto
Acción preventiva	Realizar un estudio detallado de herramientas gráficas orientadas al dibujo de grafos y sus posibilidades de integración en el proyecto			**Fecha:** Abril/Mayo 2013

Acción correctiva	1. Implementación del código completo con todos los cálculos matemáticos dentro del proyecto	
Estado	Identificado	
Conclusiones sobre las acciones tomadas	El estudio exhaustivo encontró una librería que permitía el dibujo de los grafos HDT con "pocos" ajustes matemáticos posteriores sobre las posiciones iniciales obtenidas.	

Todos estos riesgos se han ido revisando periódicamente a lo largo de la duración del proyecto.

El **cronograma resultante de la planificación y estimación de tiempos de las tareas identificadas** se muestra en la imagen adjunta:

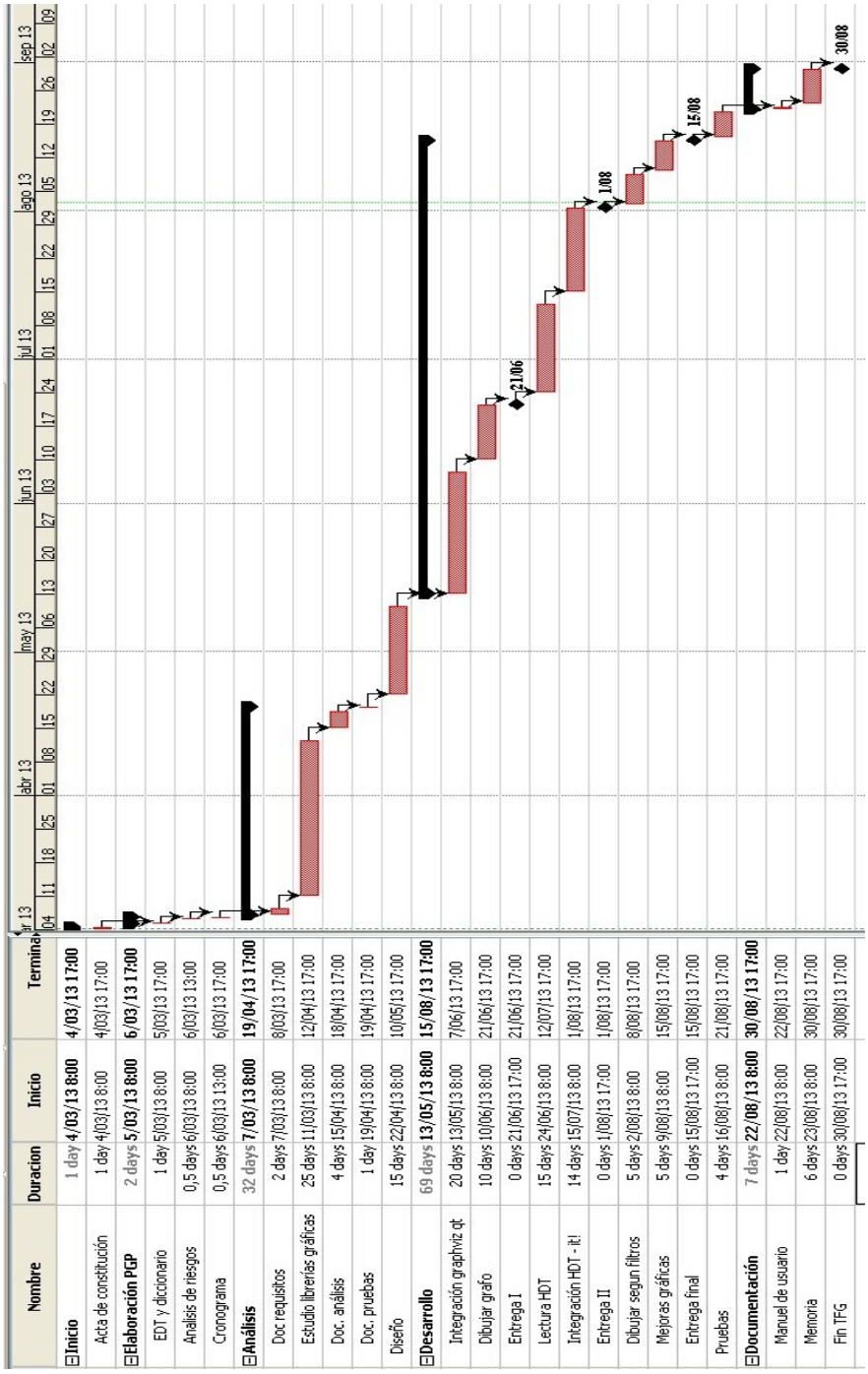


Figura 13- Planificación del proyecto

5.2.3 EJECUCIÓN

El propósito de esta fase es completar el trabajo definido durante la planificación, así como cumplir con los objetivos del proyecto. Resumiendo, se trata de obtener los entregables definidos dentro de los plazos (y coste, no considerado en este caso) y cumplir cualquier otro objetivo establecido para el proyecto.

El detalle de la ejecución se muestra en el punto "*Ejecución de proyecto*" de la presenta memoria.

5.2.4 SEGUIMIENTO Y CONTROL

Dar seguimiento y controlar significa medir el rendimiento del proyecto de acuerdo con el plan definido, y aprobar las solicitudes de cambio (no se han dado en el proyecto), incluyendo acciones correctivas de la reparación de defectos.

En el caso del presente proyecto se definieron hitos de control mensuales, donde se revisaron fundamentalmente las siguientes variables:

- % de avance en horas y plazos
- Estado de los Riesgos y acciones correctivas

Punto de control 1: *31/03/2013*

Estado	Este periodo	Periodo anterior	Comentarios
Estado medio		N/A	
Planificación		N/A	
Técnico		N/A	Problemas con la integración con Graphviz

Principales hechos durante este periodo
Elaboración de Documento de requisitos y validación
Iniciar estudio de librerías gráficas: grado de avance 40%
Actividades clave planificadas para el siguiente periodo
Cerrar estudio de librerías gráficas
Problemas y Necesidades Detectadas
La única alternativa a valorar es Graphviz. El resto no se ajustan a las especificaciones
Problemas con las pruebas iniciales de uso de Graphviz: requiere revisión de librerías cgrah y graph para determinar cuál se integra mejora para nuestras necesidades. Los métodos a utilizar son diferentes en función de la librería seleccionada.

Punto de control 2: *30/04/2013*

Estado	Este periodo	Periodo anterior	Comentarios
Estado medio			
Planificación			
Técnico			Integración OK

Principales hechos durante este periodo
Documento de análisis
Cierre de estudio de librerías gráficas: integración básica con Graphviz OK (se ha decidido utilizar la alternativa *cgraph*)

Actividades clave planificadas para el siguiente periodo
Documento de diseño
Inicio de fase de desarrollo
Problemas y Necesidades Detectadas
Integración Graphviz más costosa en horas que lo estimado inicialmente, no supone un desvío en la planificación.

Punto de control 3: *31/05/2013*

Estado	Este periodo	Periodo anterior	Comentarios
Estado medio			
Planificación			
Técnico			

Principales hechos durante este periodo
Documento de diseño cerrado
Inicio de fase de desarrollo
Actividades clave planificadas para el siguiente periodo
Avance desarrollo del 30%

Punto de control 4: *28/06/2013*

Estado	Este periodo	Periodo anterior	Comentarios
Estado medio			
Planificación			
Técnico			

Principales hechos durante este periodo
Avance de fase de desarrollo 20%. Previsto un 30%
Actividades clave planificadas para el siguiente periodo
Avance de fase de desarrollo 70%
Problemas y Necesidades Detectadas
Problemas con el uso de las librerías de HDT. La compilación para Windows no está documentada. El tiempo para obtener las librerías compiladas que se puedan utilizar en el proyecto está siendo elevado. Se concertará una reunión con los tutores para poder resolver los problemas de forma más rápida que vía mail.

Punto de control 4: *31/07/2013*

Estado	Este periodo	Periodo anterior	Comentarios
Estado medio			
Planificación			
Técnico			

Principales hechos durante este periodo
Avance de fase de desarrollo 65%. Previsto un 70%
Actividades clave planificadas para el siguiente periodo
Avance de fase de desarrollo 100%
Problemas y Necesidades Detectadas
Resueltos lo problemas con las librerías HDT.

Punto de control 4: *30/08/2013*

Estado	Este periodo	Periodo anterior	Comentarios
Estado medio			
Planificación			
Técnico			

Principales hechos durante este periodo
Fin de proyecto: Avance de fase de desarrollo 100%. Realización de pruebas y documentación asociada

5.2.5 CIERRE

Esta fase del proyecto se centra en la verificación final de ajuste a los requisitos iniciales, la obtención de la aceptación final por parte de los tutores y la redacción de las conclusiones y lecciones aprendidas. Toda esta información se refleja en el punto de “Conclusiones”.

6. EJECUCIÓN DEL PROYECTO

6.1 ANÁLISIS

6.1.1 DESCRIPCIÓN DE REQUISITOS

Adjunto se incluye el listado de requisitos contenidos dentro del Documento de requisitos:

Requisitos funcionales

RF-1: La aplicación permitirá obtener información gráfica, en forma de grafo (nodos y arcos) de un HDT seleccionado. Se utilizará como referencia la información de las ternas de los que consta dicho formato: sujeto, predicado, objeto. Prioridad: Alta.

RF–2: Se realizarán búsquedas sobre el HDT, teniendo en cuenta que las búsquedas tomarán como referencia los campos *Sujeto, Predicado y Objeto* que ya existen en la interfaz gráfica *HDT it!.* Prioridad: Alta.

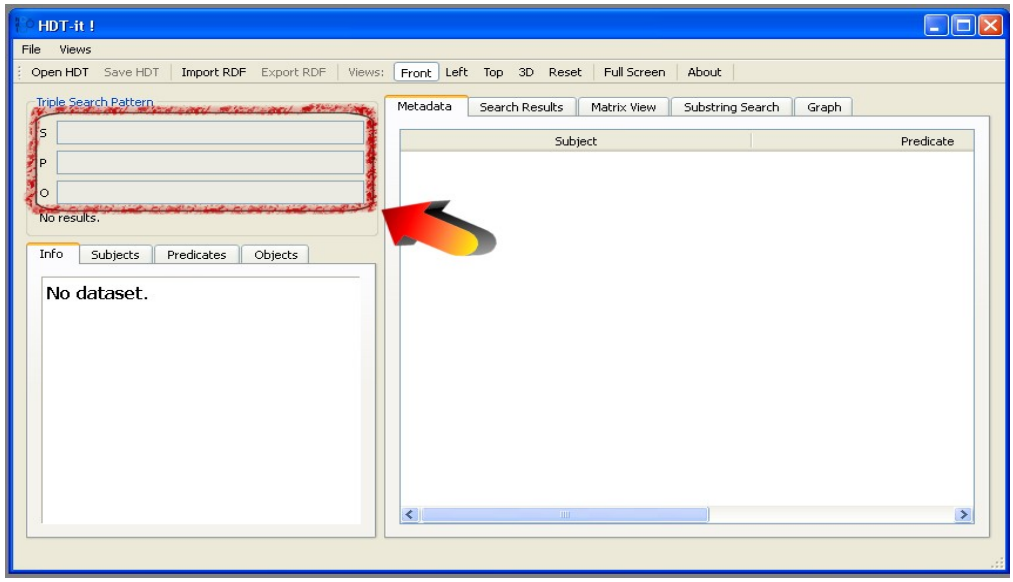


Figura 14 – Pantalla principal de HDT it! con filtros

RF-3: El usuario podrá realizar nuevas búsquedas pulsando en los nodos de los que está compuesto el grafo. Prioridad: Alta.

RF-04: Se podrá hacer *zoom* para consultar el detalle una parte del grafo. Prioridad: Alta.

RF-05: Se actualizará el grafo cuando el usuario modifique los valores de *Sujeto, Predicado y Objeto* que ya existen en la interfaz gráfica *HDT it!*

Requisitos no funcionales:

RNF-1: La herramienta se integrará dentro del proyecto HDT it!. Se incorporará una nueva pestaña denominada “Graph” donde se podrá visualizar y navegar a través del grafo resultante. Prioridad: Alta

RNF-2: Se mostrará la información de la URI del Sujeto y el Objeto en los nodos. Prioridad: Alta

RNF–3: Se mostrará la información de la URI del Predicado en el arco que une a los dos nodos relacionados. Prioridad: Alta

RNF-4: Se utilizará un código de colores para indicar al usuario si un nodo es susceptible de pulsarse para realizar una nueva búsqueda sobre él. Prioridad: Alta.

RNF-5: El entorno de desarrollo a utilizar será Qt 4.8.4.

6.1.2 MODELO DE CASOS DE USO

En este apartado se describirán los Casos de uso de la aplicación. Utilizamos los casos de uso para describir la utilización del sistema y cómo los usuarios interaccionan con nuestra aplicación. En los casos de uso se detallan paso a paso las acciones que realiza el sistema y que producen un resultado para un actor en particular.

Definición de actores:

Usuario**:** este actor representa al usuario del sistema, quien utiliza el sistema para consultar un HDT.

Diagrama de casos de uso

El objetivo de este punto se centra en representar los casos de uso existentes, las relaciones entre los mismos y los actores del sistema.

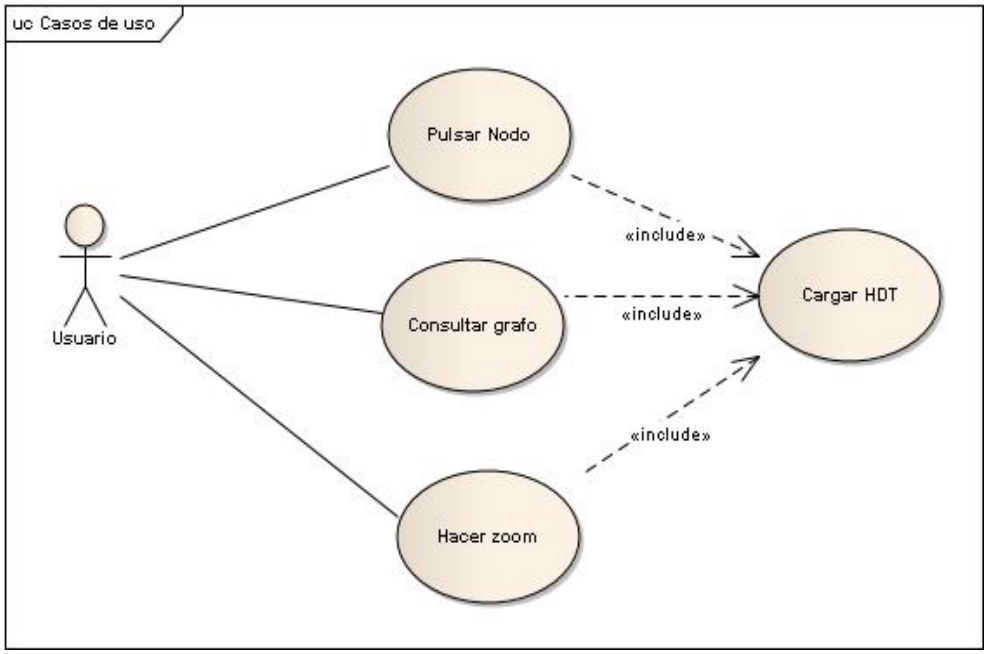


Figura 15 – Representación Diagrama de Casos de uso

CU – 01	**Cargar HDT**
Descripción	El sistema permitirá seleccionar un HDT
Requisitos asociados	RF-1
Precondiciones	El usuario habrá entrado en la aplicación HDT it!
Secuencia normal	1. El usuario selecciona un HDT 2. Si ya se ha abierto un HDT antes, el sistema borra el contenido de las estructuras de datos existentes. 3. El sistema inicializa las estructuras para el dibujo del grafo y muestra la pestaña “Graph” donde se dibujarán los grafos. 4. Si el HDT contiene más de 50 ternas (umbral configurable), el sistema filtra automáticamente por el primer sujeto encontrado y muestra el grafo correspondiente. 5. Si el HDT contiene menos de 50 ternas, se muestra el grafo completo con todos los nodos y arcos. 6. Si el usuario introduce filtros Sujeto, Objeto, Predicado, se ejecutará el caso de uso “Consultar grafo”. 7. Si el usuario pulsa un nodo, se ejecutará el caso de uso “Pulsar nodo”. 8. Si el usuario quiere acercar o alejar alguna zona del grafo, se ejecutará el caso de uso “Hacer zoom”
Excepciones	1.a Si el fichero HDT no tiene la estructura adecuada, se informará al usuario
Poscondición	Se mostrará la pestaña “Graph” donde se irán dibujando los grafos resultantes de las búsquedas en el HDT.

Caso de uso Consultar Grafo

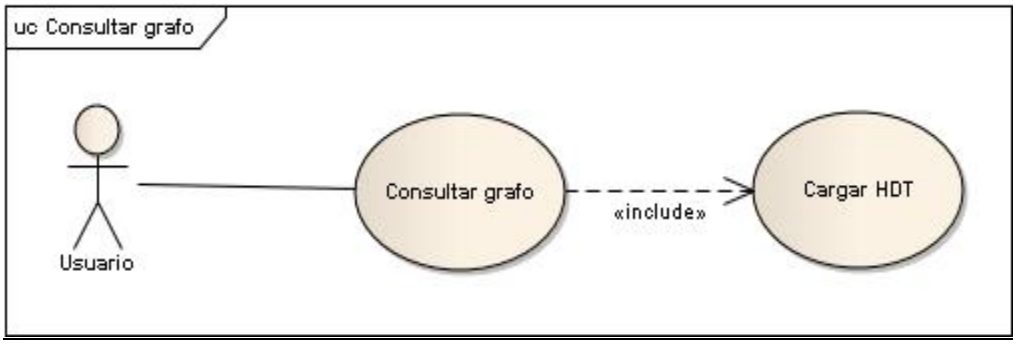


Figura 16 – Representación caso de uso "Consultar grafo"

CU – 02	**Consultar grafo**
Descripción	El sistema mostrará un gráfico con la información del HDT en formato grafo (nodos y arcos)
Requisitos asociados	RF-2 y RF-5
Precondiciones	El usuario habrá seleccionado un HDT a través de la interfaz de HDT it!
Secuencia normal	1. El usuario selecciona los valores de los filtros de Sujeto, Predicado y/o Objeto de la interfaz. 2. El sistema dibuja el grafo mostrado teniendo en cuenta los valores introducidos (la búsqueda se realiza a través de las librerías de HDT).
Excepciones	2.a. Si no se detecta información en la búsqueda, el sistema no mostrará ningún grafo. 2.b. Si los resultados son muy numerosos, se emite un mensaje de aviso y no se dibuja el grafo.

Poscondición	Se mostrará el grafo asociado a la búsqueda realizada o el mensaje de aviso adecuado

Caso de uso Pulsar nodo

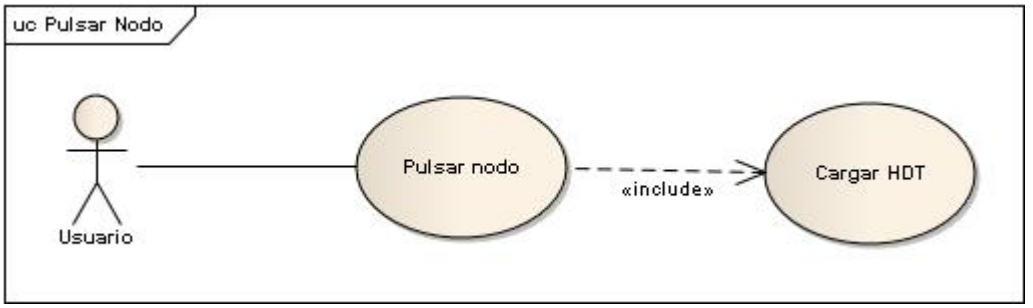


Figura 17 – Representación casos de uso "Pulsar Nodo"

CU – 03	**Pulsar nodo**
Descripción	El sistema mostrará un grafo actualizado con los nodos hijos del nodo pulsado
Requisitos asociados	RF-3
Precondiciones	El usuario habrá seleccionado un HDT a través de la interfaz de HDT it!
Secuencia normal	1. El usuario pulsa un nodo del grafo actual 2. El sistema actualiza el grafo actual con los nodos hijo del nodo pulsado (la búsqueda se realiza a través de las librerías de HDT)
Excepciones	1.a. Si no se detecta información en la búsqueda, el sistema mostrará el mismo grafo sin actualizar.
Poscondición	Se mostrará el grafo actualizado con los nodos hijos del nodo seleccionado.

Caso de uso Hacer zoom

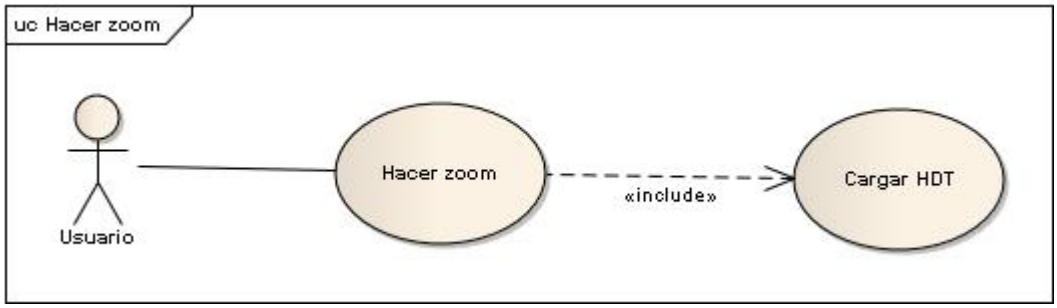


Figura 87 – Representación casos de uso "Hacer zoom"

CU – 04	**Hacer zoom**
Descripción	El sistema alejará/acercará el grafo que se muestra en la pantalla.
Requisitos asociados	RF-4
Precondiciones	El usuario habrá consultado un HDT y se habrá mostrado el grafo asociado en pantalla.
Secuencia normal	1. El usuario (con el ratón) solicita acercar o alejar una parte del grafo actual. 2. El sistema actualiza la escala del grafo, según la petición del usuario.
Poscondición	Se mostrará el grafo con el ajuste de escala solicitado

6.1.3 MODELO DE DOMINIO

El modelo del dominio muestra las clases conceptuales más significativas en un dominio de problema; es uno de los artefactos más importante creados durante la fase de análisis de un proyectos.

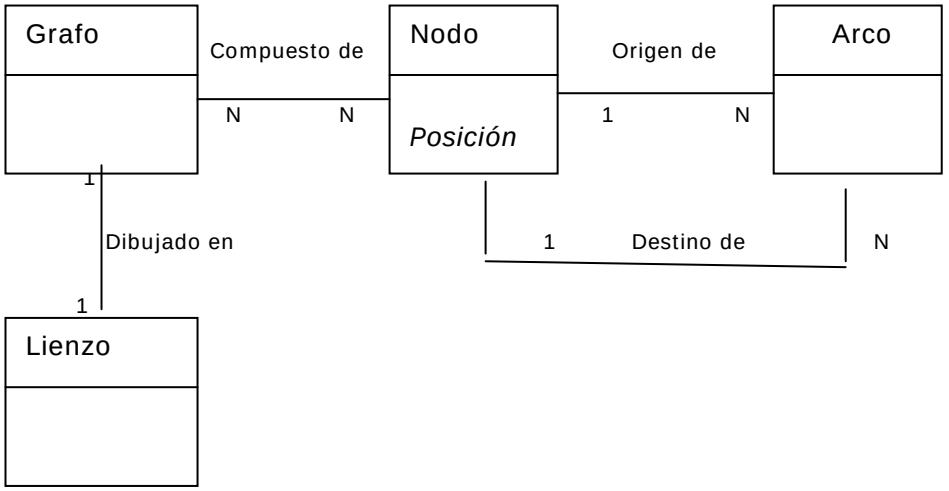


Figura 19 – Modelos de dominio

Descripción de las clases conceptuales:

- Lienzo: superficie sobre la que se dibujara el grafo.
- Grafo: conjunto de nodos y arcos que representarán la información del HDT.
- Nodo: parte fundamental del grafo HDT. Recogerá la información del sujeto o el objeto y su posición dentro del lienzo.
- Arco: relaciona dos nodos y representará la información del predicado que los une.

6.2 DISEÑO

6.2.1 ARQUITECTURA DEL SISTEMA

El proyecto se ha centrado en añadir nueva funcionalidad a la aplicación *HDT it*! ya existente. Esta herramienta se puede instalar en cualquier PC con sistema operativo Windows, Linux y/o iOs, es decir, se trata de una aplicación de escritorio que no utiliza almacenamiento en base de datos.

La mayor complejidad de su diseño se ha centrado en la algorítmica necesaria para poder asignar a cada nodo del grafo la posición más adecuada, teniendo en cuenta que también es necesario colocar los textos de nodos y arcos de la forma más legible posible.

6.2.2 DIAGRAMA DE CLASES

En la siguiente imagen mostramos el diagrama de clases correspondiente a la nueva funcionalidad añadida a HDT it!. Este diagrama permite visualizar las relaciones entre las diferentes clases involucradas en el sistema. Se muestran también los atributos y los métodos asociados a cada una de ellas.

Se han incluido algunas de las clases relevantes utilizadas de la librería de GraphViz para el cálculo de posiciones de los nodos.

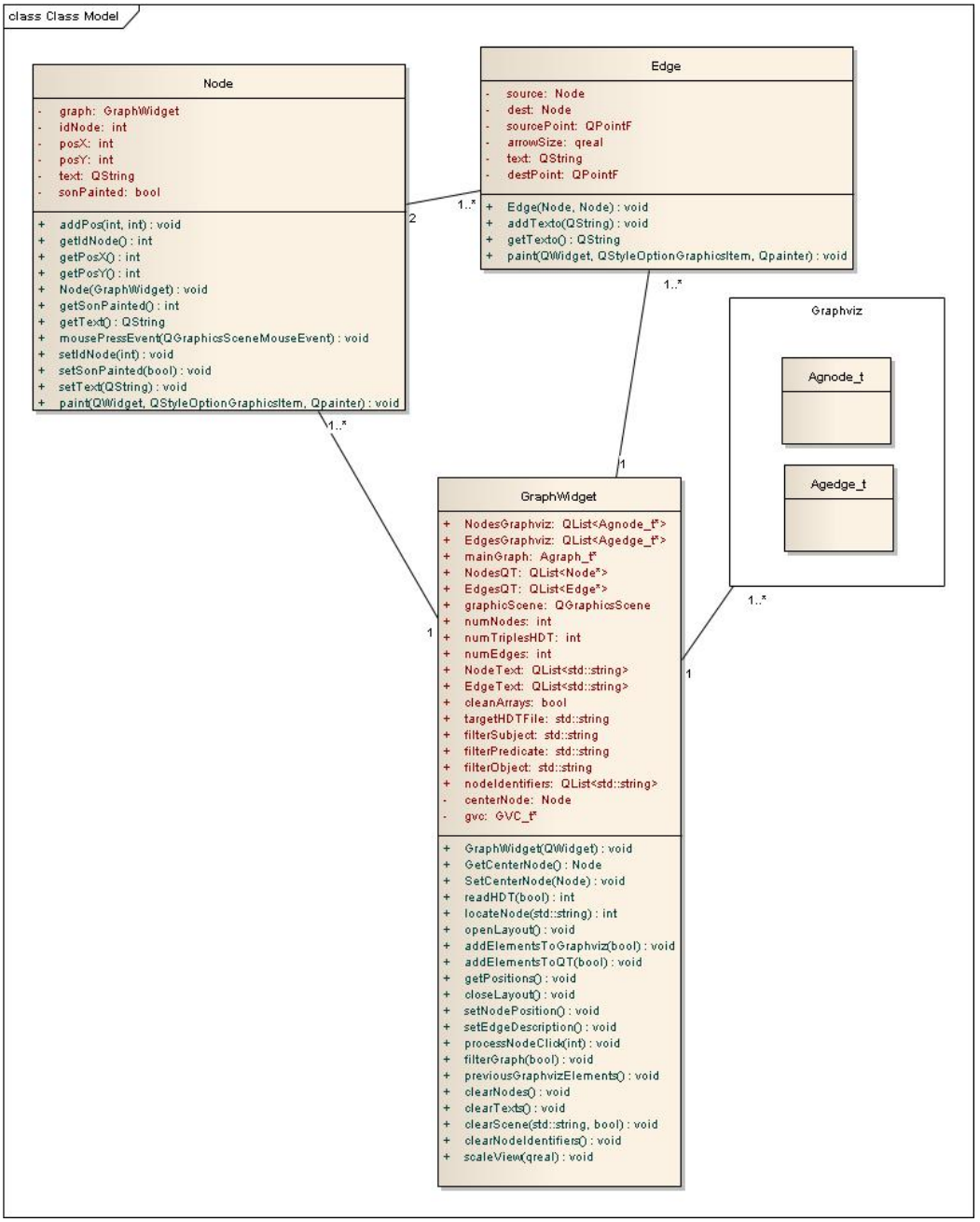


Figura 20- Diagrama de clases

6.2.3 EXPLICACIÓN DE LAS CLASES DE DISEÑO

1. NODE

Descripción:

Se trata de la clase que almacena la información de los nodos del grafo que se muestra en pantalla. Hereda de "QGraphicsItem" para poder realizar el control de los eventos en el grafo según las librerías de Qt.

Existirá una lista de objetos Node que almacenará de forma persistente (mientras se ejecuta la aplicación) la información de los nodos que se muestran en el grafo en pantalla. Se actualizará dicha lista con los nodos hijo del nodo pulsado por el usuario.

Si cuando el usuario filtra por Sujeto, Objeto y/o Predicado se obtiene un nuevo grafo, se actualizará la información de la lista de objetos Node, descartándose la correspondiente a la búsqueda anterior.

Variables miembro:

- Nombre: graph. Tipo: GraphWidget. Descripción: referencia al GraphWidget al que pertenece el nodo.
- Nombre: idNode. Tipo: int. Descripción: identificador del nodo. Permite controlar el nodo pulsado por el usuario.
- Nombre: posX. Tipo: int. Descripción: Valor en el eje x de la posición del nodo.
- Nombre: posY. Tipo: int. Descripción: Valor en el eje y de la posición del nodo.
- Nombre: text. Tipo: QString. Descripción: texto que se mostrará asociado al nodo. Se corresponde con el valor del Sujeto o el Objeto, según el nodo.
- Nombre: sonPainted. Tipo: bolean. Descripción: permite controlar si un nodo ya tiene o no sus hijos pintados en el grafo actual.

Métodos:

Node		
Descripción	Constructor de la clase	
Parámetros	*graphWidget: GraphWidget	GraphWidget al que pertenece el nodo

addPoss		
Descripción	Asigna las posiciones x e y obtenidas al nodo	
Parámetros	int x	Posición en el eje x
	int y	Posición en el eje y

getIdNode		
Descripción	Obtienen el identificador del nodo	
Salida	int	Identificador del nodo

setIdNode		
Descripción	Asigna identificador al nodo	
Parámetros	int	Identificador del nodo

getPosX		
Descripción	Devuelve la coordenada x de la posición del nodo	
Salida	int	Posición en el eje x

getPosY		
Descripción	Devuelve la coordenada y de la posición del nodo	
Salida	int	Posición en el eje y

getSonPainted		
Descripción	Devuelve el valor de la variable miembro sonPainted	
Salida	int	Valor de sonPainted

setSonPainted		
Descripción	Asigna valor de la variable miembro sonPainted	
Parámetro	int	Valor de sonPainted

getText		
Descripción	Devuelve el valor de la variable miembro text	
Salida	QString	Valor de text

setText		
Descripción	Asigna valor a la variable miembro text	
Parámetro	QString	Valor de text

mousePressEvent		
Descripción	Controla los eventos del ratón sobre los nodos. Se utiliza para lanzar la búsqueda de los nodos hijos del un nodo pulsado.	
Parámetros	*event: QGraphicsSceneMouseEvent	evento

paint		
Descripción	Responsable de pintar el nodo. Se llama automáticamente al hacer modificaciones en la scene.	
Parámetros	QPainter *painter	
	const QStyleOptionGraphicsItem *option	Opciones gráficas
	QWidget *	Widget al que pertenece el nodo

2. EDGE

Descripción:

Clase que almacena la información del arco que une dos nodos. Existirá una lista de objetos Edge que almacenará de forma persistente (mientras se ejecuta la aplicación) la información de los arcos que se muestran en el grafo en pantalla. Al pinchar sobre un nodo se actualizará dicha lista con los arcos que partan del nodo pulsado por el usuario y no se habían mostrado aún.

Si cuando el usuario filtra por Sujeto, Objeto y/o Predicado se obtiene un nuevo grafo, se actualizará la información de la lista de objetos Edge, descartándose la relativa a las interacciones anteriores del usuario.

Variables miembro:

- Nombre: source. Tipo: Node. Descripción: nodo origen del arco.
- Nombre: dest. Tipo: Node. Descripción: nodo destino del arco.
- Nombre sourcePoint. Tipo: QPointF. Descripción: punto de partida del arco.
- Nombre destPoint. Tipo: QPointF. Descripción: punto final del arco.
- Nombre: text. Tipo: QString. Descripción: texto asociado al arco. Se corresponde con la información del Predicado.
- Nombre: arrowSize. Tipo: qreal. Descripción: tamaño estándar de los arcos (se inicializa en el constructor y no se modifica).

Métodos:

Edge		
Descripción	Constructor de la clase	
Parámetros	Node *sourceNode	Nodo origen del arco
	Node *destNode	Nodo destino del arco

getText		
Descripción	Devuelve el valor de la variable miembro text	
Salida	QString	Valor de text

setText		
Descripción	Asigna valor a la variable miembro text	
Parámetro	QString	Valor de text

paint		
Descripción	Responsable de pintar el arco. Se llama automáticamente al hacer modificaciones en la scene.	
Parámetros	QPainter *painter	
	const QStyleOptionGraphicsItem *option	Opciones gráficas
	QWidget *	Widget al que pertenece el nodo

3. GraphWidget

Descripción:

Clase central de la nueva funcionalidad, hereda de QGraphicsView y es la responsable de representar en pantalla el grafo.

Variables miembro:

- Nombre: NodesGraphviz. Tipo: QList<Agnode_t*>. Descripción: lista de nodos de tipo Agnode_t que es utilizada por Graphviz para usar las funciones de la librería que permiten obtener la mejor posición de los nodos en un layout según el algoritmo "dot".

- Nombre: EdgesGraphviz. Tipo: QList< Agedge_t*>. Descripción: lista de arcos de tipo Agedge_t que es utilizada por Graphviz para obtener las posiciones de los arcos.

- Nombre: mainGraph. Tipo: Agraph_t. Descripción: grafo manejado por Graphviz.

- Nombre: NodesQT. Tipo: QList<Node*>. Descripción: lista de nodos de tipo Node que es utilizada para mantener internamente la información de los nodos del grafo mostrado en pantalla.

- Nombre: EdgesQT. Tipo: QList<Edge*>. Descripción: lista de arcos de tipo Edge que es utilizada para mantener internamente la información de los arcos del grafo mostrado en pantalla.

- Nombre: graphicScene. Tipo: QGraphicsScene. Descripción: "lienzo" que permite colocar y manejar gráficos 2D en Qt.

- Nombre: numNodes. Tipo: int. Descripción: número de nodos mostrados en pantalla.

- Nombre: numEdges. Tipo: int. Descripción: número de arcos existentes en el grafo mostrado en pantalla.

- Nombre: numTriplesHDT. Tipo: int. Descripción: número de ternas encontrados tras una búsqueda en un HDT.

- Nombre: NodeText. Tipo: QList<std::string>. Descripción: textos asociados a los nodos, incluye tanto los Sujetos, como los Objetos de cada terna.

- Nombre: EdgeText. Tipo: QList<std::string>. Descripción: textos asociados a los arcos, incluye los Predicados de cada terna.

- Nombre: cleanArrays. Tipo: boolean. Descripción: permite limpiar estructuras de memoria cuando se abre un HDT nuevo.

- Nombre: targetHDTFile. Tipo: std::string. Descripción: contiene la ruta al fichero HDT que se ha abierto a través de la aplicación.

- Nombre: filterSubject. Tipo: std::string. Descripción: texto introducido por el usuario para filtrar en el grafo por el campo Sujeto.

- Nombre: filterSubject. Tipo: std::string. Descripción: texto introducido por el usuario para filtrar en el grafo por el campo Sujeto.

- Nombre: filterPredicate. Tipo: std::string. Descripción: texto introducido por el usuario para filtrar en el grafo por el campo Predicado.

- Nombre: filterObject. Tipo: std::string. Descripción: texto introducido por el usuario para filtrar en el grafo por el campo Objeto.

- Nombre: nodeIdentifiers. Tipo: QList<std::string>. Descripción: Contiene la lista de identificadores ÚNICOS de los nodos, que permiten saber el número total de estos en el grafo, ya que en la búsqueda a través de las librerías de HDT se obtienen las ternas, pero no los nodos diferentes que existen (el mismo nodo puede estar en muchas ternas como Objeto o como Sujeto).

- Nombre: centerNode. Tipo: Node. Descripción: nodo central del grafo (necesario en QT, aunque no tiene efecto en la representación gráfica).

- Nombre: gvc. Tipo: GVC_t. Descripción: lienzo utilizado por Graphviz.

Métodos:

GraphWidget		
Descripción	Constructor de la clase	
Parámetros	QWidget*w	Widget al que pertenecerá el GraphWidget

GetCenterNode		
Descripción	Devuelve el valor de la variable miembro centerNode	
Salida	Node	Nodo central del grafo

SetCenterNode		
Descripción	Asigna valor a la variable miembro centerNode	
Parámetro	Node	Nodo central del grafo

readHDT		
Descripción	Realiza una búsqueda con los filtros introducidos sobre un HDT cargado	
Parámetro	bool deleteOldTexts	Indica si es necesario eliminar los textos de la scene tras una nueva búsqueda en un HDT.
Salida	int	Permite conocer si se han encontrado o no resultados en la búsqueda realizada. El resultado 0 indica que no hay datos, el resultado -1 indica que hay demasiados para representarlos y el resultado positivo indica las ternas encontradas.

locateNode		
Descripción	Permite conocer en qué posición de la lista de nodos se encuentra un nodo a partir de su texto.	
Parámetro	std::string nodeText	Texto asociado al nodo
Salida	int	Posición ocupada por el nodo con el texto pasado por parámetro

openLayout		
Descripción	Para poder utilizar las funciones de Graphviz es necesario tener creado el lienzo, que se inicializará indicando el tipo de grafo a crear, en nuestro caso Agdirected (grafo “dirigido”).	

addElementsToGraphviz		
Descripción	Almacena la información de nodos y arcos que se enviará a las estructuras de datos que requieren las librerías de Graphviz.	
Parámetro	boolean fatherNode	Dependiendo de su valor, se actualizará o no su información con los nodos ya existentes en las anteriores búsquedas.

addElementsToQT		
Descripción	Traslada a los objetos Node y Edge las posiciones calculadas por Graphviz y otros datos, creando los nodos y arcos definitivos que formarán parte del grafo y asignándoles identificadores únicos que servirán para expandir los hijos de un nodo determinado.	
Parámetro	boolean fatherNode	Dependiendo de su valor, se actualizará o no su información con los nodos ya existentes en las anteriores búsquedas.

getPositions	
Descripción	Obtiene las posiciones de cada nodo utilizando las funciones de las librerías de Graphviz

closeLayout	
Descripción	Permite eliminar el “lienzo” utilizado por Graphviz, las estructuras NodesGraphviz, EdgesGraphviz y graphicScene.

setNodePosition	
Descripción	Realiza ajustes geométricos complejos a partir de las posiciones iniciales devueltas por las librerías de Graphviz, para conseguir que los nodos tengan una distribución más clara en el grafo y que los textos que se muestran queden centrados bajo la imagen del nodo.

setEdgePosicion	
Descripción	Realiza ajustes geométricos complejos para los textos de los arcos. No es necesario calcular la posición de los arcos, ya que viene determinada por la que se ha adjudicado a los nodos, pero la colocación del texto debe tener en cuenta todos los textos colocados en los arcos anteriores para impedir (en lo posible) que se solapen y resulten ilegibles. Esta información sobre los textos se mantiene dinámicamente durante toda la vida del grafo.

processNodeClick		
Descripción	Permite la búsqueda de los hijos del nodo seleccionado por el usuario, es decir, filtra por sujeto dentro del HDT en el grafo actual. Si se encuentran datos en la búsqueda, se añadirán los hijos a las estructuras existentes para volver a calcular las posiciones del grafo resultante. Será necesario actualizar las posiciones de los textos de los nodos y los arcos del grafo.	
Parámetro	int IdNodeClick	Identificador del nodo pulsado por el usuario

filterGraph		
Descripción	Es el método principal para dibujar un grafo, desde él se invocan prácticamente el resto de métodos. En resumen, abre e inicializa las estructuras necesarias, busca en el HDT, obtiene las posiciones de los nodos a través de Graphviz y calcula las posiciones más adecuadas para los textos de los nodos y los arcos.	
Parámetro	bool firstPainting	Indica si el grafo se pinta por primera vez

previousGraphvizElements	
Descripción	Añade a las estructuras NodesGraphviz y EdgesGraphviz la información de los nodos Qt que ya se están mostrando en pantalla, para que también se puedan tener en cuenta a la hora de recalcular las posiciones de los nodos con Graphviz.

clearNodes	
Descripción	Elimina la información contenida en las estructuras NodesQT y EdgesQT.

clearTexts	
Descripción	Elimina la información contenida en las estructuras NodeText y EdgeText.

clearScene		
Descripción	Elimina la información contenida en la scene	
Parámetro	boolean warningScene	Especifica si la escena que se va a crear es para dibujo de grafos o para alertas.
Parámetro	std::string message	Mensaje a mostrar cuando la escena es de alerta

clearNodeIdentifiers	
Descripción	Elimina la información contenida en la estructura nodeIdentifiers

scaleView		
Descripción	Ajusta la escala del grafo (acercando o alejando el zoom) según lo solicitado por el usuario.	
Parámetro	qreal scaleFactor	Escala
Salida	GraphWidget	Gráfico actualizado

6.2.4 DIAGRAMAS DE SECUENCIA

En los siguientes puntos realizaremos la especificación de los casos de uso a través de los diferentes diagramas de secuencia.

CU – 01	**Cargar HDT**
Descripción	El sistema permitirá seleccionar un HDT
Precondiciones	El usuario habrá entrado en la aplicación HDT it!
Secuencia normal	1. El usuario selecciona un HDT 2. Si ya se ha abierto un HDT antes, el sistema borra el contenido de las estructuras de datos existentes. 3. El sistema inicializa las estructuras para el dibujo del grafo y muestra la pestaña “Graph” donde se dibujarán los grafos. 4. Si el HDT contiene más de x ternas (umbral configurable), el sistema filtra automáticamente por el primer sujeto encontrado y muestra el grafo correspondiente. 5. Si el HDT contiene menos de x ternas, se muestra el grafo con todos lo nodos y arcos.
Excepciones	1.a Si el fichero HDT no tiene la estructura adecuada, se informará al usuario
Poscondición	Se mostrará la pestaña “Graph” donde se irán dibujando los grafos resultantes de las búsquedas en el HDT

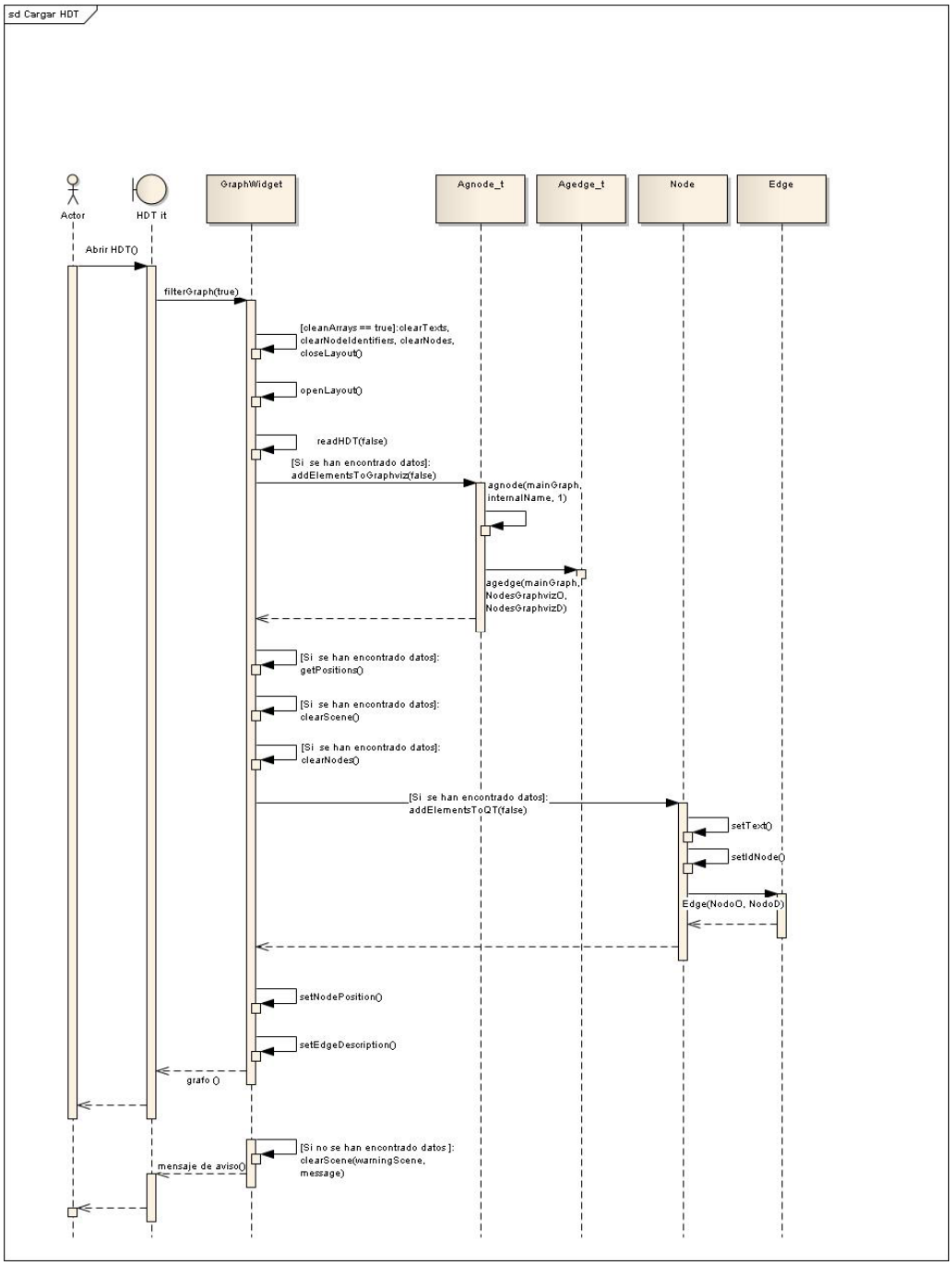


Figura 21- Diagrama de secuencia "Cargar un HDT"

CU – 02	**Consultar grafo**
Descripción	El sistema mostrará un grafico con la información del HDT en formato grafo (nodos y arcos)
Precondiciones	El usuario habrá seleccionado un HDT a través de la interfaz de HDT it!
Secuencia normal	1. El usuario selecciona los valores de los filtros de Sujeto, Predicado y/o Objeto de la interfaz 2. El sistema dibuja el grafo mostrado teniendo en cuenta los valores introducidos (la búsqueda se realiza a través de las librerías de HDT)
Excepciones	2.a. Si no se detecta información en la búsqueda, el sistema no mostrará ningún grafo. 2.b. Si los resultados son muy numerosos, se emite un mensaje de aviso y no se dibuja el grafo.
Poscondición	Se mostrará el grafo asociado a la búsqueda realizada

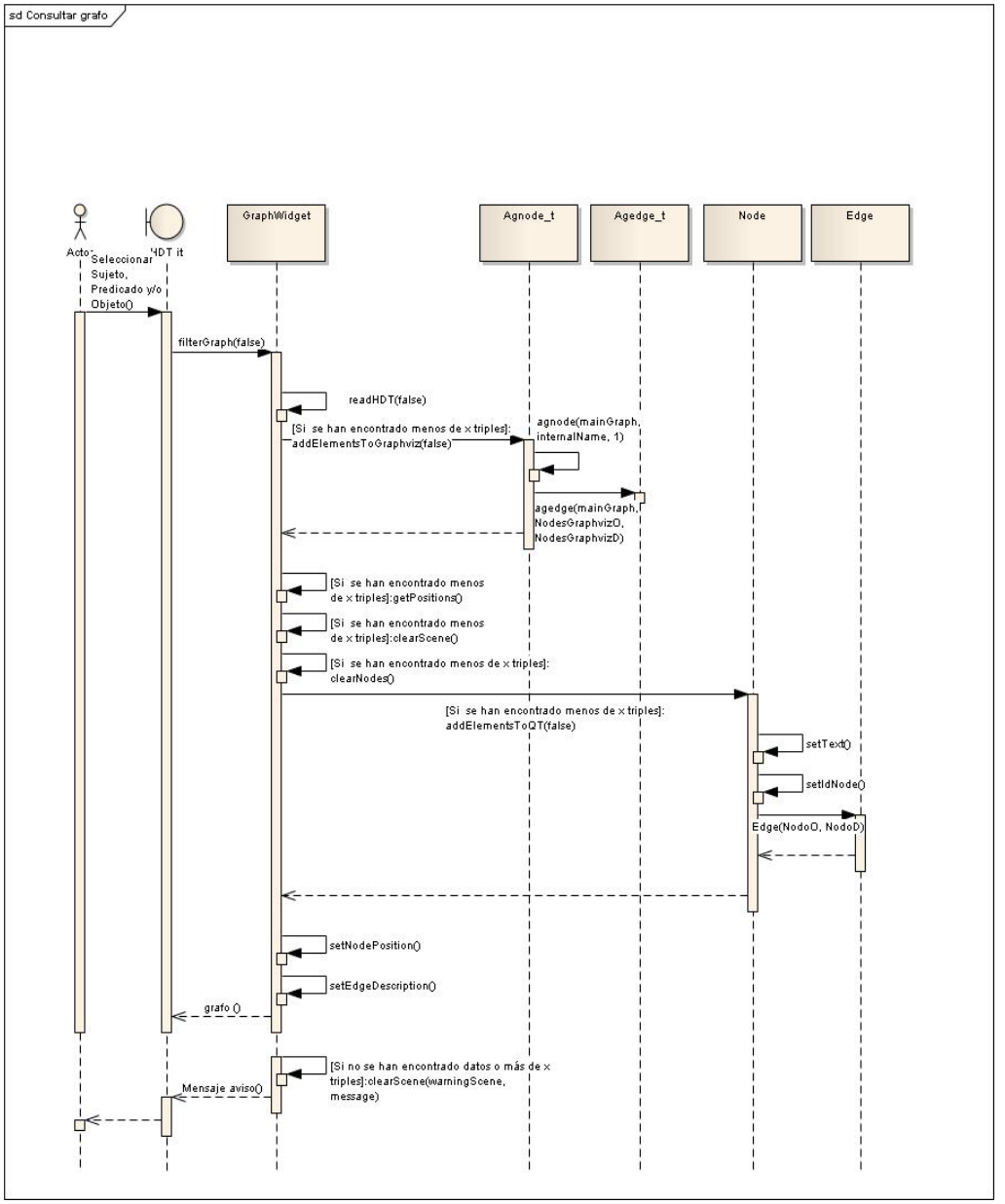


Figura 22- Diagrama de secuencia "Consultar grafo"

CU – 03	**Pulsar nodo**
Descripción	El sistema mostrará un grafo actualizado con los nodos hijos del nodo pulsado
Precondiciones	El usuario habrá consultado un HDT y se habrá mostrado el grafo asociado en pantalla.
Secuencia normal	1. El usuario pulsa un nodo del grafo actual 2. El sistema actualiza el grafo actual con los nodos hijo del nodo pulsado (la búsqueda se realiza a través de las librerías de HDT)
Excepciones	1.a. Si no se detecta información en la búsqueda, el sistema mostrará el mismo grafo sin actualizar.
Poscondición	Se mostrará el grafo actualizado con los nodos hijos del nodo seleccionado.

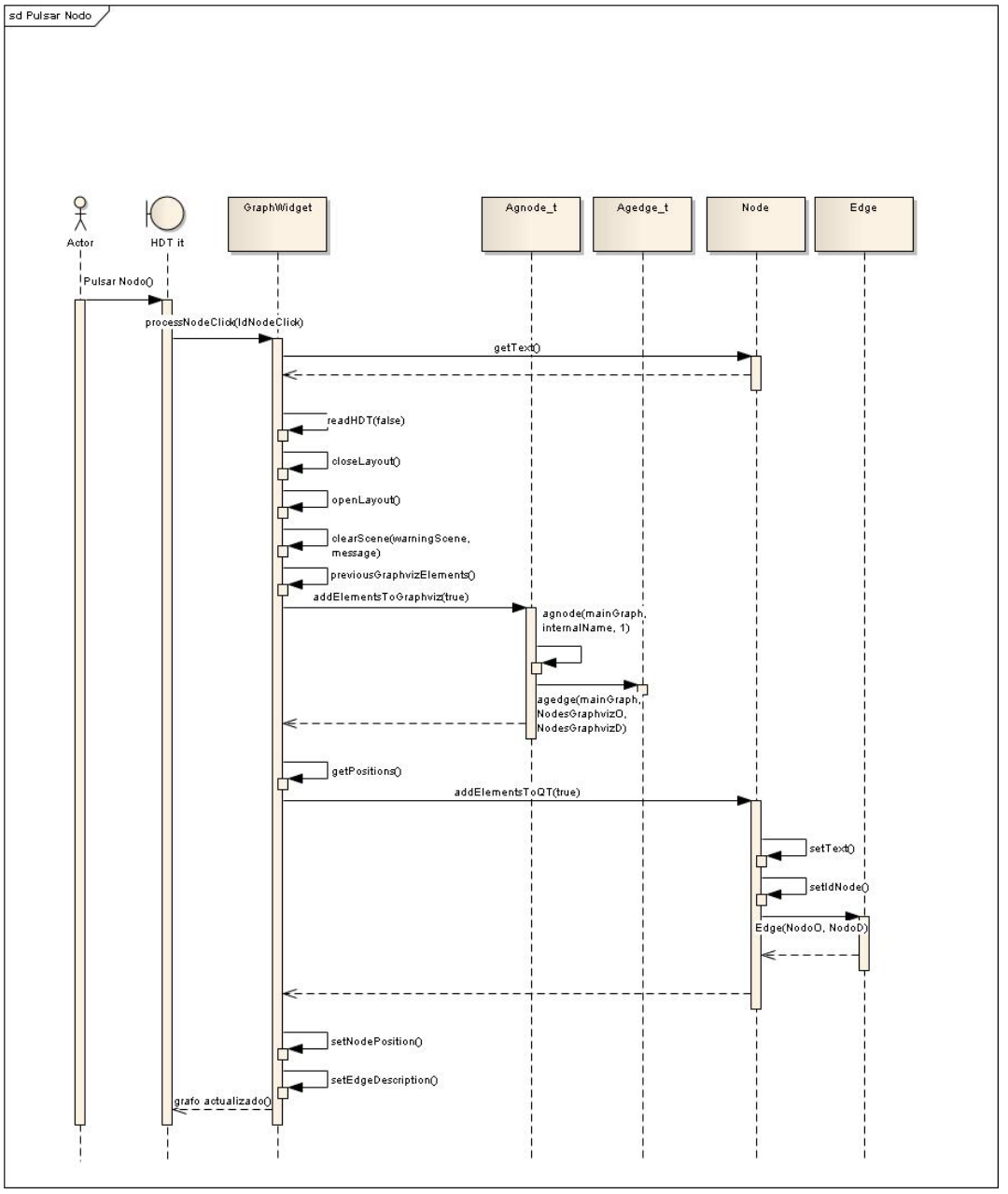


Figura 23- Diagrama de secuencia "Pulsar nodo"

CU – 04	**Hacer zoom**
Descripción	El sistema alejará/acercará el grafo que se muestra en la pantalla
Precondiciones	El usuario habrá consultado un HDT y se habrá mostrado el grafo asociado en pantalla.
Secuencia normal	1. El usuario (con el ratón) solicita acercar o alejar una parte del grafo actual 2. El sistema actualiza la escala del grafo, según la petición del usuario.
Poscondición	Se mostrará el grafo con el ajuste de escala solicitado

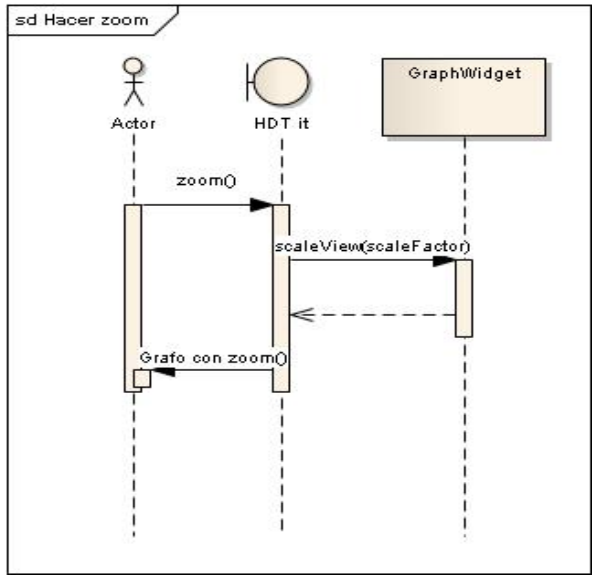


Figura 24- Diagrama de secuencia "Hacer zoom"

6.3 DESARROLLO

6.3.1 ENTORNO DE DESARROLLO: QT

Qt[16] es un *framework* para el desarrollo de aplicaciones multiplataforma creado por la compañia Trolltech y que actualmente es propiedad de Nokia, la función más conocida de Qt es la de la creación de interfaces de usuario, sin embargo no se limita a esto, ya que también provee varias clases para facilitar ciertas tareas de programación como el manejo de *sockets*, soporte para programación multihilo, comunicación con bases de datos, manejo de cadenas de caracteres, entre otras.

Qt utiliza C++ de manera nativa, pero ofrece soporte para otros lenguajes como Python mediante PyQt, Java mediante QtJambi, o C# mediante Qyoto. Qt es un *framework* muy poderoso, comparable con Swing de Java o .NET de Microsoft, además ofrece una suite de aplicaciones para facilitar y agilizar las tareas de desarrollo, las aplicaciones que componen esta suite son:

- **Qt Assistant**: Herramienta para visualizar la documentación oficial de Qt.
- **Qt Designer**: Herramienta WYSIWYG para crear interfaces de usuario.
- **Qt Linguist**: Herramienta para la traducción de aplicaciones.
- **Qt Creator**: IDE para el lenguaje C++, pero especialmente diseñado para Qt, integra las primeras dos herramientas mencionadas.

Qt está disponible bajo las siguientes licencias [17]:

- Qt GNU GPL v. 3.0: Utilizada para el desarrollo de aplicaciones de código abierto. Si se realizan cambios al código fuente de Qt, estos tienen la obligación de liberarse.

- Qt GNU LGPL v.2.1: Permite el desarrollo de aplicaciones privativas bajo ciertas restricciones. Si se realizan cambios al código fuente de Qt, estos tienen la obligación de liberarse.

- Commercial: Es la única licencia con costo, es utilizada para el desarrollo de aplicaciones propietarias, incluye soporte y la posibilidad de liberar las aplicaciones desarrolladas bajo cualquier licencia. Si se realizan cambios al código fuente de Qt, estos no tienen la obligación de ser compartidos.

El propósito de Qt es permitir a los desarrolladores construir aplicaciones multiplataforma a partir de una misma base de código de manera rápida y sencilla. Una aplicación Qt es una aplicación nativa, por la tanto se ve y se siente como tal. Qt provee un API sencilla y divertida de utilizar, permite que los desarrolladores tengan una alta productividad y ofrece herramientas potentes y sencillas.

Actualmente las plataformas de escritorio sobre las cuales las aplicaciones creadas con Qt funcionan son, Windows, Mac OS X y Linux/Unix X11. También es posible ejecutar aplicaciones Qt en plataformas embebidas Windows CE, Symbian, Maemo/MeeGo, Linux Embebido.

Existen versiones del kit de desarrollo de software para las tres principales plataformas de escritorio: Windows, Mac OS X y Linux/Unix X11.

6.3.2 ALGORITMOS PRINCIPALES

A lo largo del desarrollo del proyecto se han encontrado algunos problemas cuya solución requiere algoritmos más o menos complejos, que se detallan a continuación. Se indica también el método o métodos que implementan las operaciones descritas:

A.- Lectura de las ternas desde el fichero HDT

La recuperación de información desde el HDT es en principio sencilla, ya que la aplicación HDT it! ofrece una función de búsqueda que admite como parámetro los datos de filtro y devuelve un iterador con todas las ternas que han cumplido las condiciones. Esta operación de lectura se ha ido refinando para tener en cuenta varios factores más:

- Debe decidirse si se borran las ternas leídas previamente, filtrando nuevo desde cero, o en cambio se añaden los datos de la nueva búsqueda a los que teníamos ya cargado, por ejemplo al expandirse los hijos de un nodo cuando se pulsa en él.
- En la apertura inicial de ficheros grandes, deben hacerse dos lecturas del HDT. La primera es un simple “vistazo” para determinar que el fichero supera la cantidad de ternas que permitimos renderizar de un solo golpe. Si el fichero es lo bastante pequeño, la segunda lectura lo cargará en su totalidad, pero si supera un tamaño configurable, se filtrará automáticamente la información cargada utilizando el sujeto de la primera terna encontrada.

- Por las mismas razones que en el punto anterior, si los resultados obtenidos tras un filtrado superan cierta cantidad de ternas, se avisa al usuario de que debe refinar su búsqueda. Ni los algoritmos ni las estructuras de memoria tienen problema en trabajar con conjuntos de miles de ternas al mismo tiempo, pero el grafo obtenido resulta ilegible por razones de simple espacio.

Véase int GraphWidget::readHDT(bool deleteOldTexts)

B.- Carga de las estructuras de datos de Graphviz

Para representar un grafo son necesarios dos conjuntos de estructuras de datos: los relacionados con las librerías Graphviz y los que se encargan de enlazar con las clases que ofrece el entorno QT. La operación que tratamos en este punto utiliza el primer conjunto, y el objetivo es convertir la información de las ternas extraídas del HDT en una serie de nodos y arcos que Graphviz pueda entender. Como resultado obtendremos las coordenadas **iniciales** donde dibujar estos elementos, y el algoritmo tiene tres grandes fases.

- Se recopilan todos los sujetos y objetos contenidos en las ternas y se crea una lista única de nombres de nodos. Esta criba de nombres repetidos nos deja en memoria una estructura conteniendo todos los nodos reales que va a haber en el grafo, en posiciones ordenadas.
- A partir de la lista única de nombres del paso anterior, se crean objetos nodo de Graphviz y se asignan sus nombres internos.
- Finalmente, utilizando tanto la información de las ternas como las estructuras de los dos pasos anteriores, se crean los objetos arco de Graphviz. Para establecer la relación entre cada arco y los dos nodos que une, se busca en la lista única de nodos la posición de cada sujeto y objeto, y una vez conocidas las posiciones se enlaza el arco actual con los nodos que corresponden a esas posiciones.

Hay dos variantes de este proceso. La primera está enfocada a cargar la información tras la carga de un fichero externo o el filtrado de este mediante la interfaz de HDT it!, mientras que la segunda realiza el trabajo equivalente cuando se ha pinchado en un nodo para agregar sus hijos al grafo. Los dos casos se distinguen por un parámetro que indica si hay o no un “nodo padre”, y en ambos casos se obtienen las estructuras de datos necesarias para que Graphviz pueda asignar sus coordenadas

Véase: void GraphWidget::addElementsToGraphviz(boolean fatherNode)

C.- Carga de las estructuras de datos de QT

Este algoritmo es similar al anterior, ya que hay un paralelismo entre el juego de arcos y nodos que usa Graphviz y el juego de arcos y nodos que hay que proporcionar más adelante a QT para dibujar el grafo en el objeto escena (llamado a veces "lienzo"). Dejando aparte que los objetos que se crean son de distintas clases, las diferencias en cuanto a procesamiento son dos:

- Ya no es necesario crear la lista única de nombres de nodos, porque se puede reutilizar la del procedimiento anterior.
- La forma de relacionar arcos con sus nodos es análoga, ya que es necesario recuperar la posición que los nodos han obtenido en la lista única antes de enlazarlos con el arco correspondiente siguiendo la estructura de las ternas, pero se aprovecha este proceso para marcar los nodos que ya tienen pintados sus hijos, y que por tanto no serán pinchables cuando el usuario interactúe con el grafo.

Véase: void GraphWidget::addElementsToQT(boolean fatherNode)

D.- Asignación de posiciones para los nodos

Aunque el trabajo que hacen las librerías de Graphviz es imprescindible, ya que "ven" el grafo en su conjunto y deciden su distribución en el lienzo, las posiciones iniciales necesitan bastantes correcciones si queremos tener un aspecto gráfico legible y útil:

- Es necesario calcular la máxima diferencia que puede darse en el eje de ordenadas, entre el nodo al que Graphviz ha asignado la posición más alta y el que ha quedado colocado más abajo. Esto nos da una idea exacta de la altura del grafo y, utilizando las constantes que indican el tamaño del lienzo, nos permite centrarlo verticalmente.
- Aunque según su documentación el grafo debería crearse en sentido descendente, las librarías Graphviz lo crean en sentido ascendente, por lo que se ha tenido que "dar la vuelta" a esas coordenadas utilizando un complemento aritmético.
- Para que el nombre del nodo quede siempre centrado bajo el círculo pinchable que lo representa, se introduce un desvío a la izquierda cuya cuantía en píxeles depende del tamaño en caracteres de ese nombre. Los nombres de sujeto/objeto se truncan al llegar a cierto tamaño (se indica si están truncados con puntos suspensivos), y los nodos con un mismo padre se separan a una distancia fija para evitar en lo posible que estos nombres se superpongan.

Véase: void GraphWidget::setNodePosition()

E.- Asignación de posiciones para los textos de los predicados

Finalmente, el algoritmo más complejo que ha tenido que implementarse corresponde a una tarea aparentemente trivial, pero que no lo es en absoluto: la colocación en el lienzo gráfico de los textos correspondientes a los arcos (predicados).

Para que el grafo resulte mínimamente legible hay que intentar que los predicados no se solapen entre sí, ni con los nombres de sujeto u objeto, y esto es especialmente difícil sabiendo que la línea del arco va a dibujarse donde las librerías de Graphviz decidan colocarla.

Además, así como para los textos de los nodos se puede llegar a una solución de compromiso (el nombre se coloca siempre bajo el nodo, y se truncan nombres largos y se separan nodos hermanos entre sí para que los textos no se solapen), con los predicados hay una casuística mucho más complicada.

La solución adoptada, que no es perfecta pero permite leer los predicados en la mayoría de los casos, utiliza la siguiente filosofía:

- Como valor inicial, se intenta colocar el predicado a una altura intermedia entre el nodo objeto y el nodo sujeto. Con esto se evita, al menos en general, que los textos de los arcos choquen con textos de nodos, ya que Graphviz coloca los nodos siguiendo un esquema “generacional” (una misma altura para los padres, otra para los hijos, otra para los nietos, y así sucesivamente), aunque por supuesto puede haber colisiones cuando un nodo padre tiene arcos al mismo tiempo con su hijo y con su nieto.
- Ese valor inicial debe renegociarse cuando el grafo contiene varios arcos “hermanos” (arcos que parten del mismo nodo padre y van a distintos nodos hijos, que Graphviz ha colocado a la misma altura) que están lo bastante próximos como para que sus textos se solapen.
- Para saber si un predicado choca con otro, se mantienen en memoria las alturas que están “ocupadas” y, cuando Graphviz asigna la misma altura a dos predicados próximos, se introducen pequeñas variaciones (+/- 10 píxeles, +/- 20 píxeles, etc.) hasta alcanzarse un hueco libre.
- Finalmente, el tamaño del texto del predicado y la inclinación del arco (que depende de la separación en el eje X de los nodos que relaciona) obligan a introducir variaciones horizontales, algo más complejas que las que se utilizan para centrar los textos de los nodos.

Véase: void GraphWidget::setEdgeDescription()

6.4 PRUEBAS

En la etapa de pruebas del software se crean una serie de casos de prueba con el objetivo de detectar los posibles errores de codificación. La prueba no asegura la ausencia de errores, sólo demuestran la existencia de los mismos.

Por lo tanto, el principal objetivo de esta etapa se centra en diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo.

Se denominan pruebas de caja blanca [18] a las que se realizan sobre las funciones internas de un módulo, clase, etc. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del modulo, las de caja blanca están dirigidas a las funciones internas.

Las **pruebas de caja blanca** se llevan a cabo en primer lugar. En este caso se han ido realizando de forma simultánea que el desarrollo del código, de forma que cada método, clase, etc. se ha ido probando de forma autónoma. Se ha intentado siempre que se recorrieran todos los posibles caminos, analizando tanto las operaciones lógico-aritméticas, como la definición y uso de las variables y bucles.

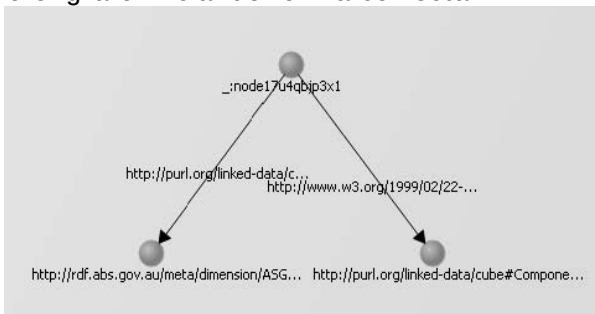
Para la realización de las **pruebas de caja negra** se han diseñado diferentes casos de prueba que se irán detallando en los puntos posteriores junto al resultado de los mismos tras su ejecución.

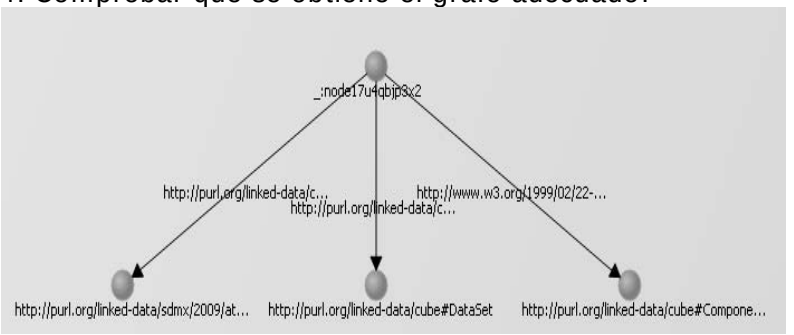
El diseño de los casos de prueba se realiza durante la fase de análisis, pero su ejecución se realiza en esta etapa, por lo que, por simplificar, mostramos en este punto toda la información.

CP – 01	**Cargar HDT incorrecto**
Descripción	Abrir un HDT incorrecto
Condiciones de ejecución	Disponer de un HDT incorrecto
Secuencia de Entrada	1. Solicitar la apertura del HDT
Resultado esperado	Mensaje en pantalla avisado del error en el fichero.
Ejecución y resultados obtenidos	Fecha ejecución: 16/08/2013 Resultado: Correcto

CP – 02	**Cargar HDT correcto con menos de 50 ternas**
Descripción	Abrir un HDT que contenga menos de 50 ternas (umbral actual)
Condiciones de ejecución	Disponer de un HDT con menos de los ternas definidos en el umbral
Secuencia de Entrada	1. Solicitar la apertura del HDT
Resultado esperado	Grafo en pantalla con todos los nodos y arcos dibujados
Ejecución y resultados obtenidos	Fecha ejecución: 16/08/2013 Resultado: Correcto

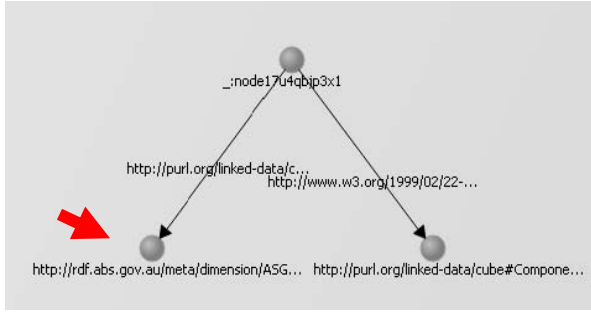
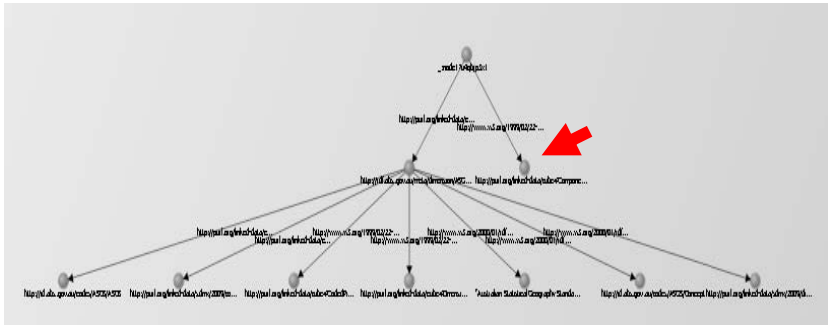
CP – 03	**Cargar HDT correcto con más de 50 ternas**
Descripción	Abrir un HDT que contenga más de 50 ternas (umbral actual)
Condiciones de ejecución	Disponer de un HDT con más de las ternas definidas en el umbral
Secuencia de Entrada	1. Solicitar la apertura del HDT
Resultado esperado	Grafo en pantalla con todos los nodos y arcos dibujados del primer nodo encontrado.
Ejecución y resultados obtenidos	Fecha ejecución: 16/08/2013 Resultado: Incorrecto. No se recupera toda la información asociada el primer nodo. Se requiere una revisión del método correspondiente. Fecha de ejecución 17/08/2013. Resultado: Correcto.

CP – 04	**Consultar grafo**
Descripción	Realizar una consulta con el filtro Objeto, Predicado y Sujeto en el HDT "*census-australian.hdt*"
Condiciones de ejecución	
Secuencia de Entrada	1. Solicitar la apertura del HDT "*census-australian.hdt*" 2. Pulsar en la pestaña de Grafos y comprobar que se ha mostrado el grafo inicial de forma correcta 

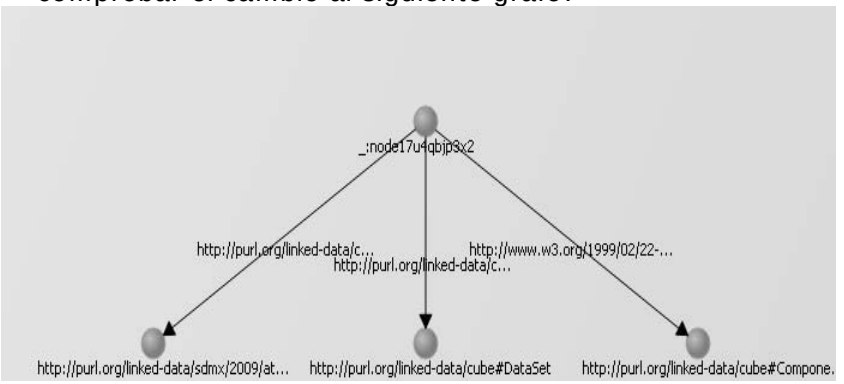
	3. Filtrar por el Sujeto: "_:node17u4qbjp3x2" 4. Comprobar que se obtiene el grafo adecuado:  5. Añadir el filtro por el predicado "*http://purl.org/linked-data/cube#attribute*" 6. Comprobar que el grafo resultante pasa a ser:  7. Eliminar el filtro del predicado y añadir el objeto "*http://id.abs.gov.au/codes/AGE5P/AGE5P*" y comprobar que no se obtiene ningún grafo. 8. Eliminar el contenido de los filtros y comprobar que se vuelve el grafo indicado en el paso 2. 9. Añadir el predicado "*http://rdf.abs.gov.au/meta/demo/measure/pop2011*" y verificar que el resultado excede las 50 ternas (no se dibuja el grafo).
Resultado esperado	Grafo actualizado de forma correcta en la pestaña de grafos.
Ejecución y resultados obtenidos	Fecha ejecución: 19/08/2013 20/08/2013 y 21/08/2013. Resultado: Correcto.

El caso de prueba anterior es un ejemplo representativo de los más de 10 diferentes casos de prueba que se realizaron para verificar el correcto funcionamiento del caso de uso "Consultar grafo" con los diferentes HDT disponibles ("*jamendo.hdt", "swdf.hdt",…)*

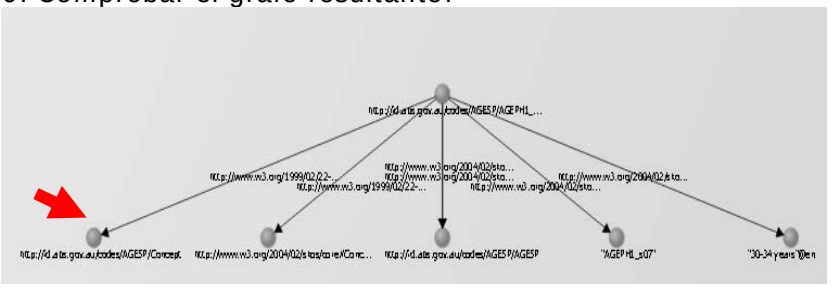
Para el caso de uso "*Pulsar nodo*" se va a incluir únicamente un caso a modo de ejemplo de los que se diseñaron para verificar esta funcionalidad.

CP – 05	**Pulsar nodo**
Descripción	Pulsar diferentes nodos obtenidos del HDT "*census-australian.hdt"*
Condiciones de ejecución	Tener cargado el HDT correspondiente en el paso 2 del caso de prueba "Consultar grafo"
Secuencia de Entrada	1. Solicitar la apertura del HDT "*census-australian.hdt*" 2. Pulsar el nodo  3. Comprobar que se obtiene:  4. Pulsar el nodo indicado en la imagen anterior y verificar que el grafo no se modifica. 5. Añadir en el filtro de Sujeto el texto "_:node17u4qbjp3x1" y verificar que volvemos a obtener el grafo indicado en el paso 2. 6. Modificar el filtro de Sujeto a "_:node17u4qbjp3x2" y

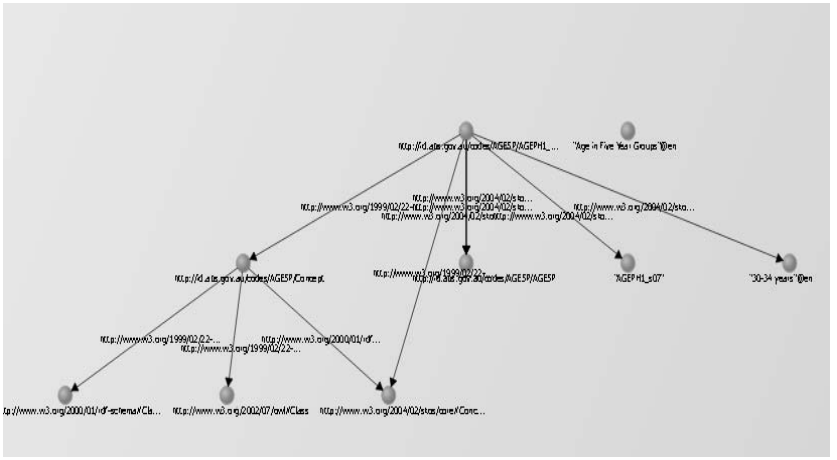
comprobar el cambio al siguiente grafo:



7. Verificar que pulsando cualquiera de los nodos en pantalla, el grafo no se ve modificado.
8. Eliminar los filtros y añadir en el sujeto http://id.abs.gov.au/codes/AGE5P/AGEPH1_s07.
9. Comprobar el grafo resultante:



10. Pulsar el nodo marcado en la imagen del paso anterior y verificar que se pasa al siguiente grafo:



Resultado esperado	Grafo actualizado de forma correcta en la pestaña de grafos.
Ejecución y resultados obtenidos	Fecha ejecución: 19/08/2013 20/08/2013 y 21/08/2013. Resultado: Correcto.

6.5 MANUAL DE USUARIO

6.5.1 ACCESO A LA APLICACIÓN

Una vez instalada la aplicación, según se muestra en el manual incluido en el CD, el acceso a la misma se realiza a través del siguiente icono:

Figura 25 – Icono de la aplicación

Tras lo cuál, se muestra la pantalla principal de la aplicación:

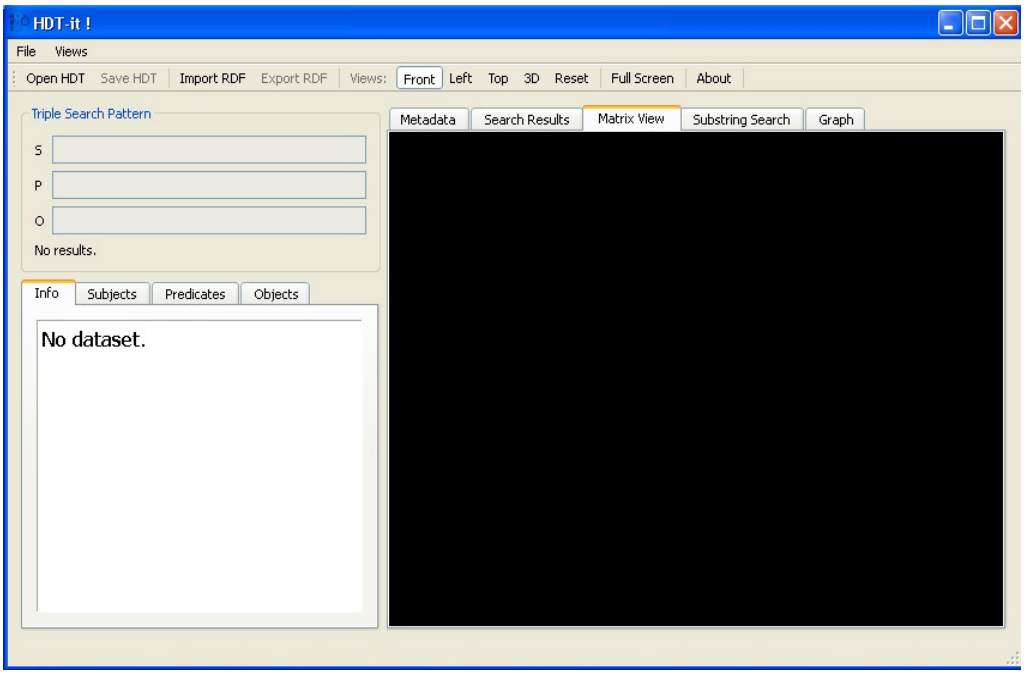


Figura 26 –Pantalla principal

6.5.2 APERTURA DE UN HDT

Para poder cargar un fichero HDT es necesario utilizar el menú "*Open HDT*" con el que se podrá seleccionar el fichero a cargar.

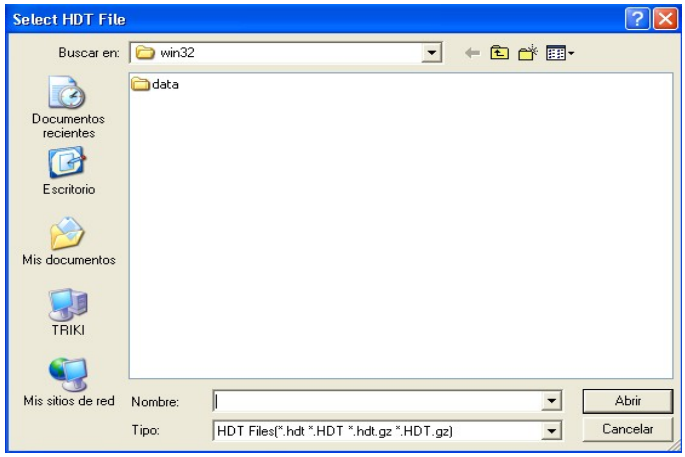


Figura 27 – Abrir HDT

Tras seleccionar el fichero, se mostrará en la pantalla principal de la aplicación la información relativa al mismo. Analizando cada una de las pestañas disponibles:

- "*Info*": resumen de la información general del archivo abierto, como número de ternas de los que consta, tamaño, compresión, etc.
- "*Subjects*": descripción de todos los sujetos que existen en el HDT.
- "*Predicates*": detalle de los predicados contenidos.
- "*Objects*": descripción de los objetos encontrados en el fichero.

El contenido de estas tres últimas pestañas se utilizará para poder realizar filtros sobre el grafo.

Del resto de pestañas disponibles, nos centraremos en describir el contenido de "Graph" que es el centro fundamental del presente trabajo.

Cuando se carga el fichero, si el HDT es correcto, se localizará un nodo del mismo y se realizará una búsqueda por *Sujeto*, para mostrar en la pestaña

"Graph" el grafo resultante (siempre que el mismo contenga menos de 50 nodos).

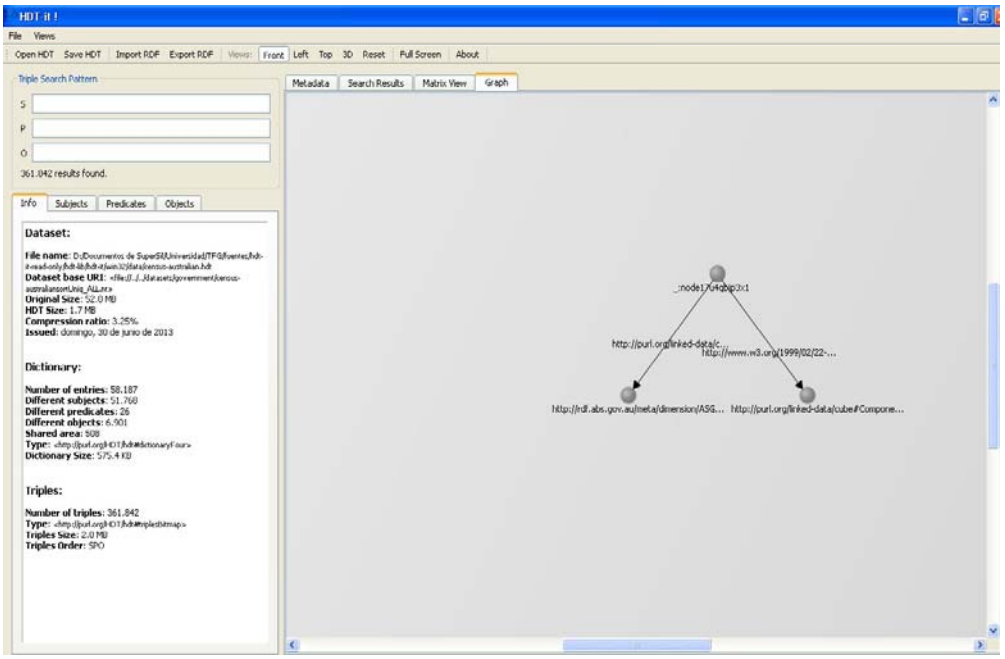


Figura 28 – Grafo inicial

6.5.3 FILTRAR POR SUJETO, PREDICADO Y OBJETO

Para poder realizar filtros por el HDT seleccionado se utilizarán las cajas de texto correspondientes, que se pueden actualizar tanto introduciendo directamente texto, como a través de las pestañas correspondientes comentadas anteriormente:

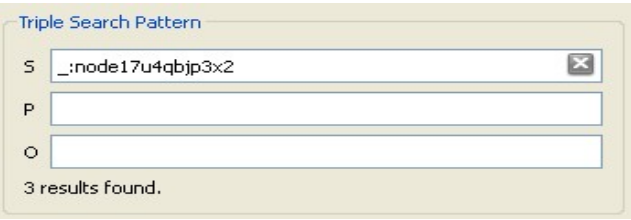


Figura 29 – Filtros

Tras realizar el filtro la pestaña “Graph” de la pantalla principal se actualizará con el grafo actualizado:

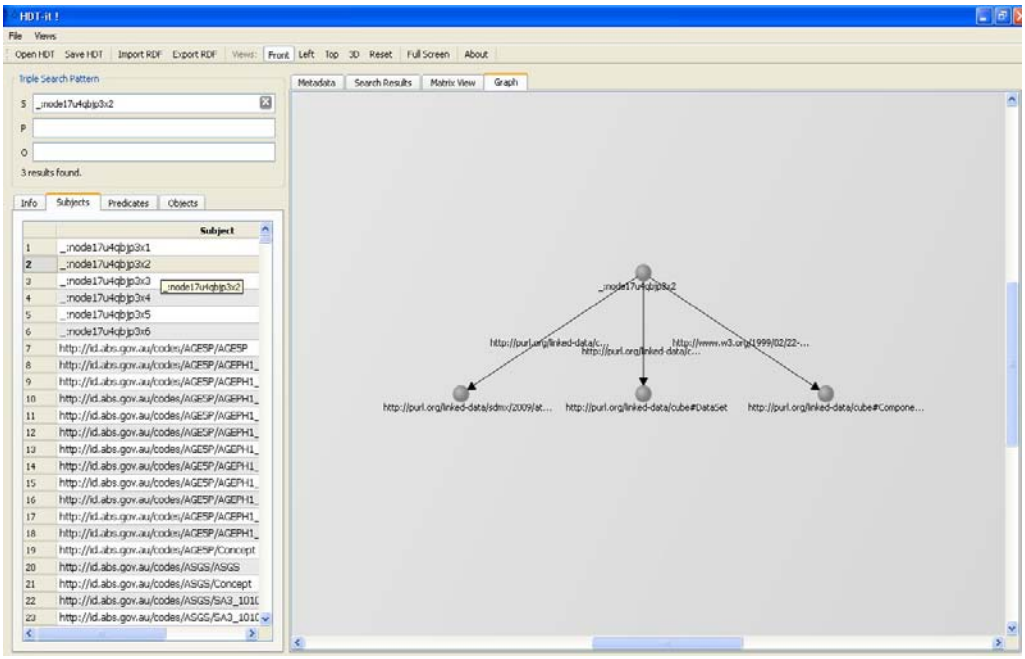


Figura 30 – Grafo con filtros

6.5.4 NAVEGACIÓN DENTRO DEL GRAFO

Para poder navegar en el grafo realizando consultas directas sobre los nodos del mismo, es suficiente con pulsar el nodo que se quiere consultar con el ratón. Si ese nodo es el Sujeto de otros nodos del HDT, se actualizará el grafo con la información de los nodos hijo.

Para facilitar la navegación dentro del grafo, existe la posibilidad de realizar zoom in/out sobre partes del mismo. Para utilizar esta funcionalidad se usará también el ratón, en concreto la imagen se alejará o acercará usando la rueda del mismo. El resultado se puede apreciar en las siguientes imágenes:

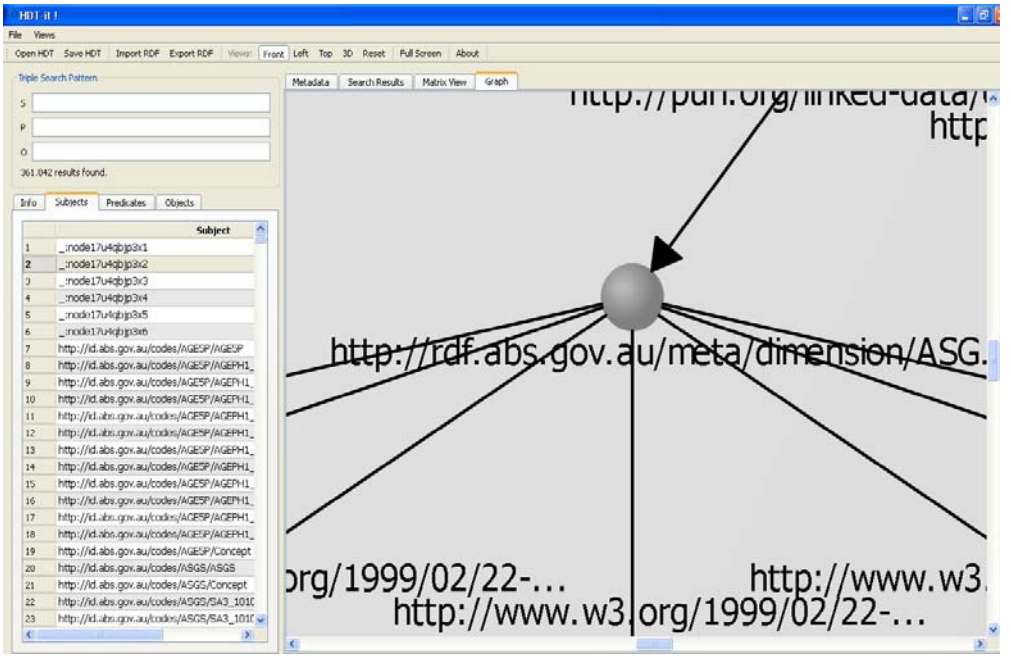


Figura 31 – Zoom in

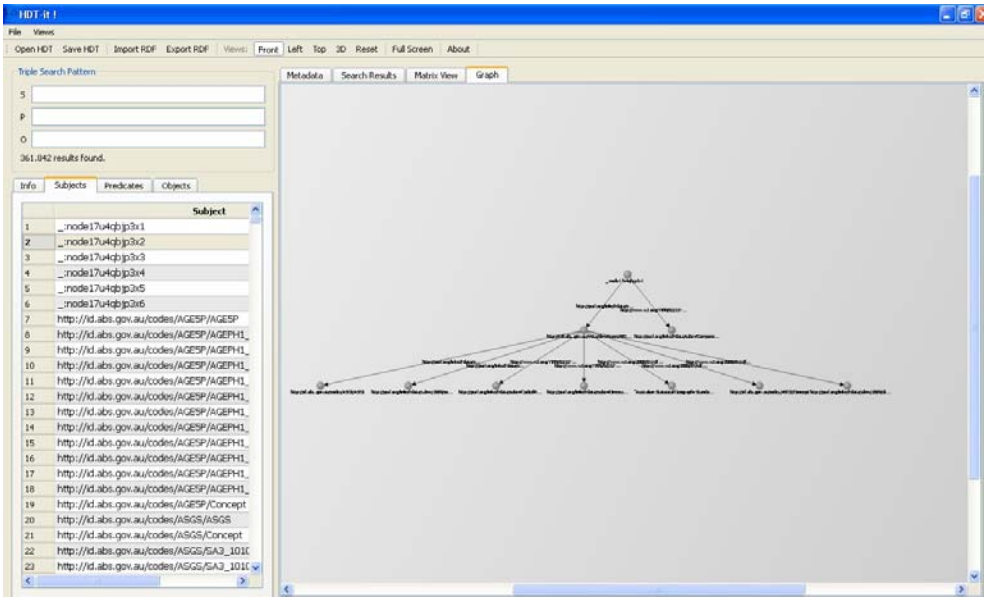


Figura 32 – Zoom out

7. CONCLUSIONES

7.1 PRINCIPALES DIFICULTADES

Las principales dificultades detectadas se han centrado fundamentalmente en aspectos técnicos. Las podemos resumir en los siguientes puntos:

- Desconocimiento del entorno de desarrollo: para poder integrar el dibujo de los grafos dentro de la herramienta HDT it! ha sido necesario utilizar Qt. Un entorno de desarrollo que no tiene una gran extensión dentro de la comunidad, lo que supone más trabajo a la hora de localizar información o resolver los problemas detectados.

- La herramienta HDT it! no utiliza la última versión de Qt, lo que dificultó la integración de Graphviz dentro de la herramienta. Esto obligó a analizar en detalle la conveniencia de utilizar las librerías cgraph o graph de esta herramienta (no pueden usarse ambas simultáneamente) para solventar las incompatibilidades.

7.2 OBJETIVOS ALCANZADOS

El principal objetivo del presente Trabajo Fin de Grado era conseguir una interfaz navegable para Big Semantic Data centrada especialmente en ficheros con formato HDT. Este punto se ha alcanzado de una forma satisfactoria.

Por otro lado, este proyecto me ha permitido profundizar en la filosofía de la Web Semántica, sus objetivos y posibilidades futuras, campo sobre el que no tenía una experiencia profesional previa.

7.3 MEJORAS FUTURAS

Como se ha comentado, una de las principales dificultades de este proyecto se ha centrado en la adecuada distribución en pantalla de los elementos del grafo, especialmente los textos asociados a los nodos y arcos. Es por eso que las principales mejoras detectadas se centran fundamentalmente en aspectos gráficos relacionados con este punto:

- Los algoritmos diseñados podrían mejorarse para ampliar el abanico de casos detectados o refinar los ya existentes. Otras posibilidades de mejora en este aspecto se centrarían en la posibilidad utilizar diferentes colores para los predicados u otros elementos del grafo.

- También se podrían diferenciar de forma gráfica los nodos que no van a tener hijos si se pulsan, de los que sí, ya que así evitaríamos que el usuario tuviera que pulsar el nodo para conocer esta información.

- Utilizar *tooltips* para visualizar el texto completo si se pasa con el ratón por el nodo o el arco.

Si nos centramos en las posibilidades a la hora de dibujar el grafo inicial en la carga de un HDT, podría ser útil cargar en cada ejecución un nodo sujeto inicial diferente. Ahora mismo se utiliza siempre el primer sujeto obtenido tras llamar a la función *search* para utilizarlo como base en la búsqueda que generará el primer grafo. Esto ofrecería diferentes visiones del HDT en cada apertura del mismo fichero.

8. BIBLIOGRAFIA

8.1 BIBLIOGRAFÍA

- **"¿Qué es la Web semántica?" y "Proyecto RDF/HDT"**

[1] w3c: www.w3.org/2001/sw (última consulta 30/08/2013)

[2] "La Web semántica". José Luís Arceiz Baquero. Curso de Doctorado: Ingeniería Web.

[3] Introducción a Big Data: http://www.enriquedans.com/2011/10/big-data-una-pequena-introduccion.html (última consulta 30/08/2013)

[4] RDF: http://www.hipertexto.info/documentos/rdf.htm (última consulta 30/08/2013)

[5] J.D. Fernández, M. Arias, M.A. Martínez-Prieto, and C. Gutiérrez. Management of big semantic data. In R. Akerkar, editor, Big Data Computing, capítulo 4. Taylor and Francis/CRC, 2013.

[6] Web RDF/HDT: http://www.rdfhdt.org/ (última consulta 30/08/2013)

[7] "Aprendiendo a nadar en el diluvio de datos". Curso de Introducción a la Web de datos. Miguel A. Martínez-Prieto y Javier D. Fernández.

[8] Web GrapViz: http://www.graphviz.org/ (última consulta 30/08/2013)

[9] "Using Graphviz as a Library". (cgraph version). Emden R. Gansner. February 28, 2013.

[10] J.D. Fernández, M.A. Martínez-Prieto, C. Gutiérrez, and A. Polleres. Binary RDF Representation for Publication and Exchange (HDT). W3C Member Submission, 2011.

[11] J.D. Fernández, M.A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF representation for publication and exchange. Journal of Web Semantics, 19:22-41, 2013.

[12] M.A Martínez-Prieto, M. Arias, and J.D. Fernández. Exchange and Consumption of Huge RDF Data. Proceedings of 9th Extended Semantic Web Conference (ESWC 2012), pp. 437-452, 2012.

- **“Metodología de gestión del proyecto”**

[13] PMI Web: www.pmi.org (última consulta 30/08/2013)

[14] Introducción a la dirección de proyectos: http://www.crisoltic.com/2011/08/introduccion-la-direccion-de-proyectos.html (última consulta 30/08/2013)

[15] Preparación para el examen PMP. Rita Mulcahy, PMP. Séptima edición.

- **“Ejecución del proyecto”**

[16] Manual Qt http://www.zonaqt.com/tutoriales/tutorial-b%C3%A1sico-de-qt-4 (última consulta 30/08/2013)

[17] Manual de Qt http://qt-project.org/ (última consulta 30/08/2013)

[18] Diseño de Casos de prueba http://angelmolsoftware.blogspot.com.es/2008/11/pruebas-del-software-caja-blanca-y-caja.html (última consulta 30/08/2013)