UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA EN INFORMÁTICA

TRABAJO DE FIN DE GRADO

Herramienta de visualización para un simulador de propagación de enfermedades contagiosas

AUTOR: Lorena Camino Rey

TUTORES: David Expósito Singh
Maria-Cristina Marinescu

Leganés, 4 de Septiembre de 2012

Agradecimientos

Me gustaría agradecer a mis padres, Mari Carmen y Antonio, todo el sacrificio que han realizado desde que nací para que pudiera llegar este momento de mi vida. Sin su apoyo y el de mi hermana, María, no hubiera sido posible llegar hasta aquí. También agradecer a mi novia, Alba, toda la paciencia y confianza que siempre me ofrece, gracias por creer siempre en mí.

Agradecer también al resto de mi familia todo lo que me han aportado durante estos años. Especialmente a mi abuelo, Antonio, que aunque se fue hace tiempo cuando yo ni siquiera imaginaba que algún día iba a escribir estas líneas, ha sido una de las personas más importantes de mi vida.

Gracias a todos mis amigos, los que hace tiempo que no veo y los que están ahí día a día, todos son muy importantes. Amigos entre los que por supuesto también incluyo a mis compañeros de estos últimos años: David, Víctor, Sandra, Mario y Ana, gracias por todos esos buenos momentos.

Por último, agradecer también al tutor de este Trabajo de Fin de Grado, David, el haberme dado la oportunidad de realizar este proyecto. Gracias a él y a Gonzalo por ofrecerme su ayuda en todo momento, y por la paciencia y comprensión que me ha demostrado siempre.

"Para volar, hay que crear el espacio de aire libre necesario para que las alas se desplieguen. Es como tirarse en un paracaídas, necesitas cierta altura antes de saltar. Para volar hay que empezar corriendo riesgos. Si no quieres, quizás lo mejor sea resignarse y seguir caminando para siempre."

Jorge Bucay, "*Las alas son para volar" (Déjame Que Te Cuente)*

Tabla de contenidos

Índice de Ilustraciones

Índice de Tablas

1 Introducción

En este primer capítulo introductorio se ofrece una visión global del proyecto realizado y documentado en la presente memoria, incluyendo las motivaciones que llevaron a su desarrollo y los objetivos perseguidos.

1.1 Motivación

Se propone el desarrollo del presente Trabajo de Fin de Grado (TFG) como continuación del Proyecto Fin de Carrera (PFC) *"Simulación de la transmisión de una enfermedad contagiosa en entornos urbanos"*, desarrollado y presentado por Gonzalo Martín Cruz en la Universidad Carlos III de Madrid [1] en el año 2010, en el que se desarrolló la herramienta *EpiGraph* [2, 3]. Este proyecto simula la evolución y contagio del virus de la gripe A (H1N1) en entornos urbanos. Para poder observar visualmente este proceso de contagio es necesario obtener una muestra reducida de la población y ver las relaciones existentes entre los miembros de la comunidad, éste es el fin del presente TFG: poder visualizar mediante un grafo la propagación del virus de la gripe A.

En un grafo los vértices representan los miembros de la población y las aristas las relaciones existentes entre ellos. La visualización de grafos presenta algunas limitaciones en la actualidad, relacionadas con el tamaño y complejidad de los mismos. Existen herramientas de visualización (algunas de ellas se analizan en este documento), pero no solucionan completamente este problema, ya que en grafos de gran tamaño es complicado distinguir claramente los nodos y las relaciones existentes entre ellos.

Por todo lo comentado anteriormente, se procede al desarrollo de una herramienta que permita obtener una muestra reducida de la población en la que se puedan observar claramente los nodos y las relaciones existentes entre ellos, para así poder analizar de manera visual la propagación del virus a lo largo del tiempo. Se debe señalar también que esta herramienta se podrá utilizar para cualquier tipo de estructura de datos que se pueda representar mediante un grafo..

1.2 Objetivos

Una vez expuestos los motivos con los que se inició la idea y el desarrollo de este proyecto, se establecen una serie de objetivos principales y específicos que debe cumplir la aplicación que se genere como consecuencia de este proyecto.

1.2.1 Objetivo principal

El objetivo principal del proyecto es desarrollar una herramienta que permita el análisis de grafos de manera visual. Se deben poder observar los nodos y las relaciones existentes entre ellos de manera clara.

1.2.2 Objetivos específicos

Además del objetivo principal surgen una serie de objetivos específicos necesarios para lograr el objetivo principal. Estos objetivos específicos que se tratarán a lo largo del desarrollo del proyecto son los siguientes:

- Visualizar de forma clara el grafo que representa el estado de la población y las relaciones existentes.
- Obtener una muestra representativa del grafo original, permitiendo variar el porcentaje de muestreo al usuario.
- Permitir realizar en un proceso unificado un muestreo de los datos originales y una visualización de esa muestra.
- Obtener un archivo con el formato adecuado para la visualización del grafo a partir de un archivo con los datos representados por una matriz dispersa en esquema *Compressed Spase Column* (CSC, véase el Anexo E), es decir, matriz dispersa comprimida por columnas.
- Permitir obtener a partir del archivo de visualización del grafo un archivo con los datos en formato CSC.
- Asociar a los vértices y aristas distintas propiedades que permitan definir su estado en ese momento, así como representarlos visualmente.
- Permitir al usuario la configuración de las propiedades asociadas a los vértices y aristas.

Para conseguir todos estos objetivos se deben seguir una serie de pasos que se definen a continuación:

- **Revisar el estado de la cuestión:** se analizarán las técnicas de muestreo de grafos y las aplicaciones para la visualización de grafos existentes en la actualidad.
- **Análisis:** definir los requisitos que debe cumplir el sistema, comprobar si es factible su desarrollo, y estudiar los posibles caminos a seguir.
- **Diseño:** definir adecuadamente y claramente la arquitectura y el entorno del sistema.

- **Planificación:** realizar una estimación del tiempo, costes y esfuerzo necesarios para el desarrollo del proyecto, definiendo una planificación del uso de recursos al comienzo del mismo.
- **Desarrollo:** implementación de la herramienta que cumpla con todas las necesidades definidas anteriormente.
- **Pruebas:** comprobar el correcto funcionamiento del sistema, realizando su labor de la manera adecuada.
- **Presupuesto:** descripción detallada del presupuesto del proyecto.

1.3 Estructura del documento

A continuación, se expone un breve resumen descriptivo del contenido de cada una de las secciones de este documento, para facilitar así la lectura del mismo:

- **Capítulo 1: Introducción.** Capítulo introductorio que sirve como presentación general del proyecto, indicando las motivaciones que han llevado a su desarrollo y los objetivos del mismo.
- **Capítulo 2: Estado de la cuestión.** Descripción y comparación entre distintas herramientas de visualización de grafos existentes en la actualidad. También se hace un análisis de las posibles técnicas de muestreo de grafos a utilizar en la aplicación.
- **Capítulo 3: Descripción del sistema.** Descripción de todos los detalles del desarrollo del sistema: metodología utilizada, entorno de desarrollo, arquitectura software, requisitos, análisis de clases, formato de la interfaz, etc. En este apartado se definen de manera específica todas las características del proyecto que se va a desarrollar.
- **Capítulo 4: Planificación.** En esta sección se incluirá una estimación de la planificación del proyecto y del cálculo de costes. Al ser estimaciones, los datos incluidos en este apartado han podido sufrir variaciones a lo largo del desarrollo del proyecto.
- **Capítulo 5: Pruebas.** En este apartado se definen las pruebas que se realizarán para verificar el correcto funcionamiento de la aplicación, comprobando que se satisfacen los requisitos definidos previamente. Además, se incluirá un análisis posterior sobre las pruebas de rendimiento realizadas en los subsistemas más importantes del sistema, ya que además de comprobar el correcto funcionamiento es importante obtener un comportamiento aceptable del sistema en cuanto al rendimiento.
- **Capítulo 6: Caso de estudio: "Propagación de una enfermedad contagiosa".** En este apartado se utiliza la aplicación desarrollada para analizar gráficamente cómo se propagaría la gripe A en una población. De esta manera se muestra la naturaleza de la aplicación, que es ofrecer una continuación al PFC *"Simulación de la transmisión de una enfermedad contagiosa en entornos urbanos"*.
- **Capítulo 7: Presupuesto.** Sección en la que se recoge una descripción detallada del presupuesto del proyecto, indicando los gastos estimados para llevarlo a cabo con éxito.
- **Capítulo 8: Conclusiones y trabajos futuros.** Resumen de las conclusiones obtenidas tras el desarrollo del proyecto, y los posibles trabajos futuros que permitan la evolución y mejora de la herramienta desarrollada en este trabajo.

2 Estado de la cuestión

Este capítulo ofrece una revisión del estado de la cuestión referente a las técnicas de muestreo de grafos existentes en la actualidad y a las herramientas disponibles para la visualización de los mismos. En el apartado de las técnicas de muestreo de grafos se prestará especial atención a analizar los algoritmos que obtengan muestras representativas del grafo original, conservando las propiedades del mismo. En cuanto a las aplicaciones de visualización de grafos se valorarán diversos aspectos, tales como el formato del archivo de visualización, las opciones de visualización que ofrezca la herramienta, la manera de disponer en pantalla los elementos del grafo, las limitaciones que tenga, etc.

2.1 Herramientas de visualización de grafos

Como ya se ha comentado anteriormente, a la hora de visualizar los grafos es importante poder distinguir claramente los estados de los vértices y las conexiones existentes entre ellos, para así poder analizar el estado de la propagación de la enfermedad en ese momento, que es la finalidad de este proyecto. Por tanto, se va a prestar especial atención a satisfacer este aspecto, además de tener en cuenta las opciones de visualización que ofrece cada herramienta, las limitaciones existentes y el formato que debe tener el archivo para su visualización.

Las aplicaciones más interesantes que se han valorado para su utilización en este proyecto son Matlab [4], NodeXL [5], Gephi [6] y R-Project [7]. A continuación, se expone una breve descripción de cada herramienta y finalmente se hace una comparación entre ellas.

- **Matlab.** Es un entorno de programación para el desarrollo de algoritmos, el análisis de datos, la visualización y el cálculo numérico. Es una herramienta muy potente, por lo que es ideal para todo tipo de cálculos matemáticos. Debido a que el formato de representación del grafo inicial utiliza las matrices dispersas, esta aplicación permitiría manejar estos datos de manera muy eficiente. Las ventajas de esta herramienta están asociadas a su potencia y opciones de cálculo, se pueden obtener las propiedades de los grafos de manera fácil y manejarlos con sencillez. Por otra parte, quizá el apartado visual sea el punto flaco, ya que esta herramienta está más orientada al manejo matemático que a la visualización de los datos. Además, como desventaja hay que señalar que Matlab no es una aplicación gratuita, aunque sí está bastante extendida dentro del mundo de la ingeniería y las matemáticas.
- **NodeXL.** Esta herramienta es un complemento para Microsoft Excel que permite visualizar y analizar grafos. Los datos se tratan en una plantilla que se maneja de la misma manera que el resto de datos en Excel, por

lo que la interfaz para gestionar los datos del grafo y la visualización es muy intuitiva para cualquier usuario mínimamente familiarizado con la tecnología. Esta herramienta está más orientada a la visualización que a la gestión de los datos, por lo que ofrece gran variedad de opciones para la representación del grafo en pantalla, aunque también permite obtener distintas propiedades matemáticas del grafo. La potencia de esta herramienta es más limitada que en Matlab, ya que empeora su rendimiento con grafos muy grandes. Es por esto que presenta algunas restricciones a la escalabilidad de los grafos, sin embargo para la finalidad de este proyecto es suficiente la potencia que ofrece. Los formatos que utiliza esta herramienta para la visualización de grafos, además del propio formato de archivos de Microsoft Excel, son:

  - **UCINET [8].** Este formato está orientado a su integración con la aplicación del mismo nombre, cuya función principal es mapear, editar y analizar redes sociales mediante su representación en grafos. El tipo de archivo utilizado es un archivo de texto con extensión .txt, en el que se define la matriz que representa el grafo.
  - **Pajek [9].** Este formato es el utilizado por su herramienta homónima. Es una aplicación similar a UCINET, su función es la misma. El tipo de archivo utilizado tiene extensión .net, y en él se definen los vértices existentes, y posteriormente las aristas mediante una matriz.
  - **GraphML [10].** Es utilizado para estandarizar el formato de archivos para la representación de grafos. El tipo de archivo es XML, en él se definen los vértices y las aristas con sus propiedades asociadas. Este formato ofrece más opciones de visualización que los anteriores, ya que permite definir propiedades para los nodos y las aristas. Además es un formato más universal al utilizar una sintaxis basada en XML, muy usado en la actualidad. La extensión de los archivos de este tipo archivo es .graphml.

- **Gephi.** Esta aplicación permite la visualización y análisis de grafos. Es una herramienta de código abierto y gratuita, disponible para los principales sistemas operativos. Presenta algunas limitaciones en la visualización respecto a NodeXL, ya que la forma de disponer en pantalla los elementos del grafo es menos clara. Presenta una interfaz muy intuitiva para gestionar los datos de los vértices y aristas, además se pueden obtener las propiedades matemáticas básicas del grafo. Ofrece las mismas limitaciones que NodeXL respecto a la escalabilidad de grafos, ya que disminuye su rendimiento ante un gran tamaño de los mismos, aunque suficiente para los datos que se van a manejar inicialmente en este proyecto. Esta aplicación dispone de un formato propio con extensión .gephi, pero además permite la utilización de otros formatos como son:

  - **CSV.** Este formato representa los datos de la matriz separados por comas. Multitud de herramientas, como por ejemplo Microsoft Excel, permiten la exportación de datos a este tipo de archivos.
  - **Pajek.** Este formato es el asociado a la aplicación con el mismo nombre y tiene extensión .net. Tal y como se ha explicado anteriormente es una herramienta dedicada a la visualización y gestión de grafos.
  - **UCINET.** Como ya se ha señalado en el párrafo anterior este formato está asociado a una aplicación llamada UCINET, que se dedica a la edición y visualización de grafos. La extensión válida para Gephi es .dl.
  - **GraphML.** Gephi permite la importación de archivos con este formato, que pretende ser una estandarización para la representación de grafos. La extensión válida es .graphml.
  - **Otros formatos.** Admite la importación de datos con otros formatos de archivo menos populares, como por ejemplo: GraphViz, GDF, GEXF, GML, TLP, VNA, etc.

  En el enlace a la documentación de Gephi existente al final de este documento [6] se puede encontrar amplia información sobre el funcionamiento de la herramienta y todos los formatos de archivo que soporta.
- **R-Project.** Es un proyecto GNU que dispone de entorno y lenguaje de desarrollo propio (llamado R), orientado a la manipulación de datos, cálculo y visualización gráfica. Ofrece herramientas tanto para el manejo intermedio de los datos, especialmente mediante matrices, como para su visualización final en pantalla. Es muy utilizado entre la comunidad estadística, ya que dispone de un amplio abanico de herramientas estadísticas y gráficas, además de permitir que los usuarios desarrollen paquetes que extiendan su funcionalidad. Incluso, dentro del cálculo numérico, puede ser una alternativa muy válida a Matlab.

A continuación se resumen en una tabla las ventajas y desventajas de cada una de las herramientas descritas, teniendo en cuenta las necesidades de este proyecto.

Herramienta	Ventajas	Desventajas
Matlab	✓ Potencia de cálculo ✓ Opciones matemáticas ✓ Disponible para los sistemas operativos más populares	× Necesaria licencia para la instalación × Menos orientado a la visualización
NodeXL	✓ Fácil instalación ✓ Interfaz intuitiva ✓ Variedad de opciones de visualización ✓ Visualización aceptable ✓ Escalabilidad visual aceptable ✓ Herramienta en evolución	× Aplicación en fase beta × Necesario tener instalado Microsoft Excel 2007 ó 2010 × Potencia de cálculo limitada × Opciones matemáticas limitadas
Gephi	✓ Fácil instalación ✓ Disponible para los sistemas operativos más populares ✓ Herramienta gratuita y de código abierto ✓ Visualización aceptable ✓ Escalabilidad visual aceptable	× Aplicación en fase beta × Potencia de cálculo limitada × Opciones matemáticas muy limitadas × Opciones de visualización limitadas
R-Project	✓ Dispone de herramientas de manejo de datos muy potentes ✓ Permite extender la funcionalidad mediante paquetes ✓ Ofrece herramientas de visualización ✓ Potencia de cálculo ✓ Herramienta gratuita y de código abierto	× Menos orientado a la visualización de grafos

Tabla 1. Ventajas y desventajas de las herramientas de visualización de grafos

Para terminar este análisis de las posibles herramientas de visualización de grafos a utilizar en este proyecto se incluye una tabla comparativa entre todas ellas, en lo referente a las propiedades que se necesitan cubrir en este proyecto. Las propiedades valoradas son las siguientes:

- **Escalabilidad visual:** capacidad de las representaciones visuales y de las herramientas de visualización de mostrar claramente grandes conjuntos de datos.
- **Gestión de datos:** capacidad de gestionar (añadir, editar y eliminar) los datos relativos a vértices y aristas del grafo.
- **Propiedades del grafo:** variedad de propiedades matemáticas del grafo que se pueden obtener.
- **Propiedades de visualización:** variedad de propiedades visuales editables relativas a los vértices y aristas del grafo (color, forma, etc).
- **Escalabilidad de software:** capacidad de la herramienta de manejar interactivamente grandes conjuntos de datos (grafos con decenas o cientos de miles de vértices y aristas).

- **Escalabilidad analítica:** capacidad de los algoritmos matemáticos de manejar de forma eficiente grandes conjuntos de datos.

Por su parte, los posibles valores usados para puntuar cada propiedad son: alto, medio-alto, medio, medio-bajo o bajo, según se ajusten a las necesidades del proyecto.

	Matlab	NodeXL	Gephi	R-Project
Escalabilidad visual	Medio	Medio	Medio	Medio
Gestión de datos	Medio-alto	Alto	Alto	Medio-alto
Propiedades del grafo	Alto	Medio-alto	Medio	Alto
Propiedades de visualización	Medio	Alto	Medio-alto	Medio-alto
Escalabilidad de software	Alto	Medio	Medio	Alto
Escalabilidad analítica	Alto	Medio-bajo	Medio	Alto

Tabla 2. Comparación entre las herramientas de visualización de grafos

2.1.1 Herramienta de visualización seleccionada

Tras valorar lo expuesto en este apartado, se ha decidido utilizar el formato GraphML como el formato de los archivos de visualización de grafos que genere la aplicación, ya que es el formato más universal, siendo aceptado por varias aplicaciones de visualización de grafos. De entre las herramientas a utilizar para la visualización de estos archivos, se van a utilizar principalmente dos: NodeXL y Gephi.

La herramienta primaria que se utilizará en las pruebas de funcionamiento y en la definición del formato de los archivos generados será NodeXL. Esto se debe a que es una herramienta muy orientada a la visualización y fácilmente instalable, ya que es un complemento de Microsoft Excel. No supone un problema que se requiera tener instalado este programa para poder utilizar NodeXL, ya que es una aplicación muy popular y ya disponible en la mayoría de equipos que se venden en la actualidad. Además, Excel permite el manejo de hojas de cálculo de gran tamaño, estando estos límites dentro de aceptable para este proyecto [14].

Gephi se utilizará para comparar la visualización obtenida en NodeXL, a la vez que servirá para calcular las propiedades matemáticas de los grafos en el caso de que por problemas de escalabilidad NodeXL no pueda ofrecer esta característica.

2.2 Técnicas de muestreo de grafos

Aunque con la herramienta de visualización de grafos que se utilice se debe obtener una escalabilidad visual aceptable, es inevitable que con grandes volúmenes de datos (millones de nodos o vértices) esta visualización pierda claridad. Es por esto que es necesario utilizar una muestra reducida de estos grandes grafos para poder visualizar claramente las propiedades de los mismos. Estas muestras deben tener unas propiedades similares al grafo original, para así ser una muestra representativa del mismo.

Para obtener las muestras de las que se hablan en el párrafo anterior se debe utilizar un algoritmo eficiente para nuestro propósito, que además, como ya se ha comentado, conserve la naturaleza del grafo inicial. Existen varios posibles algoritmos a utilizar bastante populares en la actualidad para muestrear grafos. Se exponen a continuación:

- **Breadth-First Sampling (BFS) [11].** Es el algoritmo de muestreo de grafos clásico. Se usa ampliamente en el estudio de redes sociales online, analizando por ejemplo el comportamiento de los usuarios. En líneas generales, este algoritmo comienza con un vértice aleatorio y va añadiendo sus nodos vecinos utilizando una política FIFO sin repetir nodos hasta que se llega al coste total. El problema de este algoritmo es que puede crear una muestra localizada en una parte del grafo, por lo que ésta no será representativa del grafo completo.
- **Metropolis-Hasting Random Walk (MHRW) [11].** Este algoritmo también es ampliamente utilizado en el muestreo de grafos. Trabaja de la siguiente manera:
  1. Se elige un nodo inicial aleatorio (u).
  2. Se selecciona uno de los nodos vecinos de ese vértice (v).
  3. Se genera aleatoriamente un valor ρ entre 0 y 1. Si este valor es menor que la división de k_u/k_v, siendo k_u el grado del vértice inicial y k_v el grado de su nodo vecino, se añade el vértice v a la muestra, sino el algoritmo sigue utilizando en el nodo u.
  4. Se realiza este proceso hasta que se llega al coste total del grafo.

  El problema de este algoritmo es que el grafo debe ser conexo y no tener islas, ya que éstas no se alcanzarán.
- **Random Jump (RJ) [11].** Este algoritmo es similar al MHRW, la diferencia principal es que con el algoritmo RJ existe la opción de saltar a un vértice aleatorio, con lo que se puede acceder a los nodos aislados.
- **Albatross [11]:** este algoritmo pretende solucionar el problema que tienen los otros algoritmos definidos aquí, que no convergen hacia un muestreo uniforme ya que están orientados a elegir vértices con un alto grado, exceptuando el algoritmo MHRW que sí realiza un muestreo uniforme aunque siempre teniendo en cuenta que el grafo sea conexo. Albatross se basa en el algoritmo MHRW junto con la estrategia RJ, con lo que se permitiría acceder a nodos aislados en grafos no conexos,

realizando así un muestreo uniforme. El algoritmo se basa en recorrer los nodos adyacentes a uno dado, siempre teniendo opciones de saltar aleatoriamente a otra zona del grafo, lo que aseguraría no realizar un muestreo local en el que sólo se utilizara una parte del grafo.

- **Frontier Sampling [11].** Este algoritmo es totalmente diferente a los demás ya que se basa en recorrer las aristas, y no los nodos. De esta manera se van añadiendo a la muestra del grafo las aristas escogidas. No es una técnica de muestreo especialmente adecuada para este proyecto, ya que en la práctica es más compleja y menos versátil.

2.2.1 Técnica de muestreo seleccionada

Después de valorar los distintos algoritmos de muestreo a utilizar, se ha decidido emplear el algoritmo Albatross [11]. Este algoritmo ofrece mejoras sobre los otros expuestos, ya que supone una combinación de algunos de ellos. Albatross se basa en el algoritmo MHRW ya definido en esta sección, pero incluye la técnica de RJ, lo que hace que se pueda realizar un salto a otra zona del grafo, evitando realizar muestras de manera local que sólo exploren una parte del grafo. Esto supone una gran mejora para poder realizar un muestreo uniforme, ya que los grafos no tienen por qué ser conexos.

El funcionamiento básico de esta técnica de muestreo es el siguiente:

1. Se define un valor ρ, que en el caso de este proyecto será 0,02.
2. Se elige un vértice al azar que será la semilla inicial.
3. Se genera un valor aleatorio comprendido entre 0 y 1.
   3.1. Si este valor es menor que ρ: se elige aleatoriamente un vértice y se añade a la lista de nodos de la muestra si no está ya en ella.
   3.2. Si el valor es mayor o igual que ρ: se elige al azar un nodo vecino al vértice seleccionado en ese momento, y se genera un valor β aleatorio comprendido entre 0 y 1.
      3.2.1. Si β es menor que el cociente del grado del nodo actual y su vecino seleccionado: se añade el nodo vecino a la lista de nodos de la muestra si no está ya en ella.
      3.2.2. Si β es mayor o igual que el cociente del grado del nodo actual y su vecino seleccionado: se añade el vértice actual a la lista de nodos de la muestra si no está ya en ella.
4. Se repite el punto 3 hasta que la muestra tenga el número de vértices necesarios.

Una descripción más detallada de este algoritmo se puede observar en la Ilustración 1.

Algorithm 1 Albatross Sampling

```
cost ← 0
sample set S ← empty
select a random vertex v as the initial seed
while cost < Total-Cost do
  generate α from uniform distribution U[0, 1]
  if α < p then
    choose a new random vertex u
    add u to the sample set S
    if u is visited for the first time then
      cost ← cost+Jump-Cost
    else
      cost ← cost+Jump-Cost−1
    end if
    v ← u
  else
    select an edge (v, w) starting from v randomly
    generate β from uniform distribution U[0, 1]
    if β < k_v/k_w then
      add w to the sample set S
      if w is visited for the first time then
        cost ← cost+Walk-Cost
      end if
    else
      remain at v
      add v to the sample set S
    end if
  end if
end while
```

Ilustración 1. Algoritmo de la técnica de muestreo Albatross

En [11] se realiza una evaluación del rendimiento de este algoritmo en comparación con otras técnicas de muestreo, comprobando los buenos resultados de esta técnica en cuanto a robustez y convergencia.

Para implementar este algoritmo será necesario utilizar alguna librería de manejo de matrices en formato disperso y para generación de números aleatorios. Esto se analizará en el apartado 0 de este documento.

3 Descripción del sistema

Una vez definidos los objetivos y analizadas las herramientas y técnicas de muestreo a utilizar, se pasa a estudiar las posibles soluciones a aplicar para el desarrollo completo y correcto del proyecto, de tal manera que se satisfagan todas las necesidades planteadas inicialmente. En esta sección se analizarán estas opciones, señalando la solución elegida y los motivos. También se definirá la arquitectura software del sistema, se especificarán los requisitos que debe cumplir la aplicación, se detallará la descripción del sistema a construir, y por último se hará mención al formato de la interfaz de la aplicación.

Es de gran importancia señalar que el método de trabajo que se va a seguir durante el desarrollo es la metodología **Métrica v.3** [12]. Métrica v.3 es una metodología de planificación, desarrollo y mantenimiento de sistemas de información. Esta metodología está promovida por el Ministerio de Administraciones Públicas del Gobierno de España [13] para la sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas.

La metodología Métrica v.3 marca una serie de etapas que debe seguir un proyecto durante su desarrollo. Debido a la simplicidad de este TFG en comparación con otros proyectos de gran magnitud, se ha creído conveniente adaptar este proceso resumiendo el desarrollo en tres fases (véase Ilustración 2):

- **Fase inicial.** En la que se define el marco inicial del proyecto, con las necesidades del sistema y las posibles tecnologías a utilizar. Esta fase es un análisis de los requisitos, las restricciones iniciales y los objetivos a cumplir, de tal manera que se obtenga una visión general sobre la situación inicial comprobando si es factible el desarrollo y qué caminos se pueden seguir. Durante esta etapa se deben realizar controles de configuración y de calidad para comprobar que el desarrollo del proyecto se produce dentro de los cauces adecuados y que se cumplen las especificaciones indicadas por los tutores del proyecto, mostrándoles versiones del sistema para que valoren posibles modificaciones o mejoras.
- **Fase de desarrollo (DSI).** Durante esta fase se define de manera concreta y completa la arquitectura del sistema y todas las tecnologías que van a intervenir en el desarrollo del mismo. El DSI se compone del diseño del sistema, la implementación del mismo, y las pruebas que aseguren el correcto funcionamiento de la aplicación.
- **Fase final.** Esta fase se basa principalmente en el seguimiento y mantenimiento del sistema, realizando una batería de pruebas que corrobore que la aplicación implementada cumple con todas las indicaciones previas especificadas por los tutores del TFG, y que el sistema realiza sus funciones de manera efectiva y eficiente. Los tutores

validarán si el comportamiento del sistema es el adecuado teniendo en cuenta las necesidades existentes, permitiendo o no su implantación.

Durante el desarrollo del proyecto se han producido frecuentes reuniones con los tutores del mismo David E. Singh y Maria-Cristina Marinescu, aproximadamente una al mes. De esta manera se comprobaba que el camino seguido era el correcto, subsanando lo antes posible posibles errores que pudieran cometerse, evitando que esto afectara al proyecto de manera significativa.

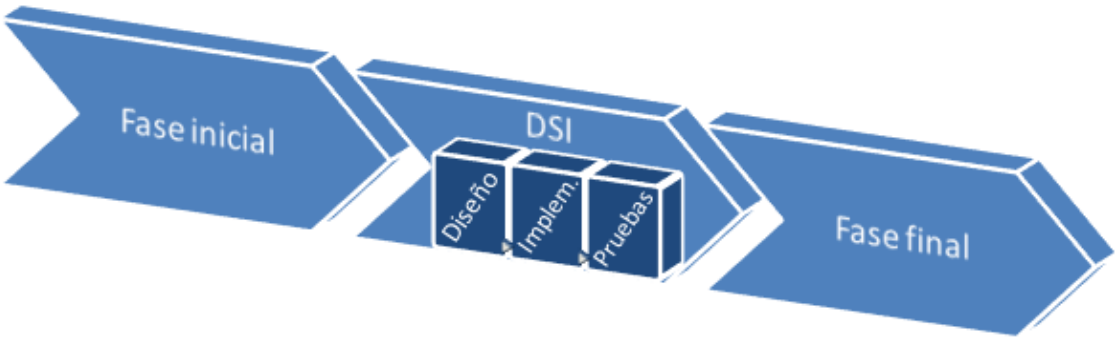

Ilustración 2. Ciclo de vida del TFG

3.1 Descripción de la arquitectura

La aplicación que se va a desarrollar está basada en varios subsistemas, formando un entramado complejo. Ya se han valorado las herramientas a utilizar referentes a la técnica de muestreo y a la herramienta de visualización, así como algunos formatos de archivo que se manejarán. Estas características se incluyen en la definición de la arquitectura del sistema.

3.1.1 Arquitectura software del sistema

Tras detallar algunas de las herramientas que se van a emplear en el desarrollo de este proyecto, se pasa a especificar una primera aproximación a la arquitectura software general de la aplicación, teniendo en cuenta que el sistema a desarrollar es una aplicación de escritorio. En la Ilustración 3 que se encuentra a continuación se resume de manera clara el proceso que sigue la aplicación para la consecución de su labor.

Ilustración 3. Arquitectura general del sistema

De forma más detallada, se tienen los siguientes elementos:

- **Componentes del sistema.** La aplicación se basa principalmente en dos componentes: el muestreo del grafo y la visualización del mismo. En el muestreo se aplicará el algoritmo Albatross como ya se ha comentado anteriormente, mientras que para la visualización del grafo será necesario generar un formato de visualización válido para ser ejecutado por las herramientas de visualización que se han elegido (NodeXL y Gephi).
- **Formato de los archivos de origen.** El formato de los archivos iniciales viene dado por el proyecto [2, 3] del que parte este TFG. El formato se basa en la representación de matrices dispersas con esquema CSC. En el Anexo E se especifica el formato concreto de este tipo de archivos.

- **Formato de los archivos generados tras el muestreo.** Durante el proceso general de visualización de grafos se genera un archivo intermedio tras el muestreo. Este archivo tiene el mismo formato CSC que los archivos de origen válidos.
- **Formato de los archivos generados para la visualización del grafo.** La parte de la aplicación que genera el archivo de visualización toma como punto de partida un archivo de muestra con formato CSC y genera un archivo de salida XML, más concretamente utiliza el formato GraphML (véase el Anexo F), que como ya se ha comentado anteriormente es un formato que se está convirtiendo en un estándar para la representación de grafos.

Un esquema de la arquitectura más detallado, tras la definición completa de requisitos y de otras características del sistema, se puede analizar en el apartado 3.6 de este documento.

3.2 Entorno de desarrollo

Como punto inicial es necesario definir el entorno que se va a emplear para desarrollar el proyecto. Teniendo en cuenta el estado de la cuestión expuesto en este documento y los requisitos que debe cumplir el sistema, se valoran varios aspectos sobre la solución que se va a adoptar para llevar a cabo este TFG. Debido a que la implementación del proyecto va a constar de tres partes (muestreo del grafo, generación del archivo de visualización del grafo, y visualización del grafo) es necesario definir cada una de las opciones de la solución para cada parte de la implementación. Los aspectos entre los que se debe decidir son:

- **Lenguaje de programación.** Teniendo en cuenta las necesidades del sistema se deben valorar los posibles lenguajes de programación a utilizar. Además de los requisitos del proyecto, hay varios puntos que analizar para elegir un lenguaje u otro: el paradigma de programación que siga (orientado a objetos, estructurado, imperativo, etc), los conocimientos propios de programación, los entornos de desarrollo disponibles para ese lenguaje, etc.
- **Sistema operativo.** Se debe elegir un sistema operativo con el que el equipo de desarrollo esté familiarizado, a la vez que sea posible su uso dependiendo del lenguaje de programación elegido.
- **Entorno de desarrollo.** Dependiendo del lenguaje de programación a utilizar y del sistema operativo elegido se debe valorar el empleo de un entorno de desarrollo u otro. Existen entornos muy populares, como por ejemplo Eclipse, NetBeans, VisualStudio, etc.
- **Otras herramientas.** Además de los aspectos anteriores es necesario definir otras herramientas que no se engloban en ninguno de los puntos anteriores, pero que de igual manera son imprescindibles para el completo desarrollo de este proyecto.

3.2.1 Lenguaje de programación

En este apartado se tratan los posibles lenguajes de programación a utilizar para el desarrollo de este proyecto. El principal motivo para elegir un lenguaje u otro es que cumpla con las necesidades del sistema, se elegirá el lenguaje de programación más adecuado para cumplir los requisitos de la aplicación a desarrollar. De manera secundaria se tendrán en cuenta los propios conocimientos de programación en estos lenguajes.

A continuación, se describen los posibles lenguajes de programación a utilizar:

3.2.1.1 Java

Java [15] es un lenguaje de programación de alto nivel orientado a objetos diseñado por *Sun Microsystems* (*Oracle Corporation*), por lo que se ajusta a las necesidades del proyecto. Las principales características de este lenguaje son las siguientes:

- Es portable, permite la ejecución de un mismo programa en distintas plataformas. Esto es debido a la JVM, que ejecuta el *bytecode* independientemente de la arquitectura hardware existente y del sistema operativo.
- La licencia que utiliza es GNU GPL [16], por lo que es software libre. Además existen entornos de desarrollo muy potentes y populares para el desarrollo de aplicaciones en Java, como son por ejemplo Eclipse y NetBeans.

3.2.1.2 .NET

.NET [17] es un framework de Microsoft orientado a objetos y compuesto por varios lenguajes de programación (principalmente C#, VB.NET y J#). Está orientado a la programación web pero también es muy adecuado para la implementación de aplicaciones de escritorio, permitiendo la creación de interfaces de manera sencilla. Las principales características de este lenguaje son las siguientes:

- Es independiente de la plataforma hardware.
- Al ser un producto de Microsoft, su instalación sólo es posible en los sistemas operativos Microsoft Windows.
- No es completamente gratuito, ya que dependiendo de las herramientas que se utilicen se necesitan licencias de pago.

3.2.1.3 C

C es un lenguaje de programación estructurado orientado a la implementación de sistemas operativos. Este lenguaje es especialmente indicado para aplicaciones que necesitan realizar su objetivo de manera eficiente, debido a que ofrece un mejor rendimiento que otros lenguajes. Las principales características de este lenguaje son las siguientes:

- Es un lenguaje multiplataforma, existen compiladores para la mayoría de sistemas existentes, lo que permite su utilización en varios sistemas operativos.
- Existen multitud de bibliotecas para este lenguaje, en concreto en este proyecto nos interesan las bibliotecas con funciones matemáticas sobre matrices que se pueden utilizar para representar los grafos.
- Las distribuciones del sistema operativo Linux tienen un compilador que permite el desarrollo de aplicaciones en C de manera gratuita. Para la implementación en otros sistemas operativos existen entornos de desarrollo gratuitos también.

3.2.1.4 C++

Este lenguaje de programación fue creado con el objetivo de extender el lenguaje C con mecanismos que permitieran la manipulación de objetos, por lo que C++ no es un lenguaje completamente orientado a objetos, sino que es multiparadigma. Las principales características de este lenguaje son las siguientes:

- La manera de programar es similar a Java, por lo que se pueden realizar aplicaciones de escritorio e interfaces fácilmente.
- Existen entornos de desarrollo gratuitos para desarrollar aplicaciones en este lenguaje.

3.2.1.5 Selección

Después de analizar los distintos lenguajes de programación que se adaptan a las necesidades del sistema a desarrollar, se han decidido utilizar Java y C.

Debido a que en la aplicación es necesario realizar un muestreo del grafo original, se cree adecuado utilizar C como lenguaje de programación para realizar este proceso. Las razones son las siguientes:

- El muestreo se realizará sobre grafos de tamaño considerable, por lo que la eficiencia del lenguaje C es recomendable para manejar este volumen de datos.
- En el proceso de muestreo será necesario manejar los grafos mediante alguna estructura de datos, las más adecuadas a priori son las matrices. En C existen bibliotecas que permiten el manejo de matrices de manera eficiente, además durante el algoritmo de muestreo que se emplee será necesario el uso de funciones matemáticas para generar números aleatorios, por lo que también será necesario utilizar bibliotecas que faciliten esta labor.
- Los conocimientos y experiencia que se tienen de este lenguaje de programación son suficientes para poder implementar esta parte de la aplicación de manera correcta.

Para el resto de desarrollo el lenguaje elegido es Java, debido a que es orientado a objetos y se ajusta a las necesidades requeridas por el sistema, pudiendo integrar el programa de muestreo que se implemente y permitiendo el desarrollo de la interfaz de manera relativamente sencilla. Las razones que hacen Java especialmente adecuado para el desarrollo de la aplicación son las siguientes:

- Al ser orientado a objetos permite la definición sencilla de una estructura para el manejo de grafos.
- Será necesaria la conversión de archivos, ya que el archivo con los datos del grafo debe ser transformado en un archivo de visualización del grafo. Java da facilidades para realizar este proceso, ya que hay APIs disponibles de *parsers* para generar y analizar algunos tipos de archivos muy populares, como por ejemplo XML y HTML.
- Es posible la integración de la parte de muestreo, invocando una el archivo ejecutable que contenga ese programa.
- Es un lenguaje muy válido para la implementación de interfaces, pudiendo utilizar APIs de ayuda incluidas en los entornos de desarrollo disponibles.

- Los conocimientos que se poseen sobre este lenguaje de programación son suficientes para llevar a cabo el desarrollo sin demasiados problemas.
- Existen varios entornos de desarrollo gratuitos y muy potentes con los que el equipo de desarrollo está muy familiarizado.

3.2.2 Sistema operativo

En este apartado se definen los posibles sistemas operativos a utilizar para el desarrollo del proyecto. La elección de uno u otro dependerá de la disponibilidad por parte del equipo de desarrollo, de su compatibilidad con el resto de elementos del entorno de desarrollo y de las limitaciones que ofrezca.

A continuación, se describen los posibles sistemas operativos a utilizar:

3.2.2.1 Microsoft Windows

Es el nombre bajo el que se denominan distintos sistemas operativos pertenecientes a la empresa Microsoft [18]. No son sistemas operativos gratuitos, están sujetos a una licencia de pago de Microsoft. Dependiendo de la versión el tipo de núcleo puede ser monolítico (versiones basadas en MS-DOS) o híbrido (versiones basadas en Windows NT). La última versión estable recibe el nombre de Windows 7.

Alguna de las características de relevancia para este proyecto son las siguientes:

- Es muy popular, está preinstalado en la mayoría de equipos que se venden a los consumidores en la actualidad.
- Se basa en un ambiente gráfico, poseyendo una interfaz intuitiva orientada a usuarios sin conocimientos elevados en tecnología.
- Posee compatibilidad con una gran variedad de aplicaciones, incluyendo entornos de desarrollo.

3.2.2.2 GNU/Linux

GNU/Linux [19] es un proyecto de sistemas operativos en los que se combina un núcleo similar a Unix (Linux) con las herramientas del proyecto GNU. Se trata de software libre, por lo que existen numerosas distribuciones: Debian, Ubuntu, Fedora, Gentoo, etc. Lon sistemas operativos basados en Linux son muy utilizados como sistemas de programación.

Las principales características que hacen este sistema operativo adecuado para el desarrollo de este proyecto son:

- Es uno de los sistemas operativos más fiables, estables y seguros.
- Puede utilizarse en modo consola o en entorno gráfico.
- Es especialmente adecuado para la programación en C, ya que incluye compiladores y herramientas para utilizar este lenguaje.
- Tiene una gran variedad de software disponible.

- Soporta diversas arquitecturas mediante la compilación cruzada, siendo un entorno adecuado para desarrollos heterogéneos.

3.2.2.3 Selección

Después de estudiar los sistemas operativos disponibles se ha decidido utilizar ambos. Para el proceso de muestreo se va a utilizar el lenguaje C, siendo más sencilla la implementación de esta parte en Linux. Por su parte, al realizar el resto de la aplicación en Java, parece más sencillo utilizar Microsoft Windows ya que existen versiones disponibles de los entornos de desarrollo que se pueden utilizar para este lenguaje.

En concreto, la distribución de GNU/Linux que se va a utilizar es Ubuntu [20], en su versión 12.04 TLS. Los motivos de esta elección son los siguientes:

- Implementar el muestreo de grafos en un sistema operativo GNU/Linux facilitará el desarrollo en C, además de evitar posibles problemas de compatibilidad o complicaciones que pudieran surgir al utilizar bibliotecas de manejo de matrices y generación de número aleatorios.
- Ubuntu es libre y de código abierto, siendo su instalación completamente gratuita.
- Ubuntu está orientado a usuarios con conocimientos medios en informática, estando enfocado a la facilidad de uso por parte del usuario. Esto ayudará a evitar problemas secundarios durante el desarrollo de esta parte del proyecto.
- El equipo de desarrollo posee experiencia en el uso de anteriores versiones de esta distribución de GNU/Linux, por lo que se evitan complicaciones innecesarias de interacción con otro sistema operativo.

En lo referente al uso de Microsoft Windows para el resto del proyecto, se debe señalar que la versión que se va a utilizar es *Windows 7 Home Premium Service Pack 1*. Las razones que llevan a esta elección son las siguientes:

- Se cree necesaria la utilización de este sistema operativo para el desarrollo en Java, existiendo ya una familiaridad de uso con las versiones de los entornos de desarrollo para Microsoft Windows.
- Windows 7 es el sistema operativo ya instalado en los equipos de los que dispone el equipo de desarrollo.
- Posee compatibilidad con todas las herramientas de visualización de grafos que se van a utilizar en este proyecto.

3.2.3 Entorno de desarrollo

En esta sección se van a valorar qué entornos de desarrollo se adaptan mejor a los lenguajes de programación y sistemas operativos elegidos. Para la parte de desarrollo que se va a realizar en Java tendremos en cuenta dos IDEs muy populares como son NetBeans [22] y Eclipse [23], y también se prestará atención a un editor de texto muy completo: Notepad++ [24]. En lo referente a la parte que se va a

implementar en Ubuntu con el lenguaje C, se van a valorar dos editores de texto: Emacs [25] (que es más potente) y Gedit [26] (que es más sencillo para principiantes), ya que la complejidad de esta parte no es muy elevada, posibilitando su desarrollo con un editor de texto en lugar de un IDE más sofisticado; posteriormente la compilación se hará por consola utilizando el compilador GCC [21].

A continuación se exponen algunas características de estos entornos y editores de texto:

3.2.3.1 NetBeans

Es un IDE multiplataforma desarrollado por Sun Microsystems (Oracle Corporation), que tiene una licencia de código abierto y libre, por lo que su instalación es gratuita. Es un entorno de desarrollo pensado especialmente para las implementaciones en Java, por lo que se ajusta perfectamente a este proyecto. Además posee una interfaz muy intuitiva para desarrolladores que estén familiarizados con otros IDEs. Para este proyecto, es de especial interés que NetBeans tiene integrado un *plug-in* para facilitar el diseño de interfaces de manera visual.

3.2.3.2 Eclipse

Eclipse es un IDE multiplataforma creado originalmente por IBM pero desarrollado en la actualidad por la Fundación Eclipse. Al igual que NetBeans, es gratuito y muy potente. Eclipse ha logrado ser más popular que NetBeans en estos últimos años dentro del entorno universitario, debido quizá a que es un IDE más completo. Existe la posibilidad de instalar *plug-ins* muy variados para añadir funcionalidades específicas al entorno de desarrollo, como por ejemplo el control de versiones dentro de un equipo de desarrollo, diseño de interfaces de manera visual, etc.

3.2.3.3 Notepad++

Es un editor de texto con soporte para una gran variedad de lenguajes, entre los que se incluyen Java, C, XML, C++, Pascal, Python, Lisp, etc. Es una aplicación gratuita, ya que se distribuye bajo la licencia GNU GPL. Esta herramienta sólo está disponible para Microsoft Windows y es muy potente. Es similar al bloc de notas pero incluyendo mucha más funcionalidad, por lo que es muy útil para proyectos que no tienen mucha complejidad.

3.2.3.4 Emacs

Emacs es un editor de texto multiplataforma que forma parte del proyecto GNU. Es un editor muy potente, que ha conseguido gran popularidad entre programadores experimentados. Tal y cómo se define en su página web [25]: *"Emacs es un editor extensible, personalizable, auto-documentado y de tiempo real".* Este editor de texto está integrado en la versión de Ubuntu que se va a utilizar en este proyecto. La desventaja principal es que si no se está familiarizado con este editor puede presentar una dificultad extra a programadores principiantes.

3.2.3.5 Gedit

Es un editor de texto multiplataforma diseñado como un editor de texto de carácter general. Es una aplicación muy simple y sencilla de usar, similar a otros

editores de texto convencionales. Incluye opciones atractivas para los programadores, como coloreado de la sintaxis, posee pestañas que permite la edición de varios archivos simultáneamente, numeración de líneas, y un sistema de *plug-ins* que permite añadir características a la aplicación. Es uno de los editores de texto integrado en la versión 12.04 TLS de Ubuntu, que es la que se va a utilizar en parte de este proyecto.

3.2.3.6 Selección

Como ya se ha comentado anteriormente, la parte de muestreo va a ser implementada bajo el sistema operativo *Ubuntu 12.04 TLS*. Debido a la relativa simplicidad del programa a desarrollar para muestrear grafos (no requiere interfaz y su único objetivo es aplicar un algoritmo de muestreo sobre un grafo, generando un archivo de salida con los datos del nuevo grafo de menor tamaño obtenido) no se ha creído necesario utilizar un sofisticado IDE, es por ello que se han valorado dos editores de texto ya disponibles en el sistema operativo para la implementación de esta parte. Finalmente, el editor elegido ha sido Gedit, debido principalmente a su sencillez, ya que la experiencia del equipo de desarrollo con Emacs es escasa, lo que podría provocar dificultades innecesarias. A pesar de ser un editor muy sencillo, Gedit ofrece todas las características necesarias para implementar esta parte del proyecto. Posteriormente será necesario compilar los archivos creados en Gedit con el compilador GCC, también integrado como herramienta en la versión de Ubuntu con la que se va a trabajar.

En cuanto a la parte de desarrollo en Windows 7, se ha decidido utilizar principalmente el IDE de Eclipse, ya que es con el que más familiarizado se encuentra el equipo de desarrollo. Concretamente, la versión utilizada será *Eclipse Java EE IDE Helios Service Release 2*. Posteriormente, para realizar la interfaz del sistema se ha decidido usar *NetBeans IDE*, ya que posee una característica ya instalada para facilitar el diseño de interfaces de manera *drag and drop* y generando código de manera automática. Así se facilitara la implementación de la interfaz de manera considerable, ahorrando tiempo y dificultades. La versión que se utilizará será *NetBeans IDE 7.1.2*. Finalmente, para la edición de texto de manera puntual o la visualización de archivos generados por la aplicación durante las pruebas de funcionamiento, se empleará el editor Notepad++, debido principalmente a su sencillez y rapidez. La versión que se utilizará será *Notepad++ v5.9.6.2*.

3.2.4 Otras herramientas

Existen otras aplicaciones que se van a utilizar en el desarrollo de este proyecto, que no se engloban en los apartados anteriores pero también tienen una gran importancia. Es por ello que se exponen a continuación, indicando la función que van a cumplir en el marco de este TFG.

3.2.4.1 MinGW

Debido a que la parte de muestreo de grafos se va a implementar en Ubuntu en lenguaje C, es necesario crear un ejecutable de ese programa que lo sea también en Microsoft Windows, ya que el resto de la aplicación que contiene la interfaz y la funcionalidad básica va a ser desarrollada bajo este sistema operativo. Es por esto que

es necesario compilar el programa que se desarrolle en C para la plataforma Win32, que es la API que utiliza Windows. Debido a esto se analizan dos posibles herramientas que puedan realizar esta labor: MinGW [27] y Cygwin [28]. Finalmente se utilizará MinGW, debido a que se pueden integrar las librerías necesarias de manera sencilla. En concreto, la versión que se va a utilizar es la *v.4.7.0*.

3.2.4.2 Microsoft Office

Microsoft Office [29] es una suite ofimática de Microsoft que ofrece varias herramientas de oficina. Los programas que se van a utilizar en este proyecto son los siguientes:

- **Microsoft Word 2007.** Procesador de textos muy popular que se utilizará para desarrollar la documentación de este TFG.
- **Microsoft Excel 2007.** Hoja de cálculo utilizada para crear el presupuesto y analizar los datos de las pruebas ejecutadas. Además esta herramienta es necesaria para la visualización de los grafos generados por la aplicación, ya que la herramienta principal que se va a utilizar para este fin es un complemento de este programa.
- **Microsoft PowerPoint 2007.** Programa dedicado a realizar presentaciones mediante diapositivas. Se utilizará para realizar la presentación de este TFG.
- **Microsoft Visio 2010.** Con esta herramienta se realizarán los diagramas y figuras que se necesiten incluir en este documento para representar procesos de manera gráfica.
- **Microsoft Project 2010.** Es un gestor de proyectos que se utilizará para la estimación y contabilidad de los recursos necesarios para desarrollar este TFG, como por ejemplo el tiempo y los costes monetarios.

Para la mayoría de herramientas la versión a utilizar es *Microsoft Office 2007 v.12.0.6661.5000 Service Pack 3*, debido a que es la versión ya instalada en los equipos disponibles. Las demás versiones serán descargadas de la librería MSDN de Microsoft [30], esto es posible debido al acuerdo existente entre Microsoft y la UC3M. La versión a utilizar de Microsoft Visio 2010 y Microsoft Project 2010 será la *v14.0.6123.5001 (32 bits)*.

3.2.5 Configuración del entorno de desarrollo

Tras el análisis de todas las opciones anteriores de configuración del entorno de desarrollo, se ha obtenido una combinación de todas ellas que hará posible el correcto desarrollo de este TFG. Las decisiones tomadas se han basado principalmente en los siguientes criterios:

- Disponibilidad de la herramienta para el equipo de desarrollo.
- Facilidad de uso y experiencia previa con la herramienta.
- Compatibilidad con otras herramientas que se van a emplear en el desarrollo del proyecto.

De esta manera, la configuración final del entorno de desarrollo se resume en la Tabla 3.

	Sistema operativo	Lenguaje de programación	Entorno de desarrollo	Otras herramientas
Muestreo de grafos	Ubuntu 12.04 TLS	C	Gedit y GCC	MinGW y Microsoft Office
Generación del archivo de visualización de grafos y diseño de la interfaz de la aplicación	Windows 7	Java	NetBeans, Eclipse y Notepad++	

Tabla 3. Entorno de desarrollo completo utilizado

3.3 Definición del sistema

En esta sección se definen en detalle los componentes del equipo de trabajo y las características técnicas de los equipos que se van a utilizar en el desarrollo de este TFG.

3.3.1 Equipo de trabajo

Existen varios tipos de recursos que se van a utilizar a lo largo del desarrollo de este proyecto, que son los siguientes:

- **Recursos hardware:** son los recursos tangibles que se van a emplear durante el desarrollo de este proyecto, es decir, todos los elementos físicos informáticos involucrados en este TFG. Estos recursos se especifican a lo largo del presente documento.
- **Recursos software:** son todos los componentes lógicos informáticos necesarios para el desarrollo del proyecto. En general, son todas las aplicaciones informáticas que se van a emplear durante este TFG. Estos recursos se especifican a lo largo del presente documento.
- **Recursos humanos:** en este caso el desarrollo del proyecto es individual, por lo que el equipo de desarrollo estará compuesto por el alumno encargado de la elaboración del TFG y los tutores del mismo. En el apartado 3.3.1.1 se define más en detalle este equipo de trabajo.
- **Otros:** existen otros recursos que no se pueden englobar dentro de estos tres grupos, pero que también están involucrados en el desarrollo del proyecto.

3.3.1.1 Descripción del equipo de trabajo

El equipo de desarrollo de este TFG está compuesto por los siguientes integrantes:

- **Lorena Camino Rey:** estudiante de Grado en Ingeniería en Informática con especialidad en Ingeniería de Computadores y alumna encargada del presente TFG.
- **David Expósito Singh:** tutor de este TFG, profesor del Departamento de Informática e integrante del grupo de Investigación de Arquitectura de Computadores, Comunicaciones y Sistemas (ARCOS) [31] de la Escuela Politécnica Superior Universidad Carlos III de Madrid.
- **Maria-Cristina Marinescu:** tutora de este TFG, profesora del Departamento de Informática e integrante del grupo de Investigación de Arquitectura de Computadores, Comunicaciones y Sistemas (ARCOS) [31] de la Escuela Politécnica Superior Universidad Carlos III de Madrid.

Debido al carácter individual de este TFG, Lorena Camino Rey realizará a la vez las distintas tareas necesarias para el desarrollo del proyecto:

- *Jefe de proyecto:* gestionando, planificando y coordinando las tareas que componen el proyecto.

- *Gestor de calidad:* asegurando en todo momento que el desarrollo del proyecto es el adecuado y que los productos generados cumplen con los estándares de calidad acordados.
- *Analista:* definiendo los requisitos de manera completa y coherente tras las necesidades marcadas inicialmente por los tutores.
- *Diseñador:* diseñando el sistema completo que se va a desarrollar, cumpliendo con los requisitos definidos en el análisis.
- *Programador:* implementando los distintos aspectos de la aplicación cumpliendo con todas las pautas definidas en el análisis y el diseño.
- *Gestor de pruebas:* definiendo y realizando pruebas específicas y globales que verifiquen el correcto funcionamiento de cada módulo del sistema.
- *Gestor de implantación:* configurando e instalando el sistema final producto de todas las fases de desarrollo.

3.3.2 Entorno tecnológico

En este apartado se detallan las características de los equipos que se van a utilizar en el desarrollo de este proyecto, así como el software involucrado durante el mismo.

3.3.2.1 Hardware

Los equipos empleados para el desarrollo del sistema serán los siguientes:

- **Ordenador portátil Acer**
  - **Modelo:** Acer Aspire 5750.
  - **Procesador:** Intel Core i5-2410M/2,3 Ghz (Dual Core).
  - **Memoria:** 4 GB DDR3 SDRAM.
  - **Disco duro:** 750 GB Serial ATA-300 (5400 rpm).
  - **Sistemas operativos:** Microsoft Windows 7 Home Premium Edición 64 bits y Ubuntu 12.04 TLS.
  - **Tarjeta gráfica:** tarjeta integrada Intel HD Graphics 3000.
  - **Monitor:** Acer pantalla integrada 15,6 pulgadas.
  - **Teclado:** teclado integrado.
  - **Ratón:** touchpad integrado.
- **Ordenador de sobremesa Asus**
  - **Modelo:** AS V3-M2A69DG
  - **Procesador:** AMD Athlon 64 X2 Dual Core Processor 4800+/2,50 GHz.
  - **Memoria:** 4 GB DDR2
  - **Disco duro:** 250 GB Serial ATA (722 rpm).
  - **Sistemas operativos:** Microsoft Windows XP Home Edition Service Pack 3.
  - **Tarjeta gráfica:** ATI Radeon Xpress 1250 (700 MB)
  - **Monitor:** LG Flatron L1511S.
  - **Teclado:** Microsoft Natural PS/2 Keyboard.

  - **Ratón:** Samsung MO-170B.
- **Impresora**
  - **Modelo:** HP PSC 1350 All-in-one Printer series.
- **Router**
  - **Modelo:** Observa Telecom AW4062.

3.3.2.2 Software

El software a utilizar durante el desarrollo del proyecto es el siguiente:

- **Eclipse Helios [23]:** IDE multiplataforma, con disponibilidad para la programación en varios lenguajes. Es una herramienta gratuita muy popular y potente, muy utilizada por los programadores de Java. Permite la adición de características avanzadas y específicas mediante la instalación de *plug-ins*.
- **NetBeans 7.1.2 [22]:** entorno de desarrollo multiplataforma, especialmente orientado a la programación en Java. Es una herramienta similar a Eclipse.
- **Notepad++ v5.9.6.2 [24]:** editor de texto con soporte para una gran variedad de lenguajes de programación y que dispone de multitud de opciones.
- **Gedit [26]:** editor de texto incluido en la distribución de Ubuntu 12.04 TLS que ofrece soporte para varios lenguajes de programación y que ofrece muchas opciones.
- **GCC [21]:** colección de compiladores GNU del lenguaje C, que está integrado en Ubuntu 12.04 TLS.
- **MinGW [27]:** implementación de los compiladores GCC para la plataforma Win32. Va a ser necesario para compilar el programa de la parte de muestreo de manera que sea ejecutable en Microsoft Windows.
- **Microsoft Office 2007 [29]:** conjunto de herramientas ofimáticas que se utilizarán durante el desarrollo de este proyecto para redactar la presente memoria del TFG, realizar el presupuesto y crear la presentación que se realizará del proyecto.
- **Microsoft Visio 2010:** aplicación que permitirá la creación de figuras, diagramas y esquemas a incluir en este documento.
- **Microsoft Project 2010:** software destinado a la gestión de proyectos. Permite llevar un seguimiento, gestionar recursos y tareas, desarrollar planes, etc.
- **COCOMO II [33]:** aplicación utilizada para realizar una estimación de costes utilizando un modelo matemático.

A continuación, se puede observar un esquema que representa el entorno tecnológico que se va a utilizar:

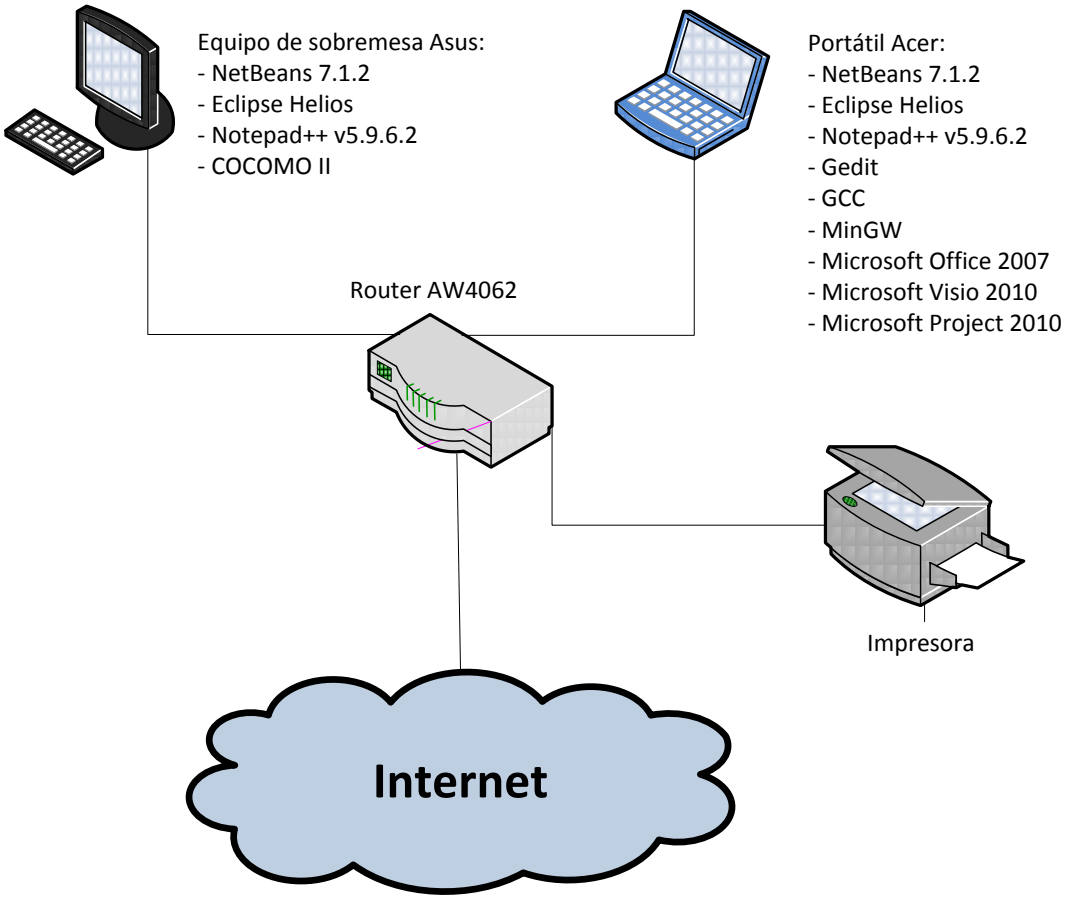

Ilustración 4. Entorno tecnológico

3.4 Análisis de requisitos

En este apartado se definen los requisitos software, que son los requisitos que debe cumplir el sistema. Estos requisitos están basados en las indicaciones dadas por los tutores y en las restricciones que irán surgiendo a lo largo del desarrollo de este TFG.

La identificación de requisitos se extraerá mediante reuniones con los tutores y conversaciones a través de correo electrónico con Gonzalo Martín Cruz, cuyo PFC es el precursor de este TFG [2, 3]. Además de los requisitos que surjan de las necesidades indicadas en estas conversaciones y reuniones, otra parte de los requisitos serán recomendados por el equipo de desarrollo para poder satisfacer de manera correcta la funcionalidad pedida. Tras este proceso, el conjunto de requisitos recogidos definirán de manera completa la funcionalidad y las restricciones del sistema, siendo un conjunto de requisitos concreto y no ambiguo.

El formato para definir los requisitos está basado en tablas que contienen los atributos que se indican en la metodología Métrica v.3, adaptados a las características de este proyecto teniendo en cuenta la experiencia personal del equipo de desarrollo.

La definición de los requisitos del sistema evoluciona a lo largo del proyecto, ya que deben ser revisados y modificados según vayan surgiendo cambios. La colección que se expone a continuación es la versión definitiva de estos requisitos.

Existen una serie de restricciones debido a las características del proyecto que se deben tener en cuenta antes de la redacción de requisitos. Principalmente, hay que tener en cuenta que el proyecto se engloba dentro de un TFG, por lo que los recursos existentes son muy limitados, ciñéndose a los que posea el alumno encargado del TFG y los que tenga disponibles en la UC3M, que es la universidad en la que se realiza el proyecto. Es por esto, que una de las restricciones más importantes es ajustarse a los recursos hardware y software de los que se disponga.

3.4.1 Formato de especificación de requisitos

Teniendo en cuenta todo lo señalado anteriormente se pasa a definir la plantilla utilizada para la definición de requisitos:

IDENTIFICADOR	XXX-xxx		
NOMBRE			
DESCRIPCIÓN			
ESTABILIDAD			
PRIORIDAD	☐Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☐ Alumno
CLARIDAD	☐ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☐ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☐ Deseable ☐ Opcional		

Tabla 4. Plantilla de requisitos

Los atributos incluidos en la plantilla de requisitos son los siguientes:

- **Identificador:** cada requisito tendrá un identificador unívoco, que permita especificar el requisito de manera clara y concreta para facilitar el orden y la búsqueda del mismo. El identificador se compondrá de tres letras seguidas de tres dígitos (XXX-xxx). La primera parte del identificador podrá ser **RUC** si se trata de un requisito de usuario capacidad o **RUR** si es un requisito de usuario restricción. La segunda parte del identificador está compuesto por un número correlativo que empezará la numeración en 001 y se incrementará con cada requisito.
- **Nombre:** definición clara y concisa de la naturaleza del requisito.
- **Descripción:** exposición breve sobre qué trata el requisito.
- **Estabilidad:** indica el período de invariabilidad del requisito durante el desarrollo del proyecto.
- **Prioridad:** define la importancia relativa del requisito en el proyecto. Los valores posibles son "*alta*", "*media*" o "*baja*".
- **Fuente:** indica el origen del requisito, indicando si ha sido propuesto por los tutores del proyecto o por el alumno. Los requisitos que han sido indicados por el autor del PFC precursor de este proyecto han sido incluidos como requisitos propuestos por los tutores.
- **Claridad:** indica si el requisito tiene una o varias interpretaciones posibles. Los valores posibles son "*alta*", "*media*" o "*baja*".
- **Verificabilidad:** indica el grado de comprobación de si el requisito ha sido añadido en el diseño de la aplicación, para ser integrado en la implementación posterior. Los valores posibles son "*alta*", "*media*" o "*baja*".
- **Necesidad:** indica la incidencia que tiene el requisito en la funcionalidad del sistema, indicando el nivel de indispensabilidad del requisito en el proyecto. Los posibles valores son:
    - Necesario: el requisito es indispensable para el desarrollo del proyecto
    - Deseable: es un requisito importante para el buen desarrollo del proyecto, pero no indispensable.

- Opcional: la importancia del requisito para el buen desarrollo del proyecto es baja, pero supondría la adición de una característica más para el mismo.

3.4.2 Requisitos de capacidad

IDENTIFICADOR	RUC-001		
NOMBRE	Muestreo de grafos.		
DESCRIPCIÓN	El sistema permitirá al usuario obtener una muestra de menor tamaño de un grafo inicial especificado por él.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 5. Requisito de capacidad RUC-001

IDENTIFICADOR	RUC-002		
NOMBRE	Generación de archivo de visualización de grafos.		
DESCRIPCIÓN	El sistema permitirá al usuario generar un archivo válido para la visualización del grafo mediante las herramientas de visualización de grafos utilizadas, a partir de otro archivo de representación del grafo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 6. Requisito de capacidad RUC-002

IDENTIFICADOR	RUC-003		
NOMBRE	Generación de archivo de representación del grafo.		
DESCRIPCIÓN	El sistema permitirá al usuario generar un archivo de representación del grafo a partir de un archivo de visualización del mismo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 7. Requisito de capacidad RUC-003

IDENTIFICADOR	RUC-004		
NOMBRE	Configuración de atributos.		
DESCRIPCIÓN	Se permitirá al usuario configurar los atributos de visualización de los vértices y las aristas del grafo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 8. Requisito de capacidad RUC-004

IDENTIFICADOR	RUC-005		
NOMBRE	Información sobre el sistema.		
DESCRIPCIÓN	El usuario podrá acceder a un apartado que especifique información sobre los creadores de la aplicación y la naturaleza de la misma.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 9. Requisito de capacidad RUC-005

IDENTIFICADOR	RUC-006		
NOMBRE	Formato de archivo de representación de grafos.		
DESCRIPCIÓN	El sistema permitirá la introducción de archivos de representación de grafos, para realizar un muestreo y/o generación de un archivo de visualización del mismo, que tengan las siguientes características: • Representación del grafo mediante una matriz dispersa con esquema CSC (véase Anexo E). • Extensión del archivo "*.dat*".		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 10. Requisito de capacidad RUC-006

IDENTIFICADOR	RUC-007		
NOMBRE	Formato de archivo de visualización de grafos.		
DESCRIPCIÓN	El sistema permitirá la generación de archivos de visualización de grafos con las siguientes características: • Representación del grafo mediante un archivo de tipo *Graphml*, con sintaxis XML (véase Anexo F). • Extensión del archivo "*.graphml*".		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 11. Requisito de capacidad RUC-007

IDENTIFICADOR	RUC-008		
NOMBRE	Formato de archivo con estados de los vértices.		
DESCRIPCIÓN	El sistema permitirá la introducción de archivos con los estados de los vértices, necesarios para realizar un muestreo y/o generación de un archivo de visualización del mismo, que tengan las siguientes características: • Indique los valores de los vértices mediante un valor numérico, y estén separados por un espacio entre ellos. • Extensión del archivo "*.dat*".		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 12. Requisito de capacidad RUC-008

IDENTIFICADOR	RUC-009		
NOMBRE	Configuración de los vértices de los grafos.		
DESCRIPCIÓN	El usuario podrá configurar los siguientes atributos de los vértices de los grafos: • Número de opción de configuración. • Color. • Forma. • Tamaño. • Opacidad. • Ruta de la imagen. • Visibilidad. • Etiqueta. • Color de la etiqueta. • Posición de la etiqueta. • Texto emergente.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 13. Requisito de capacidad RUC-009

IDENTIFICADOR	RUC-010		
NOMBRE	Configuración de las aristas de los grafos.		
DESCRIPCIÓN	El usuario podrá configurar los siguientes atributos de las aristas de los grafos: • Número de opción de configuración. • Color. • Ancho. • Estilo. • Opacidad. • Visibilidad. • Etiqueta. • Color de la fuente de la etiqueta. • Tamaño de la fuente de la etiqueta.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 14. Requisito de capacidad RUC-010

IDENTIFICADOR	RUC-011		
NOMBRE	Información al usuario.		
DESCRIPCIÓN	Se mantendrá informado al usuario sobre los procesos que realice la aplicación, así como de información relevante para él.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☐ Alta ☑ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 15. Requisito de capacidad RUC-011

IDENTIFICADOR	RUC-012		
NOMBRE	Datos para muestreo y visualización de grafo.		
DESCRIPCIÓN	Los datos obligatorios para realizar el muestreo de un grafo y la generación del archivo de visualización del mismo son: • Ruta del archivo con la representación del grafo. • Ruta del archivo con los valores de los estados de los vértices. • Porcentaje del muestreo a realizar. • Tener almacenada en el sistema la configuración de los atributos de los vértices y las aristas del grafo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 16. Requisito de capacidad RUC-012

IDENTIFICADOR	RUC-013		
NOMBRE	Datos para generar archivo de representación de grafo.		
DESCRIPCIÓN	Los datos obligatorios para generar el archivo de representación de un grafo son: • Ruta del archivo de visualización del grafo en formato *Graphml*. • Nombre del archivo que se va a generar. • Nombre del archivo con los estados de los vértices que se va a generar. • Tener almacenada en el sistema la configuración de los atributos de los vértices y las aristas del grafo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 17. Requisito de capacidad RUC-013

IDENTIFICADOR	RUC-014		
NOMBRE	Datos para añadir una opción de configuración un vértices y aristas		
DESCRIPCIÓN	Los datos obligatorios para añadir una opción de configuración para vértices y/o aristas son: • Número de opción de configuración.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 18. Requisito de capacidad RUC-014

IDENTIFICADOR	RUC-015		
NOMBRE	Formato de archivos de configuración de vértices y aristas.		
DESCRIPCIÓN	El formato de los archivos de configuración de vértices y aristas será el siguiente (véase Anexo G): • Cabecera (primera línea): con los nombres de los atributos separados por el carácter de separación "/". • Una línea por cada opción de configuración que exista, de tal manera que aparezcan los valores de los atributos indicados en la cabecera del fichero en ese mismo orden y separados también por el carácter de separación "/". • El primer valor de cada línea con una opción de configuración debe ser el número que identifica a esa opción de configuración. • Existirá un valor por defecto que se utilizará para los vértices o aristas que no se correspondan con ninguna de las opciones de configuración almacenadas. El identificador de la opción de configuración por defecto será el carácter "*" en lugar de un número.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☐ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 19. Requisito de capacidad RUC-015

IDENTIFICADOR	RUC-016		
NOMBRE	Configuración de los atributos de manera interactiva.		
DESCRIPCIÓN	El usuario podrá configurar los atributos de los vértices y las aristas de los grafos de manera interactiva, pudiendo realizar las siguientes acciones: • Cargar desde los archivos de configuración de vértices y aristas una configuración almacenada previamente. • Añadir una opción de configuración. • Modificar una opción de configuración. • Eliminar una opción de configuración. • Guardar la configuración creada o modificada en los archivos de configuración adecuados.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 20. Requisito de capacidad RUC-016

IDENTIFICADOR	RUC-017		
NOMBRE	Datos para cargar una configuración previa.		
DESCRIPCIÓN	Los datos necesarios para cargar una configuración previa de los atributos de los vértices y aristas son los siguientes: • Ruta del archivo con la configuración de los vértices. • Ruta del archivo con la configuración de las aristas.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 21. Requisito de capacidad RUC-017

IDENTIFICADOR	RUC-018		
NOMBRE	Datos para guardar una configuración de atributos.		
DESCRIPCIÓN	Los datos necesarios para guardar una configuración de los atributos de los vértices y las aristas son los siguientes: • Nombre del archivo de configuración de vértices. • Nombre del archivo de configuración de aristas.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 22. Requisito de capacidad RUC-018

IDENTIFICADOR	RUC-019		
NOMBRE	Valor del porcentaje de muestreo.		
DESCRIPCIÓN	El valor del porcentaje de muestreo puede estar comprendido entre 0% y 100%.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 23. Requisito de capacidad RUC-019

IDENTIFICADOR	RUC-020		
NOMBRE	Valor del atributo de configuración "número de opción de configuración".		
DESCRIPCIÓN	Los posibles valores del atributo "número de opción de configuración" son: • Por defecto. • Un valor numérico comprendido entre 0 y 20.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 24. Requisito de capacidad RUC-020

IDENTIFICADOR	RUC-021		
NOMBRE	Valor del atributo de configuración “color”.		
DESCRIPCIÓN	Los posibles valores del atributo “color” son: • Black. • Blue. • Brown. • Fuchsia. • Green. • Lime. • Olive. • Orange. • Red. • Silver. • White. • Yellow. • Purple.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 25. Requisito de capacidad RUC-021

IDENTIFICADOR	RUC-022		
NOMBRE	Valor del atributo de configuración “forma”.		
DESCRIPCIÓN	Los posibles valores del atributo “forma” son: • Circle. • Disk. • Sphere. • Square. • Solid Square. • Diamond. • Solid Diamond. • Triangle. • Solid Triangle. • Label. • Image. • Un valor numérico comprendido entre 1 y 11.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 26. Requisito de capacidad RUC-022

IDENTIFICADOR	RUC-023		
NOMBRE	Valor del atributo de configuración “tamaño”.		
DESCRIPCIÓN	El valor del atributo “tamaño” puede ser un valor numérico comprendido entre 1 y 100.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 27. Requisito de capacidad RUC-023

IDENTIFICADOR	RUC-024		
NOMBRE	Valor del atributo de configuración “opacidad”.		
DESCRIPCIÓN	El valor del atributo “opacidad” puede ser un valor numérico comprendido entre 0 y 100.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 28. Requisito de capacidad RUC-024

IDENTIFICADOR	RUC-025		
NOMBRE	Valor del atributo de configuración “ruta de imagen”.		
DESCRIPCIÓN	El valor del atributo “ruta de imagen” debe ser la ruta de una imagen para asignarla al vértice en la visualización del grafo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 29. Requisito de capacidad RUC-025

IDENTIFICADOR	RUC-026		
NOMBRE	Valor del atributo de configuración “visibilidad”.		
DESCRIPCIÓN	Los posibles valores del atributo “visibilidad” son: • Show. • Skip. • Hide. • 0. • 1. • 2.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 30. Requisito de capacidad RUC-026

IDENTIFICADOR	RUC-027		
NOMBRE	Valor del atributo de configuración “etiqueta”.		
DESCRIPCIÓN	El valor del atributo “etiqueta” debe ser una palabra que se le asigne al vértice o arista en la visualización del grafo.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 31. Requisito de capacidad RUC-027

IDENTIFICADOR	RUC-028		
NOMBRE	Valor del atributo de configuración “color de la etiqueta”.		
DESCRIPCIÓN	Los posibles valores del atributo “color de la etiqueta” son: • Black. • Blue. • Brown. • Fuchsia. • Green. • Lime. • Olive. • Orange. • Red. • Silver. • White. • Yellow. • Purple.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 32. Requisito de capacidad RUC-028

IDENTIFICADOR	RUC-029		
NOMBRE	Valor del atributo de configuración “posición de la etiqueta”.		
DESCRIPCIÓN	El valor del atributo “posición de la etiqueta” puede ser un valor numérico comprendido entre 1 y 5.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 33. Requisito de capacidad RUC-029

IDENTIFICADOR	RUC-030		
NOMBRE	Valor del atributo de configuración “texto emergente”.		
DESCRIPCIÓN	El valor del atributo “texto emergente” debe ser una palabra que se le asigne al vértice para que aparezca durante la visualización del grafo cada vez que se desplace el puntero del ratón sobre ese elemento.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 34. Requisito de capacidad RUC-030

IDENTIFICADOR	RUC-031		
NOMBRE	Valor del atributo de configuración “ancho”.		
DESCRIPCIÓN	El valor del atributo “ancho” puede ser un valor numérico comprendido entre 1 y 10.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 35. Requisito de capacidad RUC-031

IDENTIFICADOR	RUC-032		
NOMBRE	Valor del atributo de configuración “estilo”.		
DESCRIPCIÓN	Los posibles valores del atributo “estilo” son: • Solid. • Dash. • Dot. • Dash Dot. • Dash Dot Dot. • Un valor numérico comprendido entre 1 y 5.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 36. Requisito de capacidad RUC-032

IDENTIFICADOR	RUC-033		
NOMBRE	Valor del atributo de configuración “tamaño de la etiqueta”.		
DESCRIPCIÓN	El valor del atributo “tamaño de la etiqueta” puede ser un valor numérico comprendido entre 1 y 5.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 37. Requisito de capacidad RUC-033

3.4.3 Requisitos de restricción

IDENTIFICADOR	RUR-001		
NOMBRE	Metodología Métrica v.3.		
DESCRIPCIÓN	La metodología a seguir para el desarrollo del proyecto será Métrica v.3.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☑ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 38. Requisito de restricción RUR-001

IDENTIFICADOR	RUR-002		
NOMBRE	Interfaz intuitiva.		
DESCRIPCIÓN	Para facilitar a los usuarios el uso del sistema, la interfaz debe ser intuitiva evitando añadir complejidad innecesaria.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☐ Alta ☑ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☐ Alta ☑ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 39. Requisito de restricción RUR-002

IDENTIFICADOR	RUR-003		
NOMBRE	Elementos hardware.		
DESCRIPCIÓN	El desarrollo del proyecto utilizará lo siguientes equipos: • Equipo de sobremesa Asus - AMD Athlon 64 X2 Dual Core Processor 4800+/2,5 GHz. • Portátil Acer Aspire 5750 - Intel Core i5-2410M/2,3 GHz (Dual Core).		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 40. Requisito de restricción RUR-003

IDENTIFICADOR	RUR-004		
NOMBRE	Elementos software.		
DESCRIPCIÓN	El software a utilizar para el desarrollo del proyecto será: • NetBeans 7.1.2 • Eclipse Helios • Notepad++ v5.9.6.2 • Gedit • GCC • MinGW • Microsoft Office 2007 • Microsoft Visio 2010 • Microsoft Project 2010 • COCOMO II		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 41. Requisito de restricción RUR-004

IDENTIFICADOR	RUR-005		
NOMBRE	Tiempo de ejecución de los procesos.		
DESCRIPCIÓN	El tiempo de ejecución de los procesos que se llevan a cabo en la aplicación (muestreo y generación de archivo de visualización o de representación de grafo) debe mantenerse dentro de unos límites aceptables.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☐ Alta ☑ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 42. Requisito de restricción RUR-005

IDENTIFICADOR	RUR-006		
NOMBRE	Mensajes informativos.		
DESCRIPCIÓN	El sistema informará al usuario del estado de los procesos que esté realizando la aplicación mediante barras de progreso.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☐ Alta ☐ Media ☑ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☐ Deseable ☑ Opcional		

Tabla 43. Requisito de restricción RUR-006

IDENTIFICADOR	RUR-007		
NOMBRE	Mensajes de error y de aviso.		
DESCRIPCIÓN	El sistema informará al usuario de los posibles errores que cometa en el manejo de datos de la aplicación o de problemas que pudieran surgir en forma de mensajes en ventanas emergentes.		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☐ Necesario ☑ Deseable ☐ Opcional		

Tabla 44. Requisito de restricción RUR-007

IDENTIFICADOR	RUR-008		
NOMBRE	Batería de pruebas.		
DESCRIPCIÓN	Se debe definir una batería de pruebas completa para verificar el correcto y completo funcionamiento del sistema. Estas pruebas se especificarán mediante una plantilla que indique: • Identificador de la prueba. • Descripción de la prueba. • Objetivo de la prueba, qué se pretende realizando esa prueba. • Requisito que se pretende corroborar con la prueba. • Resultado esperado. • Resultado de la prueba (éxito o fracaso).		
ESTABILIDAD	Todo el proyecto.		
PRIORIDAD	☑ Alta ☐ Media ☐ Baja	FUENTE	☐ Tutores ☑ Alumno
CLARIDAD	☑ Alta ☐ Media ☐ Baja	VERIFICABILIDAD	☑ Alta ☐ Media ☐ Baja
NECESIDAD	☑ Necesario ☐ Deseable ☐ Opcional		

Tabla 45. Requisito de restricción RUR-008

3.5 Identificación de subsistemas de análisis

Con el fin de facilitar la organización y diseño de la aplicación se realiza una división del sistema de información general en distintos subsistemas, separando la funcionalidad global en varias partes. A continuación, se especifican los subsistemas que se encuentran en esta aplicación y se hace una breve descripción de los mismos:

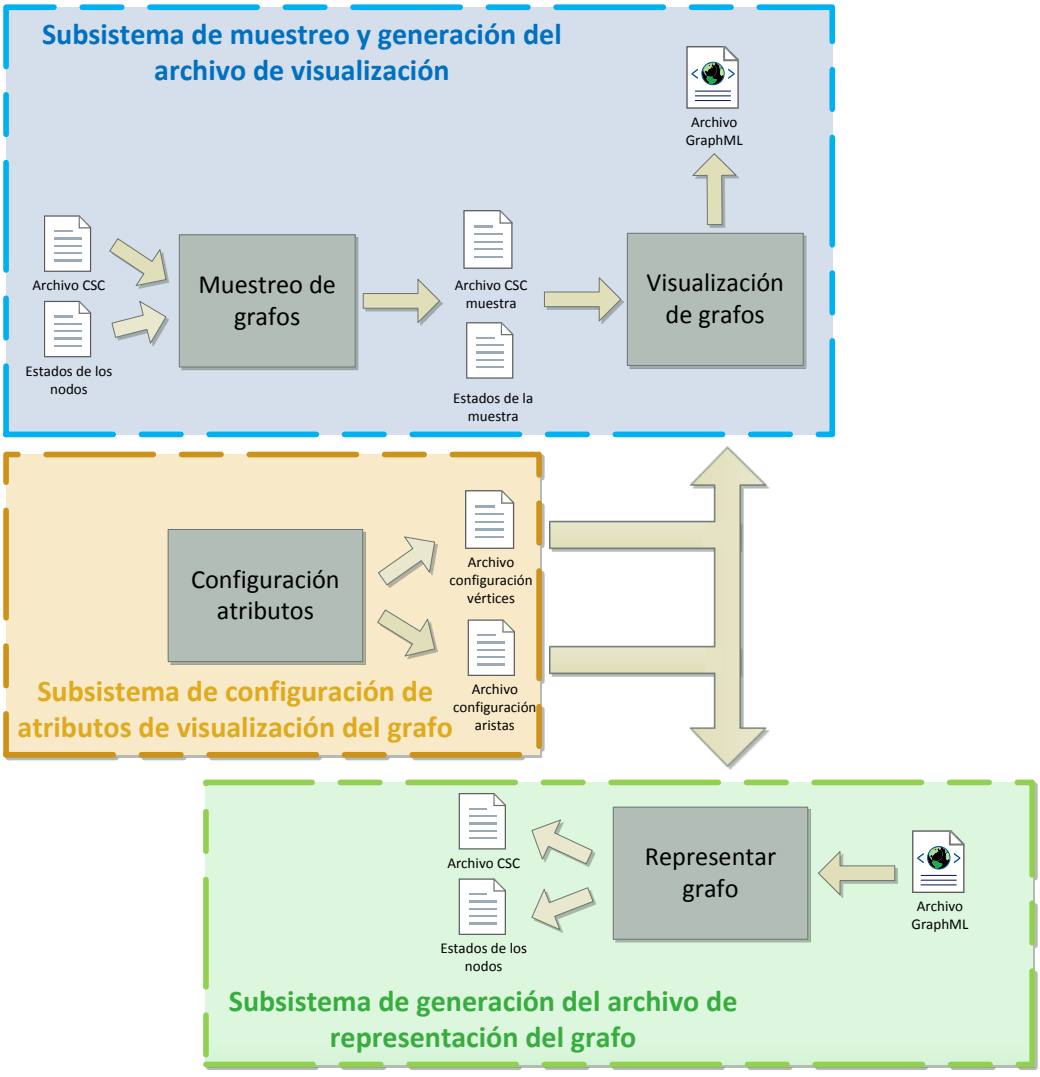

Ilustración 5. Subsistemas de la aplicación

- **Subsistema de muestreo y generación del archivo de visualización.** Este subsistema será el encargado de realizar un muestreo del grafo y seguidamente generar un archivo de visualización de esta muestra. Como ya se ha comentado en varios apartados de este documento, el formato del archivo de representación del grafo debe representar una matriz dispersa utilizando el esquema CSC (véase Anexo E), y el archivo que se generará para poder visualizar la muestra del grafo será un archivo de tipo GraphML (véase Anexo F). Además, será necesario especificar cuál es el archivo en el que se encuentran los estados de los nodos. Durante el proceso se crea un archivo intermedio con la muestra

del grafo en formato de matriz dispersa con esquema CSC, que es el mismo tipo de archivo que el aceptado para el muestreo.

- **Subsistema de generación del archivo de representación de grafo.** Este subsistema será el encargado de realizar el proceso inverso al anterior. En este caso, a partir de un archivo de visualización de grafos, de tipo *GraphML*, se generará un archivo con la representación del grafo como matriz dispersa con esquema CSC y otro archivo con el estado de los vértices de ese grafo. El valor de este estado lo tomará teniendo en cuenta la configuración de los atributos que se haya hecho.
- **Subsistema de configuración de atributos de visualización del grafo.** Este subsistema realiza una labor que es requisito previo para realizar cualquier acción en la aplicación, ya que antes de tratar los archivos de representación de grafos o de visualización de grafos, es necesario que el usuario haya definido cuál es la configuración de los atributos del grafo para así poder tratar estos archivos de manera adecuada. Si no hay ninguna configuración establecida, el sistema no añadirá valores a los atributos de los nodos y las aristas. Por tanto, este subsistema es el encargado de almacenar una configuración válida de los atributos de los vértices y las aristas de los grafos, para que el resto de los subsistemas puedan realizar correctamente su labor. Esta configuración se almacenará en dos archivos, uno para los vértices y otro para las aristas. El usuario puede modificar un archivo ya existente de manera interactiva o crear unos archivos de configuración nuevos partiendo desde cero.
- **Subsistema de información sobre la aplicación.** Este subsistema se dedicará exclusivamente a mostrar información sobre el sistema desarrollado y los autores del mismo.

A continuación, se muestra la relación existente entre los subsistemas descritos anteriormente y los requisitos que se han definido en el apartado 3.4 del presente documento:

Subsistema	Requisitos relacionados
Muestreo y visualización	RUC-001, RUC-002, RUC-006, RUC-007, RUC-008, RUC-011, RUC-012, RUC-019, RUR-001, RUR-002, RUR-003, RUR-004, RUR-005, RUR-006, RUR-007, RUR-008
Generación archivo del grafo	RUC-003, RUC-006, RUC-007, RUC-008, RUC-011, RUC-013, RUR-001, RUR-002, RUR-003, RUR-004, RUR-005, RUR-006, RUR-007, RUR-008
Configuración de atributos	RUC-004, RUC-009, RUC-010, RUC-011, RUC-014, RUC-015, RUC-016, RUC-017, RUC-018, RUC-020, RUC-021, RUC-022, RUC-023, RUC-024, RUC-025, RUC-026, RUC-027, RUC-028, RUC-029, RUC-030, RUC-031, RUC-032, RUC-033, RUR-001, RUR-002, RUR-003, RUR-004, RUR-005, RUR-006, RUR-007, RUR-008
Información del sistema	RUC-005

Tabla 46. Relación entre subsistemas y requisitos de usuario

3.6 Arquitectura del sistema

Tras la definición de los requisitos y de los subsistemas de la aplicación, se debe definir de manera más detallada la arquitectura software del sistema para así comprender cómo funcionará la aplicación de manera global. En el siguiente diagrama se detalla esta arquitectura:

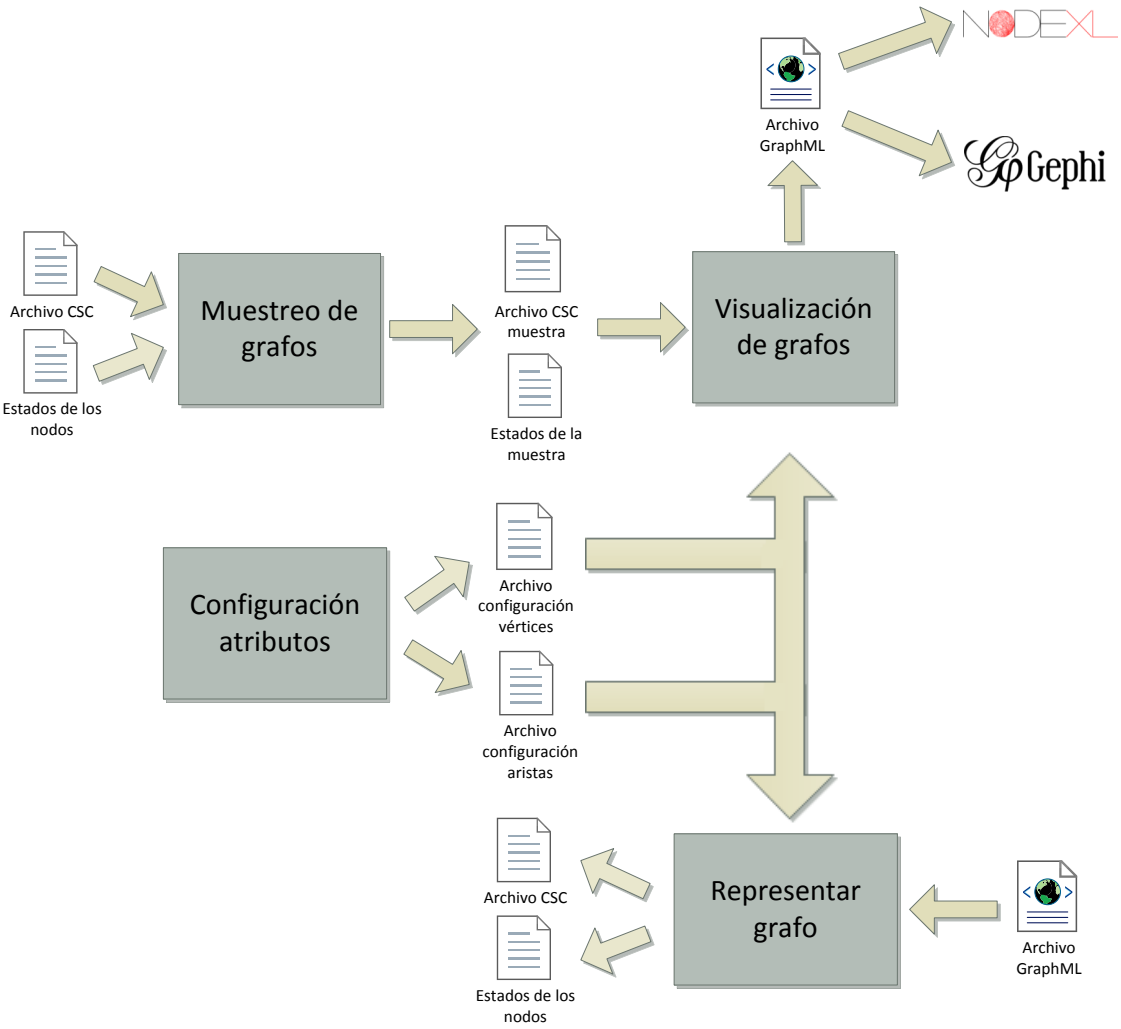

Ilustración 6. Arquitectura software completa del sistema

3.7 Análisis de clases

En este apartado se incluirá una especificación de las clases que estarán involucradas en el sistema, y que participan en el correcto funcionamiento del mismo. Como característica a señalar se debe indicar que el patrón que se pretende seguir en la implementación separará la funcionalidad de la aplicación de la interfaz de la misma, permitiendo así dividir el sistema en dos partes: la lógica de la aplicación y la vista de la misma.

A continuación se incluye un diagrama de clases que representará el sistema al completo, y posteriormente una breve descripción de cada uno de los métodos y atributos que aparecen en él.

3.7.1 Diagrama de clases

Para poder visualizar mejor el diagrama global, se divide en dos partes: las clases relacionadas con la interfaz del sistema y las que implementan la funcionalidad:

Ilustración 7. Diagrama de clases de la parte funcional del sistema

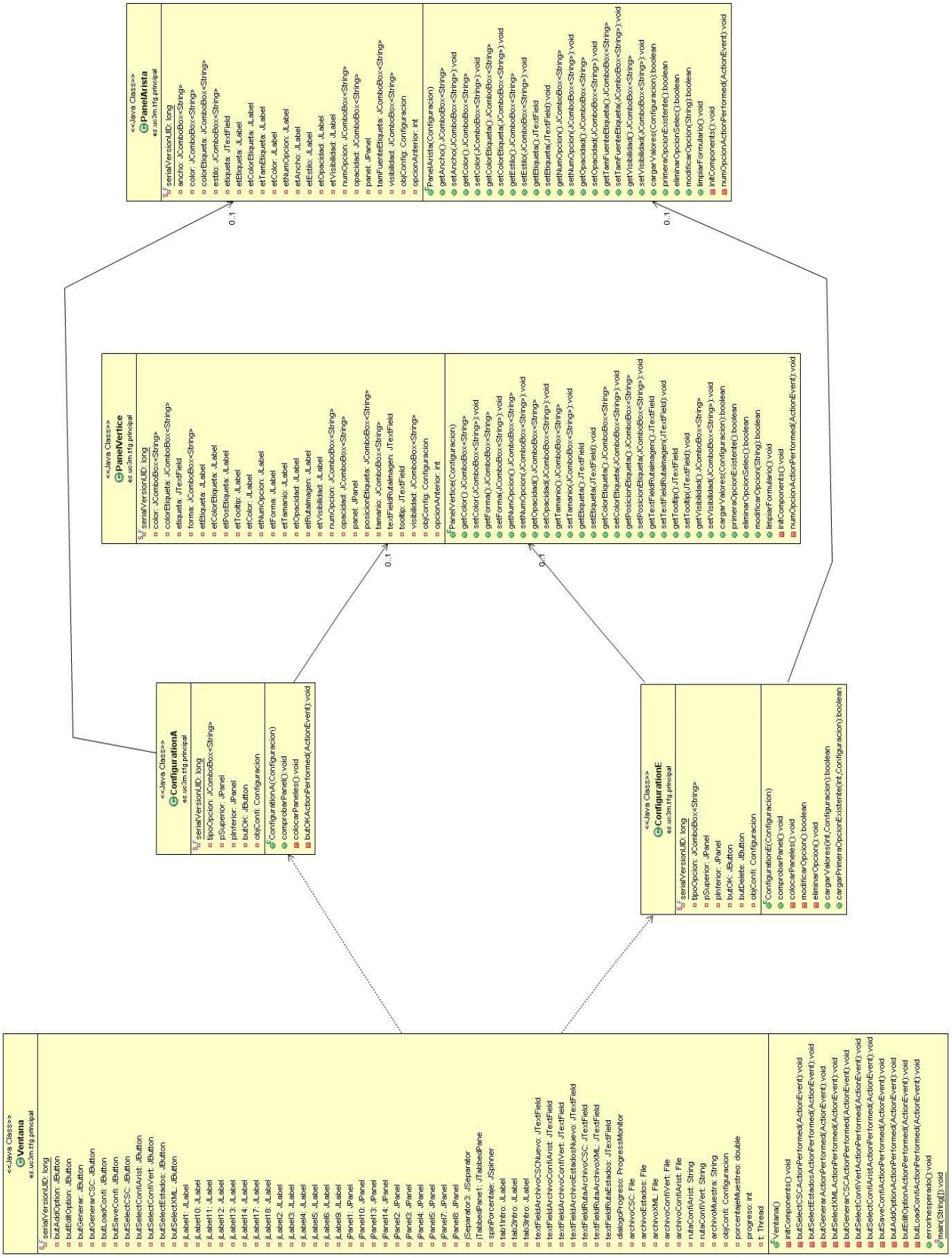

Ilustración 8. Diagrama de clases de la parte de la interfaz del sistema

3.7.2 Identificación de clases

Hay dos partes diferenciadas en este sistema: el muestreo y la generación de archivos. Por tanto se identifican las clases existentes en ambas fases, ya que se han implementado de manera separada y en entornos de desarrollo distintos.

- **Muestreo:**

Este programa está desarrollado en C, y consta de una clase que realiza toda la funcionalidad de muestreo. A continuación se detalla el contenido de este programa:

CLASE	ALBATROSS
Descripción	Genera un grafo de muestra con el número de nodos indicado por parámetro a partir de archivos de representación de grafo mediante matriz dispersa con esquema CSC.
Atributos	**numNodos:** número de nodos del grafo inicial. **nodosMuestra:** número de nodos ***vectorEstados:** *array* con los estados de los vértices del grafo. ***vectorEstadosMuestra:** *array* con los estados de los vértices del grafo de muestra. ***estSampFile:** nombre del fichero con los estados de los vértices de la muestra. ***sampFile:** nombre del archivo de la muestra.
Métodos	**cargarEstados:** almacena los estados en el *array* de estados desde el fichero de estados. **busquedaBinaria:** busca de manera binaria un nodo en el grafo. **busquedaNodo:** busca de manera binaria un nodo en un *array* introducido por parámetro. **aniadirNodo:** se añade un nodo de manera ordenada en un *array* introducido por parámetro. **conectividad:** devuelve el número de aristas que tiene un nodo, es decir, devuelve el grado de un nodo. **fileToCSC:** carga los datos de un grafo desde fichero a una estructura de matriz dispersa interna. **CSCToFile:** almacena en un fichero los datos desde una estructura de matriz dispersa interna. **albatrossSamp:** aplica el algoritmo Albatross para generar una muestra de un grafo.

Tabla 47. Descripción del programa de muestreo

En la implementación de este programa se han utilizado dos librerías externas:

- **GSL** [34]**:** es una librería que permite la generación de números aleatorios. Es necesario utilizarla como parte del algoritmo Albatross de muestreo.
- **CSParse** [35]**:** es una librería que implementa una serie de métodos para sistemas lineales dispersos. Se utiliza para la representación interna del grafo en el programa mediante una matriz dispersa con esquema CSC, lo

que permite el manejo de la misma con los métodos disponibles en esta librería.

- **Aplicación general:**

El esquema general de las clases es el siguiente:

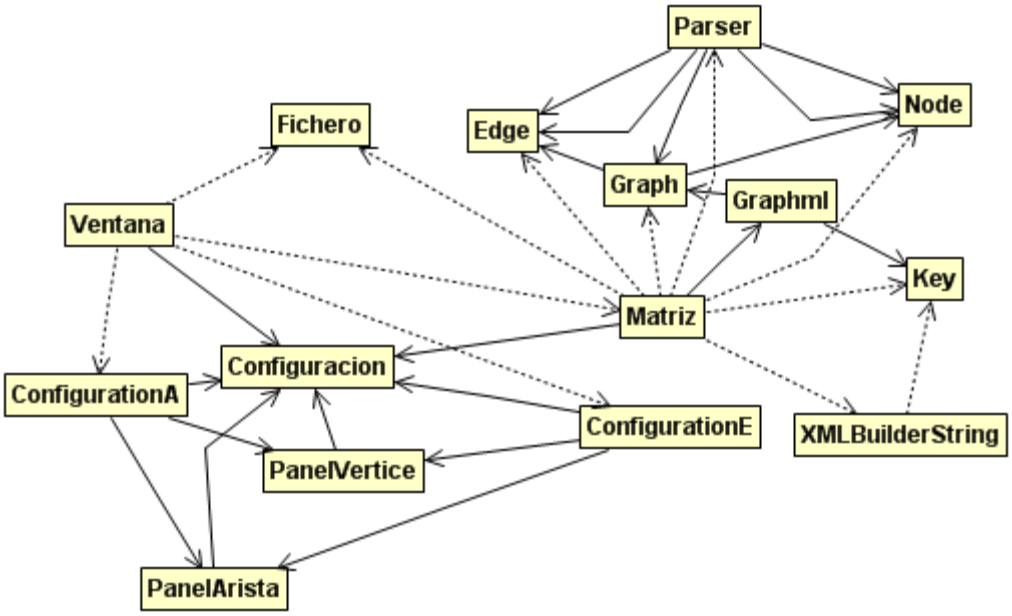

Ilustración 9. Diagrama de clases general

A continuación, se exponen mediante una serie de tablas las clases del proyecto relativas a la funcionalidad. No se ha creído necesario incluir aquí métodos y atributos sobre cómo se implementa la interfaz del sistema, lo realmente importante es el esquema de funcionamiento de la aplicación:

CLASE	NODE
Descripción	Clase que representa un vértice del grafo.
Atributos	**id:** identificador **datas:** colección con los atributos del nodo.
Métodos	**getId:** obtener el identificador. **setId:** modificar el identificador. **getDatas:** obtener la colección de atributos. **setDatas:** modificar la colección de atributos.

Tabla 48. Descripción de la clase "Node"

CLASE	EDGE
Descripción	Clase que representa una arista del grafo.
Atributos	**attr:** identificación del origen y destino de la arista. **data:** colección de atributos de la arista.
Métodos	**getAttr:** obtener el origen y destino de la arista. **setAttr:** modificar el origen y destino de la arista. **getData:** obtener la colección de atributos. **setData:** modificar la colección de atributos.

Tabla 49. Descripción de la clase "Edge"

CLASE	GRAPH
Descripción	Clase que representa el grafo.
Atributos	**attr:** atributos del grafo. **nodes:** colección con los vértices del grafo. **edges:** colección con las aristas del grafo.
Métodos	**getAttr:** obtener los atributos del grafo. **setAttr:** modificar los atributos del grafo. **getNodes:** obtener la colección de nodos. **setNodes:** modificar la colección de nodos. **getEdges:** obtener la colección de aristas. **setEdges:** modificar la colección de aristas.

Tabla 50. Descripción de la clase "Graph"

CLASE	KEY
Descripción	Clase que identifica cada uno de los posibles atributos a configurar en el grafo para aristas y vértices.
Atributos	**id:** identificador del atributo. **forr:** valor que indica si el atributo es para vértices o aristas. **attr_name:** nombre del atributo. **attr_type:** tipo de atributo.
Métodos	**getId:** obtener el identificador. **setId:** modificar el identificador. **getForr:** obtener si el atributos es para vértices o aristas. **setForr:** modificar el valor del atributo “forr”. **getAttr_name:** obtener el nombre del atributo. **setAttr_name:** modificar el nombre del atributo. **getAttr_type:** obtener el tipo de atributo. **setAttr_type:** modificar el tipo de atributo.

Tabla 51. Descripción de la clase "Key"

CLASE	GRAPHML
Descripción	Clase que representa el archivo de tipo *GraphML* que se va a generar.
Atributos	**keys:** lista de atributos configurables para vértices y aristas del grafo. **graph:** objeto del grafo.
Métodos	**getKeys:** obtener la lista de atributos. **setKeys:** modificar la lista de atributos. **getGraph:** obtener el grafo. **setGraph:** modificar el grafo.

Tabla 52. Descripción de la clase "Graphml"

CLASE	MATRIZ
Descripción	Clase que almacena los datos de la matriz que representa el grafo.
Atributos	**matrizDispersa:** lista que representa la matriz con formato disperso en forma CSC. **valores:** lista con los valores de las aristas. **graph:** objeto GraphML, que representa el archivo de visualización del grafo. **nodeName:** nombre de nodo. **estadosNodos:** lista con los estados de los nodos. **ficheroEstadosNodos:** nombre del fichero con el estado de los vértices. **objConfi:** objeto de configuración de los atributos de los elementos del grafo.
Métodos	**getObjConfi:** obtener el objeto que representa la configuración de los atributos de los elementos del grafo. **setObjConfi:** modificar el objeto que representa la configuración de los atributos de los elementos del grafo. **getEstadosNodos:** obtener la lista con los estados de los nodos. **setEstadosNodos:** modificar la lista con los estados de los nodos. **getFicheroEstadosNodos:** obtener el nombre del fichero de estados de los nodos. **setFicheroEstadosNodos:** modificar el nombre del fichero de estados de los nodos. **setNodeName:** modificar el nombre del nodo. **getNodeName:** obtener el nombre del nodo. **getGraph:** obtener el objeto GraphML. **setGraph:** modificar el objeto GraphML.

CLASE	MATRIZ
	getMatrizDispersa: obtener la matriz en formato disperso CSC. **setMatrizDispersa:** modificar la matriz en formato disperso CSC. **getValores:** obtener los valores de las aristas. **setValores:** modificar los valores de las aristas. **setConexionMatrizDispersa:** añadir una conexión entre dos vértices. **numNodos:** obtener el número de nodos de un grafo almacenado en un archivo CSC. **mostrarMatrizDispersa:** mostrar por pantalla las conexiones de la matriz dispersa. **ficheroToMatriz:** transformar los datos de un grafo en formato de matriz dispersa CSC a una lista de conexiones entre nodos. **xmlToMatriz:** transformar los datos de un grafo en formato GraphML (archivo de visualización XML) a una lista de conexiones entre nodos. **matrizToFichero:** volcar la lista de conexiones de un grafo en un fichero en formato CSC. **cargarEstadosNodos:** cargar los estados de los vértices desde un fichero. **generarGrafo:** generar un objeto de tipo Graph. **cargarConfiguracionNodo:** cargar la configuración de los atributos de un nodo dependiendo del valor que tenga asociado. **cargarConfiguracionEdge:** cargar la configuración de los atributos de una arista dependiendo del valor que tenga asociada. **generarGraphML:** genera el archivo de visualización XML.

Tabla 53. Descripción de la clase "Matriz"

CLASE	CONFIGURACION
Descripción	Clase que gestiona la configuración visual de los nodos y aristas.
Atributos	**atributosVertice:** lista con los atributos configurables para los vértices. **atributosArista:** lista con los atributos configurables para las aristas. **opcionesVertice:** lista con las opciones de configuración y los valores asociados a los atributos para los vértices. **opcionesArista:** lista con las opciones de configuración y los valores asociados a los atributos para las aristas.

CLASE	CONFIGURACION
Métodos	**getAtributosArista:** obtener la lista de atributos configurables para las aristas. **setAtributosArista:** modificar la lista de atributos configurables para las aristas. **getAtributosVertice:** obtener la lista de atributos configurables para los vértices. **setAtributosVertice:** modificar la lista de atributos configurables para los vértices. **getOpcionesArista:** obtener la lista con opciones de configuración de las aristas. **setOpcionesArista:** modificar la lista con opciones de configuración de las aristas. **getOpcionesVertice:** obtener la lista con opciones de configuración de los vértices. **setOpcionesVertice:** modificar la lista con opciones de configuración de los vértices. **addOpcionVertice:** añadir opción de configuración con los valores de los atributos para los vértices. **addOpcionArista:** añadir opción de configuración con los valores de los atributos para las aristas. **cargarValores:** carga los valores de los atributos de una opción de configuración dada. **comprobarOpcionExistente:** coprueba si la opción indicada ya existe en la configuración actual. **tieneOpcionDefecto:** comprueba si existe una opción de configuración por defecto para vértices o aristas. **modificarOpcionVertice:** modifica los valores de los atributos de una opción de configuración para los vértices. **modificarOpcionArista:** modifica los valores de los atributos de una opción de configuración para las aristas. **cargarArchivoVertices:** carga la configuración para los vértices desde un archivo. **cargarArchivoAristas:** carga la configuración para las aristas desde un archivo. **buscarConfiguracionArista:** devuelve el identificador de la opción de configuración asociada a la arista a partir de la colección de los valores de los atributos. **buscarConfiguracionVertice:** devuelve el identificador de la opción de configuración asociada al vértice a partir de la colección de los valores de los atributos. **guardarArchivo:** guarda en un archivo la configuración para vértices o aristas actual.

Tabla 54. Descripción de la clase "Configuracion"

CLASE	FICHERO
Descripción	Clase que representa un fichero de representación de grafos.
Atributos	**fr:** fichero de lectura. **br:** buffer de lectura. **dimension:** dimensión de la matriz que representa al grafo, número de vértices. **numElementosNoNulos:** número de elementos no nulos de la matriz, número de aristas. **vectorIA:** vector IA de la matriz dispersa CSC. **vectorJA:** vector JA de la matriz dispersa CSC. **vectorValores:** vector AA con los valores de la matriz dispersa con esquema CSC.
Métodos	**getDimension:** obtener la dimensión de la matriz. **setDimension:** modificar la dimensión de la matriz. **getNumElementosNoNulos:** obtener el número de elementos no nulos de la matriz. **setNumElementosNoNulos:** modificar el número de elementos no nulos de la matriz. **getVectorIA:** obtener el vectorIA. **setVectorIA:** modificar el vectorIA. **getVectorJA:** obtener el vectorJA. **setVectorJA:** modificar el vectorJA. **getVectorValores:** obtener el vector de valores. **setVectorValores:** modificar el vector de valores. **tratarFichero:** cargar en los atributos de la clase los valores de la matriz dispersa con esquema CSC desde un fichero con ese formato.

Tabla 55. Descripción de la clase "Fichero"

CLASE	XMLBUILDER
Descripción	Clase que maneja la estructura XML.
Atributos	**xml:** cadena de texto auxiliar para concatenar atributos. **root:** root del XML. **fw:** fichero de escritura. **pw:** buffer de escritura.
Métodos	**colgarNodoHoja:** colgar un nodo del documento XML que tiene etiqueta y valor. **colgarNodo:** colgar un nodo del documento XML que sólo tiene etiqueta. **colgarNodosKey:** colgar un nodo del tipo *Key*, que indican los atributos configurables del grafo. **colgarNodoConAtributos:** colgar nodo con varios atributos de par etiqueta-valor. **colgarNodoConAtributosTexto:** colgar nodo con varios atributos con la tupla tipo-etiqueta-valor. **guardarNodo:** escribe la información completa del nodo. **guardarArista:** escribe la información completa de la arista. **guardar:** guarda el objeto *xml* en un archivo.

Tabla 56. Descripción de la clase "XMLBuildeString"

CLASE	PARSER
Descripción	Implementación de un *parser SAX* [36].
Atributos	**graph:** objeto con el grafo. **graphml:** objeto con los datos del graphml. **nodoActual:** nodo actual que se maneja. **edgeActual:** arista actual que se maneja. **keyActual:** atributo actual que se maneja. **value:** valor. **tmpValue:** valor temporal. **nodes:** lista de nodos. **edges:** lista de aristas. **keys:** lista de atributos del grafo. **tieneData:** booleano que indica si el elemento actual tiene atributos. **datasAux:** variable auxiliar con los atributos de un elemento del grafo. **dataActual:** atributo actual de un elemento del grafo.
Métodos	**getGraphml:** obtener el grafo. **setGraphml:** modificar el grafo. **startElement:** inicio del elemento. **endElement:** fin del elemento. **characters:** caracteres del elemento actual.

Tabla 57. Descripción de la clase "Parser"

3.8 Definición de interfaces de usuario

En este apartado se van a especificar las interfaces que se van a utilizar en la interacción entre la aplicación y el usuario. Se revisa la navegación entre las diferentes ventanas, los elementos que las forman, sus características, disposición y la gestión de los eventos relacionados con los objetos que componen la interfaz de usuario.

La funcionalidad de esta aplicación es muy clara, es por ello que lo objetivos que se han seguido en el diseño de la interfaz pasan por no añadir complejidad innecesaria al programa, por lo que el formato de la interfaz debe ser intuitivo y poco complejo.

Se realizó una primera aproximación a la interfaz antes de comenzar la implementación, este prototipo se ha incluido en este documento como anexo (véase Anexo D).

A continuación, se muestran las distintas pantallas que definen la interfaz.

3.8.1 Pantalla de generación del archivo de visualización

Esta es la pantalla inicial de la aplicación, es la primera pestaña del menú. En ella se puede realizar un muestreo de un grafo inicial y generar el archivo de visualización de esa muestra. Para ello el usuario debe indicar cuál es el archivo en el que se encuentra el grafo representado en formato CSC y el fichero con los estados de los vértices. Además, es necesario indicar cuál es el porcentaje de muestreo que se quiere aplicar. Como se puede observar, para introducir los archivos necesarios se puede introducir de manera manual la ruta de los mismos o seleccionarlos en la carpeta en la que se encuentren. A su vez, el porcentaje de muestreo se indicará aumentando o disminuyendo en una unidad de manera interactiva el porcentaje mostrado en la pantalla. Una vez introducidos todos los datos se puede iniciar el proceso de muestreo y generación del archivo de visualización pulsando en el botón inferior de confirmación.

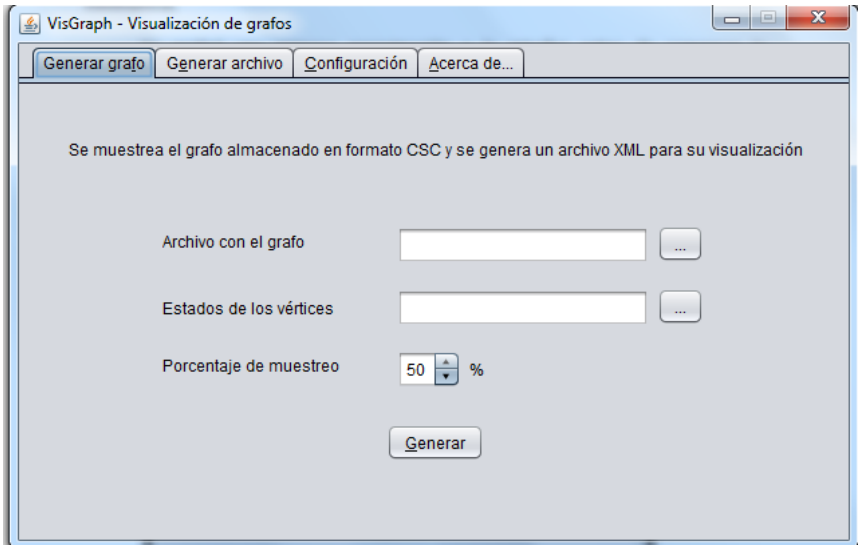

Ilustración 10. Pantalla de generación del archivo de visualización de grafo

3.8.2 Pantalla de generación del archivo de representación de grafo

Esta pantalla se corresponde con la segunda pestaña del menú. En ella se permite al usuario generar los archivos que representan el grafo y los estados de sus vértices, a partir de un archivo de visualización del mismo. Como se puede observar en la ilustración que sigue, el usuario debe indicar cuál es el archivo de visualización del grafo y dar un nombre a los archivos que se van a generar. El archivo de visualización se puede indicar introduciendo la ruta completa o seleccionando el archivo desde la carpeta en la que se encuentre. Los nombres de los archivos a generar se especifican libremente por el usuario introduciendo el texto en el campo correspondiente. Finalmente, tras haber completado todos los datos se debe iniciar el proceso de generación de los archivos pulsando el botón de confirmación inferior.

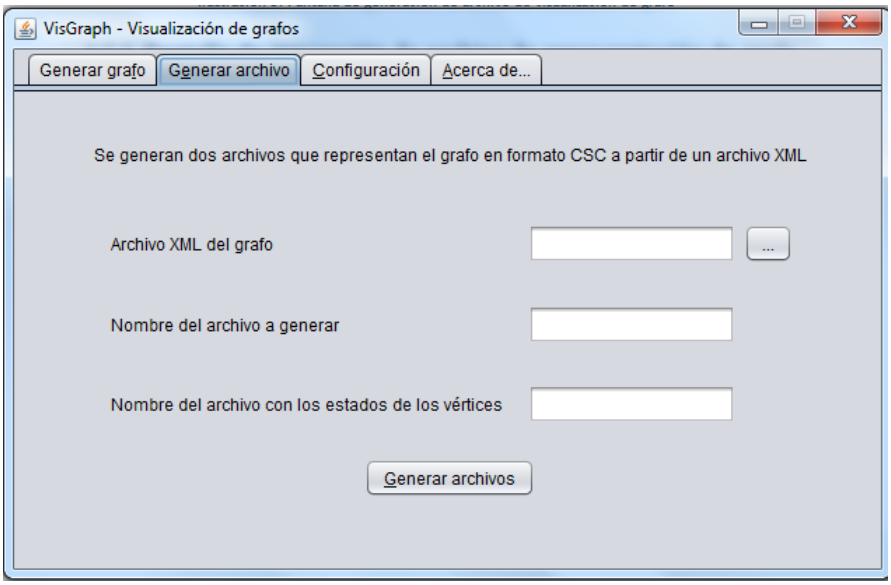

Ilustración 11. Pantalla de generación de archivo de representación de grafo

3.8.3 Pantalla de configuración de los atributos del grafo

Esta pantalla se corresponde con la tercera pestaña del menú. En ella se pueden configurar los atributos referentes a vértices y aristas. Esta configuración es necesaria para que los vértices y aristas tengan un formato que indique su estado.

En esta pantalla se trabaja con dos archivos, el de configuración de los atributos de los vértices y el de las aristas. Mediante los botones inferiores se puede:

- Cargar una configuración previa desde archivo.
- Añadir una opción de configuración.
- Modificar una opción de configuración ya existente.
- Almacenar la configuración existente en ese momento en los archivos que se indiquen.

Para cargar los archivos o para almacenar la configuración en ellos se puede introducir la ruta de manera manual o se pueden seleccionar el archivo desde la carpeta en la que esté ubicado.

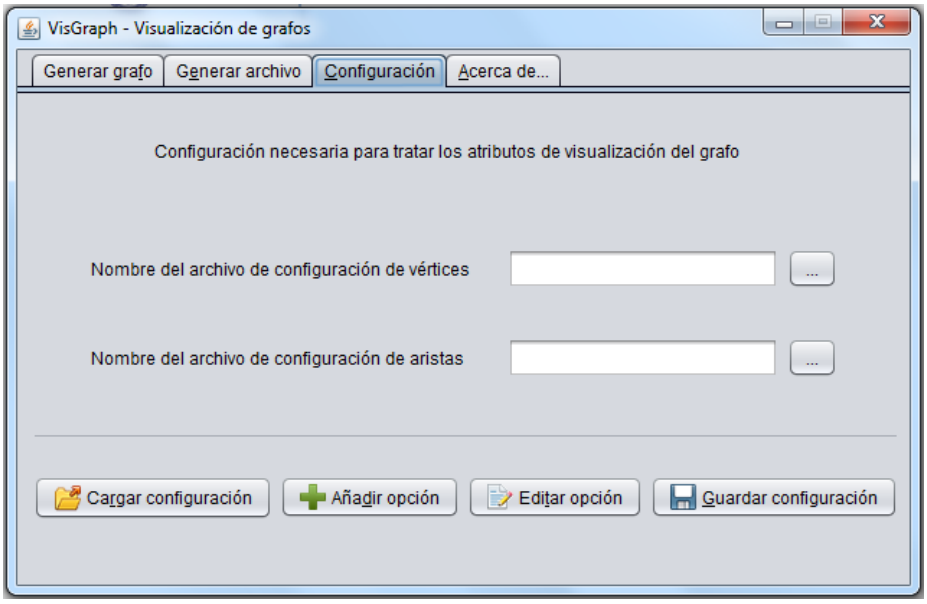

Ilustración 12. Pantalla de configuración de los atributos del grafo

3.8.4 Pantalla de información sobre la aplicación

En esta pantalla se incluye información básica sobre el objetivo de la aplicación y los autores de la misma.

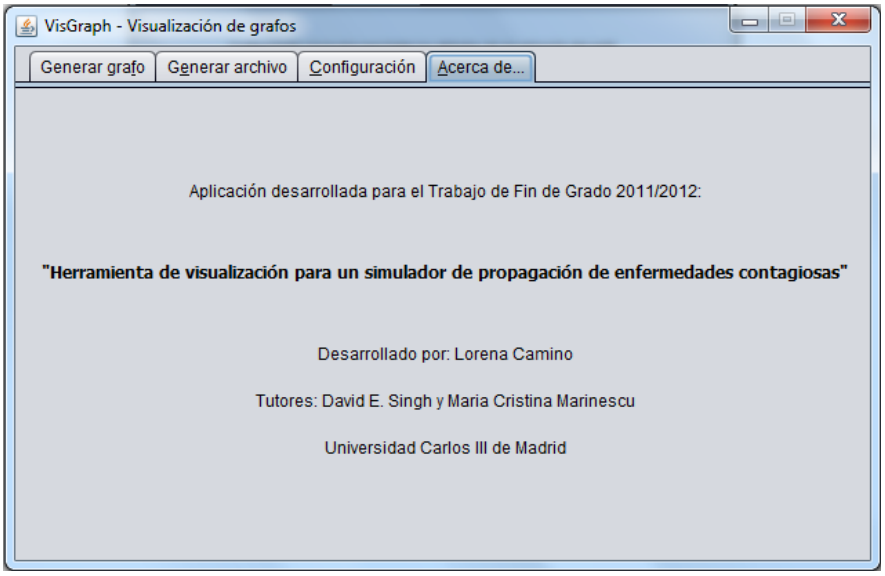

Ilustración 13. Pantalla de información de la aplicación

3.8.5 Pantalla para añadir una opción de configuración

Esta pantalla es una ventana emergente que aparecerá al darle al botón de "*añadir opción de configuración*" dentro de la pantalla de "*configuración de atributos del grafo*". En ella se pide al usuario que rellene todos los datos de la opción de configuración que quiere crear. Al terminar de rellenar los campos que se deseen se debe confirmar la acción pulsando el botón inferior de la pantalla.

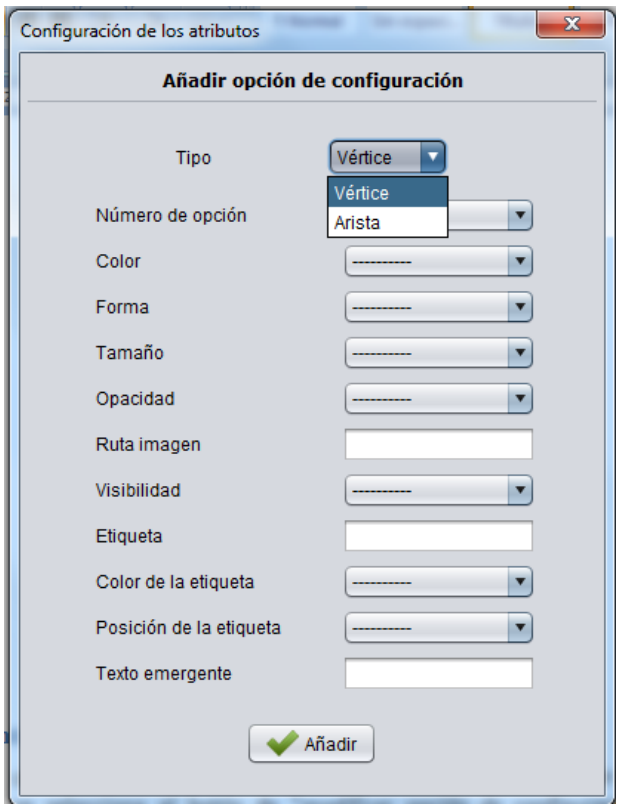

Ilustración 14. Pantalla para añadir una opción de configuración

3.8.6 Pantalla de modificación de opción de configuración

Esta pantalla aparece cuando en la ventana de "*configuración de los atributos del grafo*" se selecciona el botón de "*modificar opción de configuración*". Es una ventana emergente en la que se puede ir seleccionando las opciones de configuración existentes para vértices y aristas, y ver sus datos asociados, editándolos si se desea. Para confirmar la modificación de los datos se debe seleccionar el botón inferior de aceptación. Para borrar la opción se debe seleccionar el botón inferior de eliminación, lo que hace que desaparezca la opción de configuración seleccionada junto con todos sus atributos.

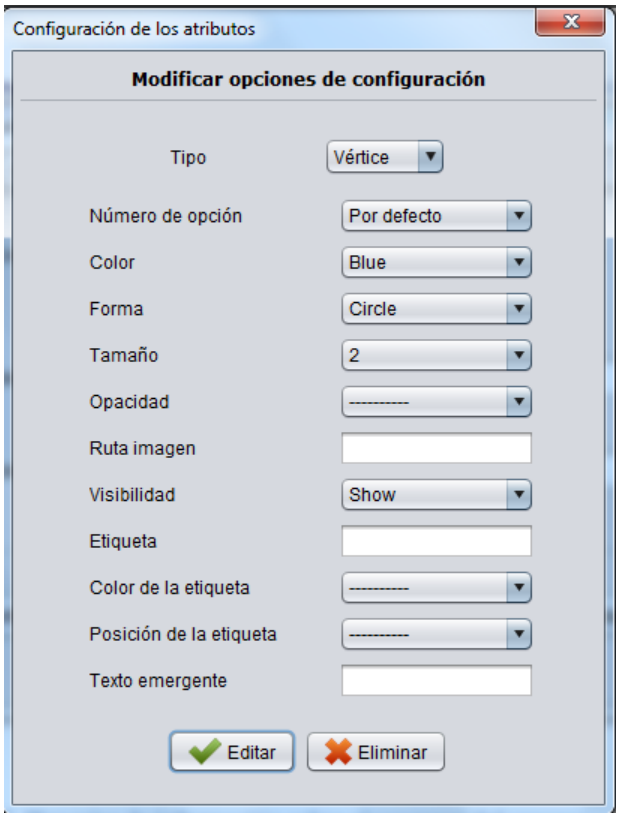

Ilustración 15. Pantalla de modificación de opciones de configuración

3.8.7 Mensajes de información sobre el progreso del proceso

Estos mensajes aparecen en una ventana emergente indicando cuál es el progreso del proceso que está realizando la aplicación en ese momento. De esta manera el usuario está informado sobre las acciones que está realizando el sistema. La manera de mostrar esta información es con una barra de progreso y un breve rótulo sobre la tarea que está llevando a cabo la aplicación en cada momento.

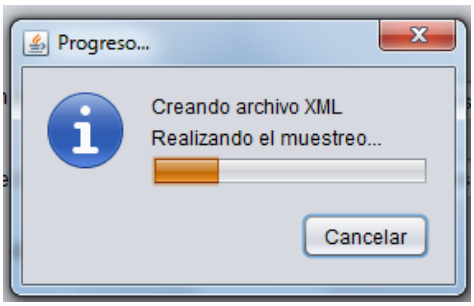

Ilustración 16. Mensaje de información del progreso del proceso

3.8.8 Mensajes de aviso del sistema

Este tipo de mensajes aparecen para informar al usuario de que debe introducir algún dato que no ha seleccionado o sobre cosas que debe tener en cuenta dependiendo de la acción que esté realizando. Se mostrará un icono representativo del aviso junto con una breve descripción del mismo y un botón que servirá para que el usuario confirme que ha leído el mensaje.

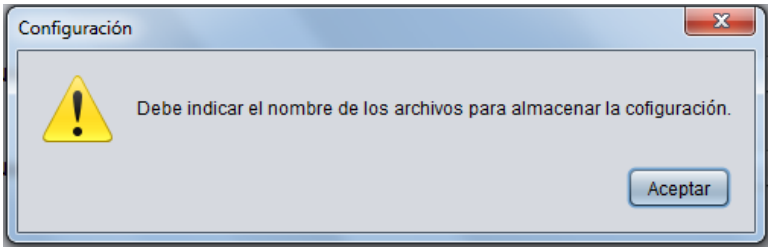

Ilustración 17. Mensaje de aviso del sistema

3.8.9 Mensajes de error del sistema

Este tipo de mensajes aparecen para informar al usuario de que ha ocurrido algún error con/durante la acción que ha seleccionado. Se mostrará un icono representativo de error junto con una breve descripción del mismo y un botón que servirá para que el usuario confirme que ha leído el mensaje de error.

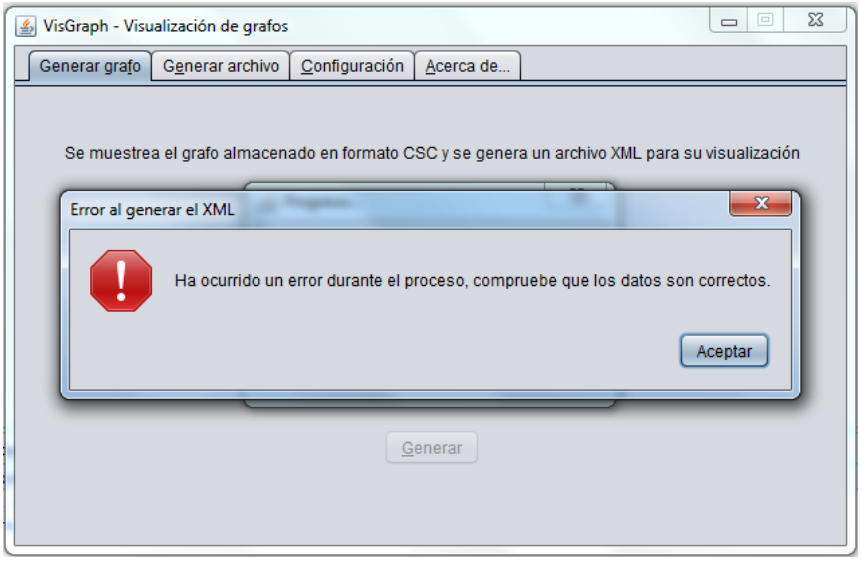

Ilustración 18. Mensaje de error del sistema

3.8.10 Mensajes de información del sistema

La función de este tipo de mensajes es simplemente la de informar al usuario de que la acción que se ha seleccionado se ha realizado con éxito. Aparece como una ventana emergente en la que se muestra un icono indicando que es un mensaje informativo, un mensaje con la acción que se ha realizado con éxito, y un botón de confirmación para indicar que el usuario ha leído el mensaje.

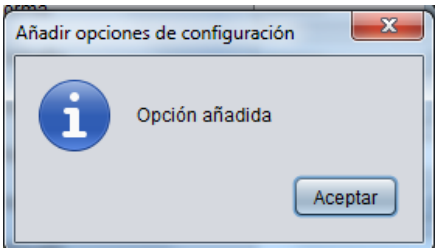

Ilustración 19. Mensaje de información del sistema

3.8.11 Mensajes de confirmación por parte del usuario

Este tipo de mensajes aparecen cuando la aplicación va a realizar alguna acción irreversible y necesita la confirmación del usuario para evitar problemas derivados. El formato del mensaje incluye un icono en forma de interrogación, un mensaje que describe la confirmación que se solicita al usuario, y dos botones para indicar si se quiere confirmar la acción o no.

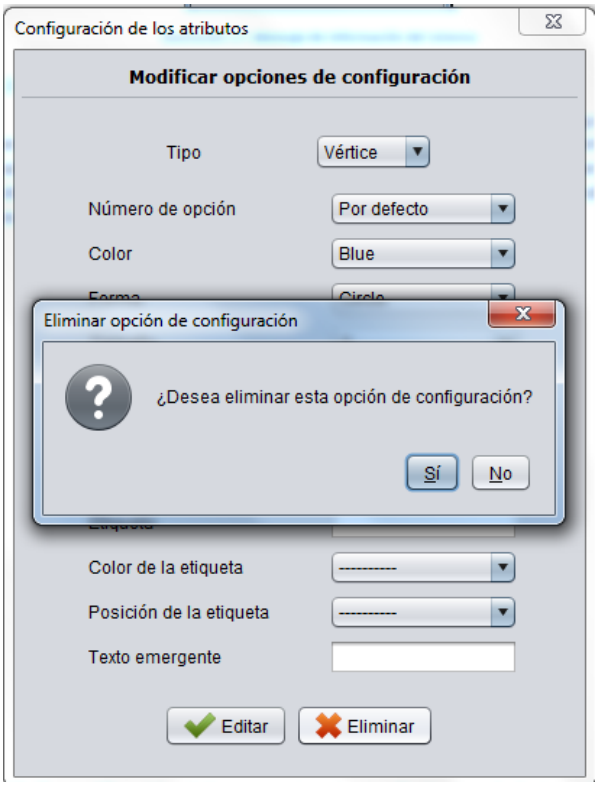

Ilustración 20. Mensaje de confirmación

3.8.12 Especificación del comportamiento dinámico de la interfaz

Tras la especificación de los distintos formatos de interfaz para cada pantalla de la aplicación, es necesario definir cómo va a ser el comportamiento dinámico de la interfaz para conocer las opciones de navegación del sistema. Como la aplicación tiene una funcionalidad muy concreta, el menú de navegación se implementa mediante pestañas, por lo que en todo momento se puede acceder a cualquiera de los

subsistemas. En el siguiente diagrama se indica esto mismo, en él se puede observar claramente la sencillez de la que se ha querido dotar a la interfaz del sistema.

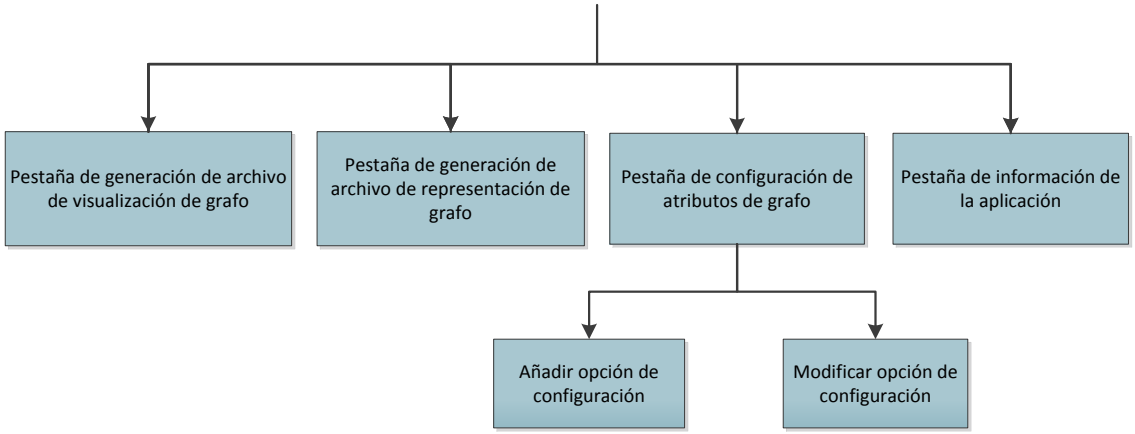

Ilustración 21. Diagrama de comportamiento dinámico de la interfaz

4 Planificación

En esta sección se detalla la planificación del proyecto, que es una estimación de las actividades, hitos y tareas que se realizarán durante el desarrollo del proyecto. Además se incluirá una estimación de costes, que resumirá los recursos hardware y software que se utilizarán en el desarrollo del sistema y el coste, tiempo y esfuerzo necesario. Al ser estimaciones servirán como referencia, pero pueden verse modificadas a lo largo de la vida del proyecto.

4.1 Planificación temporal del proyecto

Se estima que el comienzo de este proyecto datará del día 1 de febrero de 2012 y finalizará el día 4 de septiembre de 2012. En los 217 días de duración del proyecto se empleará un recurso de trabajo de 646 horas de desarrollo del proyecto.

A continuación, se exponen una serie de ilustraciones con los *diagramas de Gantt* [32] en los que se ubican las tareas que se estiman necesarias para la consecución del producto. Todo esto se organiza en un calendario, indicando su orden y duración.

4.1.1 Diagrama general

El proyecto comenzará el día 1 de febrero de 2012 y finalizará el día 4 de septiembre de 2012 con 217 días de duración, de los cuales 155 días son de trabajo. El proyecto constará de siete fases diferenciadas a lo largo del tiempo.

- Planificación (14 días).
- Gestión de la configuración (13 días).
- Estudio de viabilidad del sistema (20 días).
- Análisis del sistema de información (19 días).
- Diseño del sistema de información (56 días).
- Pruebas (25 días).
- Implantación y aceptación del sistema (8 días).

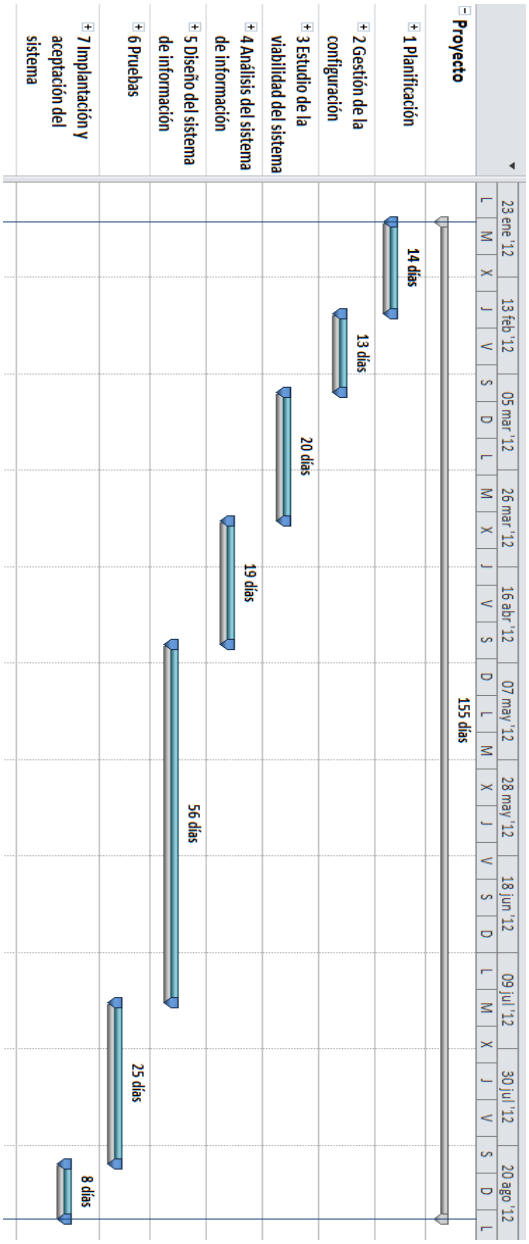

Ilustración 22. Planificación general

4.1.2 Diagrama de planificación

Esta es la primera fase y tiene una duración estimada de 14 días. Consta de las siguientes partes:

- Reunión con los tutores: 1 día.
- Estudio del plan: 3 días.
- Estudio del PFC precursor de este TFG: 2 días.
- Estudio de software de visualización de grafos: 2 días.
- Estudio de técnicas de muestreo: 2 días.
- Elaboración de la documentación: 3 días.
- Revisión de la documentación: 1 día.

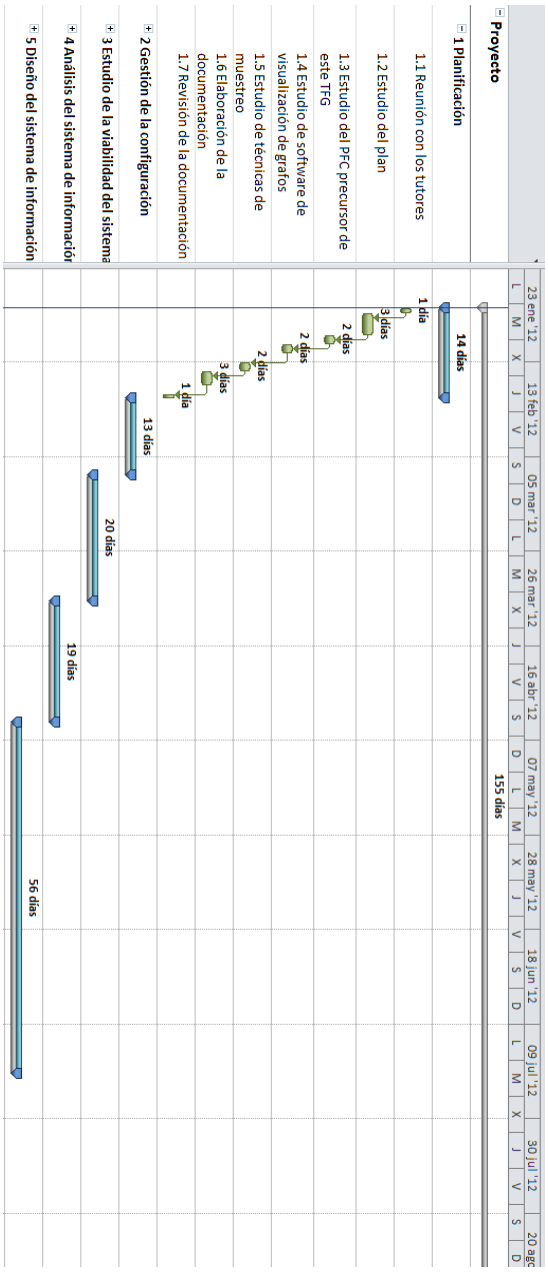

Ilustración 23. Diagrama de planificación

4.1.3 Diagrama de gestión de la configuración

Esta es la segunda fase, orientada a definir la gestión de la configuración, y tiene una duración estimada de 13 días. Está compuesta por las siguientes partes:

- Reunión con los tutores: 1 día.
- Estudio de la configuración: 5 días.
- Elaboración de la documentación: 4 días.
- Revisión de la documentación: 3 días.

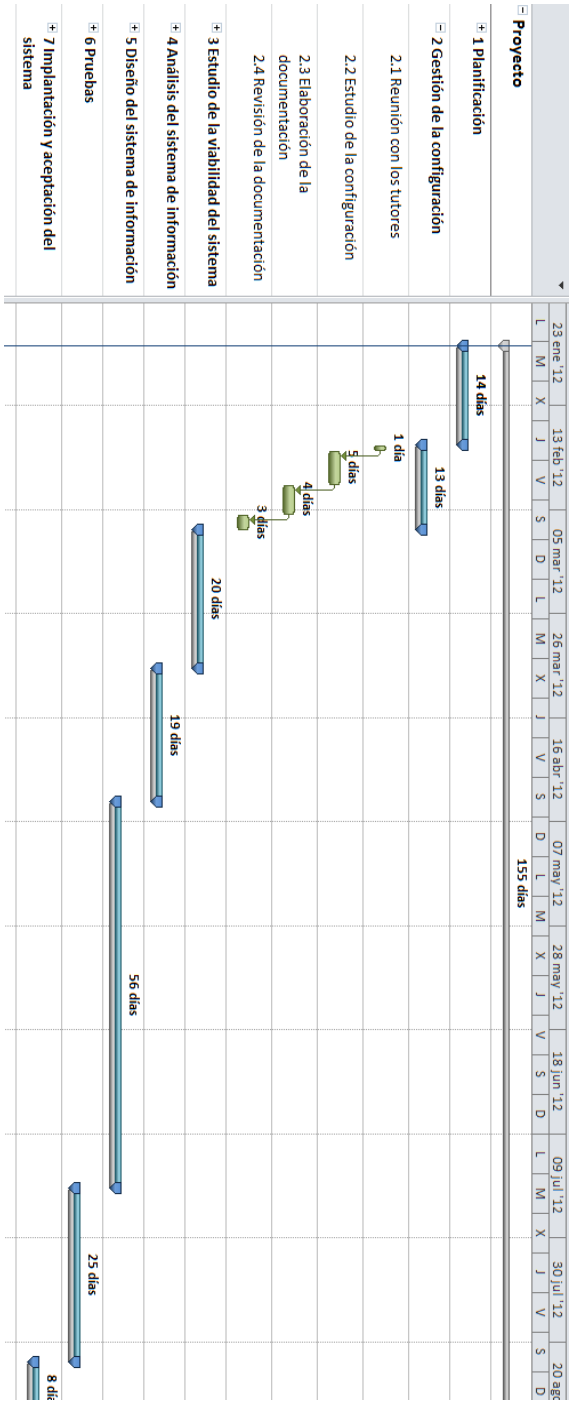

Ilustración 24. Diagrama de gestión de la configuración

4.1.4 Diagrama de estudio de viabilidad del sistema

El estudio de la viabilidad del sistema es la tercera fase y tiene una duración estimada de 20 días. Consta de las siguientes partes:

- Reunión con los tutores: 1 día.
- Estudio de las alternativas tecnológicas: 7 días.
- Estudio del entorno tecnológico: 6 días.
- Elaboración de la documentación: 4 días.
- Revisión de la documentación: 2 días.

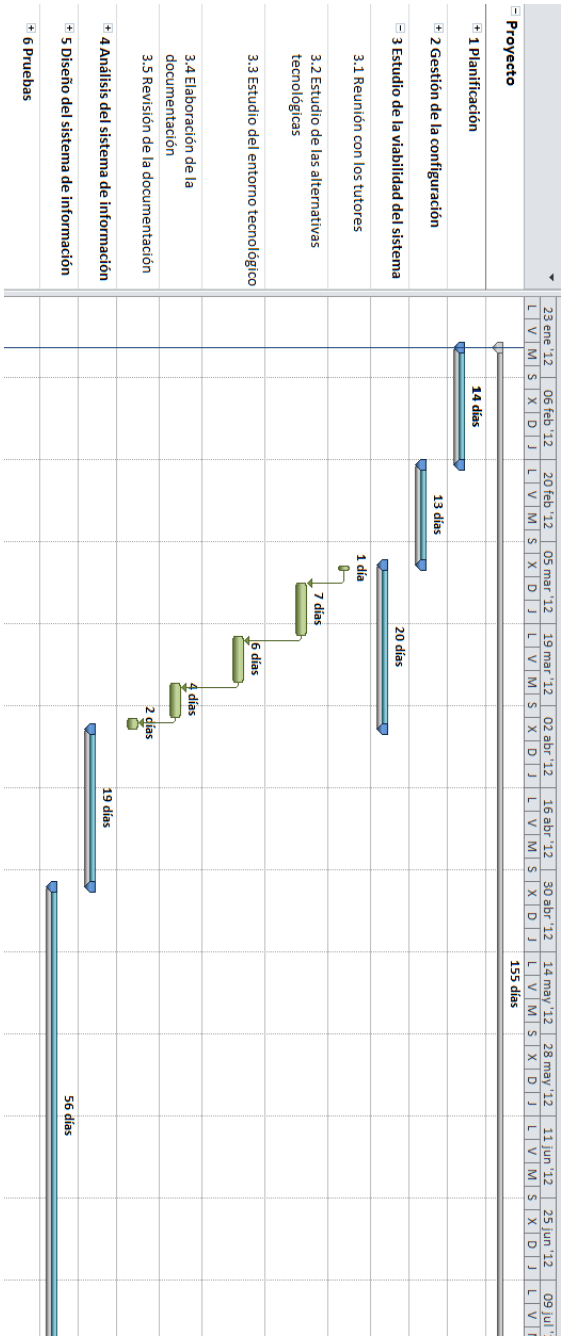

Ilustración 25. Diagrama de estudio de viabilidad del sistema

4.1.5 Diagrama de análisis del sistema de información

El análisis del sistema de información representa la cuarta fase y tiene una duración estimada de 19 días. Está formada por las siguientes partes:

- Reunión con los tutores: 1 día.
- Análisis del sistema: 7 días.
- Análisis de requisitos: 6 días.
- Elaboración de la documentación: 3 días.
- Revisión de la documentación: 2 días.

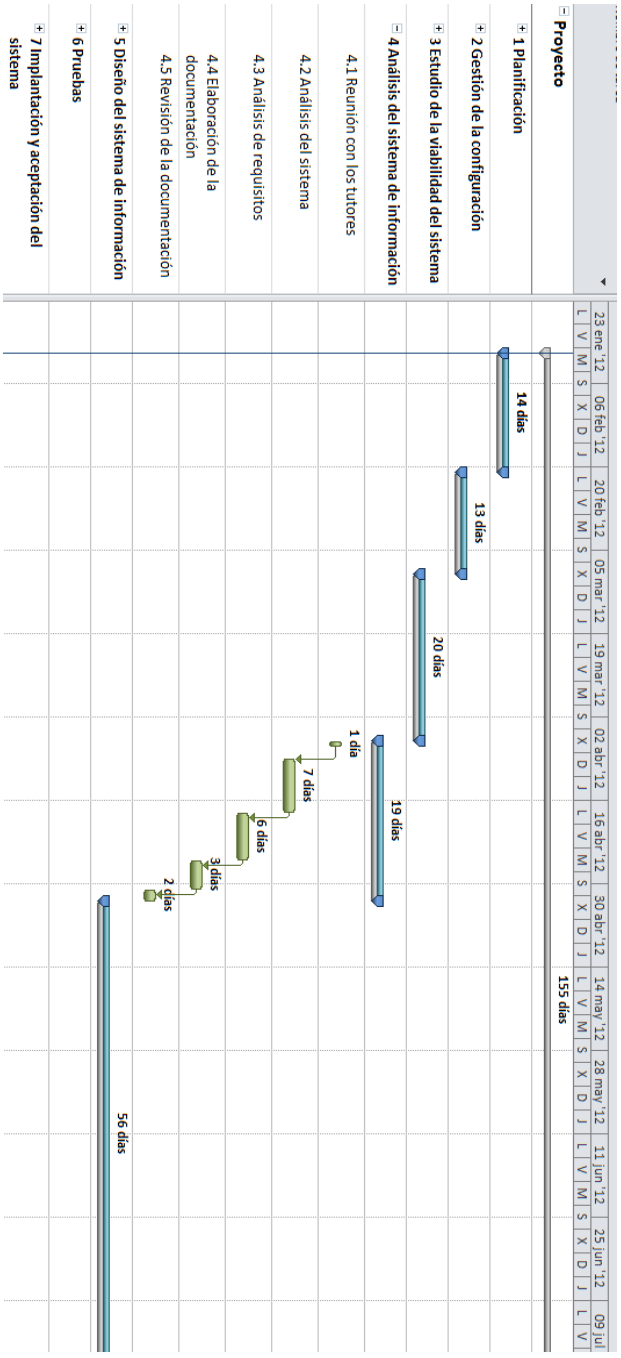

Ilustración 26. Diagrama de análisis del sistema de información

4.1.6 Diagrama de diseño del sistema de información

El diseño del sistema de información representa la quinta fase y tiene una duración estimada de 56 días. Es la que se prolonga durante más tiempo debido a que es la fase principal en la que se construye el sistema. Consta de:

- Reunión con los tutores: 1 día.
- Definición de la arquitectura del sistema: 5 días.
- Implementación del sistema: 44 días.
- Elaboración de la documentación: 4 días.
- Revisión de la documentación: 2 días.

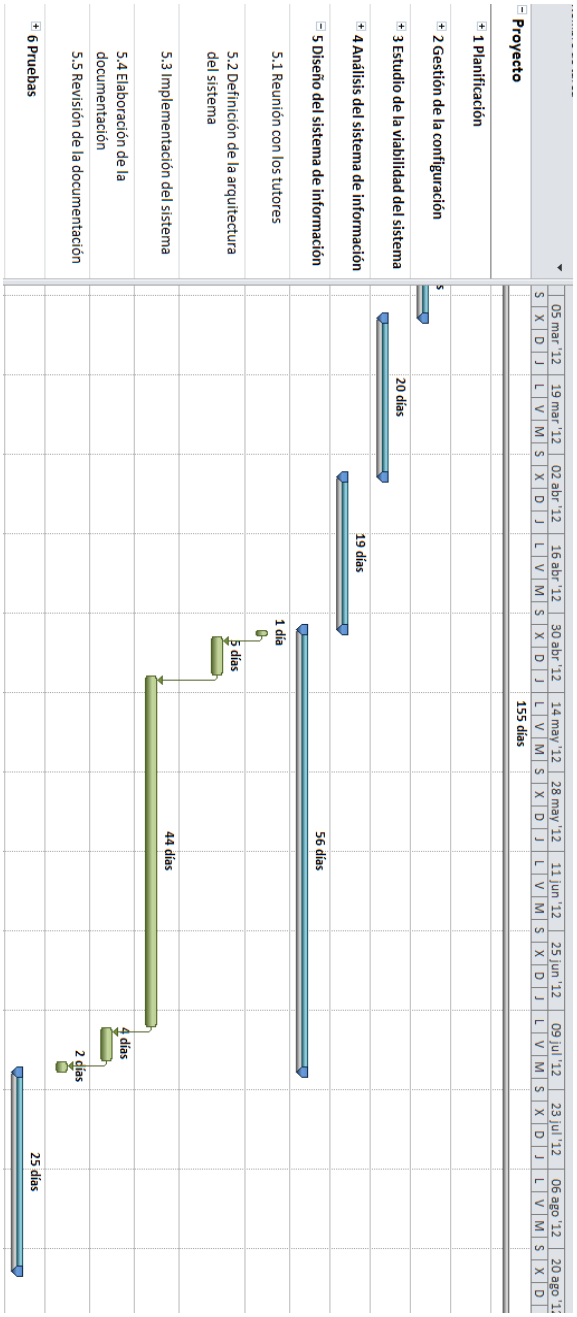

Ilustración 27. Diagrama de diseño del sistema de información

4.1.7 Diagrama de pruebas

Las pruebas del sistema se realizan en la sexta fase, y tiene una duración estimada de 25 días. Consta de las siguientes partes:

- Reunión con los tutores: 1 día.
- Definición del plan de pruebas: 4 días.
- Elaboración del plan de pruebas: 8 días.
- Análisis de los resultados obtenidos: 5 días.
- Elaboración de la documentación: 5 días.
- Revisión de la documentación: 2 días.

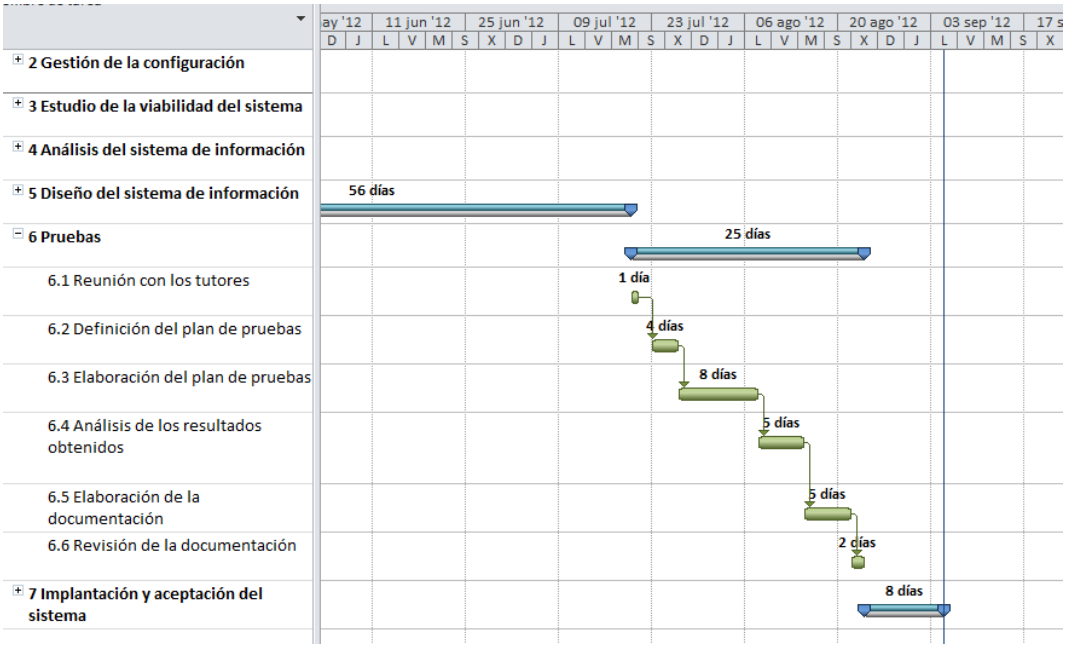

Ilustración 28. Diagrama de pruebas

4.1.8 Diagrama de implantación y aceptación del sistema

Esta es la séptima y última fase. Tiene una duración estimada de 8 días y consta de las siguientes partes:

- Entrega de la documentación y la última versión del sistema: 3 días.
- Reunión con los tutores: 1 día.
- Entrega de la documentación y versión final: 2 días.
- Aceptación del sistema: 2 días.

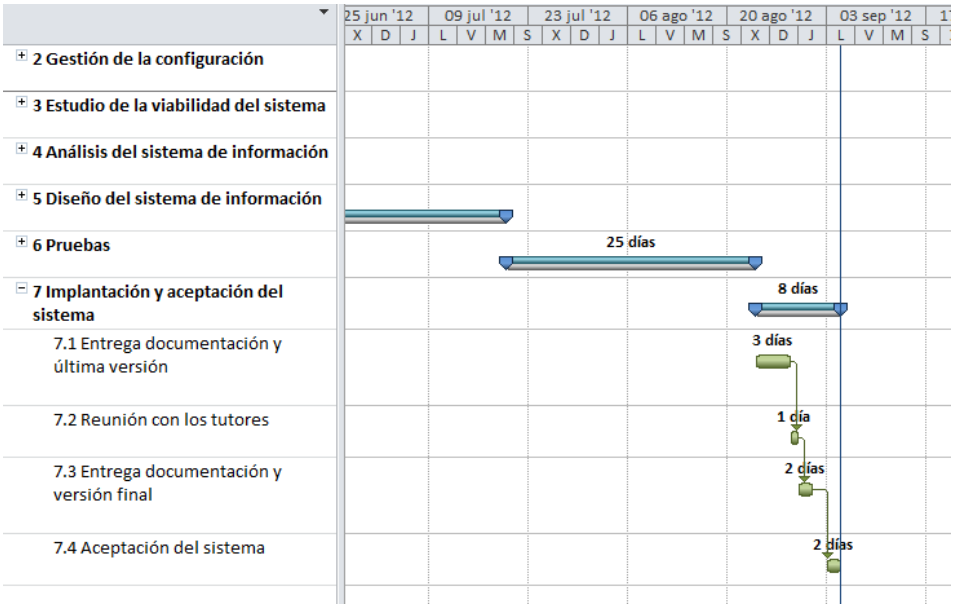

Ilustración 29. Diagrama de implantación y aceptación del sistema

4.2 Estimación de costes

Una parte muy importante de los proyectos de desarrollo de software es planificar mediante una estimación de los recursos hardware, software, de costo, esfuerzo y tiempo necesarios para lograr su finalización.

En este proyecto se utiliza **COCOMO II** [33], que es un modelo de estimación desarrollado por Barry W. Boehm a finales de los años 70 y comienzo de los 80. Basa sus estimaciones en modelos matemáticos de base empírica que tienen en cuenta el tamaño del proyecto principalmente.

COCOMO II incluye tres submodelos, cada uno de ellos ofrece mayor fidelidad a medida que se avanza en la planificación del proyecto y en el proceso de diseño:

- **El modelo de Composición de Aplicaciones:** indicado para proyectos construidos con herramientas modernas de construcción de interfaces gráficos para usuario.
- **El modelo de Diseño Anticipado:** este modelo puede utilizarse para obtener estimaciones aproximadas del coste de un proyecto antes de que esté terminada por completo su arquitectura.
- **El modelo de Post-Arquitectura:** es el modelo más detallado. Se utiliza una vez que se ha desarrollado por completo la arquitectura del proyecto.

4.2.1 COCOMO II

La versión que se utiliza en este proyecto es USC-*COCOMO II.2000.4*. Son necesarios una serie de datos para realizar la estimación, que son los siguientes:

- **Nombre del proyecto:** campo en el que se debe indicar el nombre del proyecto. En este caso se le ha asignado el de *VisGraph*.
- **Nombre del módulo:** sirve para especificar el nombre de cada módulo que componga el proyecto. En este caso existen dos partes: el de muestreo y el de generación del archivo *GraphML*. Pero para calcular la estimación con COCOMO II se va a unificar en un sólo módulo, ya que los valores de los parámetros para calcular la estimación son similares.
- **Horas por persona al mes:** el valor de horas por persona mensuales que utilizará el modelo en los cálculos. En este proyecto se va a tomar como referencia un trabajo diario de 8 horas durante 22 días al mes (8 días se corresponden con fin de semana), por lo que se especifica que utilice el valor 176 horas por persona al mes.
- **Tamaño del módulo:** indica el tamaño de cada módulo en líneas de código. Se puede calcular de tres maneras:
  - Introduciendo su valor en el campo *SLOC*.
  - Utilizando el modelo de los puntos de función.
  - Empleando el modelo de adaptación y reutilización.

Además, se permite la configuración del parámetro *Breakage*, que es el porcentaje de código que se descarta debido a la evolución y volatilidad de los requisitos.
En este proyecto se introduce directamente el número de líneas de código, estimando que son:
    - Parte de muestreo: 800 líneas.
    - Resto de la aplicación: 6000 líneas.
- **Sueldo mensual:** contiene la cantidad monetaria que percibe el desarrollador del módulo cada mes. En este caso se hace una media de lo percibido dependiendo de la fase de desarrollo y se establece un salario de 35 €/hora, ascendiendo el total mensual a 6160 €/mes.
- **Factor de ajuste de esfuerzo (EAF):** muestra el producto de los multiplicadores de esfuerzo para cada módulo. Dependiendo del modelo que se elija para el proyecto se consideran un número de factores u otro. Para el modelo *Diseño Anticipado* se utilizan 7 factores, mientras que para el de *Post-Arquitectura* son 17.
- **Riesgo:** contiene el nivel de riesgo total para el módulo.
- **Factor de escala:** se aplica al global del proyecto.

4.2.2 Estimación inicial

Se realiza una estimación inicial utilizando el modelo de *Diseño Anticipado*.

Los factores de esfuerzo que se han definido son los siguientes:

FACTOR DE ESFUERZO	DESCRIPCIÓN	VALOR	OBSERVACIONES
RCPX	Fiabilidad del producto.	NOMINAL	El sistema no es complejo y es fiable, por lo que se asigna el valor nominal. Las pérdidas son recuperables.
RUSE	Reutilización requerida.	BAJO	No se realiza reutilización.
PDIF	Dificultad de la plataforma.	BAJO	La dificultad de la plataforma es baja.
PERS	Capacidad del personal.	ALTO	La capacidad del personal es alta.
PREX	Experiencia del personal.	ALTO	La experiencia en la plataforma y lenguajes utilizados es alta.
FCIL	Facilidades.	ALTO	Se emplean varias herramientas software y el sitio del desarrollo es un mismo lugar.
SCHEDULE	Planificación temporal.	BAJO	Las restricciones temporales son flexibles.

Tabla 58. Factores de esfuerzo de la estimación inicial

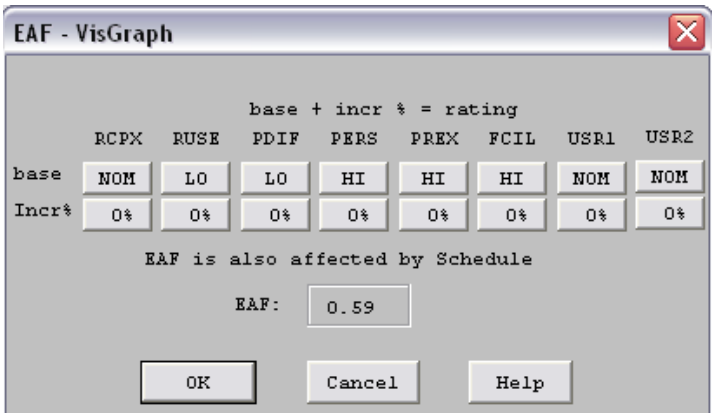

Tabla 59. Factores de esfuerzo de la estimación inicial en COCOMO II

Los factores de escala utilizados son:

FACTOR DE ESCALA	DESCRIPCIÓN	VALOR	OBSERVACIONES
PREC	Precedencia.	ALTO	Existe familiaridad en proyectos de desarrollo de software de este tipo.
FLEX	Flexibilidad del desarrollo.	BAJO	La flexibilidad es baja porque es necesario que el software se ajuste a los requisitos preestablecidos.
RESL	Resolución de riesgos.	ALTO	Se han identificado los posibles riesgos y se han establecido las medidas para resolverlos.
TEAM	Cohesión del equipo.	ALTO	No existen dificultades en la sincronización del equipo y el compromiso es compartido.
PMAT	Madurez del proceso.	NOMINAL	El nivel de madurez CMMI es nivel 3, ya que existe una buena gestión de proyectos definida.

Tabla 60. Factores de escala en la estimación inicial

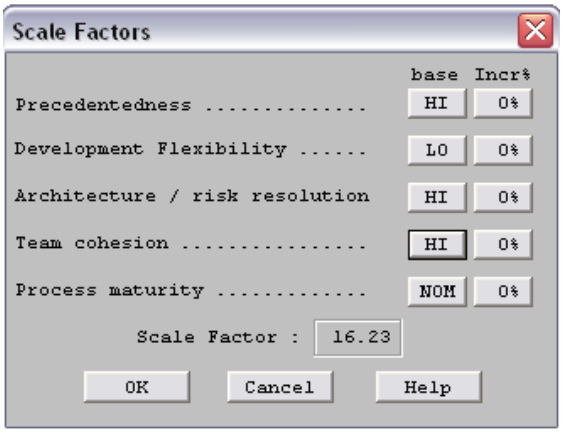

Tabla 61. Factores de escala en la estimación inicial en COCOMO II

Tras definir todos estos factores, los resultados obtenidos son los siguientes:

- **Productividad:** 579,3 SLOC por persona al mes.
- **Tiempo:** 7 meses y 27 días (el más optimista).
- **Personal:** 1,8 personas.
- **Coste:** 72314,03 €.

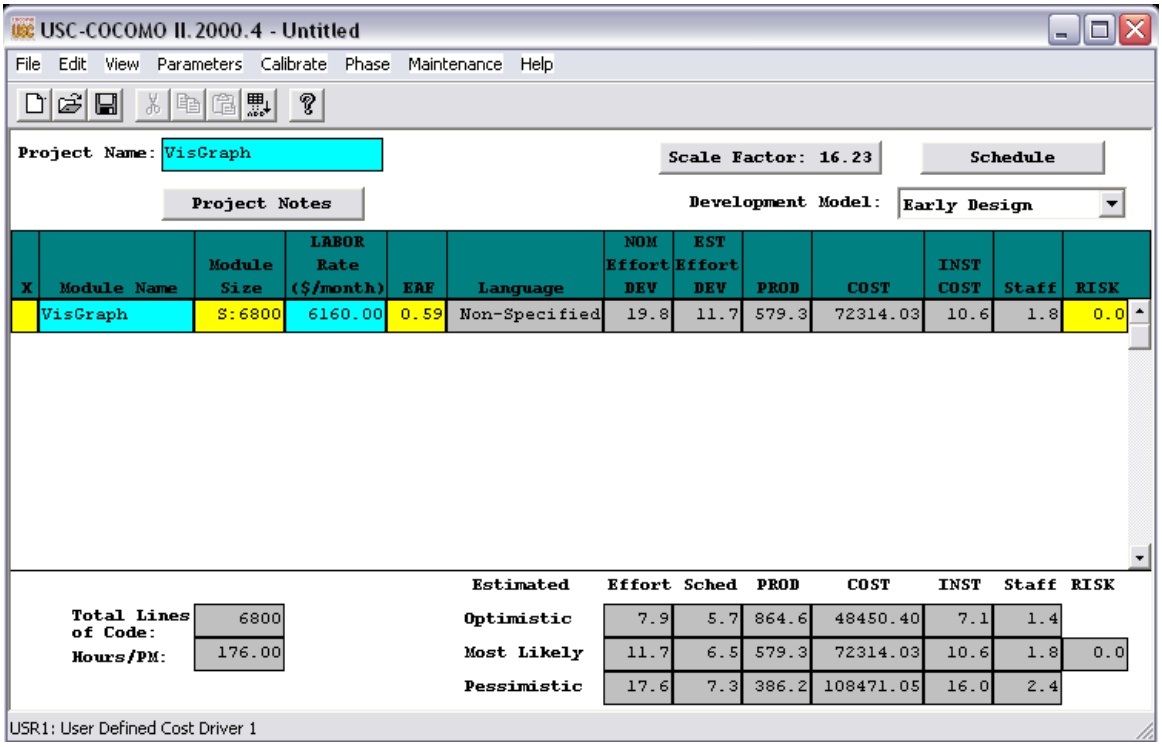

Tabla 62. Resultados de estimación inicial en COCOMO II

4.2.3 Estimación final

Se realiza una estimación inicial utilizando el modelo de *Post-Arquitectura*.

Los factores de esfuerzo que se han definido son los siguientes:

FACTOR DE ESFUERZO	DESCRIPCIÓN	VALOR	OBSERVACIONES
RELY	Fiabilidad del software.	NOMINAL	El sistema es fiable y es posible la recuperación en caso de pérdidas.
DATA	Tamaño de la base de datos.	NOMINAL	El tamaño de la aplicación y sus datos no es elevado.
DOCU	Complejidad del producto.	NOMINAL	El sistema tiene una complejidad moderada.
CPLX	Documentos del ciclo de vida.	NOMINAL	La documentación abarca todo el ciclo de vida del software.
RUSE	Reutilización requerida.	BAJO	No se realiza reutilización.
TIME	Tiempo de ejecución.	NOMINAL	Se estima que el tiempo de ejecución será mayor que del 50 % y menor del 65 %.
STOR	Restricciones de almacenamiento.	NOMINAL	Se estima que una restricción de almacenamiento del 50 %.
PVOL	Volatilidad de la plataforma.	BAJO	El sistema es muy estable.
ACAP	Habilidad del analista.	ALTO	Se estima que la capacidad del analista es alta.
PCAP	Habilidad del programador.	ALTO	Se estima que la capacidad del programador es alta.
PCON	Continuidad del personal.	MUY ALTA	Se estima una continuidad del personal muy alta.
APEX	Experiencia en aplicaciones.	ALTO	La experiencia del equipo en desarrollo de proyectos software de este tipo es alta.
LTEX	Experiencia en el lenguaje y herramientas.	ALTO	La experiencia del equipo en Java, C y las demás herramientas utilizadas es en general alta.
PLEX	Experiencia en la plataforma.	ALTO	La experiencia del equipo en la plataforma de desarrollo es alta.
TOOL	Uso de herramienta software.	ALTO	Se utilizan herramientas de ciclo de vida maduras.
SITE	Desarrollo multilugar.	ALTO	El desarrollo del proyecto se realiza en el domicilio del equipo y en la UC3M.
SCHEDULE	Planificación temporal.	BAJO	Las restricciones temporales son flexibles.

Tabla 63. Factores de esfuerzo de la estimación final

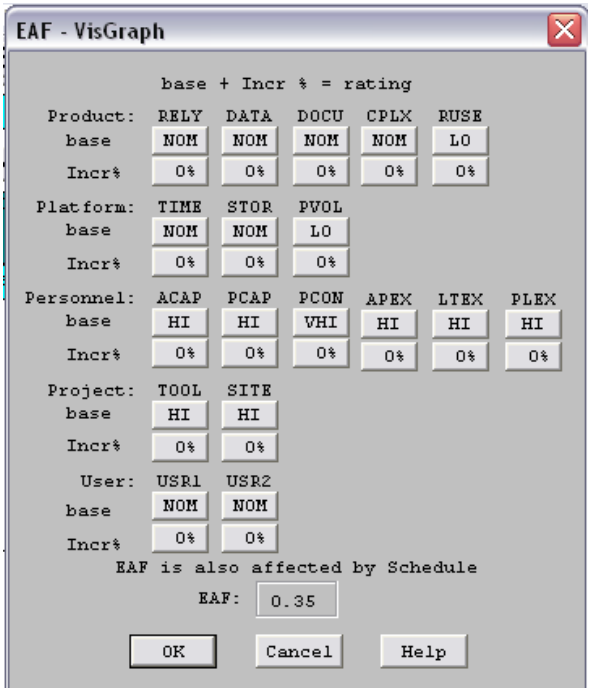

Tabla 64. Factores de esfuerzo de la estimación final en COCOMO II

Los factores de escala utilizados son:

FACTOR DE ESCALA	DESCRIPCIÓN	VALOR	OBSERVACIONES
PREC	Precedencia.	ALTO	Existe familiaridad en proyectos de desarrollo de software de este tipo.
FLEX	Flexibilidad del desarrollo.	BAJO	La flexibilidad es baja porque es necesario que el software se ajuste a los requisitos preestablecidos.
RESL	Resolución de riesgos.	ALTO	Se han identificado los posibles riesgos y se han establecido las medidas para resolverlos.
TEAM	Cohesión del equipo.	ALTO	No existen dificultades en la sincronización del equipo y el compromiso es compartido.
PMAT	Madurez del proceso.	NOMINAL	El nivel de madurez CMMI es nivel 3, ya que existe una buena gestión de proyectos definida.

Tabla 65. Factores de escala en la estimación final

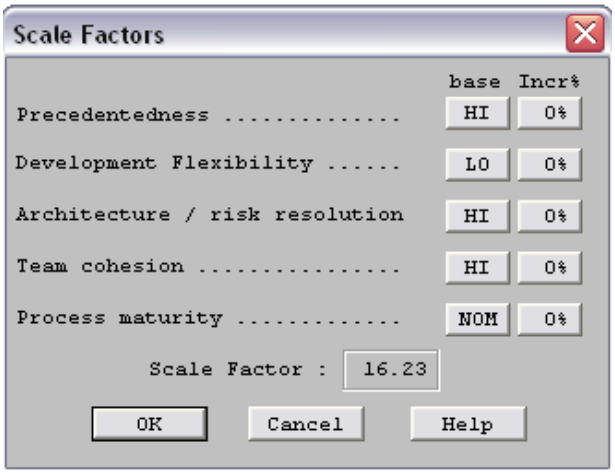

Tabla 66. Factores de escala en la estimación final en COCOMO II

Tras definir todos estos factores, los resultados obtenidos son los siguientes:

- **Productividad:** 994,7 SLOC por persona al mes.
- **Tiempo:** 5 meses y 15 días (el más optimista)
- **Personal:** 1,3 personas.
- **Coste:** 42538,63 €.

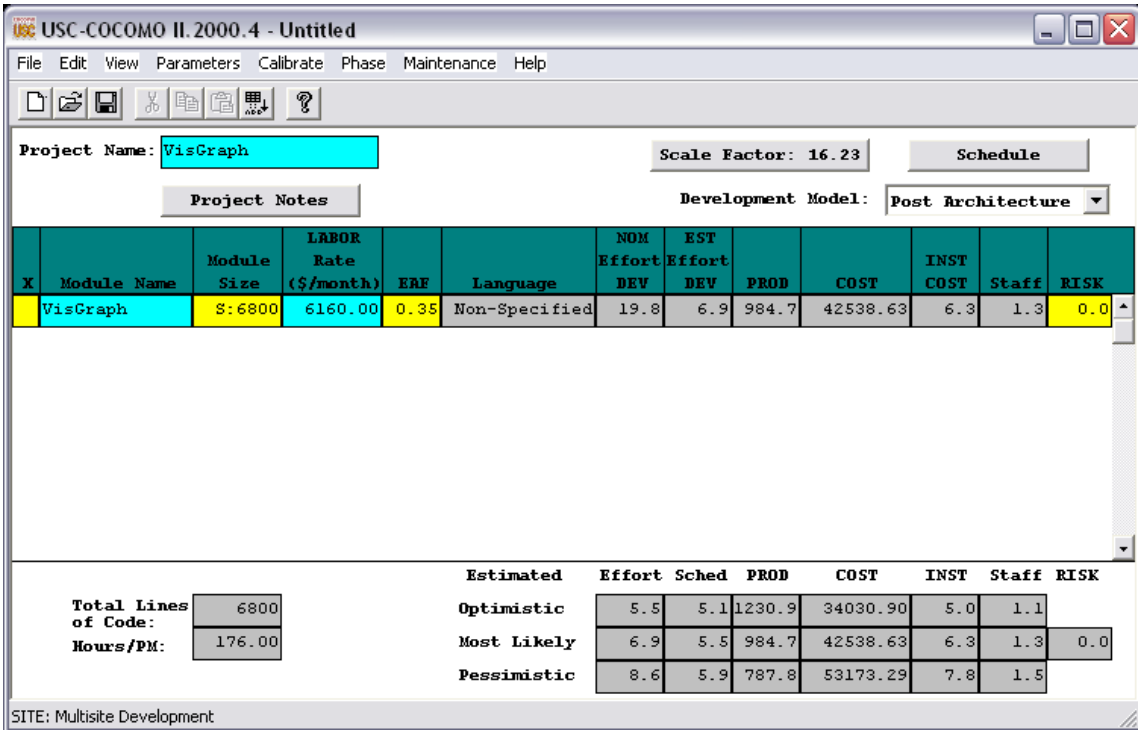

Tabla 67. Resultados de estimación final en COCOMO II

4.2.4 Evaluación de los resultados

A continuación se comparan los datos obtenidos para cada modelo:

Modelo	Productividad	Personal	Tiempo	Coste
Diseño Anticipado	579,3 SLOC/PM	1,8	7 meses y 27 días	72.314,03 €
Post-Arquitectura	994,7 SLOC/PM	1,3	5 meses y 15 días	42.538,63 €

Tabla 68. Estimaciones de desarrollo de software

El coste necesario en la estimación inicial es 72.314,03 €, lo que supone un 58% más de lo indicado con la estimación final. Esto es debido a que en un primer momento se prevé una mayor duración del desarrollo del proyecto, necesitando más personal.

En el modelo *Post-Arquitectura* se obtiene un resultado más ajustado a la realidad ya que los factores de esfuerzo son más completos, pudiendo indicar más detalladamente los conocimientos y capacidades del equipo de desarrollo. Por tanto, en la estimación final se tiene en cuenta la familiaridad de los desarrolladores del proyecto con los lenguajes y herramientas que se emplean, lo que hace que la productividad aumente en un 71% respecto a la estimación inicial, provocando que se reduzca el tiempo de desarrollo y personal necesarios, y como consecuencia el coste.

Como conclusión, indicar que se observan grandes diferencias entre lo estimado inicial y finalmente. Como ya se ha comentado en el párrafo anterior, esto entra dentro de lo comprensible, ya que en la estimación inicial no se ha podido detallar la cohesión que existe entre el equipo de desarrollo y la experiencia que posee con los lenguajes, plataformas y herramientas que se emplean en el proyecto, cosa que la estimación final sí tiene en cuenta.

Si se tienen en cuenta los datos reales en cuanto a la planificación (véase Apartado 4) y el presupuesto del proyecto (véase Apartado 7), se puede observar que:

- En lo relacionado al tiempo necesario para desarrollar el sistema, la planificación real es de 7 meses, que es un valor intermedio al obtenido en el *Diseño Anticipado* y el de *Post-Arquitectura*.
- En cuanto al coste del desarrollo del sistema, el presupuesto real es de 27.946,88 €, lo que es un 35% menos de lo que señala el modelo *Post-Arquitectura*. Esto es debido a que en el presupuesto se tiene en cuenta el coste de hardware y software, que es muy reducido ya que las herramientas utilizadas en el desarrollo son gratuitas en su mayoría.

5 Pruebas

En este apartado se definen una serie de pruebas que permitan comprobar que el sistema funciona correctamente y cumple con los requisitos definidos para el mismo.

Las pruebas se realizan en varios momentos de la vida del sistema, para permitir solucionar problemas que puedan surgir y que no se propaguen durante el desarrollo del proyecto. En concreto, se pretenden verificar los siguientes aspectos:

- El correcto funcionamiento de todos los componentes del sistema.
- El correcto funcionamiento de las relaciones entre los subsistemas de la aplicación.
- El correcto funcionamiento de la interfaz del sistema.
- El funcionamiento correcto del sistema integrado hardware y software en el entorno de operación.
- El funcionamiento del sistema es el esperado, siendo aceptado en cuanto a funcionalidad y rendimiento.

A continuación, se define el entorno de pruebas en lo referente al hardware y software, que será el mismo que el utilizado durante el desarrollo del proyecto.

5.1 Plataforma

5.1.1 Hardware

El equipamiento hardware utilizado para realizar las pruebas es el siguiente:

- **Ordenador portátil Acer**
  - **Modelo:** Acer Aspire 5750.
  - **Procesador:** Intel Core i5-2410M/2,3 GHz (Dual Core).
  - **Memoria:** 4 GB DDR3 SDRAM.
  - **Disco duro:** 750 GB Serial ATA-300 (5400 rpm).
  - **Sistemas operativos:** Microsoft Windows 7 Home Premium Edición 64 bits y Ubuntu 12.04 TLS.
  - **Tarjeta gráfica:** tarjeta integrada Intel HD Graphics 3000.
  - **Monitor:** Acer pantalla integrada 15,6 pulgadas.
  - **Teclado:** teclado integrado.
  - **Ratón:** touchpad integrado.
- **Ordenador de sobremesa Asus**
  - **Modelo:** AS V3-M2A69DG
  - **Procesador:** AMD Athlon 64 X2 Dual Core Processor 4800+/2,5 GHz.
  - **Memoria:** 4 GB DDR2
  - **Disco duro:** 250 GB Serial ATA (722 rpm).

- **Sistemas operativos:** Microsoft Windows XP Home Edition Service Pack 3.
- **Tarjeta gráfica:** ATI Radeon Xpress 1250 (700 MB)
- **Monitor:** LG Flatron L1511S.
- **Teclado:** Microsoft Natural PS/2 Keyboard.
- **Ratón:** Samsung MO-170B.

- **Impresora**
  - **Modelo:** HP PSC 1350 All-in-one Printer series.
- **Router**
  - **Modelo:** Observa Telecom AW4062.

Todas las pruebas se realizarán en el equipo Acer Aspire 5750 únicamente, salvo en las que se indique lo contrario.

5.1.2 Software

El software necesario para realizar las pruebas y documentarlas será el siguiente:

- **Sistemas operativos:** un equipo tendrá instalado Windows 7 y otro Windows XP.
- **Entorno:** ambos equipos tendrán instalado el IDE Eclipse Helios y el JDK 1.7.
- **Microsoft Office 2077:** se utilizará para realizar la documentación relativa a las pruebas.

5.2 Diseño de las pruebas

En este apartado se especifica todo lo relativo a las pruebas mínimas que debe superar el sistema para tener un comportamiento aceptable, lo que significará que cumple con las necesidades que se exponen en este documento.

5.2.1 Plantilla de las pruebas

La plantilla utilizada para las pruebas tendrá el siguiente formato:

XXX-xxx
Descripción
Objetivo
Requisitos relacionados
Resultado esperado
Resultado obtenido

Tabla 69. Plantilla de las pruebas

Los elementos que la componen son los siguientes:

- **Identificador:** se indica en la cabecera de la tabla. Está compuesto por dos partes, la primera (XXX) indicará el tipo de prueba, y puede ser de dos tipos:
  - **Prueba del sistema:** identificada por PRS.
  - **Prueba de rendimiento:** identificada por PRN.

  La segunda parte (xxx) indica el número de prueba, es un valor numérico correlativo para cada prueba de ese tipo.
- **Descripción:** una breve explicación de la prueba.
- **Objetivo:** una breve explicación de qué se pretende verificar con la prueba.
- **Requisitos relacionados:** requisitos a los que hace referencia, que están definidos en el apartado 3.4 de este documento.
- **Resultado esperado:** una breve descripción del resultado que se espera obtener con la prueba.
- **Resultado obtenido:** resultado que se ha producido al ejecutar la prueba.

5.2.2 Especificación de las pruebas

Debido a la naturaleza de la aplicación, no sólo es necesario que el sistema cumpla con su función, sino que también es importante el tiempo que emplee en ello. Se ha tenido en cuenta este hecho a la hora de elaborar las pruebas mínimas que debe superar el sistema con éxito para cumplir con las necesidades expresadas en este documento. Por tanto, las pruebas que se exponen en este punto son de dos tipos:

- **Pruebas del sistema:** comprueban el correcto funcionamiento de la aplicación sin utilizar cargas de trabajo elevadas, su función es simplemente constatar que el sistema funciona. Los requisitos que se

verifican con este tipo de pruebas son los requisitos de usuario de capacidad definidos en el apartado 3.4.2 de este documento.

- **Pruebas de rendimiento:** comprueban que los tiempos de ejecución y las cargas de trabajo que soporta el equipo en el que se instale el sistema son aceptables. Con estas pruebas se verifica que se cumple el requisito de usuario de restricción RUR-005 (véase apartado 3.4.3 de este documento). Se van a realizar pruebas de este tipo para los dos puntos importantes del sistema:
  - El muestreo del grafo
  - La generación del archivo de visualización

  Posteriormente se analizarán todos los datos obtenidos.

A continuación, se exponen de manera detallada las pruebas que se han realizado en la aplicación para verificar todo lo expuesto en los párrafos anteriores.

5.2.2.1 Pruebas del sistema

PRS-001	
Descripción	Configurar los archivos de atributos de los grafos de manera interactiva. Crear dos opciones de configuración para los vértices y otras dos para las aristas, y después almacenar los datos en los archivos.
Objetivo	Comprobar que la configuración interactiva de los atributos de los grafos funciona correctamente.
Requisitos relacionados	RUC-004, RUC-009, RUC-010, RUC-011, RUC-014, RUC-015, RUC-016, RUC-018, RUC-020, RUC-021, RUC-022, RUC-23, RUC-024, RUC-025, RUC-026, RUC-027, RUC-028, RUC-029, RUC-030, RUC-031, RUC-032, RUC-033
Resultado esperado	Configurar los atributos de los grafos sin errores y obtener los archivos de configuración. El formato de los archivos de configuración debe ser correcto.
Resultado obtenido	**Éxito**. No existen problemas durante el proceso de configuración y se generan los archivos de configuración de forma correcta.

Tabla 70. Prueba PRS-001

PRS-002	
Descripción	Obtener archivo de visualización a partir de un archivo de representación CSC de un grafo con 1000 nodos y un archivo asociado con los estados de los nodos, seleccionando un porcentaje de muestreo del 40 %.
Objetivo	Comprobar que se obtiene un archivo válido de visualización, que se puede cargar y analizar con la herramienta NodeXL.
Requisitos relacionados	RUC-001, RUC-002, RUC-006, RUC-007, RUC-008, RUC-011, RUC-012, RUC-019
Resultado esperado	Obtener un archivo de tipo *GraphML* que se puede visualizar correctamente con la herramienta NodeXL.
Resultado obtenido	**Éxito**. Se obtiene un archivo de tipo *GraphML* que se puede cargar y visualizar en NodeXL.

Tabla 71. Prueba PRS-002

PRS-003	
Descripción	Obtener archivo de representación del grafo y archivo con los estados de los vértices, a partir del archivo de tipo *GraphML* obtenido en la ejecución de la prueba PRS-002.
Objetivo	Comprobar que se obtienen los archivos CSC y de estados de los nodos obtenidos tras el muestreo en la prueba PRS-002.
Requisitos relacionados	RUC-003, RUC-006, RUC-007, RUC-008, RUC-011, RUC-013
Resultado esperado	Obtener dos archivos con extensión ".dat", que contienen la representación del grafo en formato CSC y los estados de los nodos. Los archivos representan la misma matriz que los archivos obtenidos tras la fase de muestreo en la prueba PRS-002.
Resultado obtenido	**Éxito**. Se obtienen los dos archivos esperados, ambos con el formato y los valores correctos.

Tabla 72. Prueba PRS-003

PRS-004	
Descripción	Acceso a la información sobre la aplicación.
Objetivo	Comprobar que se puede acceder al apartado que incluye la información sobre el sistema y sus autores.
Requisitos relacionados	RUC-005
Resultado esperado	Acceder a la información de la aplicación.
Resultado obtenido	**Éxito**. Se ha accedido al apartado en el que se indica la información sobre la aplicación.

Tabla 73. Prueba PRS-004

PRS-005	
Descripción	Cargar la configuración desde los archivos obtenidos en la prueba PRS-001.
Objetivo	Comprobar que se realiza correctamente la función de cargar la configuración desde archivo.
Requisitos relacionados	RUC-004, RUC-011, RUC-015, RUC-016, RUC-017
Resultado esperado	Se carga correctamente en el sistema la información de configuración de los archivos seleccionados.
Resultado obtenido	**Éxito**. Se muestra un mensaje de confirmación de carga de la configuración almacenada en los archivos.

Tabla 74. Prueba PRS-005

PRS-006	
Descripción	Modificación de los datos de una opción de configuración.
Objetivo	Comprobar que se pueden modificar los valores de los atributos de las opciones de configuración correctamente.
Requisitos relacionados	RUC-004, RUC-009, RUC-010, RUC-011, RUC-016, RUC-020, RUC-021, RUC-022, RUC-23, RUC-024, RUC-025, RUC-026, RUC-027, RUC-028, RUC-029, RUC-030, RUC-031, RUC-032, RUC-033
Resultado esperado	Modificación de la opción de configuración con éxito.
Resultado obtenido	**Éxito**. Se modifican los datos de la opción de configuración según el usuario lo ha indicado.

Tabla 75. Prueba PRS-006

PRS-007	
Descripción	Eliminación de una opción de configuración.
Objetivo	Verificar que se pueden borrar opciones de configuración existentes en la configuración de los atributos de los grafos.
Requisitos relacionados	RUC-004, RUC-011, RUC-016
Resultado esperado	Se elimina la opción de configuración, no pudiendo acceder a ella posteriormente para su modificación.
Resultado obtenido	Éxito. Se elimina la opción de configuración correctamente.

Tabla 76. Prueba PRS-007

5.2.2.2 Pruebas de rendimiento

Como ya se ha comentado, estas pruebas las vamos a separar en varios tipos, ya que posteriormente se van a analizar los resultados obtenidos individualmente. Estos subsistemas a probar son:

- Muestreo
- Generación del archivo de visualización GraphML

➢ **Pruebas de muestreo:**

PRN-001	
Descripción	Muestreo de un grafo de las siguientes características en su matriz dispersa: • Dimensión: 1000. • Elementos no nulos: 7892. • Porcentajes de muestreo: 10%, 25%, 50%.
Objetivo	Realizar el muestreo en un tiempo inferior a 3 segundos en todos los casos.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución de la prueba no sea superior a 3 segundos en ningún caso.
Resultado obtenido	Éxito. Los tiempos de ejecución son los siguientes: • Muestreo 10%: 0 segundos. • Muestreo 25%: 0 segundos. • Muestreo 50%: 0 segundos.

Tabla 77. Prueba PRN-001

PRN-002	
Descripción	Muestreo de un grafo de las siguientes características en su matriz dispersa: • Dimensión: 10000. • Elementos no nulos: 81680. • Porcentajes de muestreo: 10%, 25%, 50%.
Objetivo	Realizar el muestreo en un tiempo inferior a 7 segundos en todos los casos.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución de la prueba no sea superior a 7 segundos en ningún caso.
Resultado obtenido	**Éxito**. Los tiempos de ejecución son los siguientes: • Muestreo 10%: 0 segundos. • Muestreo 25%: 0 segundos. • Muestreo 50%: 0 segundos.

Tabla 78. Prueba PRN-002

PRN-003	
Descripción	Muestreo de un grafo de las siguientes características en su matriz dispersa: • Dimensión: 100000. • Elementos no nulos: 1291839. • Porcentajes de muestreo: 10%, 25%, 50%.
Objetivo	Realizar el muestreo en un tiempo inferior a 10 segundos en todos los casos.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución de la prueba no sea superior a 10 segundos en ningún caso.
Resultado obtenido	**Éxito**. Los tiempos de ejecución son los siguientes: • Muestreo 10%: 0 segundos. • Muestreo 25%: 0 segundos. • Muestreo 50%: 2 segundos.

Tabla 79. Prueba PRN-003

PRN-004	
Descripción	Muestreo de un grafo de las siguientes características en su matriz dispersa: • Dimensión: 500000. • Elementos no nulos: 8149283. • Porcentajes de muestreo: 10%, 25%, 50%.
Objetivo	Realizar el muestreo en un tiempo inferior a 4 minutos en todos los casos.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución de la prueba no sea superior a 4 minutos en ningún caso.
Resultado obtenido	**Éxito**. Los tiempos de ejecución son los siguientes: • Muestreo 10%: 4 segundos. • Muestreo 25%: 16 segundos. • Muestreo 50%: 60 segundos.

Tabla 80. Prueba PRN-004

PRN-005	
Descripción	Muestreo de un grafo de las siguientes características en su matriz dispersa: • Dimensión: 1000000. • Elementos no nulos: 24717236. • Porcentajes de muestreo: 10%, 25%, 50%.
Objetivo	Realizar el muestreo en un tiempo inferior a 8 minutos en todos los casos.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución de la prueba no sea superior a 8 minutos en ningún caso.
Resultado obtenido	**Éxito**. Los tiempos de ejecución son los siguientes: • Muestreo 10%: 17 segundos. • Muestreo 25%: 69 segundos. • Muestreo 50%: 4 minutos y 33 segundos.

Tabla 81. Prueba PRN-005

PRN-006	
Descripción	Muestreo de un grafo de las siguientes características en su matriz dispersa: • Dimensión: 2000000. • Elementos no nulos: 206606832. • Porcentajes de muestreo: 10%, 25%, 50%.
Objetivo	Realizar el muestreo en un tiempo inferior a 25 minutos en todos los casos.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución de la prueba no sea superior a 25 minutos en ningún caso.
Resultado obtenido	**Éxito**. Los tiempos de ejecución son los siguientes: • Muestreo 10%: 62 segundos. • Muestreo 25%: 4 minutos y 22 segundos. • Muestreo 50%: 16 minutos y 24 segundos.

Tabla 82. Prueba PRN-006

➢ Pruebas de generación del archivo de visualización GraphML:

Las pruebas de este tipo pretenden aceptar el rendimiento en cuanto a la generación del archivo de visualización. Para ello los resultados obtenidos en cuanto al tiempo de ejecución deben mantenerse dentro de los siguientes límites:

- Menor a 10 segundos si el tamaño del archivo generado es menor a 30 MB.
- Menor a 20 segundos si el tamaño del archivo generado está comprendido entre 30 MB y 80 MB.
- Menor a 1 minuto si el tamaño del archivo generado está comprendido entre 80 MB y 100 MB.
- Menor a 2 minutos si el tamaño del archivo generado está comprendido entre 100 MB y 130 MB.

Se utilizan los dos equipos de pruebas para comparar los resultados dependiendo de sus características. Teniendo en cuenta todo esto, se exponen las pruebas realizadas:

PRN-007	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 500. • Aristas: 1872. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 179 KB (7142 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): menor a 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): menor a 1 segundo.

Tabla 83. Prueba PRN-007

PRN-008	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 5000. • Aristas: 17191. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 1,65 MB (66605 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): menor a 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): menor a 1 segundo.

Tabla 84. Prueba PRN-008

PRN-009	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 15000. • Aristas: 32670. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 3,52 MB (143039 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): menor a 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): menor a 1 segundo.

Tabla 85. Prueba PRN-009

PRN-010	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 25000. • Aristas: 72544. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 7,36 MB (292658 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): menor a 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 1 segundo.

Tabla 86. Prueba PRN-010

PRN-011	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 35000. • Aristas: 124243. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 12,10 MB (477764 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): menor a 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 1 segundo.

Tabla 87. Prueba PRN-011

PRN-012	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 50000. • Aristas: 227307. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 21,30 MB (831968 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 4 segundos.

Tabla 88. Prueba PRN-012

PRN-013	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 65000. • Aristas: 351083. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 32,30 MB (1248317 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 6 segundos.

Tabla 89. Prueba PRN-013

PRN-014	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 100000. • Aristas: 291115. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 29,70 MB (1173371 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 6 segundos.

Tabla 90. Prueba PRN-014

PRN-015	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 125000. • Aristas: 419147. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 41,80 MB (1632476 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 1 segundo. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 8 segundos.

Tabla 91. Prueba PRN-015

PRN-016	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 150000. • Aristas: 570499. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 55,80 MB (2161523 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 2 segundos. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 11 segundos.

Tabla 92. Prueba PRN-016

PRN-017	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 175000. • Aristas: 748135. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 72 MB (2769434), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 3 segundos. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 14 segundos.

Tabla 93. Prueba PRN-017

PRN-018	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 200000. • Aristas: 935774. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 89 MB (3407345 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 3 segundos. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 19 segundos.

Tabla 94. Prueba PRN-018

PRN-019	
Descripción	Generación de un archivo GraphML a partir de un grafo de las siguientes características: • Nodos: 250000. • Aristas: 1377990. • Configuración del color en nodos y aristas.
Objetivo	Comprobar que el tiempo de ejecución se encuentra entre los límites adecuados.
Requisitos relacionados	RUR-005
Resultado esperado	El tiempo de ejecución se encuentra dentro de los límites correctos.
Resultado obtenido	**Éxito**. El archivo generado tiene un tamaño de 128 MB (4883996 líneas), y los tiempos de ejecución han sido de: • Equipo Acer Aspire (Intel Core i5 / 2,3 Ghz): 15 segundos. • Equipo Asus V3 (AMD Athlon 64 X2 / 2,5 Ghz): 68 segundos.

Tabla 95. Prueba PRN-019

5.3 Análisis de los resultados

En este apartado se van a analizar los tiempos de ejecución del sistema obtenidos en las pruebas de rendimiento y los archivos generados. Se valoran varias de las características a tener en cuenta para dar por válido el rendimiento de la aplicación, como son:

- El rendimiento obtenido durante la fase de muestreo, teniendo en cuenta el tiempo de ejecución y las propiedades de los grafos generados.
- La eficiencia del proceso de generación de archivos de visualización de grafos, valorando si los tiempos de ejecución obtenidos con distintos tamaños de grafos son aceptables.
- La validez de NodeXL como herramienta de visualización de los grafos obtenidos con nuestra aplicación. Se valora principalmente el tiempo de carga de los archivos y si la visualización de los grafos se puede realizar de manera clara, pudiendo distinguir los elementos y analizar las conexiones existentes.

5.3.1 Análisis del muestreo de grafos

En este apartado se van a estudiar los resultados obtenidos en las pruebas de rendimiento referentes al muestreo de grafos.

5.3.1.1 Análisis del tiempo empleado

Los resultados obtenidos en estas pruebas son los siguientes:

Prueba	Matriz	Porcentaje muestreo	Tiempo empleado (segundos)
PRN-001	Dimensión: 1000 Elementos no nulos: 7892 Tamaño: 47,8 KB	10%	0,12
		25%	0,12
		50%	0,14
PRN-002	Dimensión: 10000 Aristas: 81680 Tamaño: 603 KB	10%	0,19
		25%	0,23
		50%	0,31
PRN-003	Dimensión: 100000 Elementos no nulos: 1291839 Tamaño: 10,30 MB	10%	0,48
		25%	0,62
		50%	2
PRN-004	Dimensión: 500000 Elementos no nulos: 8149283 Tamaño: 71 MB	10%	4
		25%	16
		50%	60
PRN-005	Dimensión: 1000000 Elementos no nulos: 24717236 Tamaño: 215 MB	10%	17
		25%	69
		50%	273
PRN-006	Dimensión: 2000000 Elementos no nulos: 206606832 Tamaño: 751 MB	10%	62
		25%	262
		50%	984

Tabla 96. Tiempo de ejecución en fase de muestreo

Se puede observar como los tiempos necesarios aumentan con el tamaño del grafo y en función del porcentaje de muestreo. Como referencia indicar que para obtener una muestra del mayor archivo utilizado, que tiene un tamaño de 751 MB y dos millones de vértices, el tiempo necesario no es superior a 17 minutos. En la gráfica siguiente se puede observar cómo varía el tiempo de ejecución dependiendo del número de nodos del grafo de muestra generado:

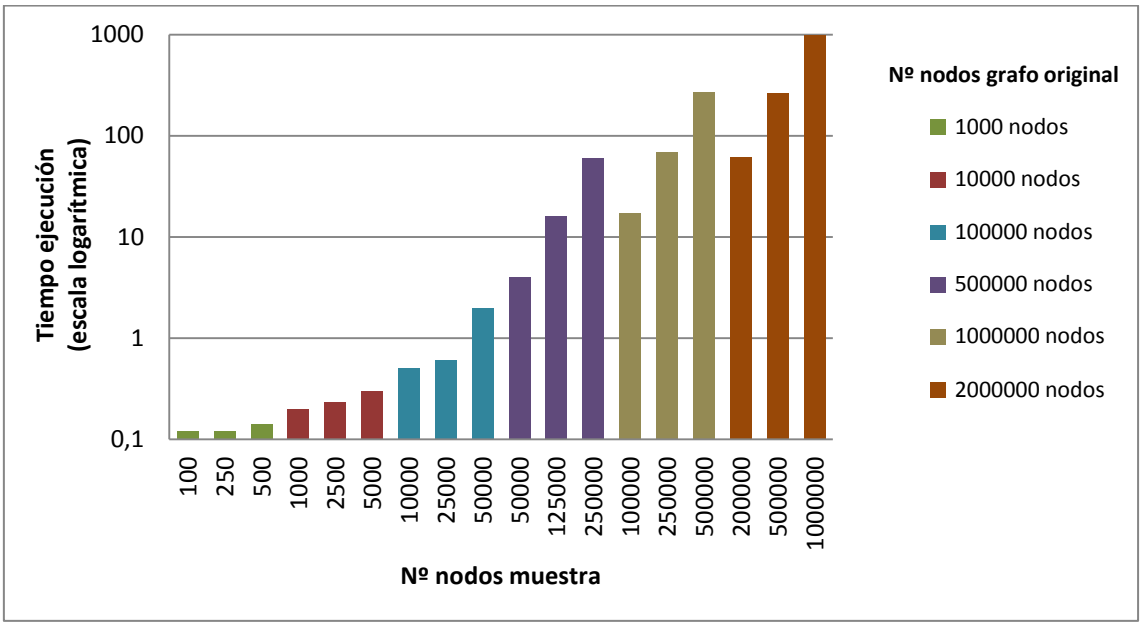

Ilustración 30. Tendencia tiempo de muestreo

Se puede observar que la evolución del tiempo de muestreo es exponencial según el número de vértices que tenga la muestra del grafo.

En conclusión, según los datos arrojados por las pruebas de muestreo, se puede apreciar que la técnica de muestreo es muy aceptable en cuanto a rendimiento, ya que el tiempo necesario es reducido en comparación con el tamaño de grafo a muestrear. El tiempo de muestreo depende principalmente del número de nodos del grafo a generar, siendo menos importante el número de vértices del grafo original. Esto se puede comprobar en la Ilustración 31, ya que se puede observar que en los puntos en los que los grafos de muestra tienen un número de vértices similar pero varía el número de nodos del grafo original, se obtienen tiempos similares, es decir, el número de nodos del grafo original no produce un impacto importante en el tiempo de muestreo. En la siguiente ilustración se señalan los siguientes aspectos:

1. Con un grafo de 500000 nodos, se necesitan 16 segundos para generar una muestra con 125000 vértices. De igual manera un grafo con 1000000 de nodos necesita 17 segundos para generar una muestra de 100000 vértices. El tamaño de la muestra es similar, por lo que el tiempo de ejecución también.
2. Con un grafo de 500000 nodos se necesitan 60 segundos para generar una muestra de 250000 vértices. A su vez, si el grafo tiene 1000000 de nodos necesita 69 segundos para generar una muestra del mismo tamaño. Por

su parte, un grafo de 2000000 de nodos necesita 62 segundos para generar una muestra de 200000 vértices. Como en el caso anterior, se puede observar que el tiempo de ejecución varía principalmente en función del tamaño de la muestra, no del tamaño del grafo original.

3. Con un grafo de 1000000 de nodos se necesitan 273 segundos para generar una muestra de 500000 vértices. Si el grafo tiene 2000000 de nodos, el tiempo necesario para generar una muestra del mismo tamaño es de 262 segundos. Tal y como se ha comentado en los dos casos anteriores, el tiempo de muestreo depende del tamaño de la muestra.

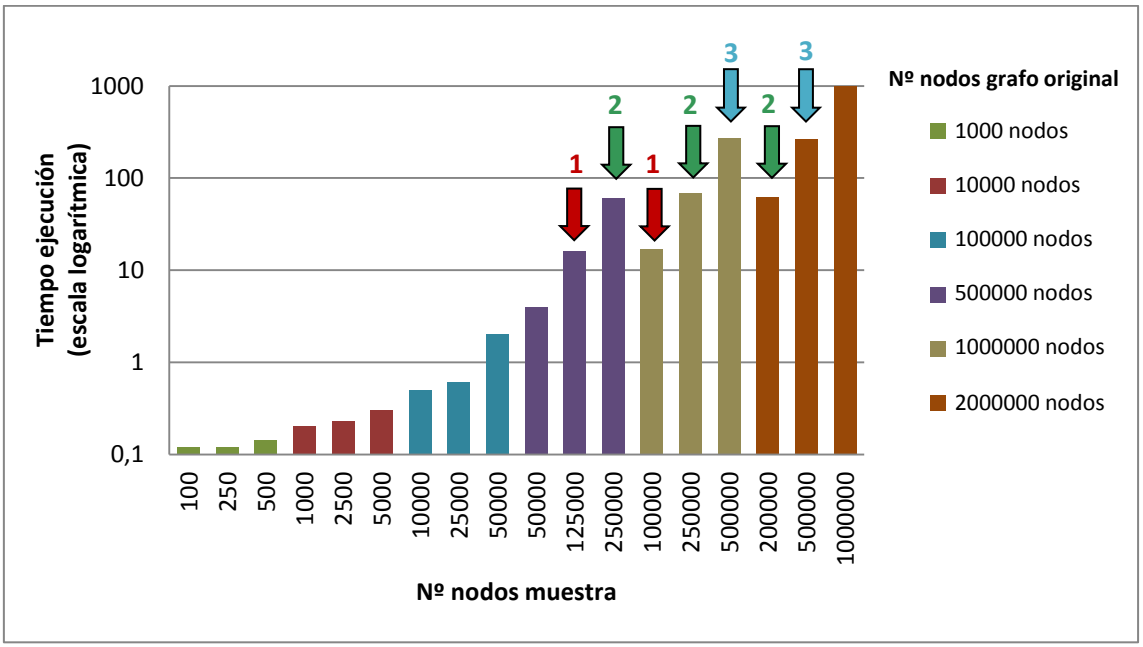

Ilustración 31. Evolución tiempo de muestreo según número de nodos del grafo

En definitiva, el rendimiento de la fase de muestreo es más que aceptable, cubre las necesidades del sistema y es un punto fuerte para los requisitos de este.

5.3.1.2 Análisis de las propiedades matemáticas de los grafos

Para estudiar las características de los grafos de muestra generados se van a utilizar los archivos de visualización de tipo *GraphML* generados durante las pruebas de rendimiento y las aplicaciones software NodeXL y Gephi, que son las que nos darán los valores de ciertos atributos de los grafos como son:

- **Número de vértices**.
- **Número de aristas**.
- **Grado medio**, que es la media de los grados de cada vértice.
- **Componentes conexos**, que son el número de agrupaciones de nodos conectadas entre sí de modo conexo.
- **Diámetro**, representa la mayor distancia de entre todos los pares de nodos del grafo.
- **Longitud media del camino**, es la distancia media del grafo entre todos los pares de nodos.

- **Densidad del grafo**, depende del número de aristas que tenga. Si el valor de la densidad se acerca a cero es que tiene pocas aristas, por lo que se denomina al grafo *disperso*. Si por el contrario el grafo contiene muchas aristas, el valor de la densidad se acercará a uno, diciendo que el grafo es denso.

Comparando estos valores para distintas muestras de grafos, se podrá ver la tendencia que sigue el muestreo en cuanto a las propiedades matemáticas de los grafos. Esto es importante para valorar la calidad de esta fase, porque además del tiempo empleado en realizar el proceso de muestreo también es muy importante que los archivos generados sean representativos de los grafos originales.

El grafo que se va a emplear para realizar este proceso tiene 10000 nodos y 40831 aristas. Se ha elegido esta dimensión de grafo porque tamaños mayores no son soportados por las herramientas software para el cálculo de métricas que se utilizan en este proyecto.

Los porcentajes de muestreo aplicados han sido: 10%, 25%, 35%, 50%.

A continuación, se exponen los datos obtenidos:

Primer grafo (10000 nodos)									
Propiedades	**Datos**	**Porcentaje de muestreo**							
		10%		25%		35%		50%	
Nº nodos	10000	1001	1001	2501	2501	3500	3501	5000	5001
Nº aristas	40831	1679	1682	5978	6117	10317	10014	16988	17585
Grado medio	8,166	3,355	3,363	4,781	4,894	5,897	5,723	6,796	7,035
Componentes conexos	2	1	4	4	5	4	3	3	3
Diámetro	9	21	20	14	15	12	13	12	11
Long. media camino	4,829	8,191	8,263	6,257	6,186	5,551	5,694	5,226	5,209
Densidad	0,001	0,003	0,003	0,001	0,001	0,001	0,001	0,001	0,001

Tabla 97. Propiedades grafos de muestra de un grafo con 10000 vértices

Se van a analizar tres propiedades importantes como son el grado medio, la longitud media de camino y el diámetro. De esta manera se estudiará el comportamiento del proceso de muestreo, viendo si es un muestreo estable y coherente, o si presenta altibajos e incongruencias. La densidad no presenta grandes variaciones, por ello no lo analizamos; esto es porque el grafo presenta muchas aristas.

- **Grado medio:**

Se va a comenzar a analizar el grado medio de los grafos. En las siguientes gráficas se puede observar la tendencia dependiendo del número de nodos y del número de aristas:

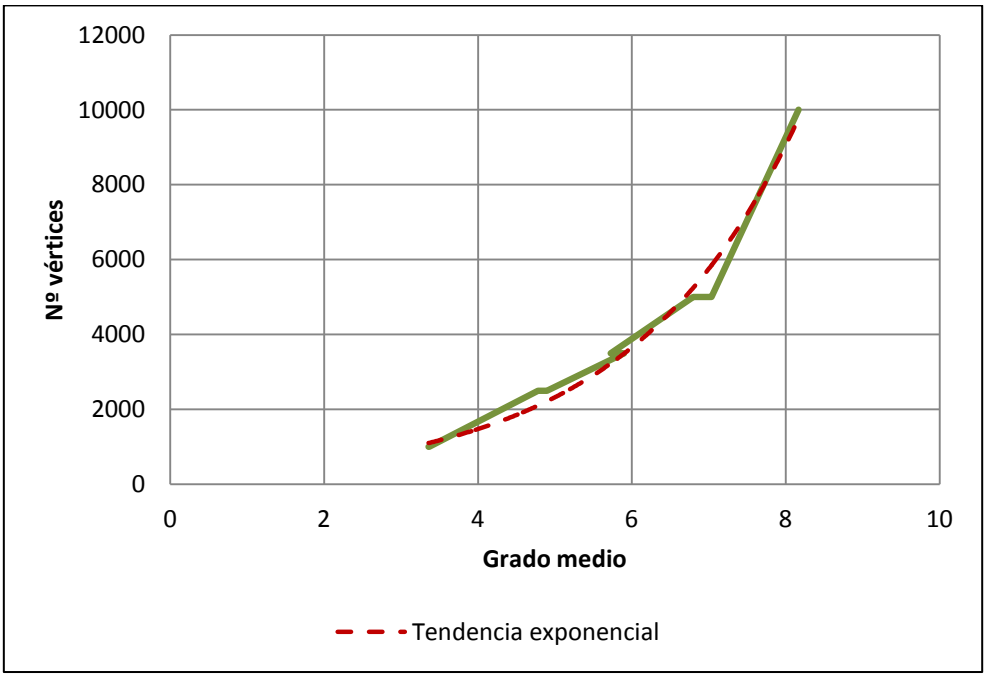

Ilustración 32. Grado medio según nº nodos - grafo de 10000 nodos

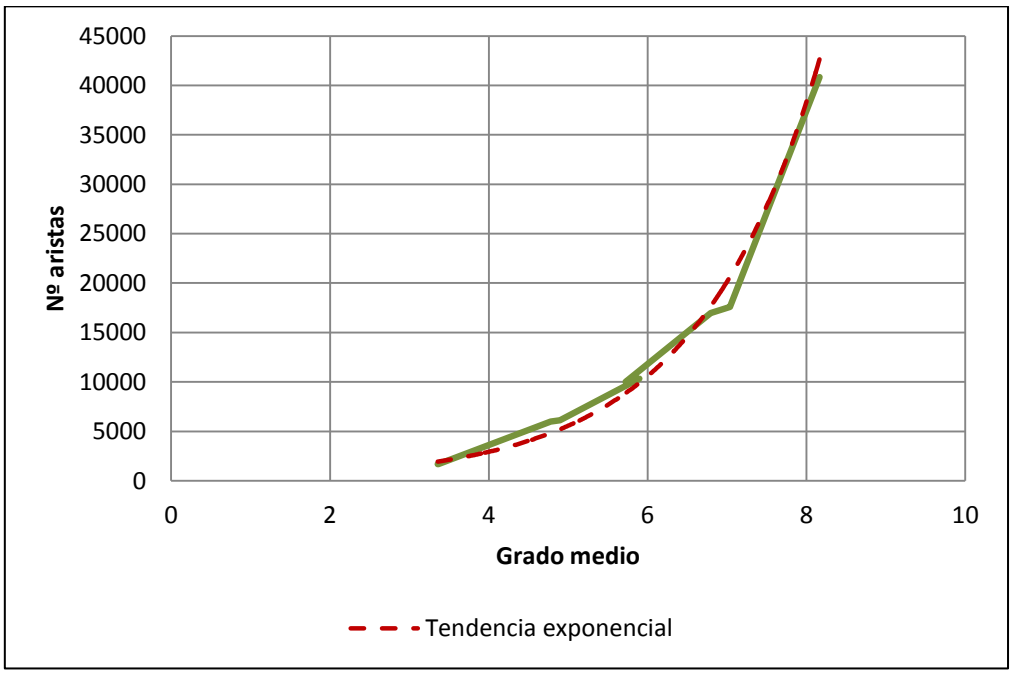

Ilustración 33. Grado medio según nº aristas - grafo de 10000 nodos

Se puede observar como las muestras generadas tienden a aumentar el grado medio, convergiendo este valor hacia el del grafo original. En ambas gráficas, se aprecia que la tendencia es exponencial, por lo que los valores de grado medio convergen exponencialmente a medida que aumentan el número de nodos y aristas.

Como parece lógico, al aumentar el número de nodos y aristas también aumenta el grado medio, ya que habrá más conexiones entre los vértices. Por tanto, la evolución de este valor de manera constante hasta llegar a datos similares a los del

grafo original hace evidente que el muestreo del grafo en este aspecto es estable y coherente.

- **Longitud media de camino:**

A continuación, se analiza el valor de la longitud media de camino, que indica la distancia media del grafo teniendo en cuenta cada par de nodos.

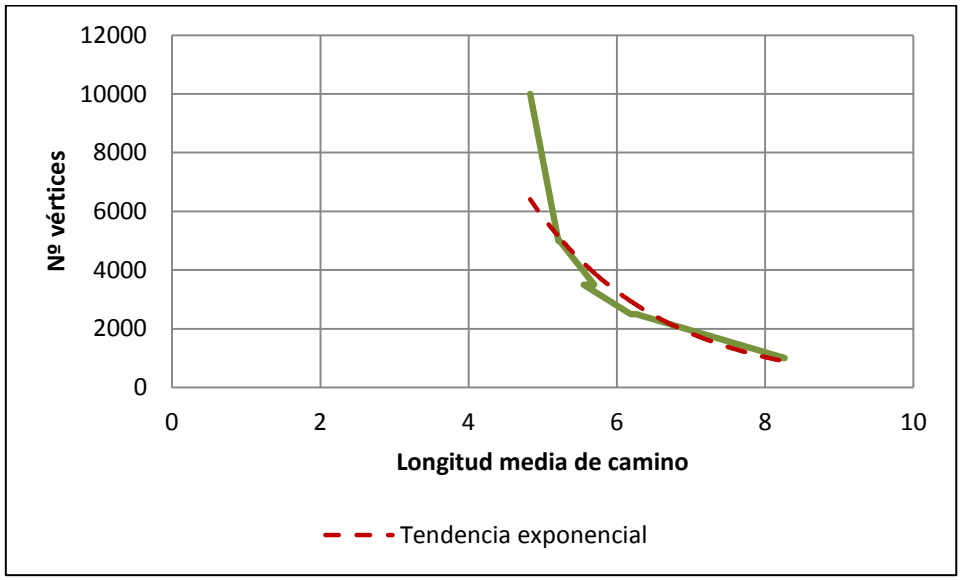

Ilustración 34. Longitud media camino según nº vértices - grafo de 10000 nodos

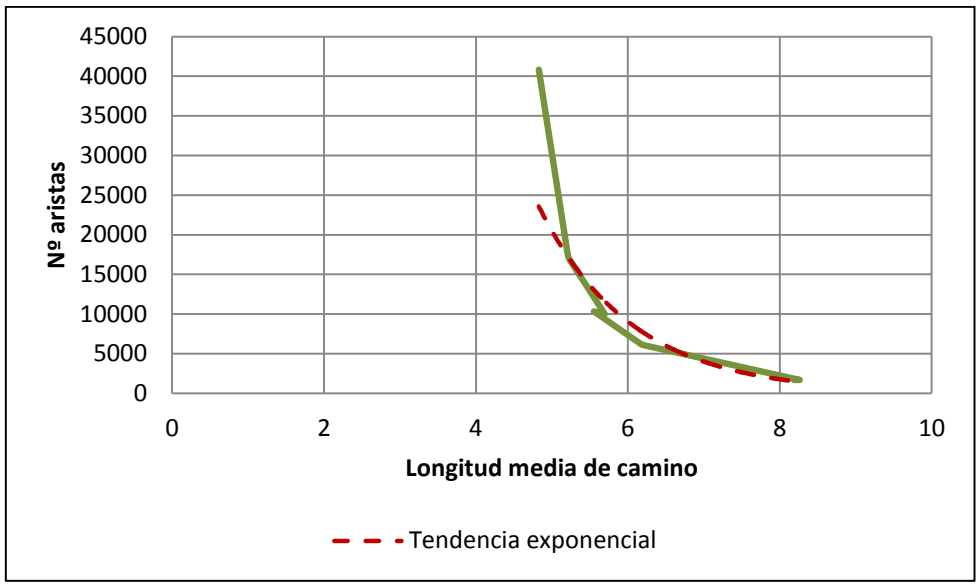

Ilustración 35. Longitud media camino según nº aristas - grafo de 10000 nodos

En las gráficas anteriores se puede observar que la longitud media de camino tiende hacia valores menores según aumentan el número de nodos y aristas. Esta evolución se ajusta bastante a una distribución exponencial. Esta tendencia es lógica ya que al aumentar el número de nodos y aristas hay más conexiones entre los vértices, lo que provoca que la longitud de camino entre cada par de vértices vaya

disminuyendo al haber más caminos disponibles. Como esta evolución se realiza de manera constante y uniforme, se puede confirmar que en lo referente a esta propiedad el muestreo también tiene efectos coherentes.

- **Diámetro:**

El diámetro indica la mayor distancia de entre todos los nodos del grafo. A continuación se muestra la evolución de este valor teniendo en cuenta el número de nodos y el número de aristas.

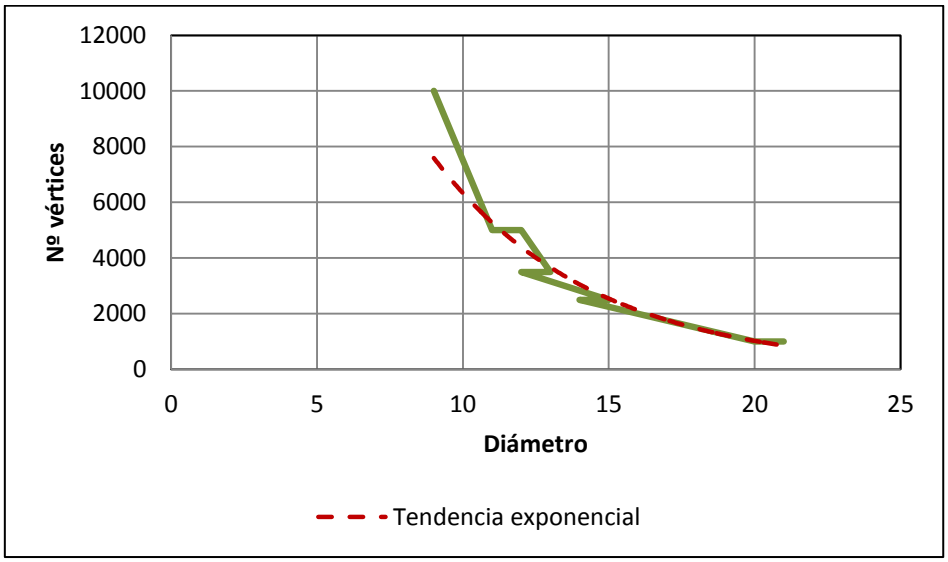

Ilustración 36. Diámetro según nº vértices - grafo de 10000 nodos

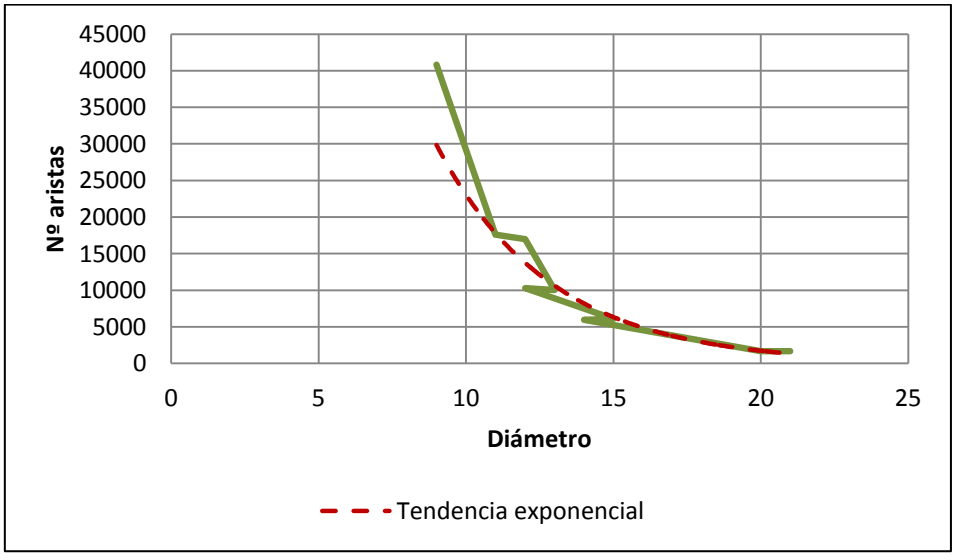

Ilustración 37. Diámetro según nº aristas - grafo de 10000 nodos

En las gráficas anteriores se aprecia como a medida que aumentan el número de nodos y aristas, disminuye el diámetro. Esta tendencia se ajusta a una exponencial, igual que ocurre con las demás propiedades. Esta evolución es lógica, ya que al

aumentar el tamaño del grafo aumentan los caminos existentes entre nodos, lo que hace que la mayor distancia entre un par de nodos sea cada vez menor.

En las gráficas se pueden apreciar pequeños altibajos, que hacen que la evolución del valor de esta propiedad no sea tan suave como ocurre con las demás propiedades. Esto no es preocupante, ya que puede ocurrir que al realizar el muestreo aparezcan agrupaciones de nodos que estén conectadas por pocas aristas, provocando que no haya muchos caminos disponibles entre algunos nodos. Pero al aumentar el porcentaje de muestreo estas irregularidades desaparecen. Este sería un punto a estudiar para posibles mejoras de la técnica de muestreo, aunque para las necesidades de este TFG el algoritmo Albatross cumple con las expectativas.

5.3.1.3 Valoración general

Tal y como hemos observado a lo largo de este apartado, la técnica de muestreo cumple con las necesidades del proyecto.

Por una parte, la calidad del muestreo en cuanto a propiedades de los grafos es aceptable, mostrando una evolución estable y coherente para las principales propiedades. Esto es importante, ya que demuestra que el proceso es uniforme, cosa que no ocurre con otros algoritmos de muestreo, tal y como se indicó en el estudio previo que se hizo de las técnicas de muestreo de grafos más populares.

En lo referente a los tiempos de ejecución del algoritmo reflejados en la Tabla 96, los resultados obtenidos son muy buenos, realizando el proceso de manera rápida. Esto demuestra que el algoritmo implementado es eficiente.

En lo referente a las herramientas que se han usado para calcular las propiedades de los grafos debemos señalar varias cosas:

- La mayoría de propiedades se han calculado con el programa Gephi, que es más potente que NodeXL en este aspecto. NodeXL tiene problemas al calcular las métricas en grafos grandes, a pesar de que los visualiza correctamente.
- La precisión de Gephi es menor, ya que sólo ofrece tres decimales de precisión. Esto en algunos casos puede ser un problema, por ejemplo con los valores muy cercanos a cero (como se puede observar por ejemplo en el valor de la densidad).

Por tanto, con las pruebas realizadas se verifica que el subsistema de muestreo de la aplicación tiene un comportamiento aceptable.

5.3.2 Análisis de generación del archivo de visualización GraphML

En esta sección se van a analizar los resultados obtenidos en las pruebas de rendimiento referentes a la generación de archivos de visualización de grafos.

Se han utilizado los dos equipos de pruebas definidos en el apartado 5.1 para poder estudiar el rendimiento dependiendo de las distintas características que poseen los equipos.

A continuación, se muestra un resumen de las pruebas de este tipo realizadas:

Prueba	Nº nodos	Nº aristas	Tamaño archivo GraphML	Tiempo equipo Acer Aspire (segundos)	Tiempo equipo Asus V3 (segundos)
PRN-007	500	1872	179 KB	<1	<1
PRN-008	5000	17191	1,65 MB	<1	<1
PRN-009	15000	32670	3,52 MB	<1	<1
PRN-010	25000	72544	7,36 MB	<1	1
PRN-011	35000	124243	12,10 MB	<1	1
PRN-012	50000	227304	21,30 MB	1	4
PRN-013	65000	351083	32,30 MB	1	6
PRN-014	100000	291115	29,70 MB	1	6
PRN-015	125000	419147	41,80 MB	1	8
PRN-016	150000	570499	55,80 MB	2	11
PRN-017	175000	748135	72 MB	3	14
PRN-018	200000	935774	89 MB	3	19
PRN-019	250000	1377990	128 MB	15	68

Tabla 98. Resumen pruebas de generación de archivos GraphML

Se puede apreciar en la Tabla 98 que el tiempo de ejecución depende principalmente del tamaño del archivo a generar, ya que cuantos más elementos haya que plasmar en el archivo *GraphML* mayor será el tiempo de ejecución. En la gráfica siguiente se aprecia visualmente este hecho:

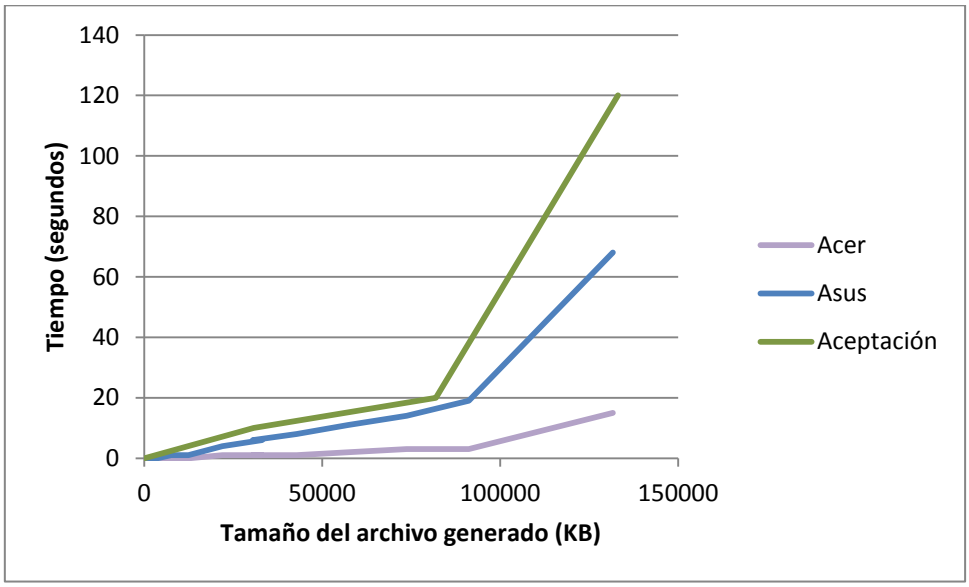

Ilustración 38. Tiempos de ejecución de generación del archivo GraphML por tamaño de archivo

Se puede observar que los dos equipos cumplen con los valores de aceptación de las pruebas de rendimiento, aunque el equipo de pruebas Asus ofrece un peor rendimiento. Este hecho puede ser debido al procesador que utiliza (AMD Athlon frente a Intel i5) y a que posee menos núcleos (dos a frente a cuatro). Por tanto, esto no es preocupante, ya los tiempos obtenidos son muy válidos en ambos equipos de pruebas.

Aunque es lógico pensar que el tamaño de archivo generado depende de los vértices y aristas a incluir en él, se incluyen a continuación las gráficas que pretenden corroborar esto:

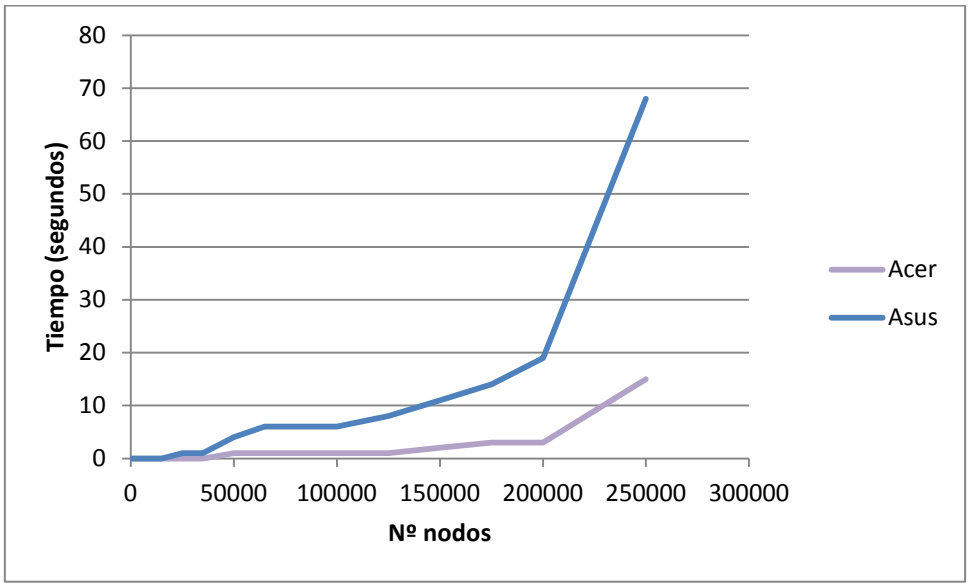

Ilustración 39. Tiempo de ejecución del archivo GraphML según nº nodos

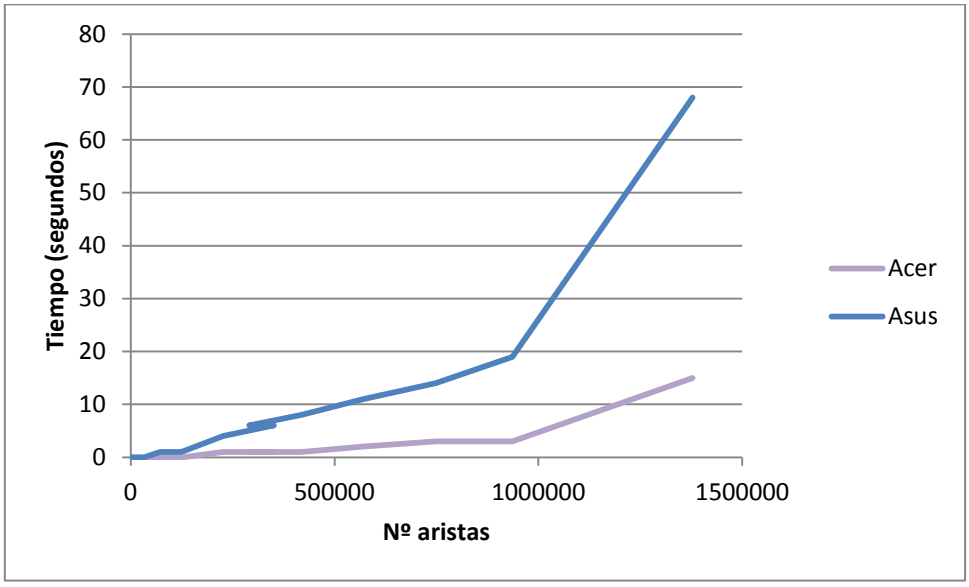

Ilustración 40. Tiempo de ejecución del archivo GraphML según nº aristas

Como ya se había indicado, la evolución del tiempo de ejecución dependiendo de las aristas y nodos del grafo generado es similar a la obtenida según el tamaño de archivo.

Se debe concluir comentando que estos tiempos son muy aceptables para la finalidad del sistema, ya que se prevé que los grafos a generar difícilmente tendrán más de 50000 vértices debido a las limitaciones de las herramientas de visualización. De hecho el archivo generado en la prueba PRN-019, con un tiempo inferior a 70 segundos en ambos equipos de pruebas, no sería soportado por NodeXL debido a las limitaciones que ofrece Microsoft Excel en el tamaño de la hoja de cálculo [14] (el archivo generado tiene más de un millón de aristas siendo este el máximo número de filas disponibles en Microsoft Excel).

5.3.3 Análisis de visualización en NodeXL

En este apartado se va a analizar el comportamiento de NodeXL al cargar los archivos generados en la fase de pruebas, tanto en lo referente al tiempo que tarda en cargar los datos en la hoja de cálculo, como la demora en mostrar la visualización del grafo. La siguiente tabla resume estos tiempos:

Nº nodos	Nº aristas	Tiempo carga archivo (segundos)	Tiempo visualización (segundos)
100	217	0	0
500	1775	0	0
1000	1092	0	1
5000	7199	3	24
7000	10989	4	50
10000	17619	5	110
15000	32642	8	265
20000	51172	13	455
25000	72544	21	1200
30000	96831	23	1380
35000	124243	38	2100
40000	155918	39	2400
45000	191986	45	2880
50000	227304	53	3600

Tabla 99. Rendimiento NodeXL

5.3.3.1 Análisis del tiempo de carga de datos

Como se puede observar en la Tabla 99, la carga de datos en NodeXL es bastante rápida, mostrándose en la hoja de cálculo los datos de un grafo de 50000 vértices y 227304 aristas en menos de un minuto. El máximo de datos que puede cargar NodeXL se limita a las restricciones de Microsoft Excel, que como se puede observar en [14] para la versión de 2007 es de algo más de un millón de filas. En la siguiente gráfica se aprecia la evolución de los tiempos de carga en relación con el número de vértices y aristas del grafo:

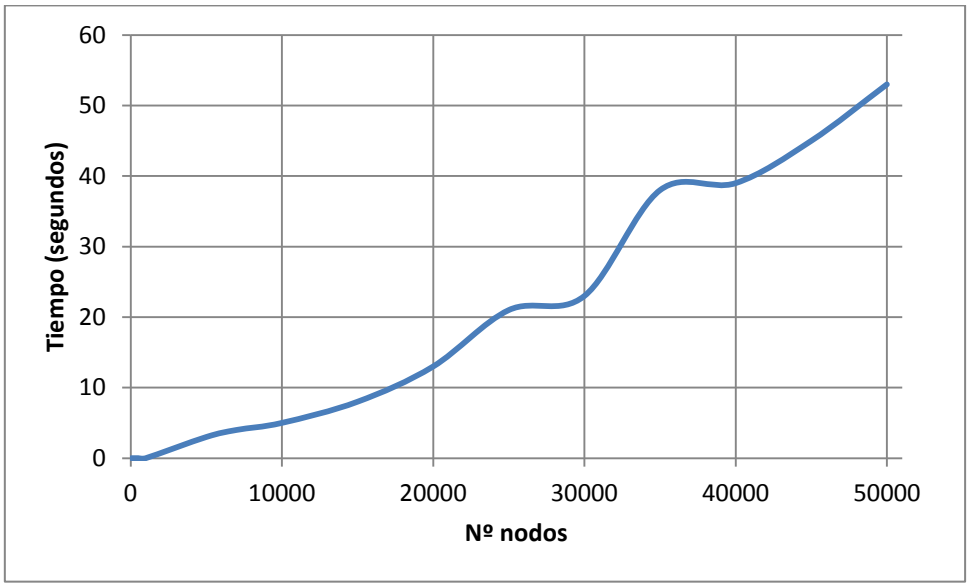

Ilustración 41. Tiempo de carga de datos en NodeXL en función del nº nodos

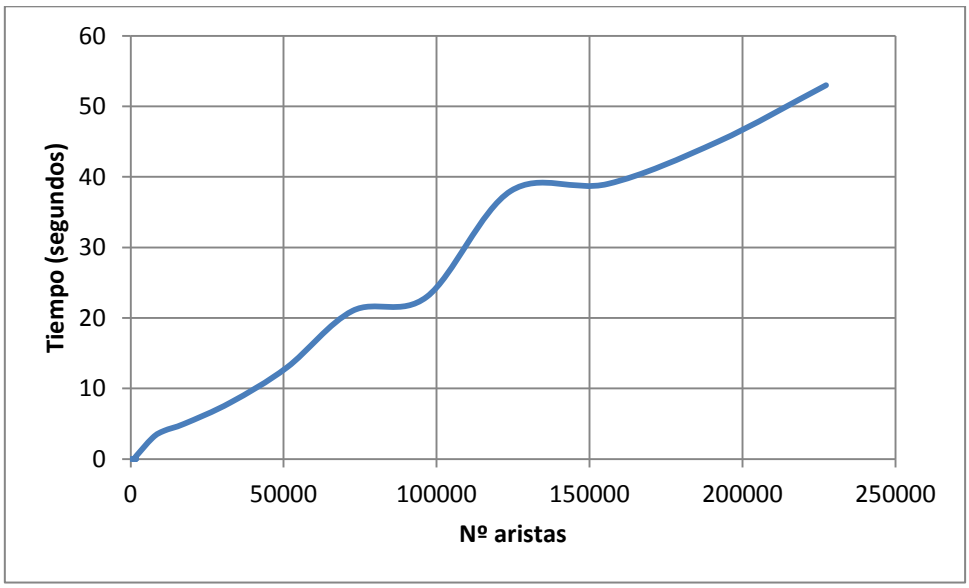

Ilustración 42. Tiempo de carga de datos en NodeXL en función del nº aristas

En las ilustraciones anteriores se puede observar una tendencia lineal con algunas fluctuaciones, por tanto es de prever que incluso grafos de alrededor de 500000 nodos se cargarían en un tiempo de entre cinco a diez minutos, lo que es un rendimiento muy bueno.

5.3.3.2 Análisis del tiempo de carga de la visualización del grafo

Ya se ha visto que el tiempo de carga de los datos es muy pequeño en comparación al tamaño de los grafos, sin embargo la visualización en pantalla es más costosa debido a la parte gráfica que implica. En las pruebas que se han realizado sólo se han configurado los atributos de color de nodos y vértices, si se configuraran más atributos seguramente el tiempo de carga sería mayor. En las siguientes ilustraciones se pueden observar los tiempos obtenidos:

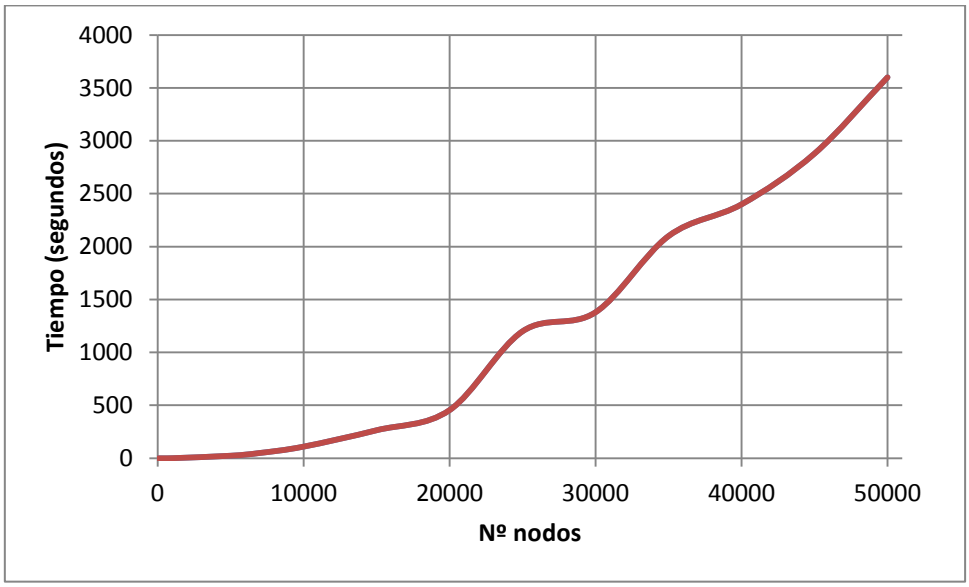

Ilustración 43. Tiempo carga la visualización del grafo en NodeXL por nº nodos

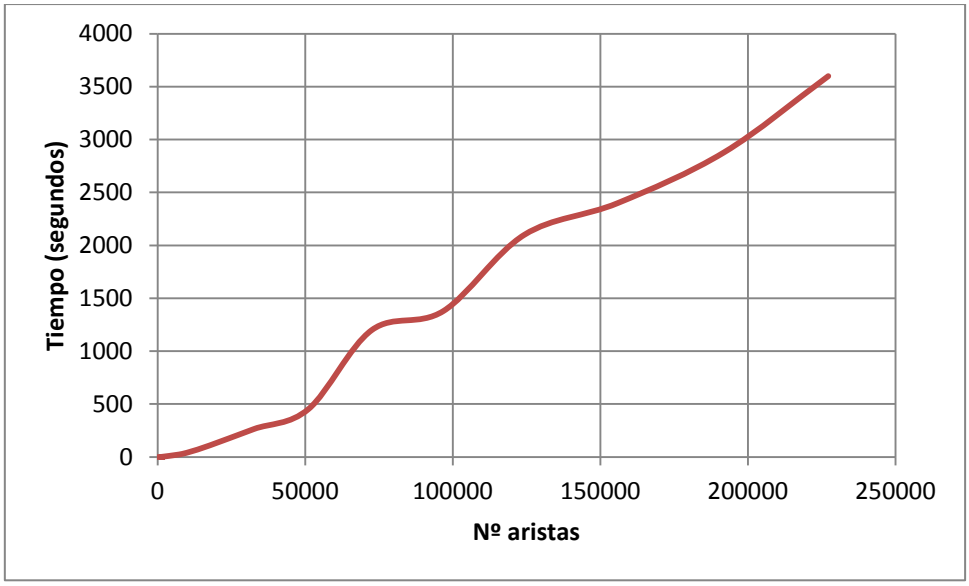

Ilustración 44. Tiempo carga la visualización del grafo en NodeXL por nº aristas

Para grafos de a partir de 10000 aristas el tiempo de carga empieza a sobrepasar el minuto, llegando a ser de casi una hora con un grafo de 50000 nodos y 227304 aristas. La tendencia que sigue es lineal, como en el caso de la carga de datos en la hoja de cálculo, pero en este caso la pendiente es mayor.

Los tiempos obtenidos son altos, pero no excesivos si se tiene en cuenta que una vez cargada la visualización se pueden manejar multitud de opciones que modifican el grafo en tiempo real. Además, para analizar grafos de una manera exhaustiva, es necesario tener una muestra no demasiado grande, ya que sino se verían todos los elementos apilados sin poder relacionarlos unos con otros

adecuadamente. Se prevé que los grafos que se van a utilizar no superarán los 50000 nodos, así que los resultados obtenidos en este análisis son aceptables.

5.3.3.3 Análisis de la calidad de visualización del grafo

En este apartado se va a valorar qué tamaño de grafo es óptimo para una visualización clara en NodeXL, es decir, con qué tamaños de grafo se visualizan claramente los elementos y se pueden analizar las relaciones existentes. Aunque en las imágenes aquí añadidas parezca que no se pueden analizar claramente las relaciones entre los nodos, en las explicaciones que se adjuntan se indicará si realmente es así o no, ya que con NodeXL hay varias opciones de visualización que te permiten hacer escala, aplicar zoom, seleccionar nodos y ver sus conexiones resaltadas en pantalla, etc. Esto no es apreciable en las imágenes aquí adjuntas, aunque sí de manera interactiva con NodeXL.

En primer lugar se va a comenzar con un grafo de 150 nodos y 381 aristas, se obtiene la siguiente visualización:

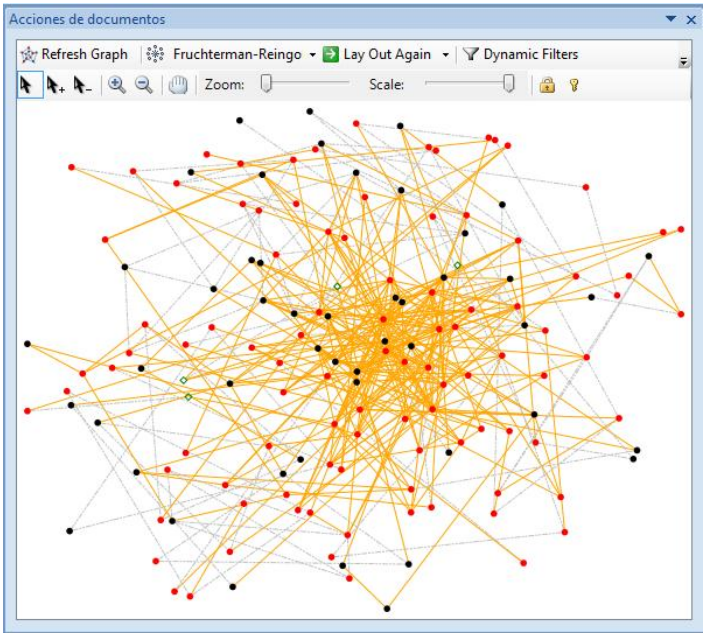

Ilustración 45. Visualización en NodeXL de grafo con 150 vértices

Con este tamaño se pueden ver los elementos de manera clara. No es necesario hacer uso de las opciones de NodeXL, en la imagen anterior se ven nítidamente todos los elementos.

El siguiente grafo que se va a mostrar es uno 250 nodos y 767 aristas:

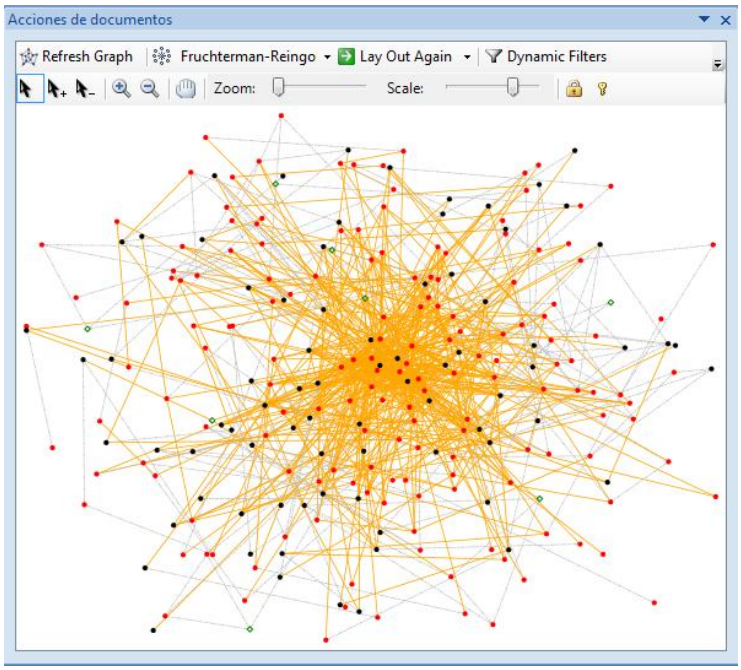

Ilustración 46. Visualización en NodeXL de grafo con 250 vértices

Se siguen pudiendo visualizar los nodos con las herramientas que ofrece NodeXL, aunque ya empieza a ser más difícil distinguir algunas conexiones.

El siguiente grafo a analizar tiene 350 vértices y 1304 aristas:

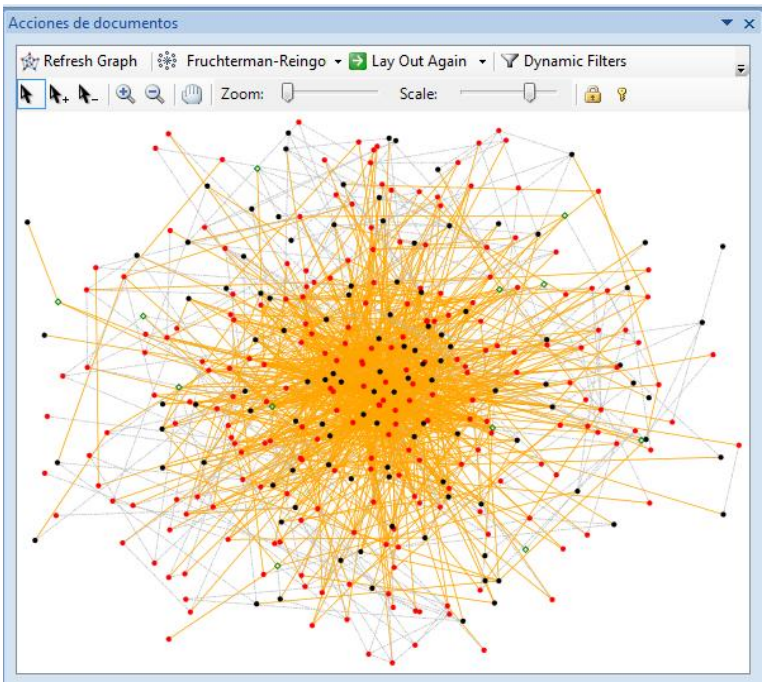

Ilustración 47. Visualización en NodeXL de grafo con 350 vértices

La visualización de los caminos entre los nodos empieza a complicarse, aunque con las opciones de visualización de NodeXL se siguen pudiendo distinguir los elementos que queramos analizar.

El siguiente grafo que se prueba a visualizar tiene 450 vértices y 1795 aristas:

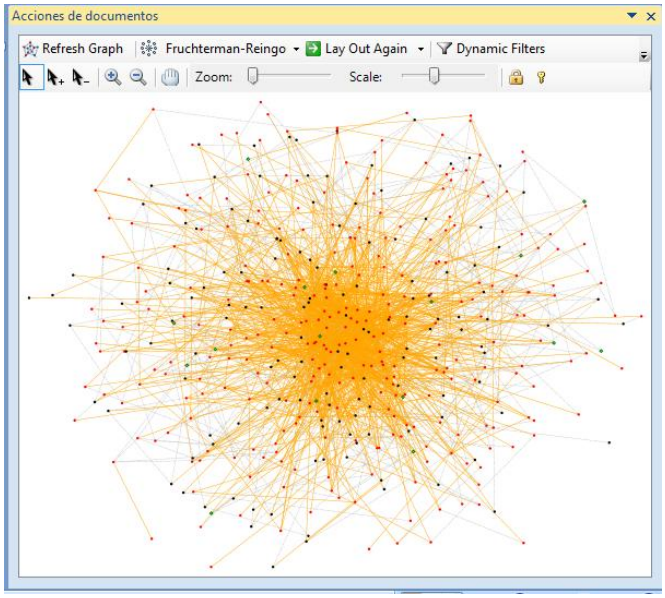

Ilustración 48. Visualización en NodeXL de grafo con 450 vértices

Se puede seguir analizando el grafo, pero en esta imagen ha tenido que emplearse la escala para facilitar la visualización. Los caminos existentes entre los nodos también son visibles de manera interactiva.

A continuación se prueba con un grafo algo más grande que tiene 600 nodos y 2535 aristas:

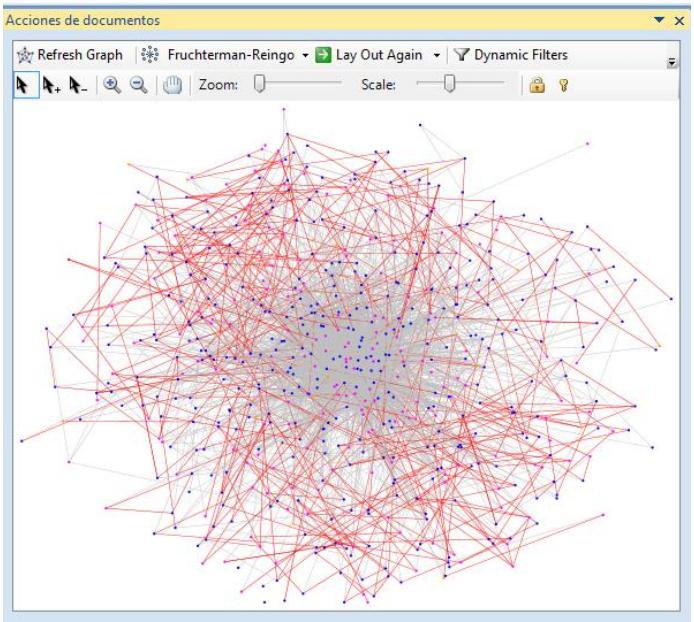

Ilustración 49. Visualización en NodeXL de grafo con 600 vértices

Ya empieza a complicarse el análisis de las conexiones, ya que para los nodos centrales (que tienen mayor grado) es complicado ver a donde se dirigen cada una de sus conexiones aunque se seleccionen y aparezcan resaltadas en la visualización.

Por tanto, si lo que se busca es poder analizar claramente los caminos existentes entre los nodos quizá éste sea el valor límite: alrededor de 3000 ó 4000 conexiones. Si por el contrario, lo que se pretende es poder analizar los vértices sin importar excesivamente conocer todas sus conexiones, se puede llegar a mayores tamaños de grafo. Se ha estipulado que un valor límite para una buena visualización en este caso sería alrededor de 7000-10000 conexiones, como ocurre en este caso:

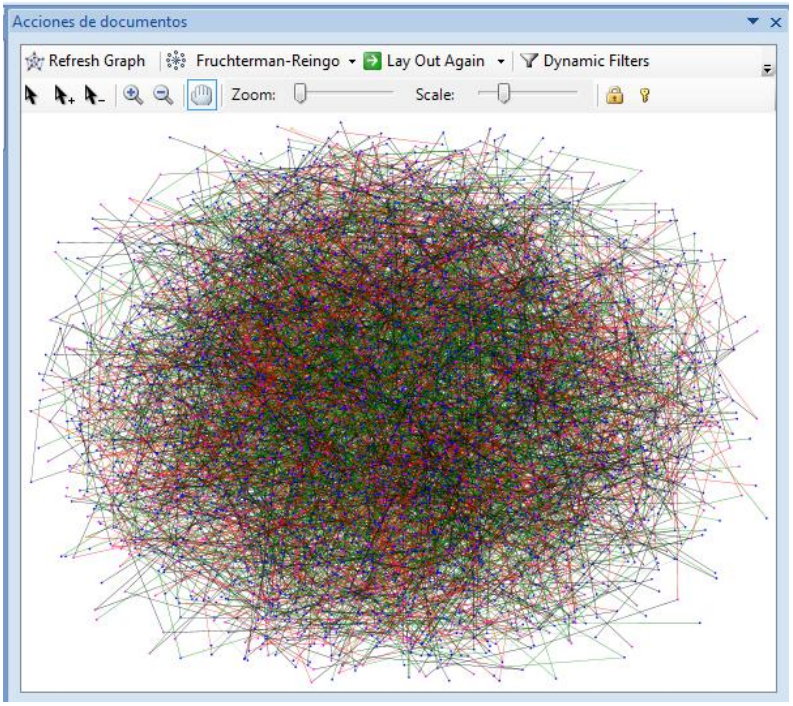

Ilustración 50. Visualización en NodeXL de grafo con 5000 vértices

Aunque en principio pueda parecer que la visualización de los nodos del grafo de la Ilustración 50 es complicada, esto no es completamente cierto, ya que aplicando el zoom y estudiando los nodos de manera más cercana se pueden seguir viendo las conexiones y los vértices implicados. Pero sí es verdad, que el límite para una buena visualización de los nodos se encuentra sobre estos valores.

5.4 Análisis de consistencia

Tras haber realizado todas las pruebas especificadas en esta sección, se va a adjuntar la matriz de trazabilidad que relaciona cada prueba de sistema con los requisitos que verifica, comprobando de esta manera que las pruebas realizadas cubren todos los requisitos de usuario de capacidad (véase apartado 3.4.2):

	PRS-001	PRS-002	PRS-003	PRS-004	PRS-005	PRS-006	PRS-007
RUC-001		X					
RUC-002		X					
RUC-003			X				
RUC-004	X				X	X	X
RUC-005				X			
RUC-006		X	X				
RUC-007		X	X				
RUC-008		X	X				
RUC-009	X					X	
RUC-010	X					X	
RUC-011	X	X	X		X	X	X
RUC-012		X					
RUC-013			X				
RUC-014	X						
RUC-015	X				X		
RUC-016	X				X	X	X
RUC-017					X		
RUC-018	X						
RUC-019		X					
RUC-020	X					X	
RUC-021	X					X	
RUC-022	X					X	
RUC-023	X					X	
RUC-024	X					X	
RUC-025	X					X	
RUC-026	X					X	
RUC-027	X					X	
RUC-028	X					X	
RUC-029	X					X	
RUC-030	X					X	
RUC-031	X					X	
RUC-032	X					X	
RUC-033	X					X	

Tabla 100. Matriz de trazabilidad requisitos de capacidad-pruebas de sistema

En lo referente a los requisitos de usuario de restricción (véase apartado 3.4.3), sólo se han definido pruebas para verificar el cumplimiento del requisito RUR-005, no se incluye la matriz de trazabilidad de este caso ya que todas las pruebas de rendimiento (apartado 5.2.2.2) tratan de cubrir este requisito. El resto de requisitos de restricción versan sobre la metodología y herramientas a utilizar, por lo que para comprobar estos requisitos basta con estudiar la documentación del proyecto y el software del mismo.

6 Caso de estudio: "Propagación de una enfermedad contagiosa"

Tras realizar las pruebas que aseguran el correcto funcionamiento y rendimiento del sistema, se ha creído importante realizar un caso de estudio sobre el tema origen de este TFG: la propagación del virus de la gripe A.

Para desarrollar este caso de estudio se utilizan datos reales de una simulación de *EpiGraph* [2, 3], con una población de 10000 personas y tomando el estado de los individuos durante distintos instantes de tiempo para comprobar la evolución de la propagación. En este apartado se mostrará el análisis de esta evolución con el sistema desarrollado en este TFG.

En primer lugar, se deben definir qué representan los valores asociados a los elementos del grafo y el color con el que se representan. El archivo en el que se encuentran los datos del grafo en formato de matriz dispersa con esquema CSC contiene los valores asociados a las aristas. Estos valores representan lo siguiente:

- **Valores 1 y 4:** relaciones laborales. Color gris.
- **Valor 2:** relaciones sociales. Color azul.
- **Valor 3:** relaciones familiares. Color oliva.

El archivo con los estados de los vértices contiene los siguientes valores:

- **Valor 0:** individuo susceptible (sano). Color verde.
- **Valor 1:** individuo en estado latente primario. Color azul.
- **Valor 2:** individuo en estado latente secundario. Color rosa.
- **Valor 3:** individuo en estado infeccioso primario. Color naranja.
- **Valor 4:** individuo en estado infeccioso secundario. Color rojo.
- **Valor 5:** individuo en estado infeccioso con tratamiento de antivirales. Color púrpura..
- **Valor 6:** individuo en estado asintomático. Color amarillo.
- **Valor 7:** individuo en estado hospitalizado. Color marrón.
- **Valor 15:** individuo en estado removido (ha superado la enfermedad). Color lima.
- **Valor 16:** individuo en estado fallecido. Color negro.

Se va a analizar la propagación con dos porcentajes de muestreo, uno con el 25%, lo que serían 2500 individuos, para poder ver de manera más o menos clara cómo evoluciona la propagación sobre una muestra reducida. Y posteriormente, se utilizará un porcentaje mayor para hacer un análisis más global.

6.1 Análisis con 2500 individuos

Los datos proporcionados por *EpiGraph* se van a dividir en seis fases para observar cómo se produce el contagio entre los miembros de la población.

6.1.1 Fase inicial del contagio

En un primer momento el estado de la población es el siguiente:

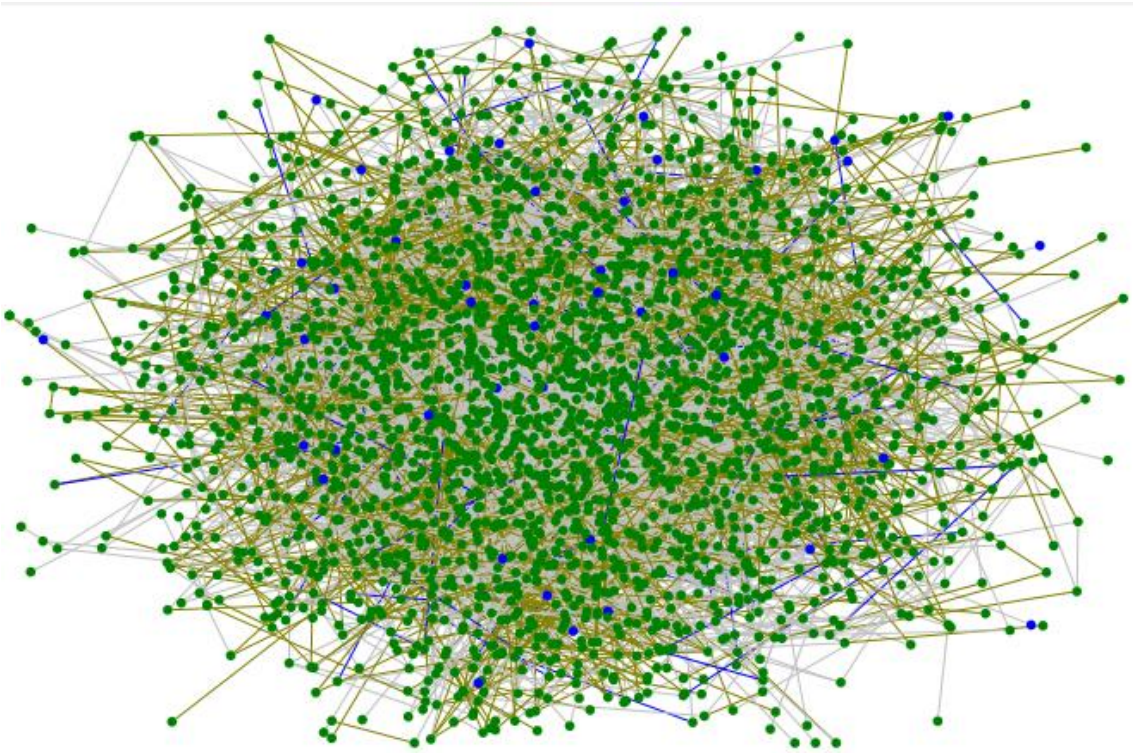

Ilustración 51. Inicio de la propagación del virus

Como se puede observar, en esta primera fase hay dos tipos de individuos:

- **Individuos sanos.** Con color verde, es el comienzo del contagio y son susceptibles de infectarse.
- **Individuos en estado latente primario**. Con color azul, es una captura del momento inicial, por lo que estos individuos son los que darán comienzo al contagio en las siguientes fases.

En cuanto a las relaciones existentes entre la población, simplemente comentar que la mayoría de las mismas son laborales (color gris) y familiares (color oliva), habiendo pocas relaciones de tipo social (color azul).

6.1.2 Segunda fase del contagio

En la siguiente fase de la propagación del virus se puede observar lo siguiente:

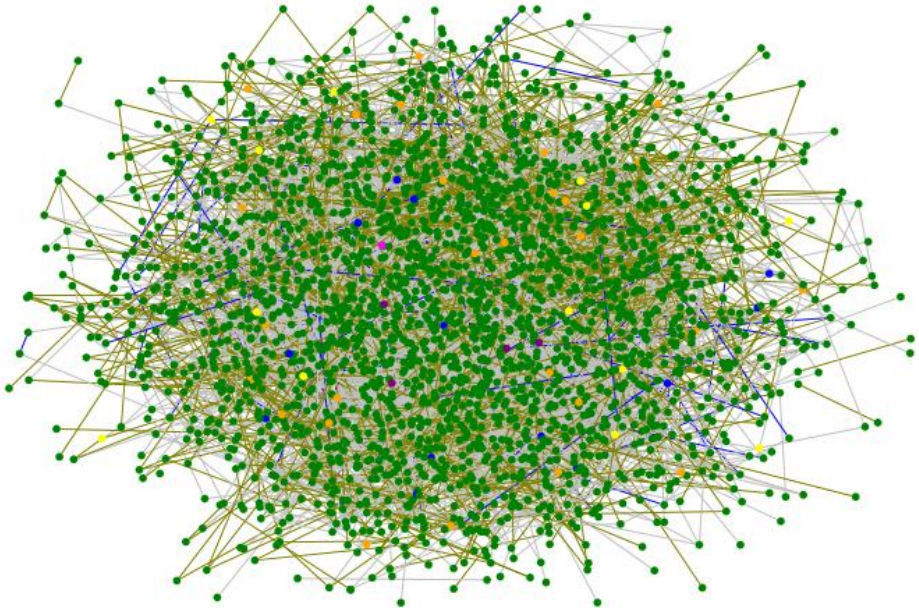

Ilustración 52. Fase 2 de la propagación del virus

En esta segunda fase empiezan a aparecer individuos asintomáticos (color amarillo), es decir, no presentan muestras de infección. Por otra parte, se puede observar el inicio del contagio, surgiendo los primeros individuos en estado latente secundario (color rosa), individuos en estado infeccioso primario (color naranja) e individuos en estado infeccioso con tratamiento de antivirales (color púrpura). Es decir, ya existe una infección latente y algunos miembros de la población comienzan a tomar medicamentos para hacer frente al virus.

6.1.3 Tercera fase del contagio

En la tercera fase de propagación la infección ya está muy presente:

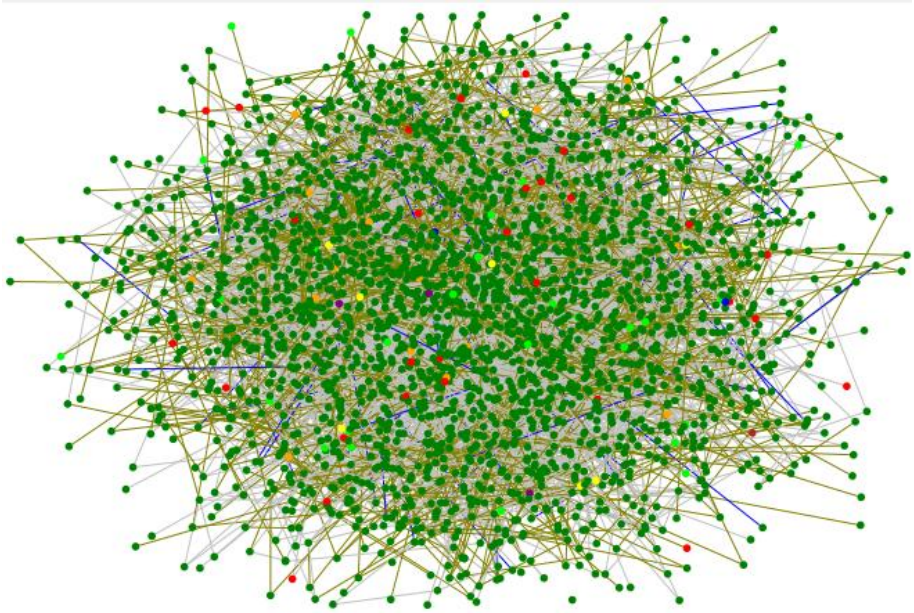

Ilustración 53. Fase 3 de la propagación del virus

En esta tercera fase la infección ya es evidente, ya que aparecen los primeros individuos en estado infeccioso secundario (color rojo), mientras que otros individuos ya han superado con éxito la enfermedad (color verde lima). Por otra parte, siguen existiendo individuos incubando el virus y algunos que toman antivirales para combatir la enfermedad.

6.1.4 Cuarta fase del contagio

En esta fase se observa lo siguiente:

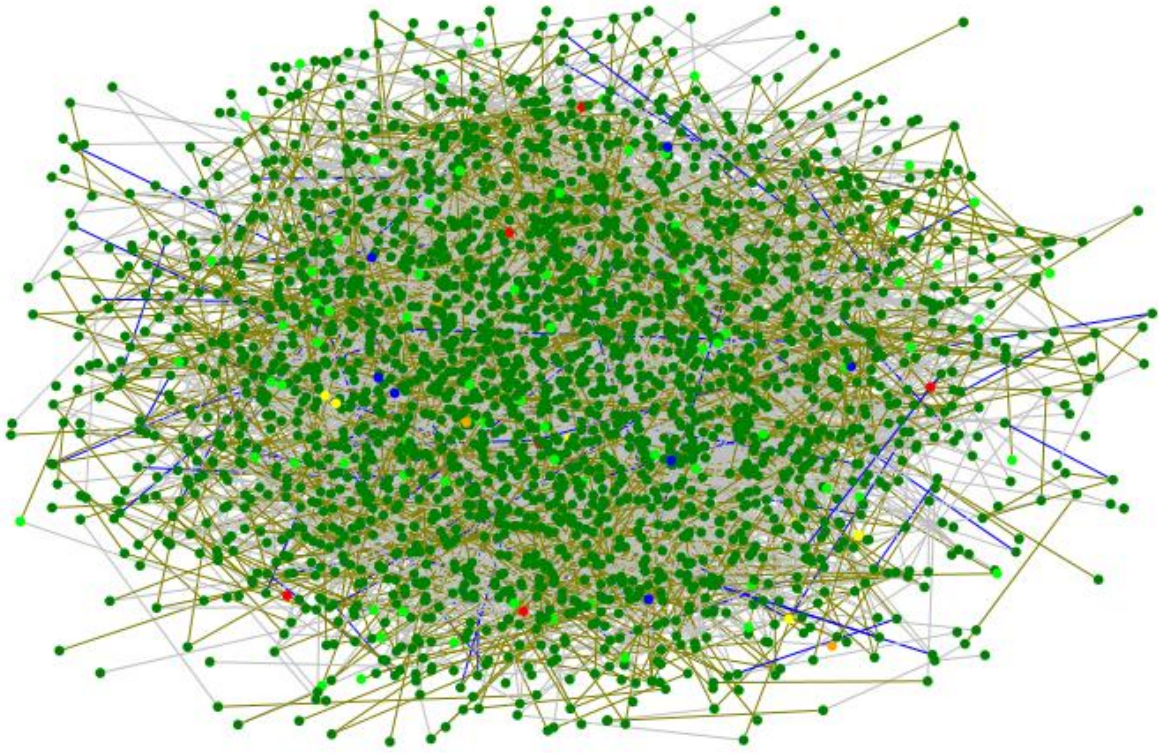

Ilustración 54. Fase 4 de la propagación del virus

En esta fase de la propagación se ve claramente como la infección comienza a remitir. Aparecen numerosos individuos que han superado la enfermedad, aunque en un porcentaje muy bajo todavía hay individuos en estado latente e infeccioso.

6.1.5 Quinta fase del contagio

En esta fase la propagación del virus se ha mantenido controlada, ya que se mantiene constante el número de individuos en estado latente o infeccioso, lo que indica que los contagios que se hayan producido habrán sido de forma aislada, quizá debido a las precauciones que haya podido tomar la población al tomar conciencia de una epidemia. También parece apreciarse un ligero aumento de los individuos que superan la enfermedad, haciéndose evidente que la propagación del virus está cerca de llegar a su fin. En la Ilustración 55 puede observarse todo lo comentado en este párrafo.

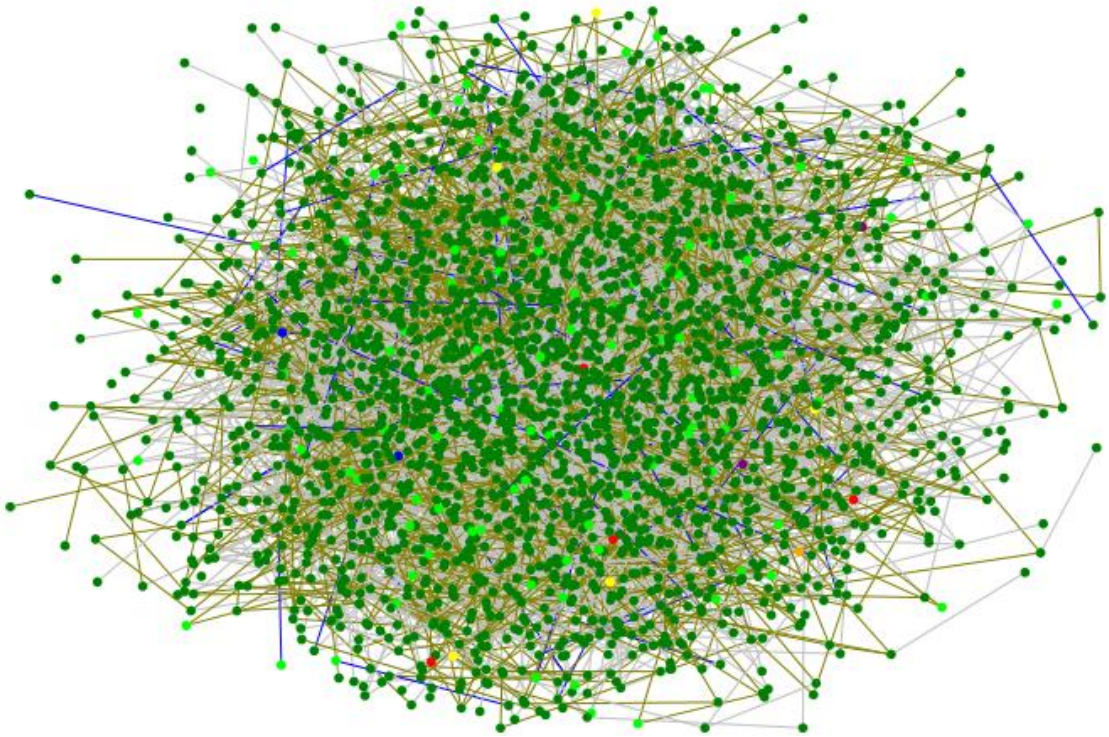

Ilustración 55. Fase 5 de la propagación del virus

6.1.6 Fase final del contagio

En la siguiente imagen se puede apreciar esta última fase de la propagación del virus:

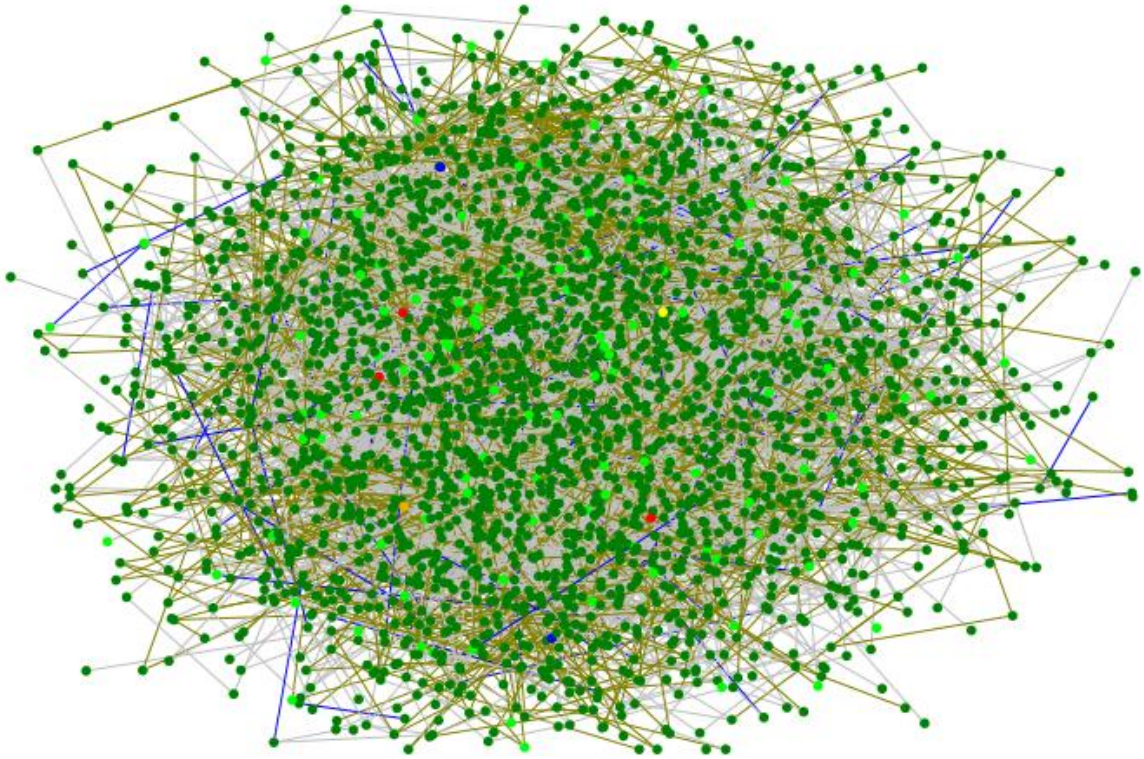

Ilustración 56. Fase final de la propagación del virus

Se puede apreciar en el grafo anterior que se ha controlado la infección, incrementándose el número de individuos curados y no aumentando el de infectados. Se puede sacar como conclusión que el contagio del virus se ha estabilizado, llegando a la fase final de propagación del mismo.

Tras haber hecho una panorámica general de cómo evoluciona el contagio de un virus dentro de una población de 2500 habitantes, se puede constatar que el proceso de propagación se realiza de la siguiente manera:

1. Algunos miembros de la población incuban el virus, siendo ellos los que darán comienzo a la infección en las siguientes fases.
2. Comienza la propagación del virus, apareciendo los primeros individuos en estado infeccioso que se encargan de contagiar a otros miembros de la población.
3. La propagación del virus llega a su momento álgido, estando infectada gran parte de la población y realizándose un inevitable contagio masivo.
4. Se controla la infección al verse alertada la población de la existencia de una epidemia. Se toman medidas para prevenir y curar la enfermedad. Una gran parte de la población infectada logra su curación, disminuyendo en gran medida el número de individuos en estado infeccioso.
5. Se mantiene el control sobre la propagación del virus, reduciéndose a un porcentaje muy bajo el número de individuos en estado infeccioso, lo que indica que la propagación del virus ha finalizado.

6.2 Análisis con 6000 individuos

En este apartado se pretende dar una visión general de cómo evoluciona la propagación del virus en una muestra con 6000 individuos. De esta manera se podrá observar el proceso de contagio identificado en el apartado 6.1 de manera más evidente, en una población mayor. Esto se resume en la siguiente ilustración:

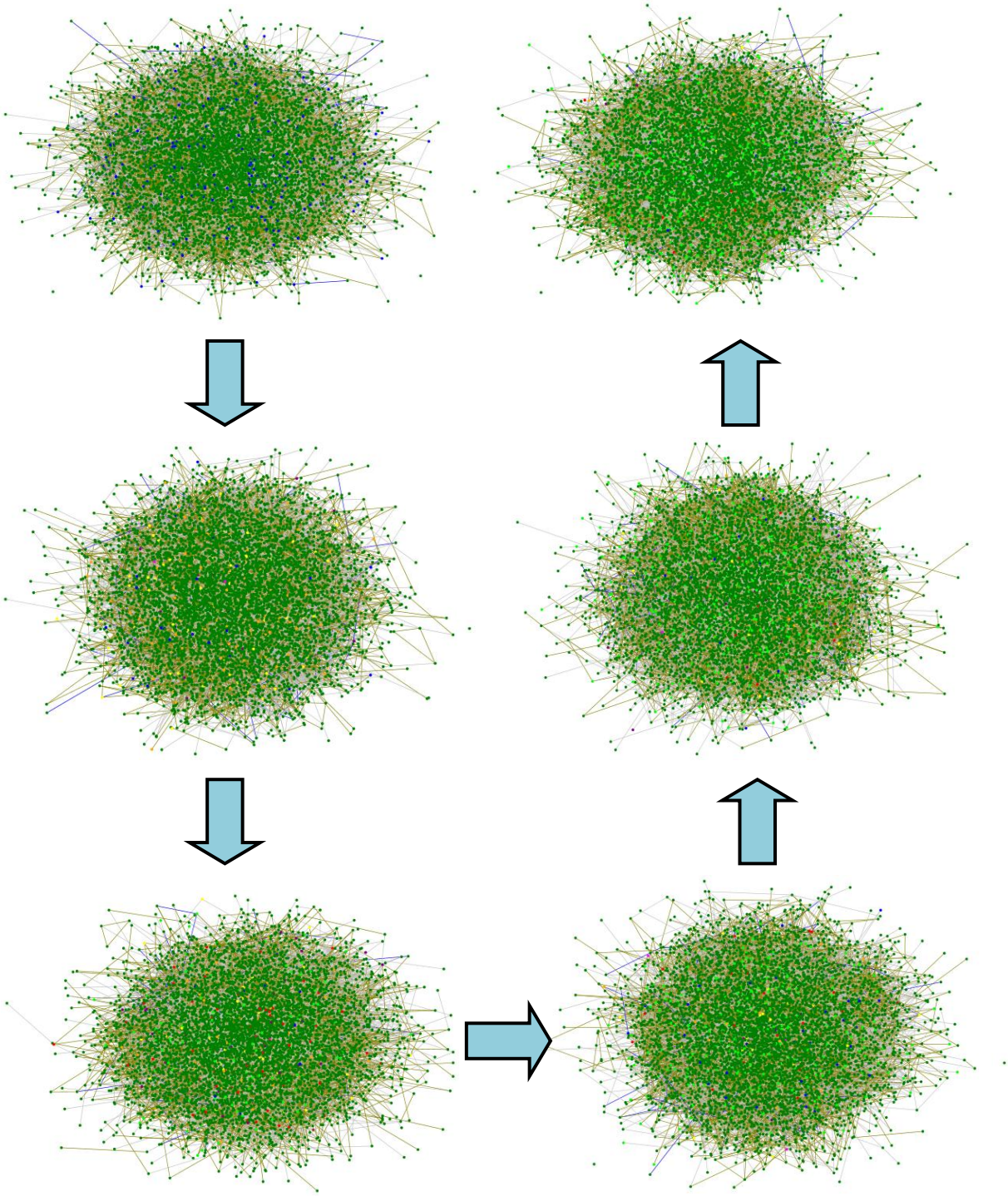

Ilustración 57. Propagación del virus en una población de 6000 individuos

En la imagen anterior se pueden ver claramente las fases diferenciadas con la muestra de 2500 individuos. En primer lugar la mayoría de la población está sana, surgiendo los primeros individuos en estado latente. En la siguiente fase da comienzo la fase de contagio, surgiendo personas en estado infeccioso. Después, se propagará el virus sin control, llegando el contagio a su punto máximo. A partir de aquí comienza la fase de cura, en la que los individuos comienzan a superar la enfermedad. Este proceso de cura continúa hasta que se controla la propagación, manteniéndose los individuos en estado infeccioso en un porcentaje muy bajo hasta llegar al final de la propagación.

Como se ha podido ver a lo largo de este apartado, el empleo del sistema desarrollado en este TFG va a ayudar a la interpretación de los resultados de propagación de la gripe que proporciona *EpiGraph*. Esto es debido a dos aspectos:

- Mediante el muestreo se simplifica el tamaño del grafo a visualizar, facilitando el análisis de los elementos del mismo.
- Es posible una configuración de la visualización, permitiendo la elección de atributos (como por ejemplo el color) para los vértices y aristas del grafo, lo que facilita la interpretación de los datos.

7 Presupuesto

En este apartado se realiza un análisis detallado del coste presupuestado para el presente proyecto. Se expone un desglose por categorías del cálculo de costes y finalmente se detalla el presupuesto final del proyecto. En todos los costes expresados en este apartado se incluyen los impuestos pertinentes.

7.1 Descripción del proyecto

Las características principales del TFG son las siguientes:

- **Autor:** Lorena Camino Rey
- **Título:** Herramienta de visualización para un simulador de propagación de enfermedades contagiosas.
- **Duración:** La fecha de comienzo del proyecto es el 1 de febrero de 2012, y la fecha de finalización prevista data del 4 de septiembre de 2012. El tiempo total de realización del proyecto se estima en 646 horas durante 155 días de trabajo.

7.2 Costes de personal

En el desarrollo del presente proyecto están involucradas tres personas:

- **David Expósito Singh**, tutor del TFG.
- **Maria-Cristina Marinescu,** tutora del TFG.
- **Lorena Camino Rey**, alumna encargada del desarrollo del TFG que cumple las funciones de analista, diseñador, programador y probador del proyecto.

El tiempo estimado, tal y como se indica en el apartado 4.1 de este documento, es de 646 horas, de las cuales, 86 horas corresponden al trabajo de los tutores. Estas horas son las relacionadas con las reuniones, la revisión de la documentación, la aceptación del proyecto, etc.

A continuación, se muestra el desglose de personal, considerando en el coste el gasto por IRFP y Seguridad Social. Para definir el coste individual se ha tenido en cuenta la experiencia que posee el alumno. Se incluye el desglose por tareas de la alumna encargada del proyecto y a continuación el resumen de los costes de personal globales.

Fase	Coste/Hora	Total horas	Total coste
Planificación y configuración	34 €	30	1.020 €
Análisis	44 €	43	1.892 €
Diseño	43 €	40	1.720 €
Implementación	28 €	248	6.944 €
Pruebas	24 €	96	2.304 €
Documentación	37 €	103	3.811 €
Total		**560**	**17.691 €**

Tabla 101. Desglose coste personal del alumno

Integrante	Rol	Coste/Hora	Horas de trabajo	Coste
David Expósito Singh	Tutor	63 €	43	2.709 €
Maria-Cristina Marinescu	Tutor	63 €	43	2.709 €
Lorena Camino Rey	Alumno			17.691 €
Total coste de personal				**23.109 €**

Tabla 102. Coste de personal

7.3 Costes de hardware

Para la realización del proyecto, la realización de las pruebas y la elaboración de la documentación han sido necesarios los equipos ya expuestos en el apartado 3.3.2.1 de este documento. El proyecto ha tenido una duración de siete meses, y teniendo en cuenta un tiempo de amortización de ocho años, como se expone en el *Real Decreto 1777/2004* [37], sólo se costeará la parte proporcional.

Por tanto, el coste de hardware es el siguiente:

Equipo	Importe	Coste[1]
Acer Aspire 5750	603,45 €	44 €
Asus AS V3-M2A69DG	538,70 €	39,28 €
Impresora HP PSC 1350	51,50 €	3,76 €
Router AW4062[2]	47,03 € mensual	329,21 €
Total coste hardware		**416,25 €**

Tabla 103. Costes de hardware

[1] Teniendo en cuenta 8 años de amortización
[2] El importe corresponde al gasto mensual de conexión ADSL

7.4 Costes de software

El coste de software está asociado a las licencias necesarias para los sistemas operativos y aplicaciones empleadas en el desarrollo de este proyecto. Se hará un desglose de la totalidad de software utilizado, pero en algunas ocasiones el coste real será inexistente debido a que es software ya incluido en los equipos adquiridos, o son herramientas descargadas gratuitamente desde la plataforma MSDN [30] aprovechándose del convenio existente entre la UC3M y Microsoft. El resto de software se estima en base al *Real Decreto 1777/2004* [37], que expone que la amortización debe ser de seis años para sistemas y programas informáticos.

Software	Precio	Coste[3]	Coste final
Microsoft Windows 7 (preinstalado en el equipo Acer Aspire 5750)	119 €	11,57 €	0 €
Microsoft Windows XP Home Edition (preinstalado en el equipo Asus AS V3)	-	-	0 €
Ubuntu 12.04 TLS	0 €	0 €	0 €
Microsoft Office 2007	93,81 €	9,12 €	9,12 €
Microsoft Project 2010 (disponible en la plataforma MSDN)	1.067 €	103,74 €	0 €
Microsoft Visio 2010 (disponible en la plataforma MSDN)	593 €	57,65 €	0 €
COCOMO II	0 €	0 €	0 €
Eclipse Helios	0 €	0 €	0 €
NetBeans 7.1.2	0 €	0 €	0 €
Notepad++	0 €	0 €	0 €
MinGW	0 €	0 €	0 €
NodeXL	0 €	0 €	0 €
Gephi	0 €	0 €	0 €
Total coste software		**182,08 €**	**9,12 €**

Tabla 104. Costes de software

[3] Teniendo en cuenta seis años de amortización

7.5 Costes de material fungible

En este apartado se exponen otros gastos asociados al material de oficina. Se exponen a continuación:

Gasto	Coste
Papel	4,10 €
Material de oficina (bolígrafos, tinta …)	57,95 €
Verbatim CD-R 700 MB (pack 25 unidades)	12,39 €
Otros gastos	75 €
Total coste material fungible	**149,44 €**

Tabla 105. Costes de material fungible

7.6 Presupuesto

El beneficio que se estima para este proyecto es bajo, ya que el objetivo del mismo es adquirir experiencia para desarrollos futuros. Por esto, se trabajará con un beneficio del 12 %. En cuanto al riesgo, se estima que también será bajo, ya que la aplicación está dirigida al Departamento de Investigación de Arquitectura de Computadores, Comunicaciones y Sistemas (ARCOS) [31] de la Escuela Politécnica Superior Universidad Carlos III de Madrid. Por tanto, se estima un riesgo del 6 %.

El presupuesto desglosado es el siguiente:

Concepto	Coste
Personal	23.109 €
Hardware	416,25 €
Software	9,12 €
Material fungible	149,44 €
Subtotal	23.683,81 €
Riesgo (6 %)	1.421,02 €
Beneficio (12 %)	2.842,05 €
Total	**27.946,88 €**

Tabla 106. Presupuesto total del proyecto

Por tanto, el presupuesto total para la realización de este proyecto asciende a **VEINTISIETE MIL NOVECIENTOS CUARENTA Y SEIS EUROS CON OCHENTA Y OCHO CÉNTIMOS** (I.V.A. incluido):

27.946,88 € (I.V.A. incluido)

8 Conclusiones y trabajos futuros

En este apartado se valoran las conclusiones que se han obtenido durante el desarrollo de este proyecto. Además, se analizan las posibles características que podrían aplicarse al sistema en un futuro.

8.1 Conclusiones

Este proyecto nació como una continuación del Proyecto Fin de Carrera de Gonzalo Martín Cruz *"Simulación de la transmisión de una enfermedad contagiosa en entornos urbanos"*, que realizó en el año 2010, y del que surgió una herramienta muy interesante: *EpiGraph*. La finalidad de este Trabajo Fin de Grado inicialmente era añadir la opción de analizar visualmente la simulación que la herramienta *EpiGraph* realiza sobre la propagación de enfermedades contagiosas. Aunque éste era el objetivo principal, a lo largo del desarrollo del proyecto se han añadido otras características que hacen que el sistema desarrollado pueda ser utilizado para otros fines igualmente válidos.

Se comenzó realizando un estudio sobre las herramientas de visualización de grafos actuales, así como de los tipos de archivo más comunes para la representación de los mismos. Debido a su validez, se decidieron utilizar estas herramientas para visualizar los grafos, poniendo como objetivo del TFG desarrollar un sistema que se encargase de adaptar grafos para que se pudieran observar los elementos claramente y configurar sus atributos.

Para visualizar los elementos de los grafos se analizó el problema de la escalabilidad visual, ya que si el grafo es muy grande es muy complicado ver nítidamente las relaciones existentes entre los nodos. Es por ello que se pensó que lo mejor era obtener muestras pequeñas que fueran representativas del grafo original. Así que se estudiaron varios algoritmos de muestreo buscando uno para implementarlo en el sistema, preocupándose principalmente de la calidad del muestreo y de la eficiencia del mismo.

Después, sólo quedaba decidir la tecnología que se iba a utilizar para el desarrollo del proyecto, valorando las necesidades del sistema y la experiencia previa que se tenía con los lenguajes y entornos de desarrollo posibles. Aunque en general ya existían conocimientos sobre el software que se iba a emplear, en algunas ocasiones ha sido necesario documentarse de manera autodidacta para poder continuar de manera correcta con el desarrollo del proyecto.

En cuanto a la metodología de trabajo utilizada, debido a que este TFG es reducido en comparación con el tamaño de otros proyectos de desarrollo de software, se ha intentado seguir el proceso que se emplea en proyectos reales, en los que se realiza una planificación, análisis del sistema de información, diseño del sistema y pruebas, aunque se han simplificado algunas etapas para adaptarse a la magnitud real de este TFG.

En esta memoria se incluyen numerosas pruebas, que son necesarias para comprobar el correcto funcionamiento del sistema, además de valorar si el rendimiento del mismo es aceptable. Aparte de esto, se ha creído conveniente añadir un caso de estudio en el que se realiza el proceso completo que ofrece la aplicación de: muestreo, generación del archivo *GraphML* y visualización de ese fichero en NodeXL con una configuración de atributos acorde a la simulación de una enfermedad contagiosa. De esta manera, se pretende que no se pierda la esencia de este TFG, que además de un proyecto de desarrollo de software tiene una parte de divulgación científica muy importante heredada de la aplicación *EpiGraph*.

Se han extraído una serie de conclusiones:

- Se ha conseguido modelar y desarrollar un sistema que permite el muestreo y generación de un archivo de visualización del grafo a partir de la representación de ese grafo en formato CSC. Por lo tanto se ha conseguido uno de los principales objetivos existentes.
- Se han conseguido visualizar los archivos generados por el sistema en distintas herramientas de visualización de grafos, pudiendo analizar los elementos del grafo de manera clara. Al lograr esto, se ha conseguido cumplir con el objetivo que se planteó inicialmente en las conversaciones con los tutores.
- Se ha conseguido obtener un rendimiento del sistema aceptable, cumpliendo con sus funciones de manera eficiente.
- Se ha conseguido realizar una herramienta útil para la aplicación *EpiGraph*, a la vez que se ha mantenido el sistema con opciones de ser una herramienta de manejo de grafos más general, pudiendo ampliarse sus características en futuras versiones.

En cuanto a la experiencia personal adquirida durante el desarrollo de este TFG se pueden exponer otras conclusiones:

- Se ha adaptado la metodología de desarrollo a seguir a la magnitud del proyecto, siguiendo un guión para poder cumplir de manera correcta y formal con los objetivos del sistema.
- Se ha hecho un estudio bastante amplio sobre la tecnología aplicada a grafos existente en la actualidad, observando que es un apartado en desarrollo y que tiene un gran potencial dentro del mundo de la informática. La representación de datos mediante grafos puede mostrar modelos que no se pueden apreciar de otra manera.
- Como ocurre en el día a día laboral, durante los proyectos surgen dificultades e imprevistos. El desarrollo de este TFG ha supuesto una manera de enfrentarse a estos problemas obligando a idear soluciones en tiempo real para no retrasar en exceso la construcción del sistema. Ha servido para adquirir una experiencia más real de cara al futuro.

8.2 Trabajos futuros

Tras el desarrollo de este TFG, se han valorado distintas líneas futuras de ampliación de esta herramienta. Se exponen a continuación:

- Se podrían ampliar los formatos de archivos de entrada y de salida válidos, de esta manera la herramienta adquiriría un matiz más general en el manejo de grafos.
- Otra posible ampliación sería añadir otros algoritmos de muestreo, dando la posibilidad al usuario de elegir el que crea más conveniente para sus fines.
- Una mejora sería aumentar el rendimiento de la aplicación en los procesos de generación de archivos. Ahora mismo el sistema ofrece una eficiencia aceptable, pero quizá en un futuro sea necesaria una mejora en este aspecto.
- Aunque no afectaría al rendimiento de la aplicación, una posible mejora sería favorecer la interfaz del sistema, haciéndola más atractiva e intuitiva. Éste no era el objetivo de este TFG, por lo que la interfaz diseñada se ha pretendido que sea intuitiva pero no se ha profundizado demasiado en ello.
- Debido a que las herramientas de visualización existentes son algo limitadas, aunque están en continuo desarrollo, se podría valorar la opción de ampliar esta aplicación dotándola de un sistema de visualización de grafos propio. Esta característica quizá es la más complicada de todas las que se exponen en este apartado, pero este campo no está liderado todavía por ninguna aplicación, por lo que la aparición de una buena herramienta software para este fin podría suponer un gran salto cualitativo.
- Otra posible mejora es rediseñar el apartado de configuración de los atributos, añadiendo más opciones al usuario y facilitándole la navegación en este subsistema de la aplicación.

9 Glosario de términos y referencias

A continuación, se resumen los acrónimos, tecnicismos y referencias que se han utilizado a lo largo de este proyecto.

9.1 Acrónimos

API Application Programming Interface (Interfaz de Programación de Aplicaciones)

ARCOS grupo de Investigación de Arquitectura de Computadores, Comunicaciones y Sistemas de la Universidad Carlos III de Madrid

BFS Breadth-First Sampling (muestreo por anchura)

CSC Compressed Sparse Column (matriz dispersa comprimida por columnas)

CSV Comma-Separated Values (valores separados por comas)

FIFO First In First Out (el primero en entrar es el primero en salir)

GCC GNU Compiler Collection (Colección de Compiladores GNU)

GEXF Graph Exchange XML Format (formato XML de intercambio de grafos)

GML Graph Modeling Language (lenguaje de modelado de grafos)

GPL General Public License (Licencia Pública General)

GSL GNU Scientific Library (Librería científica GNU)

IDE Integrated Development Environment (Entorno de Desarrollo Integrado)

IVA Impuesto sobre el Valor Añadido

JDK Java Development Kit (Equipo de Desarrollo de Java)

JRE Java Runtime Environment (Entorno de ejecución de Java)

JVM Java Virtual Machine (Máquina Virtual de Java)

MHRW Metropolis-Hasting Random Walk

PFC Proyecto de Fin de Carrera

RJ Random Jump (muestreo de salto aleatorio)

RUC Requisito de Usuario de Capacidad

RUR Requisito de Usuario de Restricción

SAX Simple API for XML (API Simple para XML)

TFG Trabajo de Fin de Grado

UC3M Universidad Carlos III de Madrid

XML Extensible Markup Language (lenguaje de marcas extensible)

9.2 Definiciones

- **Bytecode:** archivo binario que contiene un programa ejecutable generado por un compilador, su contenido es el código objeto o código máquina.
- **Conexo:** grafo en el que existe un camino entre dos vértices cualesquiera.
- **Drag and drop:** expresión utilizada en el mundo de la informática que se refiere a la acción de mover objetos con el ratón de un sitio a otro, arrastrando y soltando.
- **Grado:** en un grafo, el número de vértices adyacentes a ese nodo, es decir, el número de vértices con los que está conectado mediante una arista.
- **Islas:** conjuntos de vértices conectados entre sí pero que no tienen ninguna conexión con el resto de los nodos del grafo. Estas islas provocan que el grafo no sea conexo.
- **Parser:** analizador sintáctico que permite el análisis de la jerarquía de un archivo de entrada, construyendo una estructura de datos que será manejable por un software determinado.
- **Plug-in:** módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.
- **Tooltip:** palabra inglesa que se refiere al texto que aparece en la pantalla cuando se pasa el puntero del ratón por encima de un elemento existente en la misma.
- **Win32:** API de Microsoft que permite ejecutar aplicaciones creadas o basadas en sistemas operativos Microsoft Windows.

9.3 Referencias

[1] **Universidad Carlos III de Madrid.** Universidad pública emplazada en la Comunidad de Madrid, que cuenta con tres campus (Leganés, Getafe y Colmenarejo). Página web: http://www.uc3m.es.

[2] Gonzalo Martín, Maria-Cristina Marinescu, David E. Singh. Proyecto Fin de Carrera **"Simulación de la transmisión de una enfermedad contagiosa en entornos urbanos"**. Leganés, Madrid, Julio 2010.

[3] Gonzalo Martín, Maria-Cristina Marinescu, David E. Singh y Jesús Carretero. **"EpiGraph: A Scalable Simulation Tool for Epidemiological Studies"**, pp. 59-536. The 2011 International Conference on Bioinformatics and Computational Biology. Las Vegas, EEUU, Julio 2011.

[4] **Matlab.** Software matemático que permite la manipulación de matrices, representación de datos y funciones, etc, muy usado en universidades y centros de investigación. Página web en español: http://www.mathworks.es/.

[5] **NodeXL.** Complemento para Microsoft Excel 2007 y 2010 para el análisis, gestión y visualización de grafos. Página web: http://nodexl.codeplex.com/.

[6] **Gephi.** Aplicación gratuita y de código abierto que permite analizar, gestionar y visualizar grafos. Página web: http://gephi.org/.

[7] **R Project.** Proyecto GNU que dispone de entorno y lenguaje propio, destinado a la computación estadística y gráfica. Página web: http://www.r-project.org/.

[8] **Ucinet.** Software dedicado al análisis de redes sociales. Página web: www.analytictech.com/ucinet.

[9] **Pajek.** Aplicación para el análisis y visualización de grafos. Página web: http://pajek.imfm.si/doku.php.

[10] **GraphML.** Formato de archivo basado en XML para la representación de grafos, que pretende ser un estándar en este área. Página web: http://graphml.graphdrawing.org/.

[11] Long Jin, Yang Chen, Pan Hui, Cong Ding, Tianyi Wang, Athanasios Vasilakos, Beixing Deng, Xing Li, **"Albatross Sampling: Robust and Effective Hybrid Vertex Sampling for Social Graphs"**, en Proc. of the 3rd ACM Workshop on Hot Topics in Planet-scale Measurement (HotPlanet'11), co-located with ACM MobiSys 2011, Junio 2011.

[12] **Metodología MÉTRICA Versión 3.** Metodología MÉTRICA Versión 3: metodología de planificación, desarrollo y mantenimiento de sistemas de información. Página web de Métrica v.3 del Gobierno de España: http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P60085901274201580632&langPae=es.

[13] **Ministerio de Hacienda y Administraciones Públicas de España.** Ministerio encargado de llevar a cabo las políticas del Gobierno de España en lo referente a las entidades públicas, presupuestos y gastos. Página web: http://www.seap.minhap.gob.es.

[14] **Especificaciones y limitaciones de Excel 2007.** Restricciones que ofrecen las hojas de cálculo de Microsoft Excel 2007. Página web: http://office.microsoft.com/es-es/excel-help/especificaciones-y-limites-de-excel-HP010073849.aspx.

[15] **Java.** Lenguaje de programación orientado a objetos creado por Sun Microsystems (Oracle Corporation) en el año 1995. Página web: http://www.java.com/.

[16] **GNU GPL.** Licencia de software libre, orientada a proteger la libre distribución, modificación y uso de software. Página web: http://www.gnu.org/.

[17] **.NET.** Framework de Microsoft que es una integración de todas las herramientas necesarias para el desarrollo de aplicaciones. Página web: http://www.microsoft.com/NET.

[18] **Microsoft Windows.** Familia de sistemas operativos muy populares desarrollados por Microsoft desde 1981. Página web en español: http://windows.microsoft.com/es-ES/windows/home.

[19] **GNU/Linux.** Familia de sistemas operativos basados en un núcleo Linux con herramientas del proyecto GNU. Página web: http://www.linux.org/.

[20] **Ubuntu.** Distribución de GNU/Linux libre y de código abierto. Página web: http://www.ubuntu.com/.

[21] **GCC.** Colección de compiladores GNU, incluidos como herramienta en los sistemas operativos GNU/Linux. Página web: http://gcc.gnu.org/.

[22] **NetBeans.** Entorno de desarrollo integrado libre multiplataforma desarrollado por Sun Microsystems (Oracle Corporation) principalmente orientado a la programación en lenguaje Java. Página web: http://netbeans.org/.

[23] **Eclipse.** Entorno de desarrollo integrado de código abierto multiplataforma. Fue creado originalmente por IBM pero ahora es desarrollado por la Fundación Eclipse. Página web: http://www.eclipse.org/.

[24] **Notepad++.** Editor de texto que soporta una amplia variedad de lenguajes de programación. Se distribuye bajo los términos de Licencia Pública General de GNU, por lo que es una aplicación libre y gratuita. Sólo está disponible para Microsoft Windows. Página web: http://notepad-plus-plus.org/.

[25] **Emacs.** Editor de texto muy potente parte del proyecto GNU. Página web: http://www.gnu.org/software/emacs/.

[26] **Gedit.** Editor de texto de propósito general disponible para GNU/Linux, Mac OS X, y Microsoft Windows. Destaca por su simplicidad y facilidad de uso. Página web: http://projects.gnome.org/gedit/.

[27] **MinGW.** Minimalist GNU for Windows, es una implementación de los compiladores GCC para la plataforma Win32. Página web: http://www.mingw.org/.

[28] **Cygwin.** Colección de herramientas destinadas a proporcionar un comportamiento similar a los sistemas Unix en Microsoft Windows. Página web: http://www.cygwin.com/.

[29] **Microsoft Office.** Suite ofimática desarrollada por Microsoft con varias herramientas de oficina muy populares en la actualidad. Página web en español: http://office.microsoft.com/es-es/.

[30] **Biblioteca MSDN.** Del inglés *Microsoft Development Network*. Librería que permite la descarga online de software de Microsoft previo registro y autorización. Página web de las herramientas disponibles para la UC3M: http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=8738733a-6d9b-e011-969d-0030487d8897&vsro=8.

[31] **ARCOS.** Grupo de Investigación de Arquitectura de Computadores, Comunicaciones y Sistemas de la Universidad Carlos III de Madrid. Página web: http://www.arcos.inf.uc3m.es.

[32] José Ramón Rodríguez Bermúdez, **"Gestión de proyectos informáticos: métodos, herramientas y casos"**. UOC (Universitat Oberta de Catalunya), 2007.

[33] **COCOMO II.** Constructive Cost Model (Modelo Constructivo de Costes). Modelo matemático de base empírica que se emplea para la estimación de costes, esfuerzo y permite realizar planificaciones. Página web: http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html.

[34] **GSL.** Librería numérica para C y C++ que proporciona rutinas matemáticas para generar números aleatorios. Es software libre bajo licencia GNU GPL. Página web: http://www.gnu.org/software/gsl/.

[35] **CSParse.** Librería para C que proporciona métodos para manejar matrices dispersas. Página web: http://www.cise.ufl.edu/research/sparse/CSparse/.

[36] **SAX.** API para usar XML en Java. Página web: http://www.saxproject.org/.

[37] **Real Decreto 1777/2004,** de 30 de julio, por el que se aprueba el Reglamento del Impuesto sobre Sociedades. *Anexo: Tabla de Coeficientes de Amortización*. Página web: http://www.boe.es/boe/dias/2004/08/06/pdfs/A28377-28429.pdf.

Anexo A: Manual de usuario

Contenido

El manual de usuario contiene toda la información relevante para poder ejecutar la aplicación. Se añade una guía rápida con las principales acciones que puede realizar el usuario, añadiendo capturas de pantalla del sistema para facilitar la compresión de las funciones disponibles.

Requisitos hardware y software

Los requisitos de hardware adjuntos en este apartado son los necesarios para poder obtener ejecuciones en un tiempo mínimo, teniendo en cuenta las pruebas que se han realizado del sistema. Pero en realidad el rendimiento de la aplicación depende del tamaño de los grafos que se manejen, por lo que si se dispone de un equipo con menos recursos se podrá utilizar la aplicación normalmente pero el tiempo de ejecución quizá sea elevado.

- Procesador: Intel Core i-5 2,3 GHz o AMD Phenom II X4
- Memoria RAM: 4 GB.

En cuanto al software, si se quiere utilizar el programa de muestreo por separado, para la compilación es necesario disponer de las siguientes librerías:

- Librería GSL [34].
- Librería CSParse [35].

En lo relativo al software necesario para la ejecución de la aplicación global, es necesario disponer de:

- JRE [15], recomendada la versión jre6 o posteriores.
- Una aplicación de visualización de grafos que soporte archivos de tipo *GraphML* para poder visualizar los archivos generados por el sistema. Se recomienda *NodeXL* para poder configurar los atributos de visualización que ofrece el sistema.

Funcionalidades de la aplicación

El sistema ofrece tres funcionalidades principales, que se corresponden con las tres pestañas del menú del mismo:

- Generación del archivo de visualización de grafos.
- Generación de los archivos de representación del grafo.
- Configuración de los atributos del grafo.

A continuación, se expone de manera básica el funcionamiento de cada opción:

➢ Generación del archivo de visualización de grafos

El aspecto de este apartado de la aplicación es el siguiente:

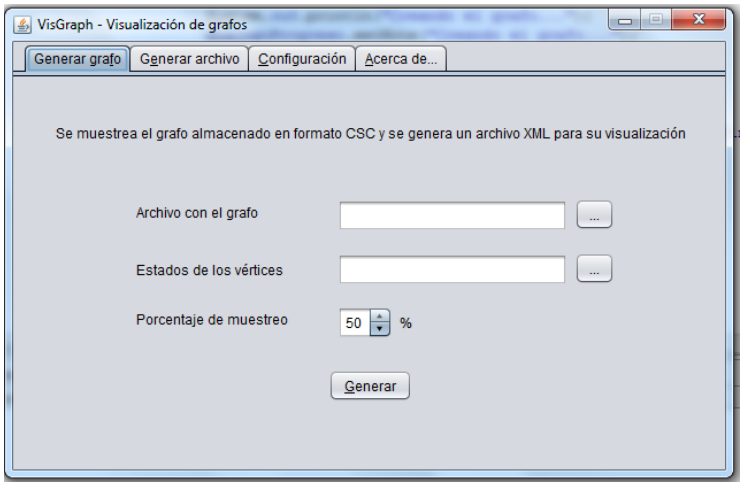

Ilustración 58. Pestaña "Generar grafo"

Para poder generar un grafo de tipo *GraphML* hay que disponer de un archivo que represente el grafo en formato CSC y otro archivo con el estado de sus vértices con el formato válido del sistema. La extensión de estos archivos debe ser *".dat"*. El proceso a seguir es el siguiente:

1. Se introduce la ruta de los dos archivos necesarios
2. Se elige el porcentaje de muestreo a aplicar
3. Se selecciona el botón "generar"

En ese momento comenzará el proceso de generación del archivo *GraphML*, mostrándose una barra de progreso que indique al usuario la etapa del proceso que se está realizando en ese momento: muestreo, leer el archivo de muestra, crear el grafo y generar el archivo. El aspecto de esta pantalla es la siguiente:

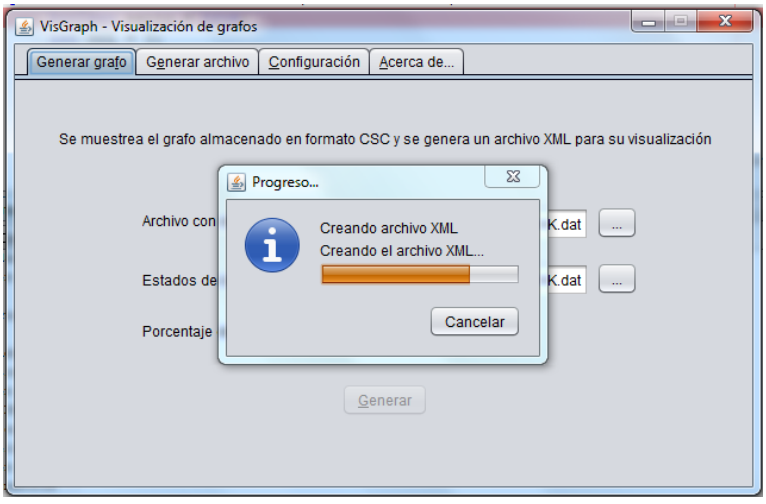

Ilustración 59. Progreso del proceso de generar el archivo GraphML

Si ocurre algún problema con la validez de algún dato de entrada o durante el proceso, el usuario será alertado con un mensaje de error. Tras el proceso de generación del archivo *GraphML* se abre la plantilla NodeXL en Excel, para permitir la importación del archivo recién creado[4].

➢ Generación de los archivos de representación del grafo

El aspecto de esta pestaña de la aplicación es el siguiente:

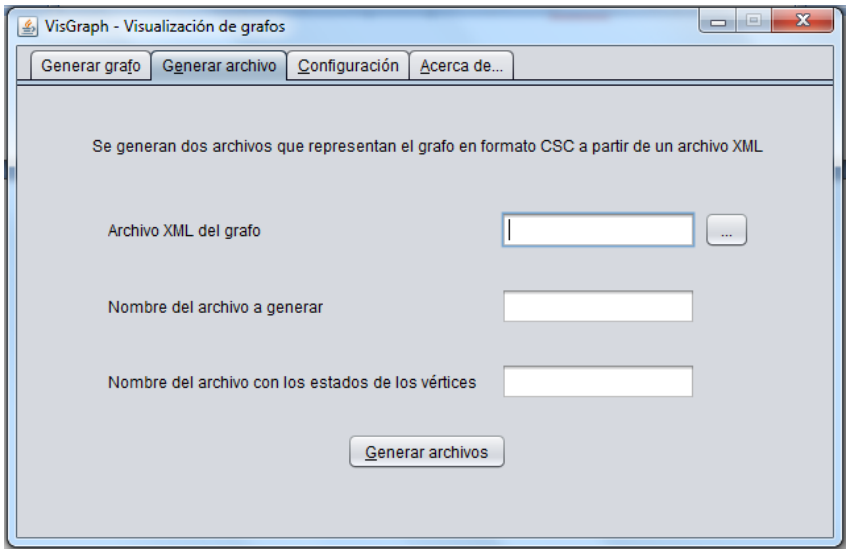

Ilustración 60. Pestaña "Generar archivo"

Este apartado permite generar a partir de un archivo *GraphML*, un archivo con formato CSC que represente mediante una matriz dispersa el grafo y otro fichero con los estados de los vértices. Es el proceso inverso al explicado en el apartado anterior. El proceso a seguir es el siguiente:

1. Se introduce la ruta del archivo *GraphML*
2. Se introducen los nombres que se quieren asignar a los archivos que se generen
3. Se selecciona el botón “generar archivos”

El usuario estará informado del estado del proceso mediante una barra de progreso similar a la expuesta en la Ilustración 59.

[4] El archivo de la plantilla de NodeXL (“NodeXLGraph.xltx”) debe encontrarse en la ruta de la aplicación, sino se abrirá Microsoft Excel para que se acceda desde ahí a la plantilla.

➢ Configuración de los atributos del grafo

El aspecto de este apartado del sistema es el siguiente:

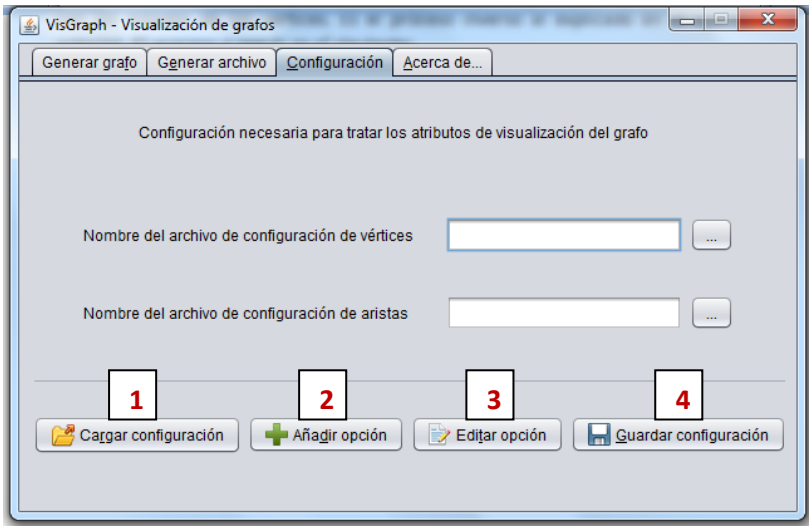

Ilustración 61. Pestaña "Configuración"

En esta pestaña se puede realizar una configuración de los atributos de los grafos o se puede cargar de una previa. A continuación, se expone el proceso a seguir para realizar cada posible acción:

- Cargar configuración anterior desde archivo

Para cargar una configuración de los atributos previa es necesario disponer de los archivos con el formato adecuado (véase Anexo G), indicar su ruta en los campos indicados para ello y seleccionar el botón “cargar configuración” (1).

- Realizar una configuración de manera interactiva

Para crear una configuración utilizando la aplicación es necesario añadir opciones de configuración pulsando en el botón “añadir opción” (2). Aparecerá una ventana emergente como la de la Ilustración 62, en la que se podrán indicar los valores para cada opción que se cree. En la parte superior se puede indicar si la opción que se quiere añadir es para los vértices o para las aristas, según se indique esto los atributos a configurar variarán.

Hay que tener en cuenta que no es obligatorio añadir valores a los atributos para crear una opción, ya que si se crea una opción de configuración vacía la herramienta de visualización usará los valores que tenga asignados por defecto para representar los elementos que se correspondan con esa opción.

Es importante señalar que para que se pueda guardar una configuración debe tener añadida una opción por defecto tanto para los vértices como para las aristas, sino el sistema no permitirá que se utilice esa configuración.

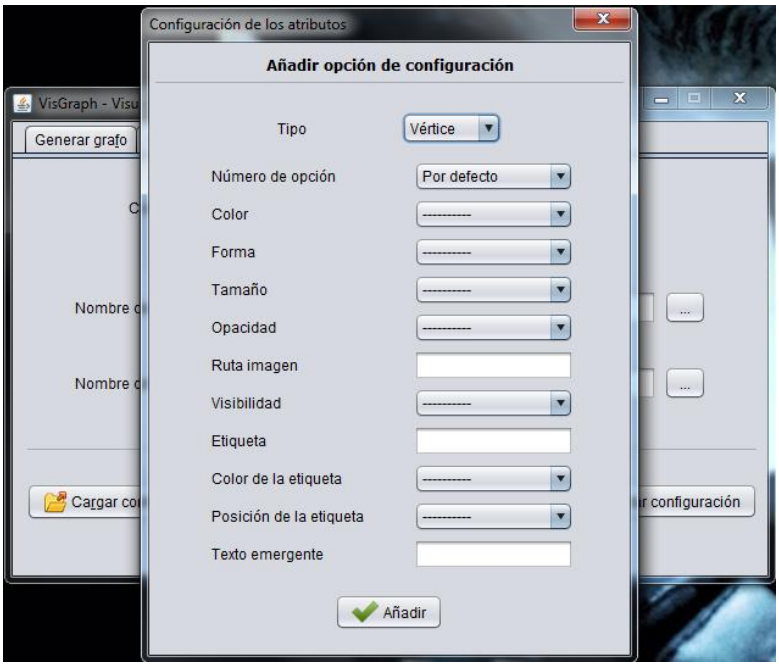

Ilustración 62. Ventana para añadir opción de configuración

Si se quieren modificar los valores o eliminar alguna opción de configuración, se debe seleccionar el botón "editar opción" (3). La pantalla que aparecerá (Ilustración 63) permitirá modificar los valores de las opciones ya creadas, pudiendo navegar por las mismas para ver sus valores actuales.

Para eliminar una opción es necesario tenerla cargada en la ventana de "editar configuración" y seleccionar el botón inferior "eliminar". El sistema pedirá la confirmación de la acción debido a la irreversibilidad de la acción.

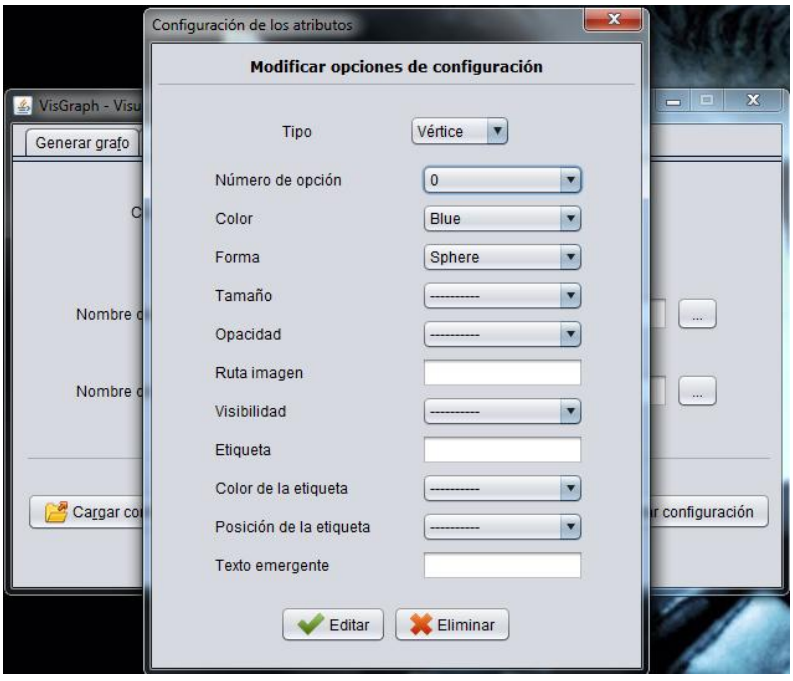

Ilustración 63. Ventana para editar o eliminar una opción de configuración

Para almacenar la configuración creada en archivos de configuración válidos para la aplicación es necesario introducir un nombre para asignarlo a los ficheros de configuración y seleccionar el botón “guardar configuración” (4).

También es posible cargar una configuración previa y modificarla tal y como se ha explicado en los párrafos anteriores.

Si se quiere obtener más información sobre la aplicación y sus autores del proyecto se puede acceder en la pestaña “Acerca de…”.

NOTA: Todos los archivos que se generen durante la ejecución de la aplicación se añadirán en la carpeta donde se encuentren los ficheros que se utilicen como datos de entrada.

Anexo B: Manual básico de NodeXL

Este anexo es una breve descripción de las acciones necesarias para visualizar grafos en NodeXL, para una explicación más detallada véase la documentación de su página web [5].

NodeXL es una plantilla disponible para Excel 20007 y 2010, por ello, los datos del grafo referentes a vértices y aristas se cargan en la hoja de cálculo, por lo que es bastante intuitivo ya que la mayoría de nosotros estamos habituados a trabajar con el paquete ofimático de Microsoft. Además de esto, NodeXL tiene la opción de obtener una serie de métricas sobre el grafo, y permite visualizar el grafo dando numerosas opciones al usuario de manera interactiva para poder analizar los vértices y las aristas entre ellos.

En primer lugar, el aspecto inicial que ofrece NodeXL al cargar su plantilla en Excel es el siguiente:

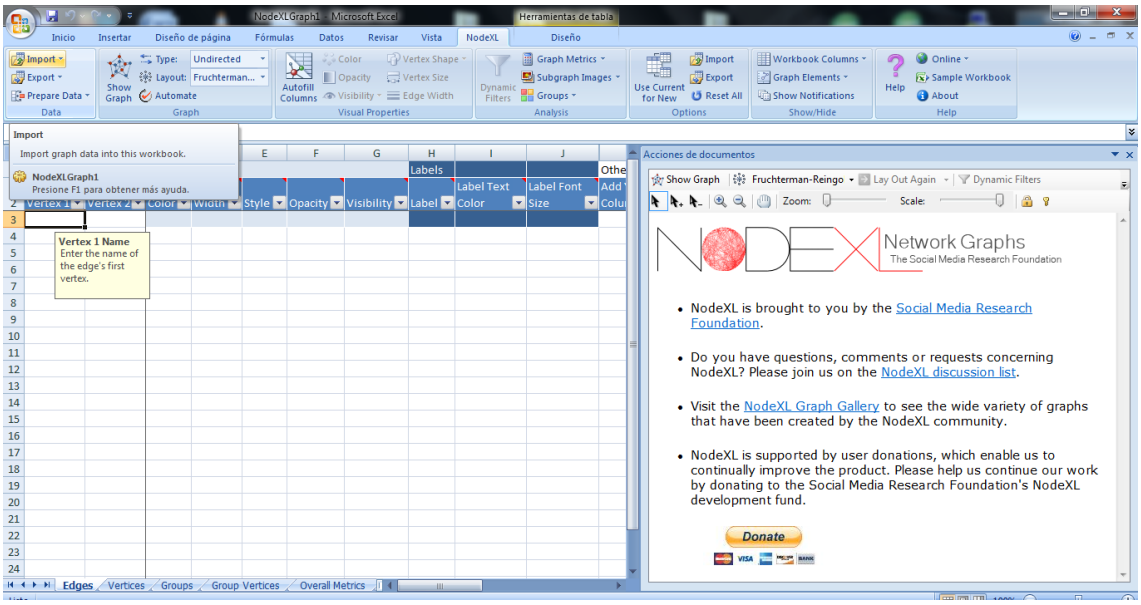

Ilustración 64. Imagen inicial de NodeXL

NodeXL permite la carga de varios tipos de archivo, tal y como se indica en el apartado 2.1 de este documento, pero el archivo que se emplea en este proyecto es de tipo *GraphML*. Así que para cargar un archivo de este tipo es necesario seleccionar el botón de "importar", tal y como se indica en la siguiente imagen:

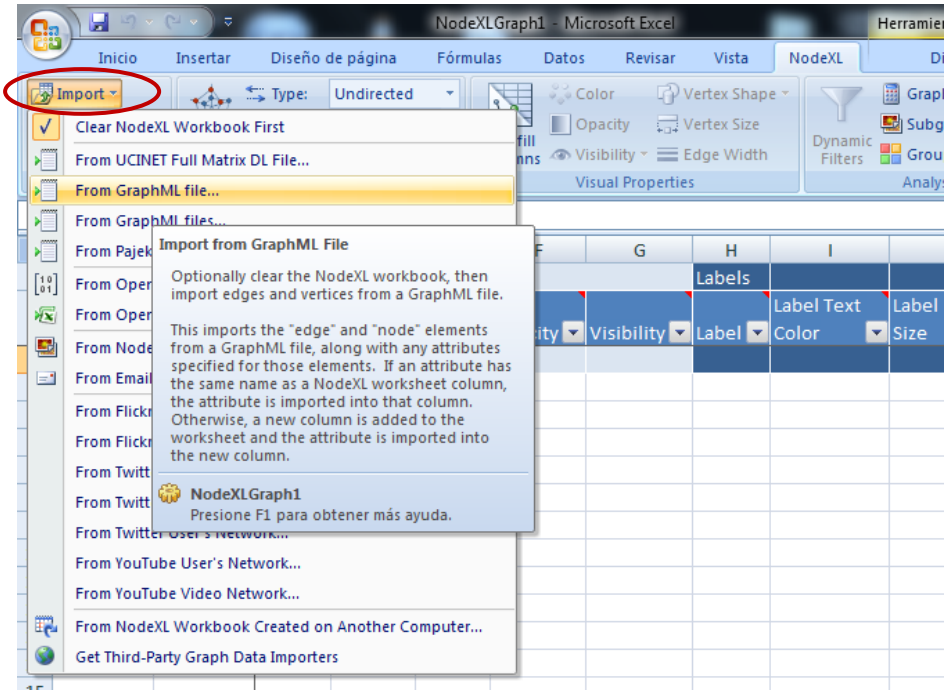

Ilustración 65. Cargar archivo GraphML en NodeXL

Una vez cargado el archivo, los datos aparecerán en la hoja de cálculo. En la parte inferior, se podrá navegar entre las distintas hojas para cada tipo de dato:

- Datos de los vértices.
- Datos de las aristas.
- Grupos de vértices.
- Métricas del grafo.

En la siguiente imagen se señala este hecho:

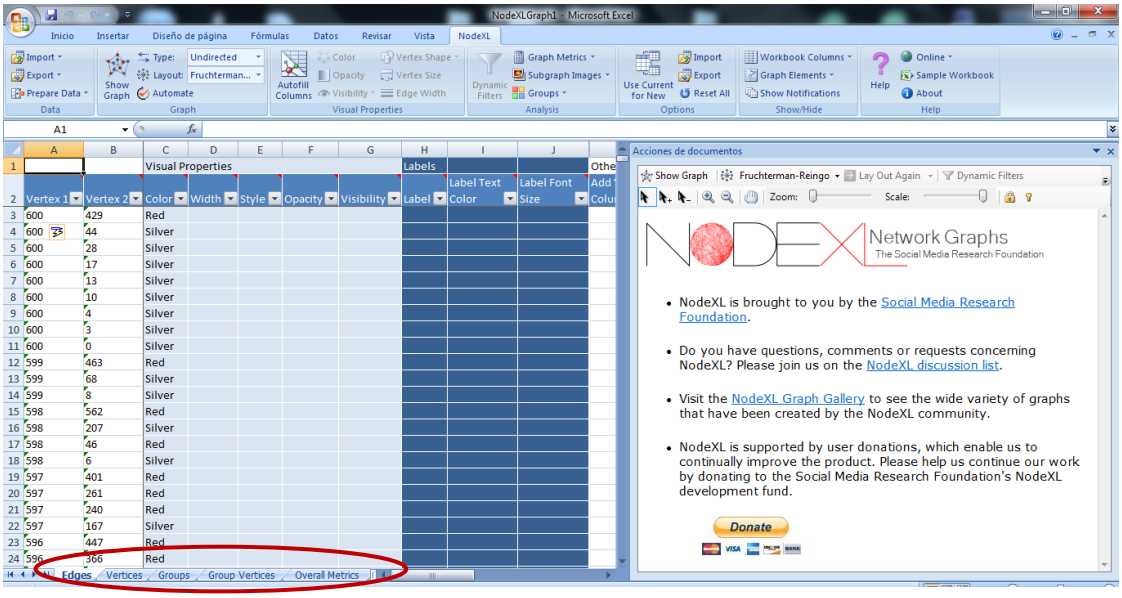

Ilustración 66. Carga de datos en NodeXL

Las opciones básicas que ofrece NodeXL son:

- Observar los datos de los vértices, aristas y grupos pudiendo modificarlos
- Calcular las métricas del grafo
- Visualizar el grafo

Para editar los datos basta con desplazarse por las hojas de cálculo y modificar los valores tal y como se haría en cualquier proyecto de Microsoft Excel.

Para calcular las propiedades del grafo es necesario seleccionar la opción "graph metrics" y en la ventana que se despliegue seleccionar qué métricas quieres calcular. En la siguiente ilustración se muestra este proceso:

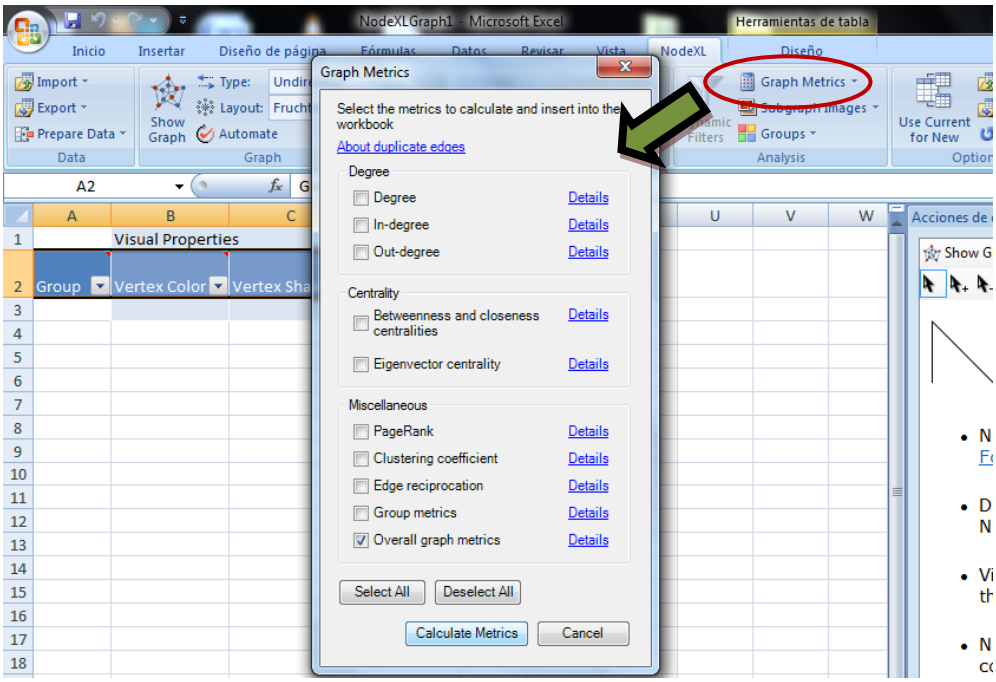

Ilustración 67. Calcular métricas en NodeXL

Finalmente, para visualizar los grafos hay que señalar la opción "Show graph" existente en el panel de visualización de la derecha, y se cargara el grafo en ese espacio.

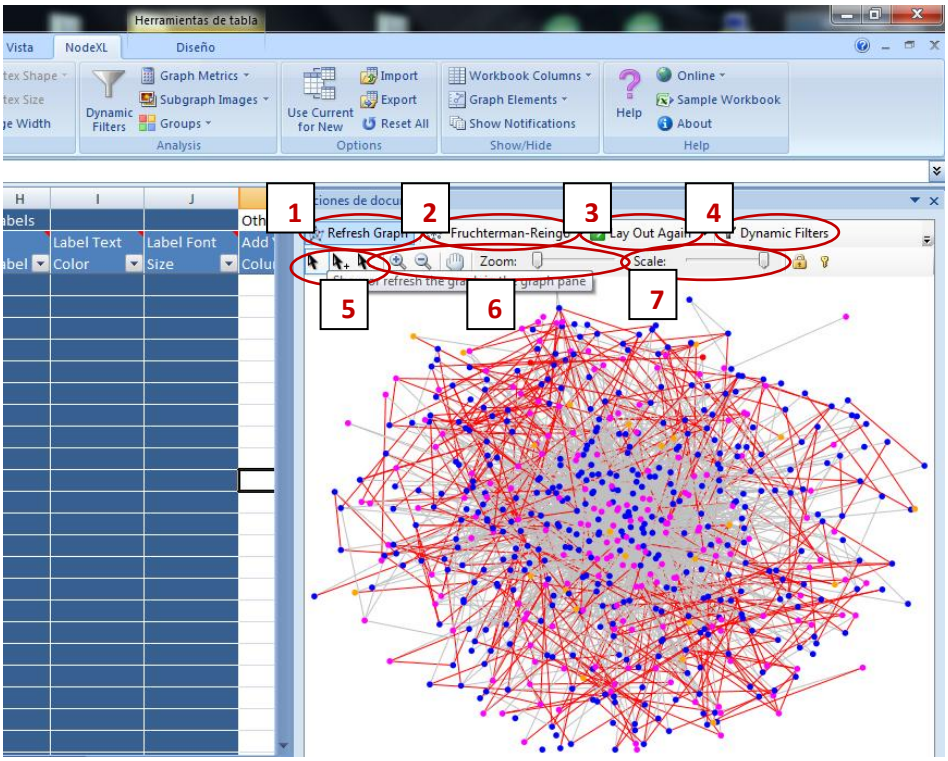

Ilustración 68. Visualización de grafos en NodeXL

Las opciones principales de visualización que ofrece NodeXL al usuario son las siguientes (véase la Ilustración 68):

1. Refrescar el grafo si se cambia alguna opción de visualización o algún valor de los vértices o aristas.
2. Cambiar el algoritmo utilizado para mostrar el grafo. Por defecto se visualiza según el *Fruchterman-Reingo*.
3. Cambiar la posición de los vértices en pantalla.
4. Aplicar filtros a las aristas para seleccionar sólo las que se encuentren entre los valores que te interesen.
5. Seleccionar vértices para ver sus aristas resaltadas en pantalla.
6. Hacer zoom sobre la imagen.
7. Cambiar la escala haciendo que las aristas y nodos tengan menos grosor, esto permite ver los elementos menos apilados.

Anexo C: Manual básico de Gephi

En este anexo se presentan algunas características básicas de Gephi para visualizar grafos, pero si se necesita un manual más avanzado se puede encontrar en su sitio web [6].

Gephi ofrece las mismas características que NodeXL, con la diferencia de que ofrece más potencia en algunos apartados, ya que es una herramienta completa desarrollada específicamente para grafos. Tiene la opción de ver y editar los datos de los grafos, obtener las propiedades de los mismos y visualizarlos en pantalla.

La pantalla de inicio de la aplicación es la siguiente:

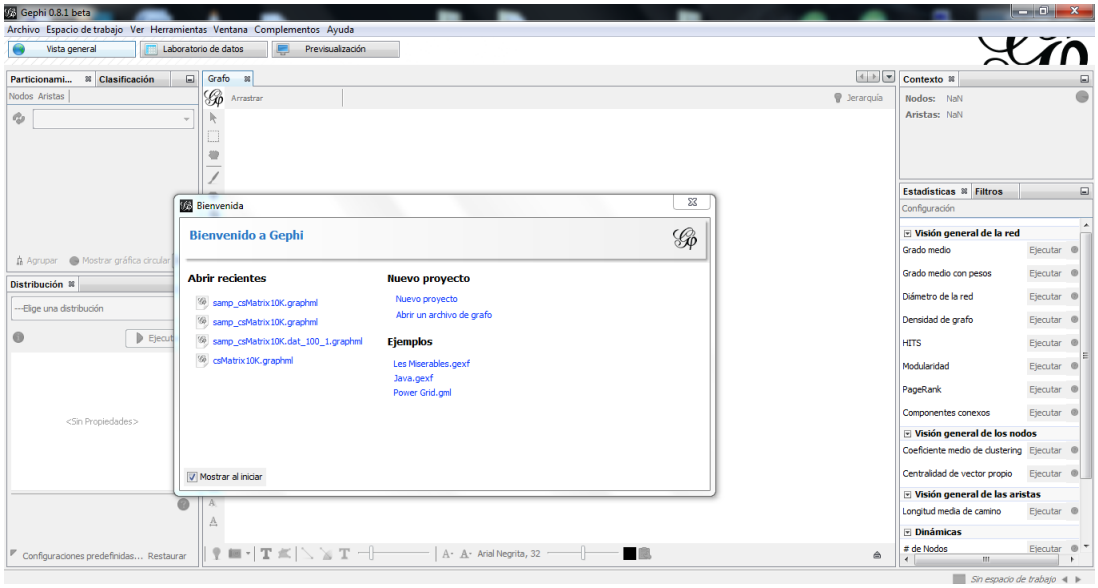

Ilustración 69. Pantalla de inicio de Gephi

Gephi soporta varios tipos de archivos, tal y como se indicó en el apartado 2.1 de este documento. Para poder cargar un grafo se debe ir al menú “archivo” en la parte superior izquierda y seleccionar “abrir”.

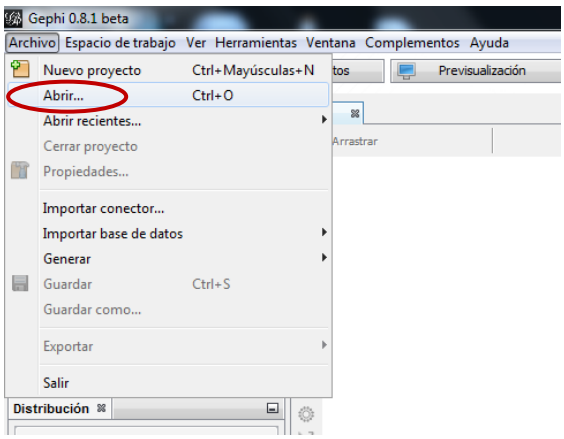

Ilustración 70. Cargar grafo en Gephi

Una vez cargado el grafo se visualizará directamente en la pantalla, tal y como se aprecia en la siguiente imagen:

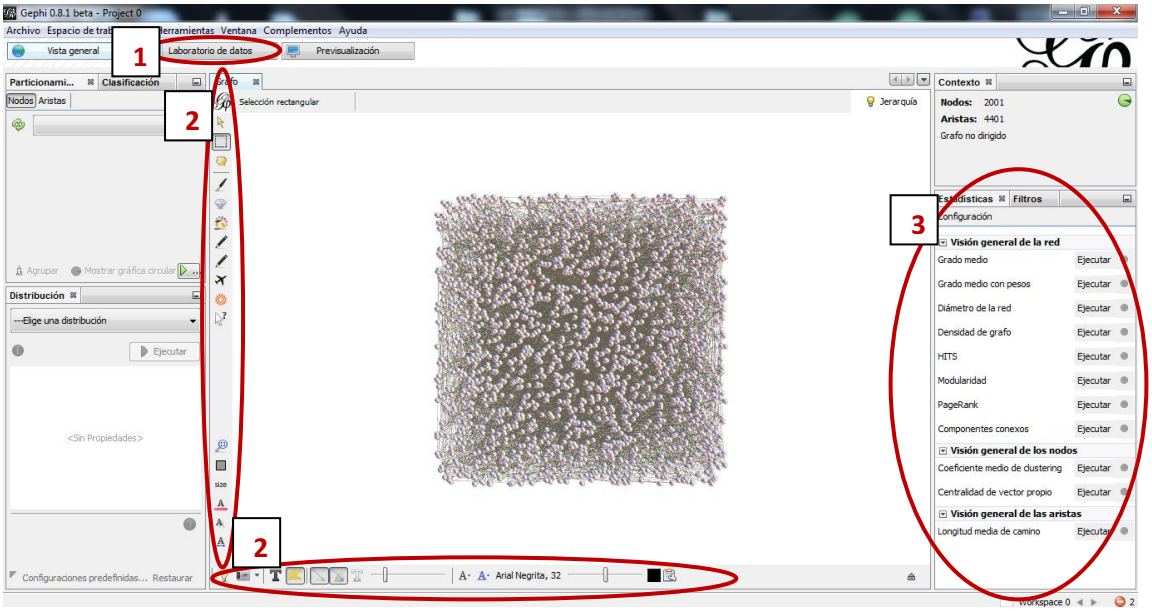

Ilustración 71. Visualización de grafo en Gephi

Las principales acciones que ofrece Gephi al usuario a partir de aquí son las siguientes:

1. Ver y editar los datos del grafo en el “laboratorio de datos”.
2. Modificar de manera interactiva sobre la visualización del grafo existente numerosas opciones del grafo.
3. Calcular las métricas del grafo.

Anexo D: Prototipo de la interfaz

El prototipo inicial de la interfaz presenta las siguientes pantallas:

- Pestaña principal, generación del archivo de visualización del grafo:

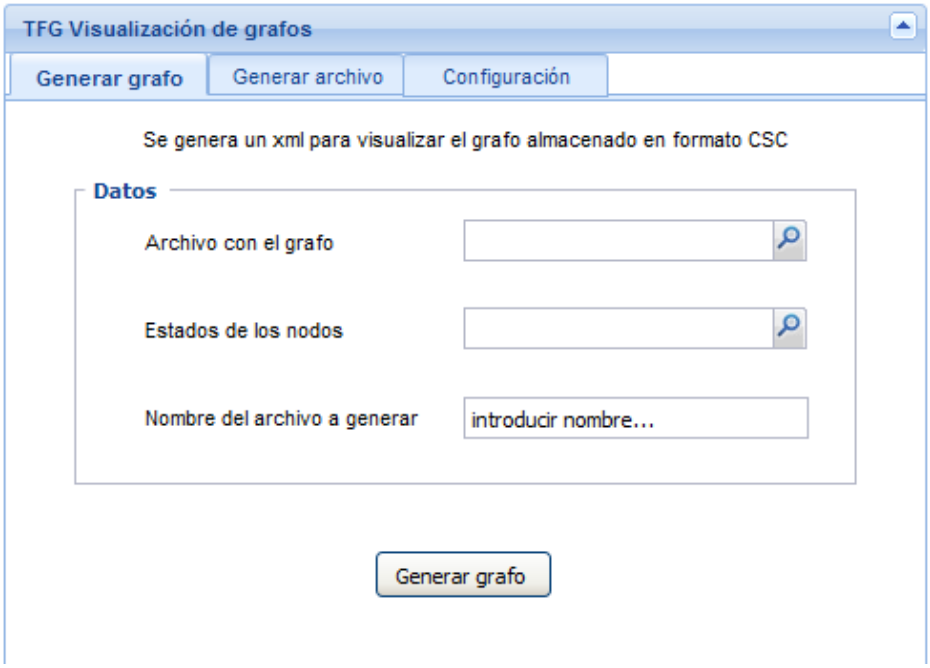

Ilustración 72. Prototipo primera pestaña de la interfaz

- Segunda pestaña, generación del archivo de representación del grafo:

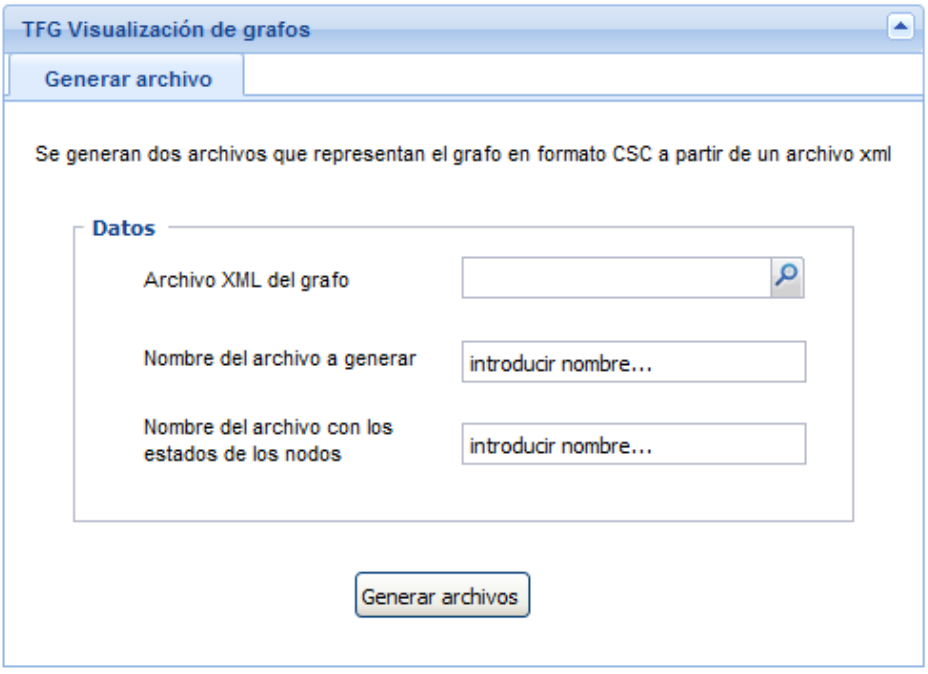

Ilustración 73. Prototipo segunda pestaña de la interfaz

- Tercera pestaña, configuración de los atributos del grafo:

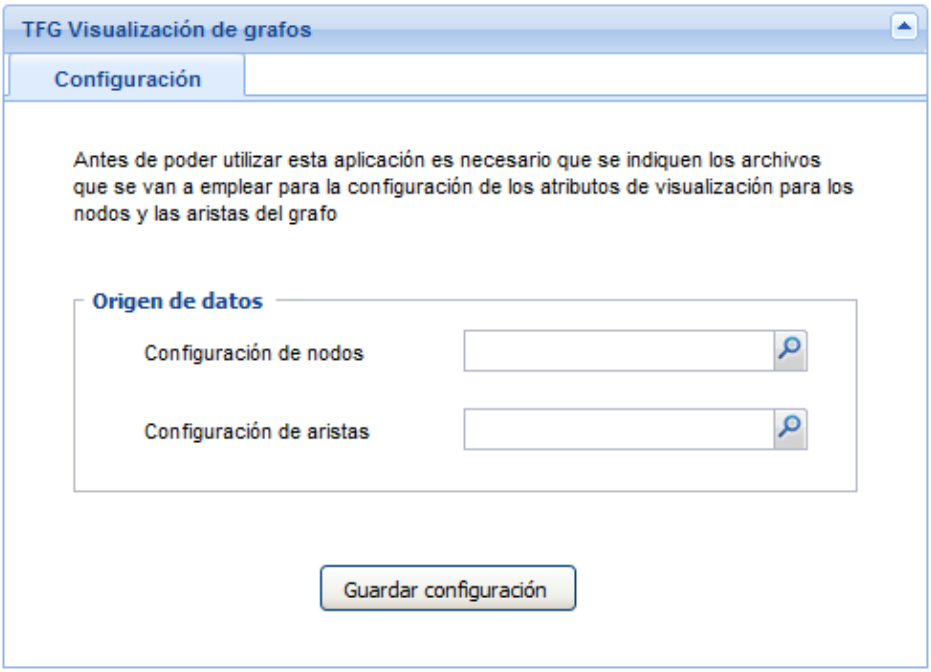

Ilustración 74. Prototipo tercera pestaña de la interfaz

Anexo E: Formato de archivo CSC

Los archivos de entrada de representación del grafo deben definir el grafo mediante una matriz dispersa con esquema CSC. El formato de este tipo de archivos es el siguiente:

- Primera línea: está compuesta por cuatro valores.
  - Número de filas de la matriz (número de vértices del grafo).
  - Número de columnas de la matriz (número de vértices del grafo).
  - Número de elementos no nulos de la matriz (Número de aristas).
  - El valor 1, no tiene relevancia para el funcionamiento del programa.
- Segunda línea: vector JA, de dimensión el número de columnas +1. La estructura es la siguiente:
  - JA (1) = 1
  - JA (j + 1) - JA (j) = número de elementos no nulos de la columna j
- Tercera línea: vector IA. De dimensión el número de elementos no nulos, contiene los números de las filas de los elementos de la matriz.
- Cuarta línea: vector AA. De dimensión el número de elementos no nulos con los valores de las aristas.

A continuación, se puede observar un ejemplo de un grafo de 10 vértices con este formato:

```
1  10 10 28
2  1 4 7 10 12 14 16 21 23 25 29
3  3 5 10 5 7 9 1 4 7 3 7 1 2 9 10 2 3 4 8 10 7 10 2 6 1 6 7 8
4  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Ilustración 75. Ejemplo formato de archivo de representación de grafo válido

La matriz de adyacencia del grafo es la siguiente:

0	0	1	0	1	0	0	0	0	1
0	0	0	0	1	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0
1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1
0	1	1	1	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	1
0	1	0	0	0	1	0	0	0	0
1	0	0	0	0	1	1	1	0	0

Anexo F: Formato de archivo GraphML

El archivo de visualización del grafo sigue una sintaxis XML, y tiene el siguiente formato:

<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
[Definición de atributos del grafo]
<graph edgedefault="undirected">
[Definición de los nodos con los valores de sus atributos]
[Definición de las aristas con los valores de sus atributos]
</graph>
</graphml>

A continuación, se muestra un ejemplo de un archivo de este tipo:

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
    <key id="V-Color" for="node" attr.name="Color" attr.type="string"/>
    <key id="V-Shape" for="node" attr.name="Shape" attr.type="string"/>
    <key id="E-Width" for="edge" attr.name="Width" attr.type="string"/>
    <graph edgedefault="undirected">
    <node id="0">
        <data key="V-Color">Green</data>
        <data key="V-Shape">Circle</data>
    </node>
    <node id="1">
        <data key="V-Color">Green</data>
        <data key="V-Shape">Circle</data>
    </node>
    <node id="2">
        <data key="V-Color">Black</data>
        <data key="V-Shape">Triangle</data>
    </node>
    <edge source="0" target="1">
        <data key="E-Color">Red</data>
        <data key="E-Width">1</data>
    </edge>
    <edge source="0" target="2">
        <data key="E-Color">Orange</data>
        <data key="E-Width">1</data>
    </edge>
    <edge source="1" target="2">
        <data key="E-Color">Orange</data>
        <data key="E-Width">1</data>
    </edge>
    </graph>
</graphml>
```

Ilustración 76. Ejemplo de archivo con formato GraphML

Anexo G: Formato de archivo de configuración

Los archivos de configuración tienen un formato similar tanto para los vértices como para las aristas. Es el siguiente:

- **Primera línea:** cabecera en la que se indican los atributos del elemento (vértice o arista) separados por el carácter "/".
- **Resto del archivo:** una línea por cada opción de configuración con el siguiente formato:
  - ***Primer elemento:*** identificador de la opción de configuración. Puede ser un valor numérico o el símbolo "*" que indica que es la opción utilizada por defecto.
  - ***Resto de elementos:*** los valores de los atributos indicados en la cabecera, en ese orden y separados por el carácter "/".

Otras características a tener en cuenta son:

- Los atributos que se han definido en el sistema para su posible configuración de manera interactiva se han extraído de los existentes en la plantilla NodeXL, que es la herramienta de visualización principal que se va a utilizar. Si se quisieran añadir otros atributos distintos habría que editar el archivo de configuración respetando las pautas que se indican aquí.
- Cada opción de configuración sólo tiene un elemento obligatorio, que es el identificador de la opción, los demás elementos pueden ser nulos, indicando que la herramienta de visualización debe realizar la acción que tenga configurada por defecto con esos elementos. Los atributos que no tengan valor asociado se indicará sin incluir ningún valor en el lugar que le corresponda pero se deben mantener los caracteres de separación establecidos.

A continuación, se puede observar un ejemplo de un archivo de este tipo:

```
1 Color/Shape/Size/Opacity/Image File/Visibility/Label/Label Fill Color/Label Position/Tooltip/
2 */Blue/Disk////Show////
3 0/Black/Sphere//////Black//
4 1/Fuchsia///////Olive//
5 2/Green/Square////////
```

Tabla 107. Ejemplo de archivo de configuración