Escuela Politécnica Superior (Jaén)

Ingeniería en Informática

Universidad de Jaén

Escuela Politécnica Superior (Jaén)

Proyecto Fin de Carrera

HERRAMIENTA BASADA EN LA COMPUTACIÓN CON PALABRAS PARA LA CARACTERIZACIÓN DE USUARIOS EN REDES SOCIALES

Alumno: Salud María Jiménez Zafra

Tutor: Prof. Dr. Luis Martínez López

Departamento: Informática

Septiembre, 2013

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Departamento de Informática

Dr. D. Luis Martínez López, tutor del Proyecto Fin de Carrera titulado: Herramienta basada en la computación con palabras para la caracterización de usuarios en redes sociales, que presenta Dª. Salud María Jiménez Zafra, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2013

El alumno

El tutor

Dª. Salud María Jiménez Zafra

D. Luis Martínez López

A mi ángel de la guarda, te quiero abuelito.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas, que de un modo u otro, han contribuido a que este proyecto haya salido adelante, especialmente:

A mis padres y a mi hermano, por su apoyo, cariño y confianza constante. Por los valores que me han inculcado y por ser mi ejemplo a seguir.

A mi abuelita, por enseñarme a afrontar las cosas con ilusión y por creer siempre en mí.

A Juan Luis, por la ayuda y comprensión que me ha brindado en todo momento.

A mi tutor, Luis Martínez, por el tiempo que me ha dedicado, por poner a mi disposición todo su conocimiento y por la ayuda tan valiosa que me ha dado para elaborar este proyecto.

A mis profesores, por todo lo que me han enseñado a lo largo de estos años y por haber sabido transmitir su entusiasmo por las asignaturas impartidas.

A toda mi familia, por su apoyo y confianza en mí.

Índice

Capítulo 1. Introducción.

1.1 Introducción al proyecto

La facilidad de comunicación y la inmediatez en la transmisión de opiniones y contenido han hecho de las redes sociales un potente instrumento social. “Las relaciones en las redes sociales se están convirtiendo en una importante tecnología para modelar el comportamiento humano”, como bien dice Ronald R. Yager en su artículo Concept Representation and Database Structures in Fuzzy Social Relational Networks [1].

La importancia del análisis de las redes sociales radica en la estructura de las relaciones entre individuos y la influencia de las mismas en los distintos fenómenos sociales. Estos vínculos facilitan la transmisión de información, el intercambio de ideas y la formación de consensos. Sin embargo, la mayoría de las personas desconocen las características de una red y la posición que los individuos ocupan en ella.

Los análisis sobre redes sociales existentes hasta el momento (segmentación de los usuarios, frecuencia de uso, principales actividades,…) representan las relaciones entre los individuos de manera cuantitativa. En numerosas ocasiones resulta complicado comprender qué significan esos términos cuantitativos. Por ello, con este proyecto se pretende dar un paso más para poder caracterizar a los usuarios y expresar sus relaciones de manera cualitativa.

Ronald R. Yager propone en el artículo mencionado anteriormente [1] dos métodos para enriquecer el modelado de las redes sociales: la representación de redes utilizando grafos difusos y el uso del paradigma de computación con palabras de Zadeh, basado en conjuntos difusos [2]. Estas ideas son la base sobre la que se desarrollará el proyecto por lo que a continuación se explican de forma resumida.

La teoría de grafos permite cuantificar los vínculos entre las personas que pertenecen a una red social y analizar la estructura de la misma. Cada individuo se puede representar como un nodo, y las relaciones entre ellos como aristas.

La computación con palabras es necesaria cuando la información disponible es demasiado imprecisa para justificar el uso de valores cuantitativos. Los seres humanos principalmente se comunican y razonan utilizando términos lingüísticos, mientras que las máquinas requieren símbolos más formales [3]. El paradigma de computación con palabras de Zadeh, basado en conjuntos difusos, proporciona un puente entre la forma de comunicación de los seres humanos y las máquinas.

El fin de este proyecto es caracterizar a los usuarios de una red social mediante la asociación de los conceptos formales de representación de la misma con los conceptos que los seres humanos utilizan habitualmente sobre las redes sociales. Para ello, se desarrollará una herramienta que, a partir de los datos de una red social, generará un grafo en el que se podrán visualizar los individuos que forman parte de la misma y las estructuras relacionales. Utilizando el paradigma de computación con palabras de Zadeh, basado en conjuntos difusos, se expresarán estas relaciones utilizando términos lingüísticos, tal y como se comunican y razonan los seres humanos.

1.2 Propósito

El propósito de este proyecto es desarrollar una herramienta que permita analizar las relaciones existentes entre los usuarios de una red social, llevando a cabo una correspondencia entre la forma de comunicación de los seres humanos y la representación formal de una red.

1.3 Objetivos

- Estudiar la representación formal de una red mediante un grafo.
- Estudiar los conjuntos difusos y su relación con la computación con palabras.
- Estudiar el paradigma de computación con palabras de Zadeh basado en conjuntos difusos.
- Estudiar herramientas para la representación de grafos.
- Desarrollar una herramienta para el análisis de redes sociales utilizando el paradigma de computación con palabras de Zadeh basado en conjuntos difusos.
- Obtener datasets relativos a redes sociales.
- Evaluar la herramienta desarrollada.
- Realizar la memoria.

1.4 Resultados

- Herramienta para la caracterización de usuarios en redes sociales.
- Explicación del método utilizado para la representación de la red.
- Beneficio de utilizar la metodología de computación con palabras.
- Evaluación crítica de la herramienta desarrollada.
- Resultados obtenidos tras aplicar la herramienta a los datasets recopilados.
- Manual de instalación.

- Manual de usuario.
- Memoria.

1.5 Estructura

Antes de profundizar en el desarrollo del proyecto, vamos a explicar brevemente los contenidos expuestos en cada uno de los capítulos en los que se estructura.

Como hemos podido ver, en este capítulo hemos realizado una breve introducción del proyecto, indicando su propósito, los objetivos a alcanzar y los resultados esperados.

En el segundo capítulo se va a profundizar en la idea que ha originado este proyecto, el modelado de redes sociales siguiendo la propuesta de Ronald R. Yager. En primer lugar, se realizará una introducción a la teoría de grafos y a la teoría de conjuntos difusos para poder comprender los términos utilizados en el resto de la memoria. A continuación, se explicará cómo llevar a cabo la representación de redes utilizando grafos difusos y la utilidad del paradigma de computación con palabras de Zadeh, basado en conjuntos difusos.

En el tercer capítulo se revisarán tres de las herramientas más utilizadas para la representación de grafos: IGraph, JUNG y Gephi. Posteriormente se justificará cuál es la herramienta que hemos elegido y se explicará en qué consiste, los algoritmos que proporciona y su utilidad en este proyecto para la visualización de grafos.

El cuarto capítulo es el de mayor extensión ya que en él se muestra el proceso completo de desarrollo. Al tratarse de un proyecto de desarrollo software, en primer lugar se realizará un breve repaso sobre las diferentes etapas de la Ingeniería del Software, para posteriormente aplicarlas a nuestra herramienta. Así, definiremos los requerimientos funcionales y no funcionales para el sistema, abordaremos el análisis del mismo y realizaremos su diseño. Una vez que tengamos todo esto claro y bien documentado, se implementará la herramienta para el análisis de redes sociales y posteriormente se realizará su validación mediante la utilización de pruebas de caja negra.

El quinto capítulo se dedicará a las conclusiones obtenidas durante el desarrollo de este proyecto.

Además, se incluirá una bibliografía en la que se enumerarán las referencias bibliográficas utilizadas, para poder consultarlas en caso de necesitar más información.

Por último, la sección final de esta memoria contiene los anexos dedicados a la instalación del servidor de bases de datos, y el manual de usuario en el que se describe paso a paso cómo utilizar la aplicación.

Capítulo 2. Análisis de redes sociales mediante grafos difusos y computación con palabras.

2.1 Introducción

Las redes sociales surgen como una nueva herramienta de comunicación entre las personas. Son un importante instrumento social, cuyo éxito radica en la facilidad para intercambiar opiniones y contenido entre los individuos que forman parte de ellas.

En ocasiones, nos encontramos en situaciones en las que resulta complicado determinar a quién tenemos que dirigirnos en función de nuestro propósito. Por ejemplo, si queremos difundir un mensaje sería interesante descubrir qué individuo tiene una mayor influencia sobre el resto o, si necesitamos utilizar intermediarios para ponernos en contacto con una persona, deberíamos buscar aquellos más fiables, es decir, que tengan una relación más fuerte.

Estos aspectos, unidos al papel fundamental de las redes sociales en la actualidad, hacen que resulte interesante el desarrollo de herramientas para analizar las mismas. Por ello, el objetivo de este proyecto es desarrollar una herramienta que permita llevar a cabo el análisis de redes sociales.

La mayoría de los análisis existentes hasta ahora, en ocasiones, resultan complicados de comprender, ya que expresan la información en términos cuantitativos. Con este proyecto se pretende facilitar la compresión de los conceptos formales utilizados en el análisis de redes sociales, usando para ello términos lingüísticos. Para ello, se va a utilizar el método propuesto por Ronald R. Yager [1] que consiste en la representación de la red mediante un grafo difuso y en la utilización del paradigma de computación con palabras de Zadeh, basado en conjuntos difusos. A continuación, se explican de forma detallada los conceptos fundamentales y el proceso llevado a cabo para el desarrollo de este proyecto.

2.2 Teoría de grafos

La Teoría de Grafos es una rama de las Matemáticas y de las Ciencias de la Computación que estudia las propiedades de los grafos. Resulta interesante para el estudio de redes sociales, ya que puede ser utilizada para analizar muchas propiedades de las estructuras sociales y proporciona operaciones matemáticas para medir esas propiedades. Permite cuantificar los vínculos entre las personas que pertenecen a una red social y analizar la estructura de la misma. Para ello, cada individuo se puede representar como un nodo y las relaciones entre ellos como aristas.

Origen

Tiene su origen en el problema de los siete puentes de Königsberg, que fue resuelto por Leonhard Euler en 1736. Königsberg (actualmente Kaliningrado, Rusia) era una ciudad de Prusia del siglo XVIII atravesada por el río Pregolya, el cual dividía el terreno en cuatro regiones distintas, que estaban unidas mediante siete puentes. El problema formulado fue el siguiente:

"*Dado el mapa de Königsberg con el río Pregolya dividiendo el plano en cuatro regiones distintas, que están unidas a través de los siete puentes, ¿es posible dar un paseo comenzando desde cualquiera de estas regiones, pasando por todos los puentes, recorriendo sólo una vez cada uno, y regresando al mismo punto de partida?*" (Ver Ilustración 1).

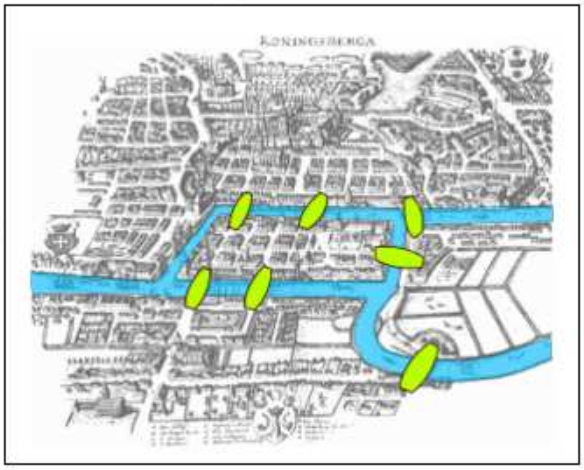

Ilustración 1: Puentes de Königsberg

Leonhard Euler propuso una ingeniosa forma de solucionar el problema. Para su demostración recurrió a una abstracción del mapa, centrándose exclusivamente en las regiones terrestres y en las conexiones entre ellas. Representó por un punto cada sector de tierra y por una línea cada puente, uniendo los distintos puntos con tantas líneas como puentes había entre dichos sectores, creando de esta forma el primer grafo de la historia. En la Ilustración 2 se puede observar dicha representación.

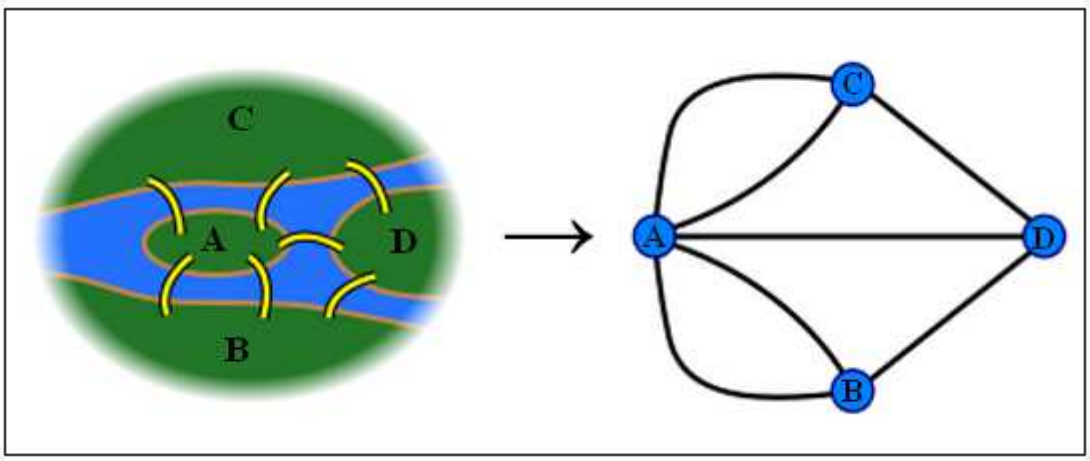

Ilustración 2: Propuesta de Euler

Euler se dio cuenta de que la cantidad de líneas de cada punto tendría que ser par, para así poder entrar y salir de dicho punto, ocupando tan sólo una vez cada puente. De esta manera demostró

que era imposible recorrer la ciudad de la forma pedida, ya que el número de líneas que llegan a cada punto es impar y esto implica que no es posible recorrerlos pasando una sola vez por cada uno de los puentes, es decir, el problema propuesto no tenía solución.

Esta abstracción del problema ideada por Euler dio pie a la primera noción de grafo.

Definición de grafo

Un grafo G es un conjunto de objetos llamados vértices o nodos, N = {n_1, n_2,…,n_m}, unidos por enlaces llamados aristas o arcos, A = { a_1, a_2,…,a_t }, que permiten representar relaciones binarias existentes entre los elementos de un conjunto. En función de la dirección de las aristas un grafo puede ser:

- **Dirigido**: las aristas tienen una dirección definida, indicando que existe una relación del nodo desde el que parte la flecha con el nodo en el que incide la misma (ver Ilustración 3).

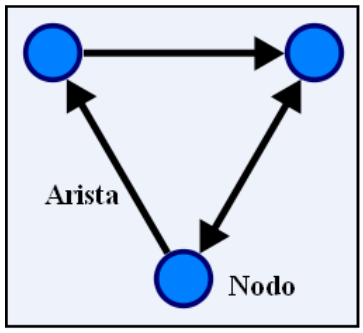

Ilustración 3: Grafo dirigido

- **No dirigido**: las aristas no tienen una dirección definida debido a que la relación se produce en ambos sentidos (ver Ilustración 4).

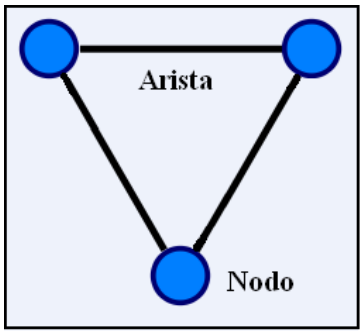

Ilustración 4: Grafo no dirigido

Hay ocasiones en las que no sólo resulta interesante la dirección de las aristas, es decir, si dos nodos están conectados o no, sino que en muchos casos es preciso atribuir a cada arista un número específico llamado peso que indique la importancia de la conexión. De este aspecto se encargan los grafos ponderados (ver Ilustración 5). Se dice que un grafo G= (N, A) es ponderado, etiquetado o con peso si cada arista (a_i, a_j) tiene asociado un valor o peso p_{ij} que viene determinado por una función de ponderación $P : A\ x\ A \rightarrow R$, donde R es el conjunto de los números reales.

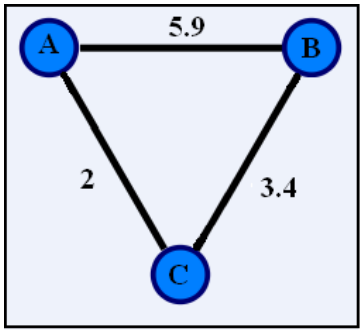

Ilustración 5: Grafo ponderado

En este ejemplo se puede observar que el peso de la arista (A, B) = 5.9, lo cual quiere decir que la importancia de la conexión entre el nodo A y el nodo B es de 5.9.

Como se ha indicado anteriormente en este proyecto nos centraremos en un tipo de grafo ponderado conocido como grafo difuso, en el que el peso de cada arista se limita al intervalo [0,1]. Este tipo de grafo resulta de gran utilidad a la hora de representar dependencias definidas de forman imprecisa.

Formalmente, un grafo difuso G(N, A) es aquel que está formado por un par de funciones $\sigma : N \rightarrow [0,1]$ y $\mu : N\ x\ N \rightarrow [0,1]$, donde para todo par n_i, n_j pertenecientes al conjunto de nodos N se verifica que $\mu(n_i, n_j) \leq \sigma(n_i) \wedge \sigma(n_j)$ [4]. La función $\sigma(n_i)$ indica el valor de cada nodo, mientras que la función $\mu(n_i, n_j)$ establece el valor de cada arista de manera que el peso de la arista que une los nodos n_i y n_j es menor o a lo sumo igual que el mínimo de los valores de los nodos n_i y n_j.

Cualquier relación R ⊆ N x N en un conjunto N puede considerarse como la definición de un grafo con un conjunto de nodos N y un conjunto de aristas A [4]. Del mismo modo, cualquier relación difusa $\mu : N\ x\ N \rightarrow [0,1]$, puede ser considerada como la definición de un grafo ponderado o grafo difuso, donde la arista (n_i, n_j) Є N x N tiene un peso $\mu(n_i, n_j)$ Є [0,1].

2.3 Teoría de conjuntos difusos

Los grafos difusos se basan en la Teoría de Conjuntos Difusos, por lo que a continuación presentaremos una breve descripción de los principales conceptos de esta teoría, que son necesarios para el desarrollo del proyecto.

El interés de la Teoría de Conjuntos Difusos se centra en modelar aquellos problemas donde los enfoques clásicos de la Teoría de Conjuntos y la Teoría de Probabilidades resultan insuficientes o no operativos. Para ello, dicha teoría generaliza la noción clásica de conjunto e introduce el concepto de "difusicidad" (fuzziness).

Conjuntos difusos y función de pertenencia

Un conjunto es una colección de objetos que comparten una serie de características o propiedades. Para determinar si un objeto pertenece o no a un conjunto determinado se utiliza el concepto de dicotomía, es decir, si el objeto pertenece al conjunto se utiliza el valor 1 y si no pertenece el valor 0. Esta decisión de clasificación puede expresarse a través de una función característica.

Sea C un conjunto en el universo X, la función característica asociada a C se define como:

$$\mathrm{C(x)} = \begin{cases} 1 \; si \; x \; \in C \\ 0 \; si \; x \; \notin \mathrm{C} \end{cases}$$

Pero en numerosas ocasiones, las clases de objetos del mundo real no tienen unos límites claros y bien definidos. Por ejemplo, persona *alta*, coche *rápido*, ordenador *potente*, etc. Como consecuencia aparecen los conjuntos difusos, como una nueva forma de representar la imprecisión y la incertidumbre. A diferencia de los conjuntos tradicionales, admiten valores intermedios en la función característica que pasa a denominarse función de pertenencia.

Un conjunto difuso puede definirse como una colección de objetos con valores de pertenencia entre 0 (exclusión completa) y 1 (pertenencia completa). Los valores de pertenencia expresan los grados con los que cada objeto es compatible con las propiedades o características distintivas de la colección. De esta manera, el grado de pertenencia de un objeto a una categoría concreta puede ser expresado por un número real en el intervalo [0,1]. Cuanto más cercano a 1 sea el grado, indicará mayor pertenencia a una categoría determinada y cuanto más cercano a 0 indicará menor pertenencia a dicha categoría.

Formalmente, podemos decir que un conjunto difuso D sobre X está caracterizado por una función de pertenencia que transforma los elementos de un dominio, espacio o universo del discurso X en el intervalo [0,1].

$$\mu_D : X \rightarrow [0, 1]$$

Así, un conjunto difuso D en X puede representarse como un conjunto de pares ordenados de un elemento genérico x, tal que x Є X, y su grado de pertenencia $\mu_D(x)$:

$$D = \{(x, \mu_D(x))/\ x\ \in X, \mu_D(x)\ \in [0,1]\}$$

En un conjunto tradicional, un elemento pertenece a un conjunto dado o bien no pertenece. En cambio, un conjunto difuso permite valores intermedios de pertenencia.

Ejemplo

Los conjuntos difusos permiten formalizar expresiones lingüísticas que contienen algún grado de ambigüedad, es decir, proporcionan un método para expresar matemáticamente conceptos como *alto, frío, rápido, fuerte ,corto, etc.* que son usados habitualmente, pero que no son precisos.

Por ejemplo, sea X = [0,220] el universo formado por los posibles valores de velocidad a los que circula un coche en autovía en km/h, y D el conjunto "coche veloz". Claramente, un coche que circula a 60 km/h no es un coche veloz, por lo que su grado de pertenencia al conjunto "coche veloz" es 0. Un coche que circula a más de 120 km/h puede considerarse un coche veloz y se le puede asignar un valor 1 para expresar el grado de compatibilidad con dicho concepto. Por tanto, los coches cuya velocidad se encuentre en el rango [121, 220] tienen un grado de pertenencia igual a 1 en el conjunto de coches veloces. La cuantificación del resto de valores se puede llevar a cabo mediante una función de pertenencia que puede ser la siguiente:

$$\mu_D(X) = \begin{cases} 0 & si\ x\ \epsilon\ [0,60] \\ 1 - \dfrac{121 - x}{61} & si\ x\ \epsilon\ (60,121) \\ 1 & si\ x\ \epsilon\ [121,220] \end{cases}$$

A continuación, se muestra en la Ilustración 6, la traducción y representación del conjunto D según la función de pertenencia indicada.

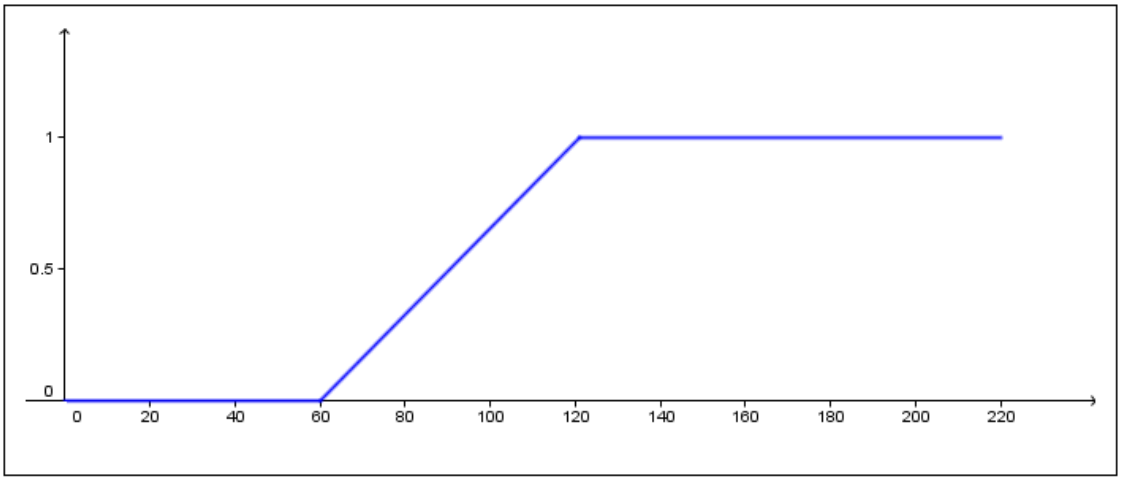

Ilustración 6: Representación del conjunto "coche veloz"

Los conjuntos difusos permiten manejar la imprecisión, lo que facilita la traducción del pensamiento humano a operaciones computacionales.

2.4 Análisis de redes sociales mediante grafos difusos

Una vez comprendidos estos conceptos, es el momento de profundizar en el método propuesto por Ronald R. Yager, que es en el que se fundamenta este proyecto.

Las redes sociales se están convirtiendo en una tecnología importante para el modelado del comportamiento humano, por lo que el análisis de las mismas resulta de gran utilidad. Por ello, Ronald R. Yager propone en primer lugar la utilización de grafos difusos para la representación de las redes, ya que van más allá de indicar si dos nodos están conectados o no y utilizan conexiones ponderadas o difusas. En segundo lugar, propone el uso del paradigma de computación con palabras de Zadeh, basado en conjuntos difusos, para proporcionar un puente entre el modelo formal de una red y los términos lingüísticos utilizados para el análisis de la misma.

2.4.1 Grafos difusos

Para la representación formal de la red social a analizar vamos a utilizar un grafo difuso no dirigido G(N, A), tal y como propone Ronald R. Yager [1]. Cada nodo representará un individuo y cada arista reflejará la relación existente entre cada par de individuos.

El concepto de relación difusa juega un papel fundamental en el modelado de un grafo difuso [5]. Sea X un conjunto de elementos, una relación difusa en X se define como $R: X\ x\ X \rightarrow [0, 1]$ donde R(x, y) indica el grado de relación entre x e y.

Toda relación difusa debe cumplir las siguientes propiedades:

1. **Reflexiva:** $R(x,x) = 1 \; \forall \, x$
2. **Simétrica:** $R(x,y) = R(y,x)$
3. **Transitiva:** $R(x,z) \geq \mathrm{Max}_y \, [R(x,y) \wedge R(y,z)]$

Trasladando este concepto a la terminología utilizada en los grafos difusos, el conjunto de elementos X será el conjunto de nodos del grafo, que se corresponderá con los miembros de una red social. En nuestro caso, para mostrar la funcionalidad que ofrece la herramienta desarrollada, se utilizarán los datos relativos a los miembros de un Sistema de Recomendación de películas. La relación difusa R(x, y) representará el peso de la relación entre el nodo *x* y el nodo *y*, que vendrá determinado por la confianza existente entre cada par de individuos y que será un valor perteneciente al intervalo [0,1], donde 1 representa el máximo nivel de confianza. Para el cálculo de la confianza entre los distintos miembros de esta red se utilizará el método "Trust Derivation" [6], propuesto por Qusai Shambour y Jie Lu, que será explicado de forma detallada posteriormente en la sección "Datos utilizados".

2.4.2 Representación formal de la red mediante grafos difusos

Los grafos, como conceptos matemáticos formales, permiten tratar toda la maraña de información que ofrecen las redes sociales. Proporcionan una forma de visualizar una red social, además de facilitar el análisis de la misma.

Tal y como se ha explicado anteriormente, en este proyecto se va a representar una red social mediante un grafo difuso no dirigido, siguiendo la propuesta de Yager. Cada nodo representará un individuo que será identificado por un número entero de forma unívoca. Cada arista reflejará la confianza, en el intervalo [0,1], existente entre cada par de usuarios.

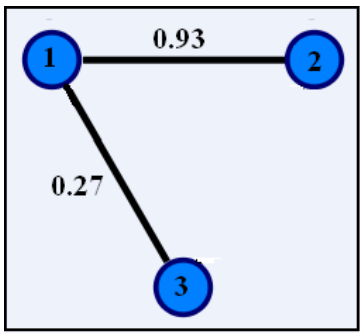

Ilustración 7. Grafo difuso no dirigido

Observando el grafo de ejemplo de la Ilustración 7, podemos ver que la red social que representa está formada por tres individuos. Entre el usuario 1 y el 2 existe una confianza de 0.93 y entre el usuario 1 y el 3 de 0.27, mientras que el usuario 2 y el 3 no se relacionan.

2.4.3 Conceptos fundamentales en el análisis de grafos

Para llevar a cabo el análisis de un grafo difuso existen tres conceptos fundamentales: fortaleza de un camino, longitud de un camino y distancia entre dos nodos. A continuación veremos en qué consiste cada uno de ellos y su aplicación en el análisis de redes sociales.

A) Fortaleza de un camino

Sea G = (X, R) un grafo difuso en el que X representa el conjunto nodos y R la función que determina la relación existente entre cada par de nodos. Un camino ρ en G es una secuencia de nodos diferentes $x_0, x_1, \dots, x_n$ cuya fortaleza viene dada por la expresión:

$$ST(\rho) = Min_{i=1\ to\ n}[R(x_{i-1}, x_i)] \quad (1)$$

donde:

ρ: es el camino del grafo formado por una secuencia de distintos nodos $x_0, x_1, \dots, x_n$.

n: es el número de conexiones o enlaces del camino.

$R(x_{i-1}, x_i)$: es la relación entre el individuo $i-1$ y el individuo i.

Se dice que dos nodos están conectados si entre ellos existe un camino ρ cuya fortaleza es mayor que cero, $ST(\rho) > 0$.

B) Camino más fuerte

A partir de esta fórmula podemos calcular el camino más fuerte para ir desde el nodo x_i hasta el nodo x_j y obtener la fortaleza del mismo. Para ello tendremos que calcular todos los posibles caminos que partiendo del nodo inicio nos permiten llegar al nodo fin, y calcular la fortaleza de cada uno de ellos, quedándonos con el de mayor fortaleza.

De ahora en adelante utilizaremos la expresión $R^k(x_i, x_j)$ para referirnos a la fortaleza del camino más fuerte entre el nodo x_i y el nodo x_j , que contiene como mucho k enlaces.

C) Longitud de un camino

Sea G = (X, R) un grafo difuso y $\rho = x_0, x_1, \dots, x_n$ un camino, el concepto de longitud de un camino según Rosenfeld [4] está definido por la suma de los inversos de los pesos de las aristas del camino:

$$L(\rho) = \sum_{i=1}^{n} \frac{1}{R(x_{i-1}, x_i)} \quad (2)$$

donde:

ρ: es el camino del grafo formado por una secuencia de distintos nodos $x_0, x_1, \dots, x_n$.

n: es el número de conexiones o enlaces del camino.

$R(x_{i-1}, x_i)$: es la relación entre el individuo $i-1$ y el individuo i.

D) Distancia entre dos nodos

La distancia entre dos nodos x e y en un grafo G = (X, R) es la longitud del camino más corto para ir de x a y.

$$\delta(x, y) = Min_{all\ paths\ x\ to\ y}\ [L(\rho)] \quad (3)$$

donde:

$L(\rho)$: es la longitud de un camino ρ que tiene como origen el nodo x y como fin el nodo y.

2.5 Computación con palabras

La computación con palabras es una metodología en la cual las palabras procedentes del lenguaje natural son utilizadas en lugar de los números para realizar operaciones lógicas y computacionales. Está basada en la teoría de conjuntos difusos y en la lógica difusa. L. Zadeh, en su paper Fuzzy logic = computing with words [2], señala dos situaciones en las que se debe computar con palabras:

- Cuando la información disponible es demasiado imprecisa para representarla con números.

- Cuando hay una tolerancia de imprecisión que puede ser utilizada para conseguir robustez, soluciones de bajo coste y un mayor acercamiento a la realidad.

El objetivo de este proyecto es extender el análisis de las relaciones en las redes sociales mediante la asociación de estos conceptos formales con aquellos con los que los seres humanos entienden las relaciones, de manera que sean comprensibles tanto para los seres humanos como para las máquinas. Para ello, es necesario establecer un puente entre la forma de comunicación y razonamiento de los seres humanos, mediante términos lingüísticos, y la de las máquinas, mediante símbolos más formales. Aquí es donde juega un papel fundamental el paradigma de computación con palabras de Zadeh, basado en conjuntos difusos [2]. Esta tecnología permite un alto nivel de cooperación hombre-máquina, proporcionando un marco en el que los conceptos e ideas pueden ser tratados de forma flexible para ambos. La idea de aplicar tecnologías basadas en conjuntos difusos dentro del dominio del análisis de redes sociales es prometedora, ya que computacionalmente el modelado de estas redes se realiza mediante relaciones matemáticas que, como ya hemos señalado, son equivalentes a los conjuntos difusos.

A continuación, presentamos algunas ideas del enfoque de computación con palabras basada en conjuntos difusos para tratar de entenderlo mejor.

Sea U un atributo que toma sus valores en el espacio Y. Por ejemplo el atributo edad que toma su valor en el conjunto $Y = \{0,1, \dots ,100\}$. En computación con palabras el concepto de valor lingüístico es algo fundamental [11]. Un valor lingüístico es una palabra utilizada para expresar el valor de un atributo U. En el caso del atributo edad, algunos ejemplos de valores lingüísticos son “viejo”, “joven”, “alrededor de 40”.

Otro concepto muy importante es el de vocabulario. Por vocabulario entendemos aquella colección de palabras que utilizamos para expresar los valores lingüísticos asociados a un atributo. Los conjuntos difusos proporcionan una herramienta para formalizar la idea de vocabulario, de manera que permita la comprensión y el tratamiento por parte de las máquinas. Si W es una palabra del vocabulario asociado con el atributo U, podemos expresar W como un subconjunto difuso F del dominio de U. Por tanto, $\forall\, y \in Y$, el grado de pertenencia de y en F indica la compatibilidad del valor y con el valor lingüístico W. De esta manera, el subconjunto difuso F proporciona una interpretación de la palabra

W comprensible para una máquina. Por ejemplo, la palabra "viejo" del atributo edad podemos expresarla como el subconjunto difuso formado por las edades comprendidas entre 70 y 100, $F = \{70, \dots, 100\}$.

Así, permitiendo que el ser humano construya un vocabulario de términos lingüísticos asociados con un atributo y que proporcione una representación para estos términos mediante conjuntos difusos, obtenemos un vocabulario que es coherente tanto para los seres humanos como para las máquinas. De esta manera se rompen las barreras existentes entre ambos, facilitando el análisis de las redes sociales. (Ver Ilustración 8).

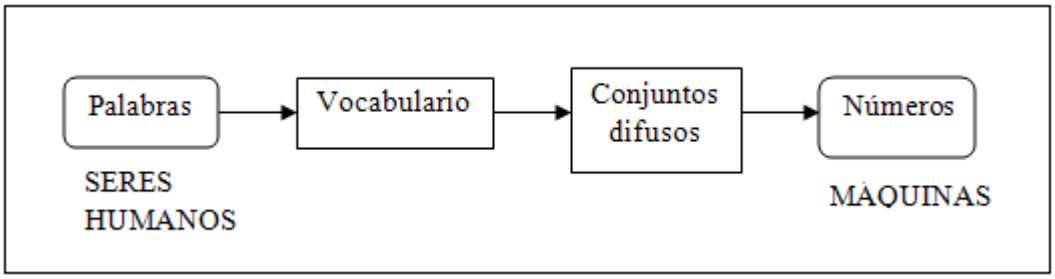

Ilustración 8: Comprensión hombre - máquina

Es necesario enfatizar que tanto la elección del vocabulario como la asociación de las palabras del mismo con conjuntos difusos, se encuentran en manos del ser humano. Al ordenador se le indica los términos que se van a utilizar y el significado en su idioma, los conjuntos difusos.

2.5.1 Modelado de términos lingüísticos para el análisis de grafos

En el análisis de relaciones en redes, existen algunos atributos para los que resulta de utilidad tener un vocabulario con términos comúnmente utilizados.

A) Término fuerte

Uno de estos atributos es la fortaleza de la conexión. Se trata de un atributo que puede tomar valores en el intervalo [0, 1] y en el que términos como *fuerte*, *débil* y *ninguna* pueden formar parte de su vocabulario asociado. En este caso, podemos definir la palabra *fuerte* como un subconjunto difuso S de [0, 1] de manera que, $\forall y \in [0, 1]$, el valor $S(y)$ indicará el grado con el que y satisface la definición del término *fuerte*. Un ejemplo de la definición del término *conexión fuerte* podría ser el mostrado a continuación en la Ilustración 9, y será el utilizado en este proyecto.

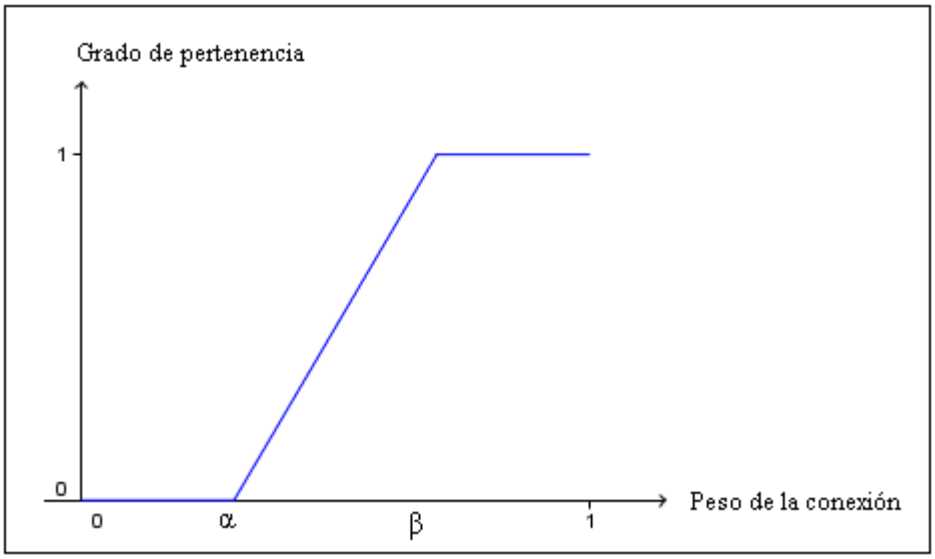

Ilustración 9: Representación del término fuerte

donde:

α: es el valor hasta el cual la fortaleza de la conexión es 0.

β: valor a partir del cual se considera que la fortaleza de la conexión es 1.

Tal y como se ha explicado anteriormente, estos valores los definirá el usuario que utilice la aplicación.

B) Camino corto

Otro atributo para el que resulta interesante tener un vocabulario es el número de enlaces de un camino. Algunas palabras asociadas con este atributo, longitud del camino, pueden ser *corto* y *largo*. En el análisis de redes sociales fundamentalmente nos interesan los caminos cortos, por lo que a continuación daremos una definición de dicho término. El concepto *camino corto*, basándonos en el número de enlaces que contiene, puede ser definido como un conjunto difuso SH de N = {1, 2,…, n}, siendo n el número de enlaces de la red. Siguiendo el sentido común, SH debería cumplir las siguientes propiedades, SH (1) = 1, SH(n)= 0, y SH (k_1) ≥ SH (k_2), si $k_1 < k_2$, es decir, para representar este término se utilizará una función monótona decreciente. Cuanto menor sea el número de enlaces de un camino mayor será su grado de pertenencia al conjunto *camino corto*. Un ejemplo de la representación de este concepto se muestra en la Ilustración 10, y será el utilizado en este proyecto.

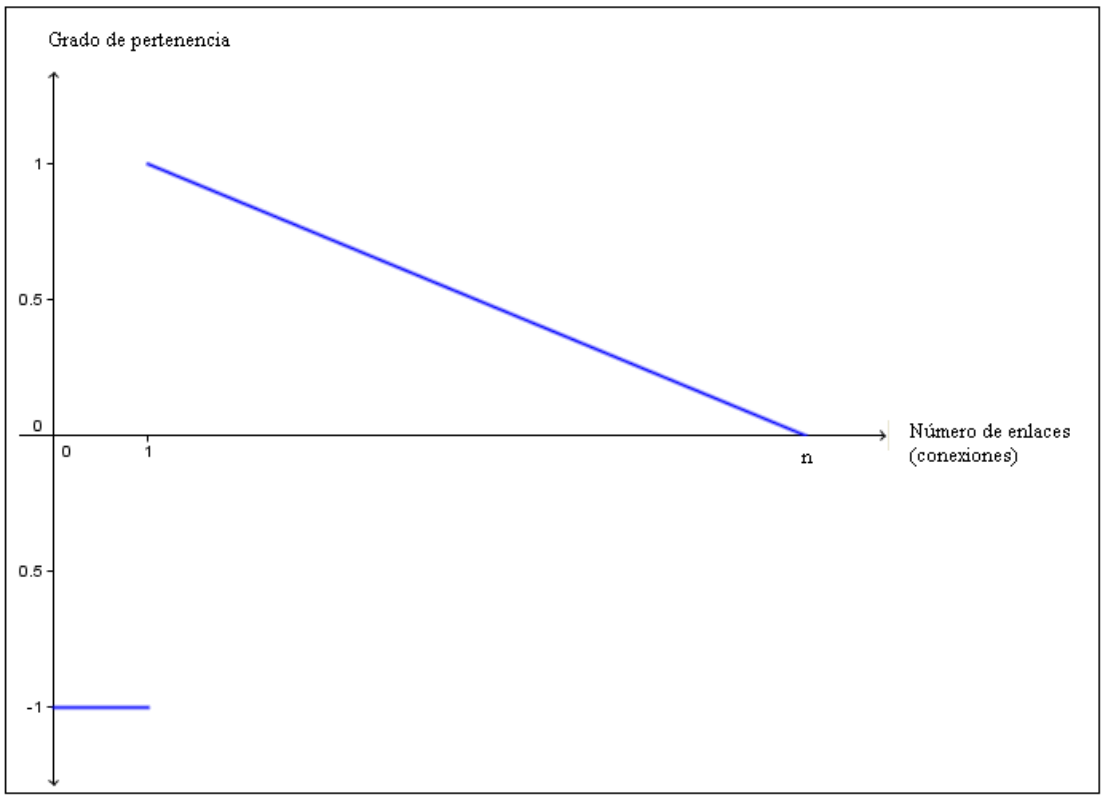

Ilustración 10: Representación del término camino corto

C) Camino más fuerte y corto

Teniendo en cuenta la definición de los términos *conexión fuerte* y *camino corto*, podemos determinar el grado con el que existe una conexión fuerte y de longitud corta entre dos nodos, definiendo de esta manera un nuevo término, el término *camino más fuerte y corto*. Para ello, utilizamos la fórmula de una integral de Sugeno [12], [13]:

$$C_1(x_i, x_j) = Max_{k=1\ to\ t}[SH(k) \wedge S(R^k(x_i, x_j))] \quad (4)$$

donde:

x_i : nodo inicio.

x_j : nodo fin.

t: número máximo de enlaces que puede tener el camino.

$SH(k)$: grado con el que un camino de k enlaces pertenece a la definición de *camino corto*.

$R^k(x_i, x_j)$: fortaleza del camino más fuerte desde el nodo x_i hasta el nodo x_j que tiene como mucho k enlaces.

$S(R^k(x_i, x_j))$: grado con el que $R^k(x_i, x_j)$ pertenece a la definición de *conexión fuerte*.

$\wedge$: t-norma, en nuestro caso utilizaremos el mínimo.

Por medio de esta ecuación podemos determinar cuál es el camino más corto y fuerte para ir desde el nodo inicio hasta el nodo fin, lo cual será de gran utilidad en el análisis de una red.

D) Centralidad

Otro concepto importante en el análisis de redes sociales es el de centralidad [14], ya que la centralidad de un nodo está relacionada con su importancia en la red. La medida de la centralidad de un nodo x_i se realiza a partir del número de nodos que están conectados a él por caminos de como mucho k enlaces.

$$C^k(x_i) = \sum_{\substack{j=1 \\ j \neq i}}^{m} R^k(x_i, x_j) \quad (5)$$

En nuestro caso, al tratarse de un grafo ponderado, podemos encontrar el problema de que un gran número de conexiones débiles, es decir, pequeños valores de $R^k(x_i, x_j)$, pueden parecer una conexión fuerte. Para solventar este problema podemos utilizar dos alternativas:

- Un método consiste en calcular el número de nodos conectados a x_i con una fortaleza de al menos α, utilizando como mucho k conexiones o enlaces.

$$C_\alpha^k(x_i) = \sum_{\substack{j=1 \\ j \neq i}}^{m} R_\alpha^k(x_i, x_j) \quad (6)$$

- Otra alternativa, quizás más apropiada para la definición de centralidad y que por tanto será la que utilicemos, sería calcular el número de conexiones fuertes utilizando como mucho k enlaces. En este caso, sería necesario utilizar la definición del concepto conexión fuerte indicada anteriormente.

$$C_\alpha^k(x_i) = \sum_{\substack{j=1 \\ j \neq i}}^{m} S(R^k(x_i, x_j)) \quad (7)$$

donde:

m : número de nodos del grafo.

k : número máximo de enlaces que puede tener el camino.

$R^k(x_i, x_j)$: fortaleza del camino más fuerte desde el nodo x_i hasta el nodo x_j que tiene como mucho k enlaces.

$S(R^k(x_i, x_j))$: grado con el que $R^k(x_i, x_j)$ pertenece a la definición de *conexión fuerte*.

2.6 Datos utilizados

Con el objetivo de mostrar la funcionalidad de la herramienta que se va a desarrollar en este proyecto, se va a representar una red social formada por los individuos de un Sistema de Recomendación de películas y una red social simulada bastante sencilla.

2.6.1 Datos sobre el Sistema de Recomendación de películas

El Sistema de Recomendación que se va a representar cuenta con 943 usuarios y 1682 películas, y un total de 100000 valoraciones que reflejan las puntuaciones que los distintos usuarios han realizado sobre las películas.

Para la representación de la red social se va a utilizar un grafo difuso no dirigido. Cada uno de los miembros del Sistema de Recomendación será representado mediante un nodo que contendrá un número entero que lo identificará de forma unívoca. Cada arista reflejará la confianza existente entre cada par de individuos, en función de las valoraciones de las películas que hayan realizado. Para el cálculo de la confianza utilizaremos el método "Trust Derivation" [6] propuesto por Qusai Shambour y Jie Lu, tal y como indicamos anteriormente, que pasamos a explicar a continuación.

Trust Derivation

Esta técnica permite medir la confianza en un usuario determinado basándose en la precisión de las predicciones obtenidas a partir de él para un usuario activo. Por ejemplo, si un usuario b ha emitido recomendaciones muy precisas para un usuario a en el pasado, entonces el usuario b debe obtener una puntuación de confianza alta por parte del usuario activo a [7].

Para obtener la confianza de un usuario a en un usuario b, en primer lugar, es necesario calcular la predicción de las valoraciones que realizaría el usuario a, utilizando el método de

predicción de Resnick [8]. Para cada $a, b \in U$, $i \in I$, la predicción de la valoración del usuario a para el ítem i basándose en el usuario b ($P_{a,i}$: $UxI \rightarrow [0,5]$) viene dada por:

$$P_{a,i} = \bar{r}_a + (r_{b,i} - \bar{r}_b) \quad (8)$$

donde:

U: conjunto de usuarios.

I: conjunto de películas.

$P_{a,i}$: predicción de la valoración del usuario a para el ítem i.

$r_{b,i} \in \{1, 2, 3, 4, 5\}$: valoración que el usuario b ha realizado del ítem i.

$\bar{r_a}$: media de las valoraciones realizadas por el usuario a.

$\bar{r_b}$: media de las valoraciones realizadas por el usuario b.

En segundo lugar, teniendo en cuenta que la confianza de un usuario en otro se va a basar en la precisión de las predicciones realizadas, se va a utilizar el método MSD (Mean Squared Differences) también conocido como error cuadrático medio. El método MSD [9] permite medir el grado de similitud del usuario a con respecto al usuario b, a partir del error de las predicciones de los ítems que han valorado ambos usuarios. Para garantizar que el valor de $MSD_{a,b} \in [0, 1]$, tenemos que normalizar las valoraciones del usuario activo ($r_{a,i}$) y las predicciones ($P_{a,i}$) dentro del rango [0, 1] utilizando el método de normalización min-max [10]:

$$v' = \frac{v - min}{max - min} * (new\_max - new\_min) + new\_min \quad (9)$$

donde:

v': valor en el nuevo rango.

v : valor en el antiguo rango.

$[min, max]$: antiguo rango. En el caso de las predicciones ($P_{a,i}$) este rango se corresponde con el intervalo [0, 5], tal y como se ha indicado anteriormente en el método de predicción de Resnick. En el caso de las valoraciones ($r_{a,i}$), *min* será el mínimo de las valoraciones realizadas por el usuario a y *max* el máximo.

[new_min, new_max]: nuevo rango, en este caso [0, 1].

Por tanto, $\forall\, a, b \in U$, el grado de similitud del usuario a con respecto al usuario b, $MSD_{a,b} \in [0, 1]$, basado en el error de las predicciones de los ítems que han valorado ambos usuarios $I_{a,b}$, viene dado por:

$$MSD_{a,b} = 1 - \frac{\sum_{i=1}^{I_{a,b}} (P_{a,i} - r_{a,i})}{I_{a,b}} \quad (10)$$

donde:

$P_{a,i}$: predicción normalizada de la valoración del usuario a para el ítem i.

$r_{a,i}$: valoración normalizada que el usuario a ha realizado del ítem i.

$I_{a,b}$: número de ítems que han valorado ambos usuarios, a y b.

Teniendo en cuenta que toda relación difusa debe cumplir la propiedad reflexiva ($R(x, x) = 1\ \forall\, x$), simétrica ($R(x, y) = R(y, x)$) y transitiva ($R(x, z) \geq Max_y\, [R(x, y) \wedge R(y, z)]$), y que el método utilizado para calcular la confianza entre dos usuarios no verifica la propiedad simétrica, la relación entre el usuario a y el usuario b se calculará de la siguiente forma:

$$R(a, b) = R(b, a) = \min\,(MSD_{a,b}, MSD_{b,a}) \quad (11)$$

donde:

$MSD_{a,b}$: confianza del usuario a en el usuario b.

$MSD_{b,a}$: confianza del usuario b en el usuario a.

2.6.2 Datos sobre la red social simulada

Debido a que la cantidad de información disponible sobre el Sistema de Recomendación de películas dificulta la visualización de las relaciones existentes entre sus miembros, se ha decidido construir una red social simulada formada sólo por 6 miembros, para poder explicar de forma más clara las distintas opciones de análisis que ofrece la aplicación. No obstante, la aplicación permitirá al usuario elegir los datos que desea visualizar para llevar a cabo un análisis de los mismos.

Capítulo 3. Herramientas para la representación de grafos.

3.1 Introducción

El objetivo de este capítulo es el estudio y comparación de tres de las herramientas más utilizadas para el análisis y visualización de grafos, IGraph, JUNG y Gephi. Una vez realizado el estudio y elegida la herramienta que se va a utilizar, pasaremos a describirla detalladamente, centrándonos en las funcionalidades que son de utilidad en nuestro proyecto.

3.2 Estudio comparativo

A continuación, se van a mostrar las características más relevantes de cada una de estas herramientas, tratando de justificar por qué hemos decido utilizar la herramienta JUNG.

3.2.1 IGraph

IGraph es una librería de código abierto, distribuida bajo licencia GPL, para el estudio y análisis de redes/grafos. Sus principales objetivos son proporcionar un conjunto de tipos de datos y funciones para facilitar la implementación de algoritmos de grafos y permitir un manejo rápido de grandes grafos con millones de nodos y arcos, todo ello por medio de un lenguaje de alto nivel como R (lenguaje para el análisis estadístico y gráfico) [22]. IGraph permite manipular tanto grafos dirigidos como no dirigidos. Por otro lado cuenta con implementaciones de problemas típicos de teoría de grafos como árboles de expansión mínima y flujo de red. También implementa algoritmos para algunos métodos de análisis estructural dentro de una red [23].

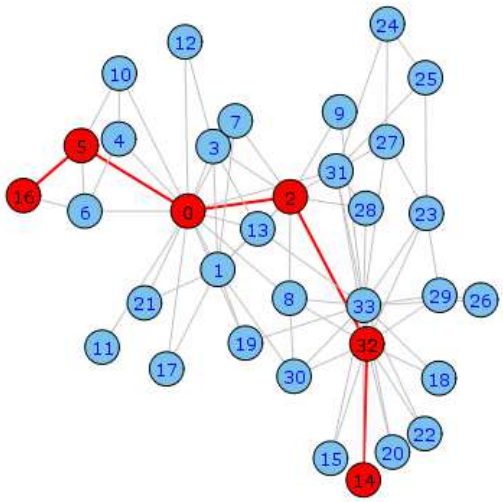

Ilustración 11: Ejemplo visualización IGraph

IGraph se puede instalar como una librería del lenguaje C, como un paquete de R, como un módulo de extensión de Python o como una extensión de Ruby.

3.2.2 JUNG

JUNG (abreviación de Java Universal Network/Graph) es un framework de código abierto que proporciona un lenguaje común y extensible para la manipulación, análisis y visualización de datos que pueden ser representados como un grafo o una red [24]. Está escrito en Java, permitiendo el desarrollo de aplicaciones basadas en él por medio de la API disponible para su uso.

Se trata de una librería que permite el trabajo con grafos, pero que necesita conocimientos de programación ya que necesariamente debe estar implementada en una aplicación Java.

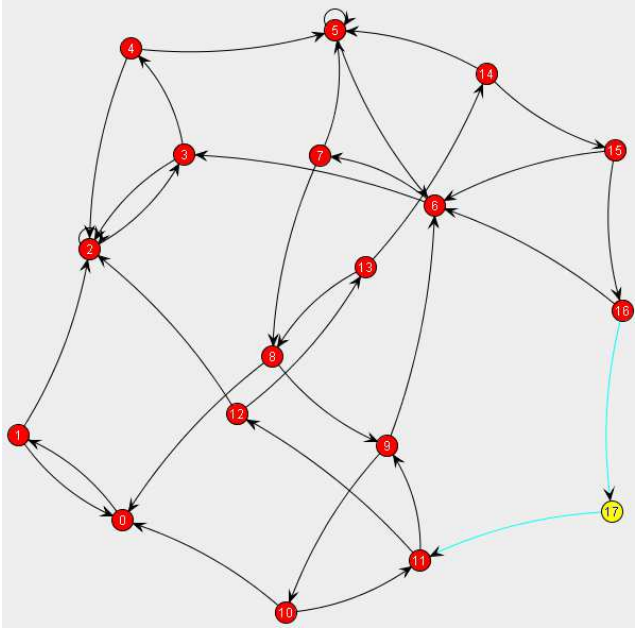

Ilustración 12: Ejemplo visualización JUNG

Las principales características de JUNG incluyen [25]:

- Soporte para una variedad de representaciones de entidades y sus relaciones, incluyendo grafos dirigidos y no dirigidos, grafos multi-modales, multígrafos e hipergrafos.

- Mecanismo para etiquetar grafos, entidades y relaciones con metadatos. Esta capacidad facilita la creación de herramientas analíticas para conjuntos complejos de datos que necesitan analizar las relaciones entre las entidades así como los metadatos asociados a cada entidad y relación.

- Implementación de algoritmos de teoría de grafos, análisis exploratorio de datos, análisis de redes sociales y aprendizaje automático.

- Un framework de visualización que permite de forma fácil la construcción de herramientas para la exploración interactiva de redes de datos.

- Mecanismos de filtrado para extraer subconjuntos de una red, permitiendo centrar la atención en una porción específica de la red.

JUNG puede ser utilizado para construir herramientas orientadas al análisis de redes/grafos o para proveer estas capacidades a sistemas existentes. Esto es posible gracias a la flexibilidad que ofrece al ser una API y no una herramienta final.

3.2.3 Gephi

Gephi es una herramienta final desarrollada en Java para la exploración, navegación y análisis de grafos [26]. Permite a los usuarios interactuar con las distintas representaciones, manipular las estructuras, las formas y los colores. Utiliza un motor renderizado 3D para mostrar las redes. Su objetivo es ayudar a los analistas de datos a realizar hipótesis, descubrir patrones, aislar singularidades en las estructuras o encontrar fallos en los datos. Soporta la representación de grafos dirigidos, no dirigidos y mixtos.

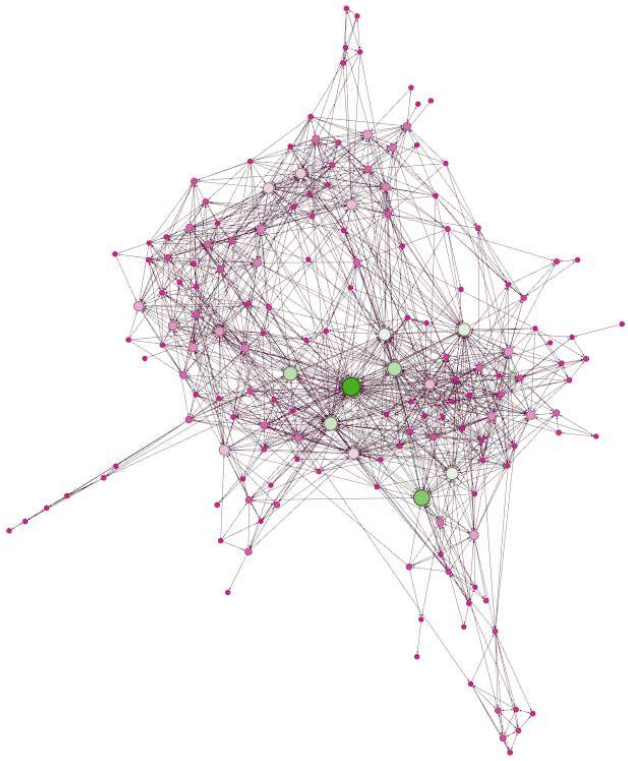

Ilustración 13: Ejemplo visualización Gephi

Destaca por ser una herramienta de código libre disponible para Windows, Linux y Mac que ofrece un conjunto enorme de posibilidades de interacción (cambiar el tamaño o color de los nodos, mostrar u ocultar sus etiquetas, cambiar la forma de las aristas, etc.).

Además, dispone del código fuente para su utilización y de una API denominada Gephi Toolkit [27] para desarrollar aplicaciones propias basadas en dicha herramienta. Esta es la parte que nos interesa en este proyecto.

3.2.4 Comparativa

En este proyecto se va a llevar a cabo el desarrollo de una herramienta para el análisis de redes sociales con tecnología Java. Por tanto, la librería IGraph queda automáticamente descartada ya que no está disponible para Java.

Nos encontramos entonces en situación de decidir entre Gephi y JUNG. Como herramienta final, Gephi ofrece numerosas opciones a la hora de interactuar con la información pero su documentación para desarrollar aplicaciones parece algo escasa. Por ello, hemos decidido utilizar JUNG ya que parece una librería bastante sencilla e intuitiva y contiene todos los métodos necesarios para desarrollar nuestra herramienta de análisis. A continuación, pasamos a describirla de forma detallada.

3.3 JUNG

JUNG (Java universal Network /Graph) es una librería de código abierto basada en Java, diseñada para el modelado, análisis y visualización de datos que se pueden representar en forma de grafos [24]. Se centra en el estudio de grafos y algoritmos relacionados con el análisis de redes sociales, en la visualización de información, en el descubrimiento de conocimiento y en la minería de datos. Sin embargo, no es específica para estos campos y se puede utilizar para muchas otras aplicaciones relacionadas con grafos y redes. Fue creada por tres estudiantes de Ciencias de la Computación y de la Información de la Universidad de California: Joshua O'Madadhain, Danyel Fisher y Scott White.

La distribución actual de JUNG incluye un gran número de algoritmos de teoría de grafos, minería de datos y análisis de redes sociales, tales como rutinas de clustering, generación aleatoria de grafos, análisis estadístico, cálculo de distancias en grafos y otras medidas de importancia. Además, proporciona un framework de visualización que facilita la construcción de herramientas para la exploración interactiva de los datos de la red.

¿Por qué surgió JUNG?

Según la documentación oficial del proyecto JUNG, se creó ante la percepción de la necesidad de una herramienta genérica, flexible y potente para la creación, manipulación, análisis y visualización de grafos y redes en Java. Aún así existen otras herramientas que se pueden adecuar más a las necesidades y capacidades específicas de otros proyectos.

¿Qué es y que no es JUNG?

JUNG es un framework que permite la creación de herramientas de diseño y visualización de grafos y redes. Puede ser utilizado tanto para la creación de pequeñas porciones de código para probar ideas como para crear herramientas con una interfaz gráfica de usuario sofisticada. No es en sí misma una herramienta final, ni lo pretende ser. Es posible construir una herramienta basada en ella pero para ello es necesario conocer el lenguaje de programación Java.

Dependencias

JUNG depende de tres librerías externas para su funcionamiento: JUnit, Colt y Common Collections.

- **JUnit** es un framework que se utiliza para las pruebas unitarias de JUNG. Se puede descargar en la dirección http://junit.org/. Permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado. Si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba. En caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

- **Colt** es una librería para la computación científica y técnicas de alto rendimiento en Java que proporciona estructuras de datos y algoritmos que permiten llevar a cabo operaciones con precisión sobre grandes cantidades de datos. Fue desarrollada en el CERN junto con otros laboratorios de investigación. Es posible descargarla en la dirección http://acs.lbl.gov/software/colt/.

- **Commons Collections** es una librería que proporciona estructuras de datos muy eficientes que permiten agilizar los procesos en los que intervienen grandes cantidades de información. Se puede descargar desde http://larvalabs.com/collections/.

En este proyecto se va a utilizar la versión JUNG 2.0, la cual ya contiene estas tres librerías. Para descargarla es necesario acceder a la siguiente página http://sourceforge.net/projects/jung/files/jung2-alpha/ .

Esta librería ofrece múltiples propiedades para el tratamiento de grandes cantidades de información, pero en nuestro caso únicamente la utilizaremos para la visualización de los grafos no dirigidos utilizados para representar a las redes sociales de estudio. A continuación, pasamos a describir las posibilidades que ofrece esta herramienta centrándonos en las que han sido de utilidad en nuestro proyecto.

3.3.1 Construcción de un grafo no dirigido

Tal y como se indicó en el capítulo anterior, en este proyecto se va a representar un grafo difuso no dirigido, también conocido como grafo difuso indirecto.

JUNG2 proporciona varias interfaces para la definición de distintos tipos de grafos:

- **Graph <V, E>**
- **DirectedGraph<V, E>**
- **UndirectedGraph<V, E>**
- **Forest< V, E>**
- **Hypergraph<V, E>**
- **KPartiteGraph<V, E>**
- **MultiGraph<V, E>**
- **Tree<V, E>**

Estas interfaces se encuentran en el paquete *edu.uci.ics.jung.graph* y permiten definir las operaciones básicas que se pueden realizar sobre un grafo:

- Añadir vértices y aristas (addVertex(V vertex), addEdge (E edge)).
- Eliminar vértices y aristas (removeVertex(V vertex), removeEdge(E edge)).
- Obtener todos los vértices y aristas del grafo (getVertices(),getEdges()).
- Determinar si dos vértices están conectados por alguna arista (isNeigbor(V v1, V v2)).
- Determinar si una arista incide en un vértice (isIncident(V vertex, E edge)).
- Obtener el número de vértices y aristas que tiene el grafo (getVertexCount(),getEdgeCount()).
- Averiguar cuáles son los vértices conectados a uno dado (getNeighbord(V vertex)).

- Determinar el número de aristas incidentes en un vértice (degree(V vertex)).

- Obtener el conjunto de aristas incidentes en un vértice (getIncidentEdges(V vertex).

Pero en este caso sólo nos interesa la interfaz UndirectedGraph<V, E>, que permite la construcción de grafos que sólo admiten aristas no dirigidas. Los vértices y aristas pueden ser objetos simples como enteros, cadenas, etc., o bien clases definidas por nosotros mismos.

En el grafo de la red social que estamos tratando en el proyecto, los vértices se corresponderán con los identificadores de los miembros de la red social, es decir, serán números enteros. Para las aristas utilizaremos una cadena de caracteres con la forma idUsuario1_idUsuario2_valor, donde idUsuario1 e idusuario2 representan los identificadores de dos usuarios entre los que existe relación, y valor el grado de confianza existente entre dichos usuarios. A continuación se muestran los métodos de la librería JUNG que se han utilizado para crear dicho grafo mediante un ejemplo básico:

```
// Creamos un grafo no dirigido, UndirectedGrap<V, E>, donde V es el tipo de los vértices y E es el tipo de las aristas.

UndirectedGraph < Integer, String> _grafo = new UndirectedSparseGraph();

//Añadimos algunos vértices de tipo entero

_grafo.addVertex((Integer) 1);
_grafo.addVertex((Integer) 2);
_grafo.addVertex((Integer) 3);

//Añadimos algunas aristas de tipo cadena con la forma idUsuario1_idUsuario2_valor

_grafo.addEdge("1_2_0.4", 1, 2);
_grafo.addEdge("2_3_0.85", 2, 3);
```

En caso de necesitar más información que un simple número o cadena, podemos definir nuestras propias clases de vértices (V) y aristas (E):

```
class MiNodo {

        int id;
        String nombre;

        public MiNodo(int id, String nombre) {

                this.id = id;
                this.nombre = nombre,
```

```
        }

        public String toString() {
                return "V"+id + " Nombre: " + nombre;

         }

}

class MiArista {

        double capcidad;
        double peso;

        public MyLink(double weight, double capacity) {

                this.peso= peso;
                this.capacidad= capacidad;

         }

         public String toString() {
                 return "E" + peso + "-" + capacidad;
        }

}
```

3.3.2 Algoritmos

JUNG2 ofrece una gran variedad de algoritmos para trabajar con grafos y redes. Estos algoritmos se clasifican según su tipo en paquetes. Actualmente esta librería cuenta con los siguientes paquetes:

- *edu.uci.ics.jung.algorithms.blockmodel*: soporte para establecer y mantener equivalencia de elementos en grafos.

- *edu.uci.ics.jung.algorithms.cluster*: mecanismos para identificación de clusters en grafos. Un cluster es una colección de objetos en la que todos son similares en algún aspecto.

- *edu.uci.ics.jung.algorithms.filters*: mecanismos de filtrado que permiten obtener subgrafos a partir de un grafo original.

- *edu.uci.ics.jung.algorithms.flow*: métodos para calcular propiedades relacionadas con el flujo en redes.

- *edu.uci.ics.jung.algorithms.generators*: métodos para generar grafos.

- *edu.uci.ics.jung.algorithms.generators.random*: métodos para generar grafos de forma aleatoria.

- *edu.uci.ics.jung.algorithms.importance*: algoritmos que permiten medir la importancia de cada vértice o arista de acuerdo con un conjunto de criterios que habitualmente se basan en el posicionamiento de ese vértice o arista en relación al resto del grafo.

- *edu.uci.ics.jung.algorithms.layout*: algoritmos para asignar coordenadas 2D a los vértices. Normalmente se utiliza para la visualización de grafos.

- *edu.uci.ics.jung.algorithms.matrix*: mecanismos para tratar los grafos como matrices.

- *edu.uci.ics.jung.algorithms.metrics*: medidas especializadas en las propiedades de un grafo.

- *edu.uci.ics.jung.algorithms.scoring*: mecanismos para la asignación de valores a los elementos de un grafo (que denotan importancia, influencia, centralidad, etc.) basados en propiedades topológicas.

- *edu.uci.ics.jung.algorithms.shortestpath*: proporciona métodos para calcular distancias y los caminos más cortos.

- *edu.uci.ics.jung.algorithms.transformation*: mecanismos para la transformación de grafos.

- *edu.uci.ics.jung.algorithms.util*: proporciona utilidades algorítmicas generales.

Para desarrollar el proyecto no ha sido necesario utilizar estos algoritmos ya que los métodos para el análisis se han implementado siguiendo la propuesta de Ronald R. Yager [1], por ello no se va a profundizar en ninguno de ellos. Para obtener más información y ejemplos de código de los distintos métodos es posible visitar la página http://jung.sourceforge.net/doc/.

Sin embargo, esta herramienta sí ha sido de gran utilidad para la representación visual de los grafos utilizados en el análisis de redes sociales. Por ello, se va a explicar en el siguiente punto, el mecanismo de visualización utilizado.

3.3.3 Visualización

El paquete de visualización de JUNG2 (edu.uci.ics.jung.visualization) proporciona distintos mecanismos para la representación de grafos utilizando la API de Java Swing. Los elementos fundamentales que necesitamos conocer para llevar a cabo dicha representación son:

1. El grafo a mostrar.

2. Un layout, es decir, una forma de localizar los vértices del grafo en el espacio de visualización. Esto es posible gracias a la interfaz Layout de JUNG2 y a sus clases relacionadas, las cuales se encuentran en el paquete *edu.uci.ics.algorithms.layout*. La función de esta interfaz es proporcionar una coordenada de localización dado un vértice del grafo.

3. Un visualizador para mostrar el grafo.

4. Un componente de Swing en el que incluir el visualizador.

Grafo

Tal y como se ha indicado en el punto 1, en este proyecto se va a construir un grafo difuso no dirigido (UndirectedGrap<V, E>) cuyos vértices representan a los individuos de una red social y cuyas aristas reflejan la confianza existente entre cada par de individuos. A partir de ahora será representado mediante la variable _grafo.

Layout

Los vértices del grafo se van a distribuir en el espacio de visualización siguiendo la forma de un círculo. Para ello se va a utilizar el layout *CircleLayout* proporcionado por JUNG2 en el paquete *edu.uci.ics.jung.algorithms.layout*.

Visualizador

A la hora de representar un grafo existen ciertos aspectos que no debemos descuidar como el color de los vértices, el grosor de las aristas y las etiquetas identificativas, ya que pueden ser de gran utilidad si queremos realizar un análisis del mismo. Por ello, para mostrar el grafo vamos a utilizar la clase *VisualizationViewer*, disponible en el paquete *edu.uci.ics.jung.visualization* , que permite aparte de definir estas características, procesar eventos de ratón producidos sobre el grafo.

JUNG2 permite definir las propiedades del visualizador. Las interfaces que necesitamos utilizar para establecer estas propiedades son RenderContext (disponible en el paquete *edu.uci.ics.jung.visualization*) y Renderer (en *edu.uci.ics.jung.visualization.renderers*). Aunque estas

interfaces ofrecen una gran variedad de métodos, nos vamos a limitar a mostrar aquellos que resultan de interés para el proyecto que estamos desarrollando:

- setVertexFillPaintTransformer(), para cambiar el color de los vértices.
- setEdgeStrokeTransformer(), para cambiar el grosor de las aristas.
- setVertexLabelTransformer() y setEdgeLabelTransformer(), para mostrar un etiqueta representativa en los vértices y aristas respectivamente.

En cada uno de estos métodos es necesario indicar como parámetro una clase Transformer que será la que contenga la información necesaria para cambiar el aspecto visual de las aristas y vértices del grafo.

En la aplicación que vamos a desarrollar, los nodos estarán etiquetados con el identificador del usuario al que representan, y por defecto, su color será verde. Pero este color se modificará a amarillo cuando sea necesario resaltarlo, por ejemplo, para mostrar los líderes de la red, para indicar el camino mínimo entre dos nodos, etc. Para ello, se ha definido el siguiente transformador que cambia el color de un nodo de verde a amarillo si éste ha sido marcado:

```
private static class VertexPaintTransformer implements Transformer<Integer, Paint> {

    private final PickedInfo<Integer> pi;

    VertexPaintTransformer(PickedInfo<Integer> pi) {

        super();

        if (pi == null) {

            throw new IllegalArgumentException("PickedInfo instance must be non-null");

        }

        this.pi = pi;

    }
```

```
        @Override
        public Paint transform(Integer i) {

                Color p = null;

               //Si el nodo está marcado se dibujará de color amarillo, sino de color verde

                if (pi.isPicked(i)) {
                   p = Color.YELLOW;
                } else {
                   p = Color.GREEN;
                }

                return p;
        }
}
```

Por otro lado, las aristas se identifican mediante una cadena de caracteres con la forma idUsuario1_idUsuario2_valor, tal y como se ha indicado anteriormente. A la hora de representar el grafo sólo nos interesa el valor de confianza existente entre los dos nodos que une, por lo que para mostrar únicamente este valor en la etiqueta de una arista se ha definido el siguiente transformador:

```
private static class EdgeLabelTransformer implements Transformer<String, String> {

    @Override
    public String transform(String s) {

       String[] cadena = s.split("_");

       return cadena[2];

    }
  }
```

Además, ha sido necesario definir un transformador que establezca el color de las aristas en negro y otro para modificar el grosor de las mismas cuando se resalte un camino, de manera que se facilite su visualización:

```
private static class EdgePaintTransformer implements Transformer<String, Paint> {

        @Override
        public Paint transform(String s) {
                return Color.BLACK;
        }
```

```
        }

        private static class EdgeStrokeTransformer implements Transformer<String, Stroke> {

                private final PickedInfo<String> pi;
                final Stroke edgeStroke1 = new BasicStroke(2.5f);
                final Stroke edgeStroke2 = new BasicStroke(1.25f);

                EdgeStrokeTransformer(PickedInfo<String> pi) {

                        super();

                        if (pi == null) {
                                throw new IllegalArgumentException("PickedInfo instance must be
non-null");
                        }

                        this.pi = pi;
                }

                @Override
                public Stroke transform(String s) {

                        if (pi.isPicked(s)) {
                                return edgeStroke1;
                        } else {
                                return edgeStroke2;
                        }
                }
          }
```

Otro aspecto que resulta de interés para nuestra aplicación es mostrar la información relativa a los distintos usuarios que forman el grafo. Para ello se ha definido un nuevo transformador que muestra un tooltip con la información de un usuario cuando el cursor del ratón es situado sobre el nodo que lo representa.

```
        private class VertexToolTipTransformer implements Transformer<Integer, String> {

                @Override

                public String transform(Integer i) {

                        return _controlador.peticionObtenerInformacionUsuario(i);
```

```
            }
        }
```

Componente de Swing

Para integrar el visualizador del grafo en nuestra aplicación es necesario un componente de Swing. En concreto, utilizaremos un JPanel que a partir de ahora se representará por la variable _pnGrafo.

Código

Una vez explicados los conceptos fundamentales que hemos utilizado para representar la información proporcionada mediante un grafo y tras indicar la funcionalidad de los transformadores definidos, vamos a incluir el código que refleja el mecanismo de visualización empleado.

```
        //Paso 1: El grafo a mostrar _grafo ha sido creado anteriormente

        //Paso 2: Creamos un layout que se encargará de localizar los vértices en el grafo siguiendo la
    forma de un círculo

        Layout<Integer, String> layout = new CircleLayout(_grafo);
         layout.setSize(_pnGrafo.getPreferredSize());

         //Paso 3: Creamos un visualizador con scroll para mostrar el grafo

          _vv = new VisualizationViewer<Integer, String>(layout);

        GraphZoomScrollPane pane = new GraphZoomScrollPane(_vv);

        BorderLayout panelMapLayout = new BorderLayout();
        _pnGrafo.setLayout(panelMapLayout);

        //Establecemos las propiedades del visualizador
        //Color de los vértices
        _vv.getRenderContext().setVertexFillPaintTransformer(new
VertexPaintTransformer(_vv.getPickedVertexState()));

        //Grosor de las aristas

        _vv.getRenderContext().setEdgeStrokeTransformer(new
EdgeStrokeTransformer(_vv.getPickedEdgeState()));

         //Color de las aristas
        _vv.getRenderContext().setEdgeDrawPaintTransformer(new EdgePaintTransformer());
```

```
        //Etiquetas de los vértices
        _vv.getRenderContext().setVertexLabelTransformer(new ToStringLabeller());

        //Etiquetas de las aristas
        _vv.getRenderContext().setEdgeLabelTransformer(new EdgeLabelTransformer());

         //Separamos la etiqueta de la arista
        _vv.getRenderContext().setLabelOffset(20);

        /Establecemos el color de las etiquetas
        DefaultEdgeLabelRenderer label = new DefaultEdgeLabelRenderer(Color.black);
              _vv.getRenderContext().setEdgeLabelRenderer(label);

        //Indicamos la posición de las etiquetas con respecto a las aristas (CENTRO)
        _vv.getRenderer().getVertexLabelRenderer().setPosition(Renderer.VertexLabel.Position.CNT
R);

        //Establecemos un tooltip para mostrar información sobre los vértices
         _vv.setVertexToolTipTransformer(new VertexToolTipTransformer());

        //Añadimos zoom al grafo
        DefaultModalGraphMouse graphMouse = new DefaultModalGraphMouse();
graphMouse.setMode(edu.uci.ics.jung.visualization.control.ModalGraphMouse.Mode.ANNOTATIN
G);
        _vv.setGraphMouse(graphMouse);



            //Paso 4: Incluimos el visualizador en un componente de Swing
            _pnGrafo.add(pane, BorderLayout.CENTER);
            _pnGrafo.setVisible(true);
```

3.3.4 Resaltado de un camino

En el punto anterior hemos hablado de que era necesario definir algunas clases transformadoras para modificar el color de los vértices y el grosor de las aristas cuando estos elementos son resaltados. Para poder resaltar un vértice o arista, en primer lugar, hay que utilizar los métodos getPickedEdgeState() y getPickedVertexState() para obtener un objeto con el estado de las aristas y vértices respectivamente. Estos métodos se encuentran disponibles en el paquete *edu.uci.ics.jung.visualization*. Una vez obtenido el estado de los elementos se utilizará el método pick(E arista, boolean estado) que resaltará la arista si estado = true y el método pick(V vértice, boolean estado) que resaltará el vértice si estado=true. En caso de que estado sea igual a false desmarcará el elemento.

A continuación se muestra el fragmento de código utilizado para resaltar los nodos y aristas en la aplicación desarrollada.

```
private void resaltarCaminoGrafo(List<Integer> camino) {

        //Resaltamos todos los nodos del camino excepto el último y todas las aristas
         for (int i = 0; i < camino.size() - 1; i++) {
                String s = _grafo.findEdge(camino.get(i), camino.get(i + 1));

                 if (s != null) {
                        //Resaltado de una arista
                        _vv.getPickedEdgeState().pick(s, true);
                }

                //Resaltado de un vértice
                 _vv.getPickedVertexState().pick(camino.get(i), true);
         }

        //Resaltamos el último nodo del camino
         _vv.getPickedVertexState().pick(camino.get(camino.size() - 1), true);

         //Volvemos a pintar el grafo
         _vv.repaint();
   }
```

Capítulo 4. Desarrollo del proyecto.

4.1 Descripción

Una vez presentados el propósito y los objetivos del proyecto y tras explicar las bases teóricas en las que se fundamenta, es el momento de entrar en profundidad en su desarrollo.

El proyecto aborda el desarrollo de una aplicación de escritorio con tecnología JAVA para el análisis de redes sociales, haciendo uso del paradigma de computación con palabras de Zadeh, basado en conjuntos difusos [2], y de la teoría de grafos. Con esto, se pretende facilitar al ser humano la comprensión de los conceptos matemáticos utilizados para llevar a cabo el análisis, estableciendo una correspondencia entre los mismos y los términos lingüísticos que habitualmente utilizan los seres humanos para comunicarse.

Una forma sencilla e intuitiva de visualizar las relaciones de una red social la proporcionan los grafos, ya que los nodos pueden representar a los individuos, y las aristas pueden reflejar la relación existente entre los mismos. Así, si existe relación entre un par de individuos, existirá una arista que una los nodos que representan a estos individuos. En este proyecto, haciendo uso de la propuesta de Ronald R. Yager [1], se utiliza un tipo de grafo conocido como grafo difuso [3], de forma que no sólo se muestra si existe relación o no entre dos individuos, sino que además se indica el peso de la relación. Para la representación del grafo obtenido a partir de la red social se ha utilizado la herramienta JUNG. El peso de la relación entre dos usuarios se puede calcular utilizando distintos criterios, tales como, número de mensajes privados enviados, número de amigos en común, número de menciones, etc.

Para poder mostrar la funcionalidad de la herramienta desarrollada se han utilizado datos relativos a los miembros de un Sistema de Recomendación de películas. El análisis se ha llevado a cabo teniendo en cuenta la confianza que tienen unos usuarios en otros, en función de las valoraciones de las películas que han realizado, tal y como se ha explicado en el capítulo “Análisis de redes sociales mediante grafos difusos y computación con palabras”, en la sección “Datos utilizados”.

Esta herramienta consta de tres componentes fundamentales en el desarrollo del proyecto:

- Una **base de datos** que incluye los datos relativos a los usuarios del Sistema de Recomendación, las películas valoradas, la confianza existente entre los usuarios, etc.

- La **librería JUNG** (Java Universal Network Graph) para la representación gráfica de la red social.

- Una **interfaz** a través de la cual se pueden analizar las propiedades de la red social representada.

Tras realizar una breve descripción del proyecto, en este capítulo, se detalla el proceso llevado a cabo para desarrollarlo. Como todo proyecto de desarrollo software debe seguir la metodología de la Ingeniería del Software, la cual se explicará a continuación.

La Ingeniería del Software consiste en el establecimiento y uso de principios y métodos firmes de ingeniería, para obtener software económico que sea fiable y que funcione de manera eficiente en máquinas reales. Pretende construir software de calidad con un presupuesto limitado y en un plazo de entrega en contextos de cambio continuo. Se compone de las siguientes etapas [15]:

- **Especificación de requerimientos:** se obtiene el propósito del sistema, así como las propiedades y restricciones impuestas sobre el mismo.

- **Análisis del sistema:** se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.

- **Diseño del sistema:** se establecen los objetivos del proyecto y las estrategias a seguir para conseguirlos.

- **Implementación:** consiste en la traducción del modelo lógico del sistema a código fuente.

- **Pruebas:** se lleva a cabo la verificación y validación del sistema.

En los siguientes puntos se profundizará en cada una de estas etapas y en cómo se han llevado a cabo durante el desarrollo del proyecto.

4.2 Especificación de requerimientos

El primer paso en el proceso de Ingeniería del Software debe ser determinar el propósito del proyecto, las propiedades que debe satisfacer y las restricciones a las que está sometido. Este es un paso muy importante en el desarrollo de un proyecto software, ya que sin conocer el propósito del proyecto ni las limitaciones a las que debe hacer frente, será realmente complicado realizar una aplicación software que cumpla dicho propósito.

Para determinar el propósito de un proyecto de ámbito comercial para una empresa se recurre a entrevistas con los clientes, observación en el puesto de trabajo, estudios de viabilidad, etc. En este caso, al tratarse de un proyecto académico, el propósito es conocido desde la concepción del mismo:

> *"Desarrollar una herramienta que permita analizar las relaciones existentes en las redes sociales, llevando a cabo una correspondencia entre la forma de comunicación de los seres humanos y la representación formal de una red."*

Una vez definido el propósito del proyecto, el siguiente paso es especificar los requerimientos del mismo. Los requerimientos de un proyecto software son el conjunto de propiedades o restricciones, definidas con total precisión, que dicho proyecto debe satisfacer. Describen los servicios que dicho software ha de ofrecer y las restricciones asociadas a su funcionamiento. Existen dos tipos diferenciados de tales requerimientos:

- **Requerimientos funcionales:** aquellos que se refieren específicamente al funcionamiento del sistema. Definen qué debe hacer el sistema.

- **Requerimientos no funcionales:** aquellos no referidos al estricto funcionamiento del sistema, sino a otros factores externos. Definen cómo debe ser el sistema.

En los dos epígrafes siguientes se definirán cuáles son estos requerimientos para el proyecto sobre el que trata esta memoria.

4.2.1 Requerimientos funcionales

Los requerimientos funcionales de un sistema software son aquellos que se encargan de describir las funcionalidades que el sistema debe proporcionar a los usuarios del mismo, para cubrir sus necesidades [16]. Describen las reacciones del sistema ante determinadas entradas y su comportamiento en diferentes situaciones.

Las funcionalidades que el sistema desarrollado ofrece, así como su forma de responder a dichas funcionalidades, se muestran a continuación. Para distinguir cada uno de los requisitos utilizaremos el identificador RF-Número_del_requisito.

1) **RF-01: Cargar un grafo.**
   El sistema permite cargar el grafo representativo de la red social que se va a analizar.

2) **RF-02: Definir el significado del término fuerte.**
   El sistema ofrece al usuario la posibilidad de definir la función de pertenencia del término fuerte, que permite determinar si una conexión es fuerte o no. Si los valores introducidos son correctos el sistema actualizará la definición de dicho término y, a partir de ese momento, todas las operaciones se realizarán en base a esa definición.

3) **RF-03: Obtener los líderes de la red social.**
   El sistema permite la obtención de los líderes de la red social que se está analizando. Para ello el usuario debe indicar el número de líderes que desea calcular, el número máximo de conexiones a tener en cuenta y el valor de fortaleza a partir del cual se tendrán en cuenta las conexiones.

4) **RF-04: Calcular el camino mínimo entre dos miembros de la red social.**
   El sistema proporciona al usuario un mecanismo para obtener el camino mínimo entre dos miembros de la red social.

5) **RF-05: Obtener el camino más fuerte que une a dos miembros de la red social.**
   El sistema permite calcular el camino más fuerte entre dos miembros de la red.

6) **RF-06: Calcular el camino mínimo más fuerte que existe entre dos miembros de la red social.**
   El sistema ofrece al usuario la posibilidad de obtener el camino mínimo más fuerte que une a dos miembros de la red, indicando, si lo desea, el número máximo de conexiones a tener en cuenta.

7) **RF-07: Obtener los posibles caminos existentes entre dos miembros de la red social.**
   El sistema permite obtener todos los caminos que conectan a dos miembros de la red ordenados de mayor a menor fortaleza. El usuario, si lo desea, puede indicar el número máximo de conexiones a tener en cuenta y el criterio de ordenación deseado. El sistema ofrece la posibilidad de ordenar estos caminos por fortaleza o por longitud, de mayor a menor o de menor a mayor.

8) **RF-08: Obtener información sobre los miembros.**
   El sistema permite obtener información sobre los miembros de la red social representados en el grafo.

9) **RF-09: Salir del sistema.**
   El sistema ofrece al usuario la posibilidad de salir de la aplicación en cualquier momento.

4.2.2 Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que describen restricciones que afectan a los requerimientos funcionales. Generalmente suelen especificar propiedades del sistema que tienen que ver con el rendimiento, velocidad, uso de memoria, plataforma, facilidad de uso, tolerancia a fallos, tiempo de respuesta, metáforas de interfaz, etc.

Estos requerimientos son tan importantes como los funcionales y pueden incluso llegar a ser críticos para la aceptación del sistema.

En este caso, al no tratarse de un proyecto de tipo empresarial o comercial, no existen restricciones organizacionales a las que someterse. Por ello, los requerimientos no funcionales que se abordarán serán los relativos al hardware y software de los equipos informáticos, para ofrecer al usuario las funcionalidades requeridas de forma eficiente, y los referentes a la interfaz gráfica entre el usuario y la aplicación.

4.2.2.1 Requerimientos del equipo informático

En este proyecto se va a desarrollar una aplicación de escritorio con tecnología JAVA y con acceso a una base de datos que trabaja a nivel local. Para que la aplicación funcione correctamente, los requerimientos del equipo informático en el que se vaya a utilizar son los siguientes:

a) Hardware

- Monitor: debe soportar una resolución de 1024x768 como mínimo para que el usuario pueda visualizar la información de forma correcta.

- Velocidad: el equipo debe ser lo suficientemente rápido para ejecutar las distintas funcionalidades de la aplicación. Cualquier microprocesador actual es capaz de cumplir con esta labor.

- Memoria: el equipo debe disponer de suficiente memoria RAM libre para realizar las operaciones que el usuario solicite a la aplicación.

- Almacenamiento: la capacidad de almacenamiento del equipo debe ser suficiente como para alojar la base de datos con la que trabaja la aplicación y permitir con holgura las transacciones entre ambas entidades.

b) Software

- Sistema operativo: Windows.

- Máquina Virtual de Java (Java Virtual Machine, JVM): debe disponer de ella para poder ejecutar la aplicación. La Máquina Virtual de Java es capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (bytecode Java), el cual es generado por el compilador del lenguaje Java. Se recomienda disponer de su última versión, JVM 7.

- Sistema de Gestión de Bases de Datos: es necesario que el equipo disponga del gestor de bases de datos MySQL, para el almacenamiento, modificación y extracción de información de la base de datos.

4.2.2.2 Requerimientos de la interfaz

Los requerimientos de la interfaz gráfica están estrechamente relacionados con la usabilidad y sus principios [17].

La usabilidad es la medida en que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico [18]. La efectividad es la precisión con la que los usuarios alcanzan los objetivos especificados. La eficiencia hace referencia a los recursos empleados (tiempo, etc.) que son necesarios para alcanzar con precisión esos objetivos. La satisfacción es la ausencia de incomodidad y la actitud positiva del usuario hacia el producto; es pues, un factor subjetivo.

Los principios generales que se pueden utilizar para aumentar la usabilidad de una interfaz se asociarán con los requerimientos no funcionales que debe cumplir la interfaz gráfica de la aplicación desarrollada, y son los siguientes:

1) **Facilidad de aprendizaje:** se refiere a las características de la interfaz que permiten a usuarios inexpertos comprender cómo utilizarla inicialmente y cómo alcanzar un alto nivel de productividad rápidamente. Para que un sistema sea fácil de aprender debe ser:

    - *Predecible:* el usuario debe poder predecir qué va a pasar tras una interacción. Para ello se utilizarán nombres claros e identificativos en las etiquetas y botones de la aplicación.

    - *Sintetizable:* los cambios resultantes de una interacción deben ser percibidos fácilmente por el usuario. Para ello se mostrará el cambio al usuario inmediatamente después de que se produzca.

    - *Familiar:* es importante diseñar el sistema de manera que los elementos de su interfaz resulten conocidos y familiares para el usuario.

    - *Consistente:* para tareas similares se mantendrá la misma estructura.

2) **Flexibilidad:** se proporcionará al usuario la capacidad de decidir cuándo iniciar y finalizar una interacción. Además, el sistema se encargará de localizar los errores y mostrárselos al usuario para que decida qué hacer con ellos.

3) **Robustez:** es el grado con el que el sistema es capaz de tolerar fallos. Está relacionada con los siguientes factores:

    - *Navegabilidad:* el usuario debe poder investigar el estado del sistema sin que ello repercuta de forma negativa en él.

    - *Persistencia:* está relacionada con la duración de un acto de comunicación y la capacidad del usuario para hacer uso de esa información. Cada vez que el usuario

realice una operación sobre la interfaz se mostrará el resultado de forma gráfica y de forma textual, de manera que pueda revisar esta información posteriormente.

- *Tiempo de respuesta:* es el tiempo que necesita el sistema para mostrar los cambios producidos tras una interacción del usuario. Se pretende que la respuesta sea prácticamente inmediata o que el tiempo de espera sea corto.

4.3 Análisis del sistema

Una vez conocido el propósito del proyecto, las propiedades que debe cumplir y las restricciones a las que debe someterse, es el momento de analizar el sistema con el objetivo de obtener un modelo del mismo que sea correcto, completo, consistente, claro y verificable. Para ello, se definirán casos de uso en base a los requerimientos previamente obtenidos y se describirán mediante el uso de narrativas.

4.3.1 Diagramas de casos de uso

Los diagramas de casos de uso sirven para mostrar las funciones de un sistema software desde el punto de vista de sus interacciones con el exterior, sin entrar ni en la descripción detallada ni en la implementación de estas funciones [19]. Los elementos que forman parte de un diagrama de este tipo son:

- **Actor:** rol que un usuario desempeña en el sistema. Modela una entidad externa que se comunica con el sistema, no tiene por qué ser una persona puede ser otro sistema. Debe tener un nombre único y puede tener una descripción asociada.

- **Caso de uso:** representa una funcionalidad del sistema. Se trata de una tarea específica que se realiza tras la orden de un agente externo, bien por la petición de un actor o por la invocación desde otro caso de uso. Se representa mediante una elipse y un texto identificativo. Para su descripción se utilizan las narrativas del sistema.

- **Relaciones:** reflejan las relaciones existentes entre actores y casos de uso, entre actores, o entre casos de uso. Se distinguen cuatro tipos de relaciones:

  - *Asociación:* es el tipo de relación más básico. Indica la invocación por parte de un actor a un caso de uso.

  - *Inclusión:* un caso de uso A está incluido dentro de los casos de uso B, C, etc., si es una parte de proceso común a todos estos [18]. Se utiliza para evitar repetir el

mismo flujo de eventos repetidas veces. Se denota con una relación que parte del caso que incluye (casos B, C, etc.) y apunta al caso incluido (caso A).

- *Extensión:* se dice que el caso de uso A extiende el B si dentro de B se ejecuta A cuando se cumple una condición determinada. A tiene que ser una caso de uso que se pueda ejecutar de forma separada de B, y debe tener el mismo actor primario que éste. Se denota con una relación que parte del caso que extiende (caso A) y apunta al caso base (caso B).

- *Generalización:* un caso de uso A es una especialización de otro caso de uso B si A realiza todo el proceso de B, más algún proceso específico. El caso de uso hijo (caso A) hereda la especificación del caso de uso padre (caso B) y además puede añadir información o redefinir el comportamiento del padre. Puede ser colocado en cualquier lugar donde aparezca el padre.

Una vez definidos los elementos fundamentales de un diagrama de casos de uso, es el momento de crear los distintos casos de uso. A la hora de realizar esta acción es importante que cada uno de los requerimientos funcionales previamente definidos, aparezca en al menos uno de los casos de uso. Por otro lado, puede haber casos de uso nuevos, en los que no aparezca ninguno de los requerimientos, ya que nos encontramos en una fase de refinamiento del sistema en la que se pretende construir un modelo detallado del mismo.

Antes de describir los distintos casos de uso mediante las narrativas asociadas, es necesario obtener los diagramas de casos de uso de nuestro sistema. El primero de ellos es un diagrama frontera. Se conoce como diagrama frontera a aquel que incluye todos los casos de uso genéricos del sistema, que podrán ser desglosados posteriormente en nuevos diagramas de casos de uso que los describan. En la Ilustración 14 se puede observar el diagrama frontera de nuestro sistema.

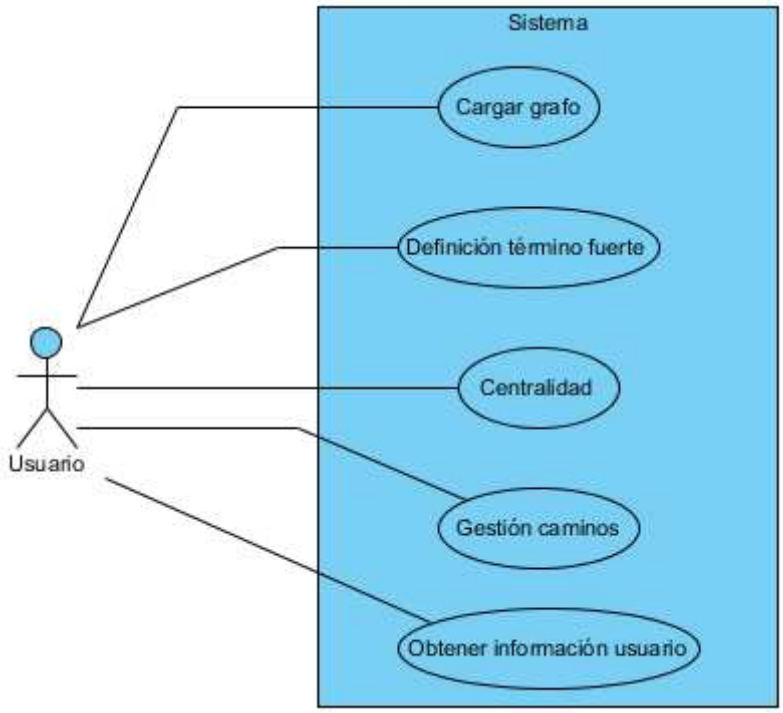

Ilustración 14: Diagrama frontera del sistema

Los casos de uso reflejados en un diagrama frontera pueden ser lo suficientemente precisos o, por el contrario, pueden necesitar ser explicados con mayor detalle. En el caso de nuestro sistema, el caso de uso "Gestión caminos" debe ser explicado con mayor profundidad. Para ello, se utilizará un diagrama de casos de uso que describirá su funcionalidad de forma más detallada (ver Ilustración 15).

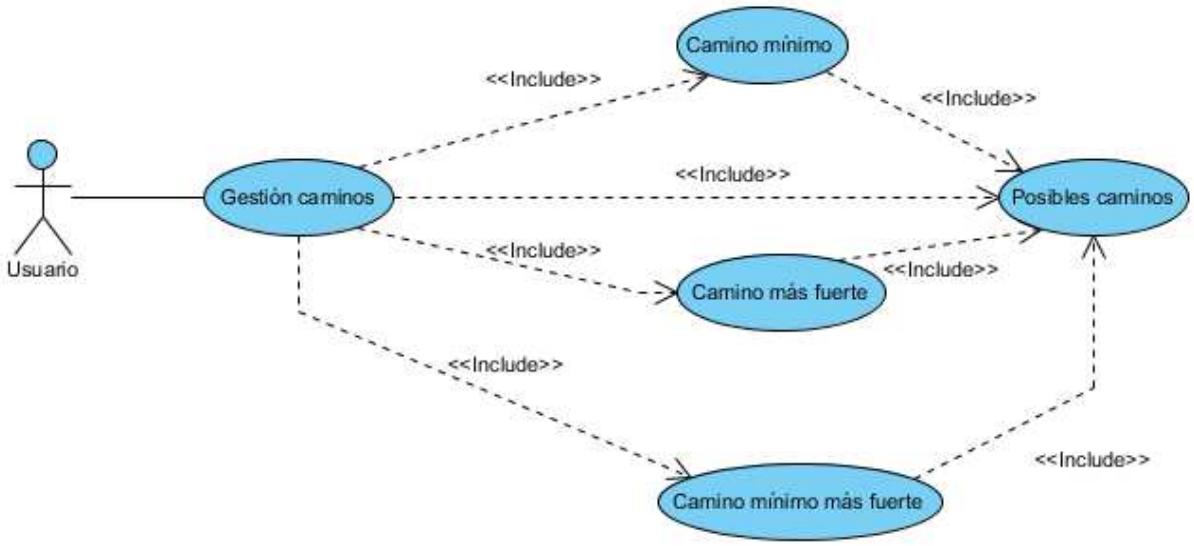

Ilustración 15: Diagrama del caso de uso "Gestión caminos"

4.3.2 Narrativas de casos de uso

Un caso de uso es una representación abstracta de una funcionalidad del sistema. La representación concreta de un caso de uso se realiza mediante la creación de uno o más escenarios que muestren todas las interacciones posibles entre el sistema y sus usuarios. Para ello se utilizan las narrativas, que se encargan de explicar detalladamente qué es necesario y qué sucede en cada caso de uso. En ellas se indica cómo actúan los actores con el sistema en cada caso particular. Están formadas por los siguientes elementos:

- **Título:** nombre del caso de uso.
- **Actores:** individuos que intervienen en el caso de uso.
- **Precondiciones:** condiciones que se deben cumplir para que se dispare o pueda utilizarse este caso de uso.
- **Escenario principal:** describe el flujo normal de eventos.
- **Escenarios alternativos:** describen situaciones concretas en las que el caso de uso no sigue el flujo normal de eventos.
- **Postcondiciones:** resultado final que se obtiene tras la ejecución de la secuencia de acciones.

A continuación, se describen de forma detallada cada uno de los casos de uso de nuestro sistema mediante la utilización de narrativas.

- **Caso de uso 1: Cargar grafo.**

Título	**Cargar grafo**	
Actores	Usuario.	
Precondiciones		
Escenario principal		**Escenarios alternativos**
1. El usuario selecciona el grafo que desea analizar y pulsa Aceptar.		
2. El sistema muestra el grafo representativo de la red social a analizar.		
Postcondiciones	El sistema muestra el grafo representativo de la red social seleccionada y las distintas opciones de análisis que se pueden realizar sobre la misma.	

Tabla 1: Narrativa "Cargar grafo"

- **Caso de uso 2: Definición término fuerte.**

Título	**Definición término fuerte**	
Actores	Usuario.	
Precondiciones	El usuario debe haber cargado algún grafo.	
Escenario principal		**Escenarios alternativos**
1. El usuario selecciona la opción “Definición término fuerte”.		
2. El sistema muestra el valor actual y solicita los nuevos valores para *alfa* y *beta* respectivamente		
3. El usuario introduce los nuevos valores para alfa y beta y pulsa en Definir.		
4a. El sistema comprueba que los datos son válidos y actualiza la definición del término fuerte.		4b. Si el valor de *alfa* y/o el de *beta* no son correctos, el sistema informa de ello al usuario y vuelve al paso 2.
Postcondiciones	La nueva definición del término fuerte queda almacenada en el sistema. Dicha información se muestra tanto en el panel “Definición término fuerte” como en el panel de salida de texto.	

Tabla 2: Narrativa “Definición término fuerte”

- **Caso de uso 3: Centralidad.**

Título	**Centralidad**	
Actores	Usuario.	
Precondiciones	El usuario debe haber cargado algún grafo.	
Escenario principal		**Escenarios alternativos**
1. El usuario selecciona la opción “Centralidad”.		
2. El sistema solicita que se indique el número *máximo de conexiones* a tener en cuenta, el *mínimo valor de fortaleza* que deben tener esas conexiones y el número de *líderes* a calcular.		
3. El usuario introduce los valores indicados y pulsa en *Calcular*.		
4a. El sistema comprueba que los datos son válidos y muestra los líderes de la red social en función del número indicado.		4b. Si alguno de los valores introducidos no es correcto, el sistema informa de ello al usuario y vuelve al paso 2.
Postcondiciones	Los líderes obtenidos se resaltan en el grafo y se muestran sus identificadores en el panel de salida de texto.	

Tabla 3: Narrativa "Centralidad"

- **Caso de uso 4: Gestión caminos.**

Título	**Gestión caminos**	
Actores	Usuario.	
Precondiciones	El usuario debe haber cargado algún grafo.	
Escenario principal		**Escenarios alternativos**
1. El usuario selecciona la opción “Camino mínimo”, “Camino más fuerte”, “Camino mínimo más fuerte” o “Posibles caminos”. 1.1. Si el usuario elige “Camino mínimo” se realiza el subcaso S4.1 (ver Tabla 5). 1.2. Si el usuario elige “Camino más fuerte” se realiza el subcaso S4.2 (ver Tabla 6). 1.3. Si el usuario elige “Camino mínimo más fuerte” se realiza el subcaso S4.3 (ver Tabla 7). 1.1. Si el usuario elige “Posibles caminos” se realiza el subcaso S4.4 (ver Tabla 8).		
Postcondiciones		

Tabla 4: Narrativa "Gestión caminos"

- *S4.1: Camino mínimo.*

<table>
<tr><th>Título</th><th colspan="2">Camino mínimo</th></tr>
<tr><td>Actores</td><td colspan="2">Usuario.</td></tr>
<tr><td>Precondiciones</td><td colspan="2">El usuario debe haber cargado algún grafo.</td></tr>
<tr><th colspan="2">Escenario principal</th><th>Escenarios alternativos</th></tr>
<tr><td colspan="2">1. El usuario selecciona la opción “Camino mínimo”.</td><td></td></tr>
<tr><td colspan="2">2. El sistema solicita el inicio y el fin del camino a calcular.</td><td></td></tr>
<tr><td colspan="2">3. El usuario introduce en inicio y fin los identificadores de los miembros para los cuales desea calcular el camino mínimo que los une.</td><td></td></tr>
<tr><td colspan="2">4a. El sistema comprueba que los identificadores introducidos se corresponden con miembros de la red social y muestra el camino mínimo.</td><td>4b. Si alguno de los identificadores introducidos no es correcto, el sistema informa de ello al usuario y vuelve al paso 2.</td></tr>
<tr><td>Postcondiciones</td><td colspan="2">El camino mínimo que une a los usuarios indicados se resalta en el grafo. Además, este camino se muestra en el panel de salida de texto.</td></tr>
</table>

Tabla 5: Narrativa "Camino mínimo"

- *S4.2: Camino más fuerte.*

Título	**Camino más fuerte**	
Actores	Usuario.	
Precondiciones	El usuario debe haber cargado algún grafo.	
Escenario principal		**Escenarios alternativos**
1. El usuario selecciona la opción “Camino más fuerte”.		
2. El sistema solicita el *inicio* y el *fin* del camino a calcular.		
3. El usuario introduce en *inicio* y *fin* los identificadores de los usuarios para los cuales desea calcular el camino más fuerte existente entre ellos.		
4a. El sistema comprueba que los identificadores introducidos se corresponden con miembros de la red social y muestra el camino mínimo.		4b. Si alguno de los identificadores introducidos no es correcto, el sistema informa de ello al usuario y vuelve al paso 2.
Postcondiciones	El camino más fuerte que une a los usuarios indicados se resalta en el grafo y se muestra dicho camino en el panel de salida de texto.	

Tabla 6: Narrativa "Camino más fuerte"

- *S4.3: Camino mínimo más fuerte.*

<table>
<tr><th>Título</th><th colspan="2">Camino mínimo más fuerte</th></tr>
<tr><td>Actores</td><td colspan="2">Usuario.</td></tr>
<tr><td>Precondiciones</td><td colspan="2">El usuario debe haber cargado algún grafo.</td></tr>
<tr><th colspan="2">Escenario principal</th><th>Escenarios alternativos</th></tr>
<tr><td colspan="2">1. El usuario selecciona la opción “Camino mínimo más fuerte”.</td><td></td></tr>
<tr><td colspan="2">2. El sistema solicita el inicio y el fin del camino a calcular y el número máximo de conexiones a tener en cuenta.</td><td></td></tr>
<tr><td colspan="2">3. El usuario introduce en inicio y fin los identificadores de los usuarios para los cuales desea calcular el camino mínimo más fuerte e indica el número máximo de conexiones a tener en cuenta.</td><td></td></tr>
<tr><td colspan="2">4a. El sistema comprueba que los identificadores introducidos se corresponden con miembros de la red social y que el valor indicado en número máximo de conexiones es correcto, y muestra el camino mínimo.</td><td>4b. Si alguno de los valores introducidos no es correcto el sistema informa de ello al usuario y vuelve al paso 2.</td></tr>
<tr><td>Postcondiciones</td><td colspan="2">El camino mínimo más fuerte que une a los usuarios indicados, teniendo en cuento el número máximo de conexiones permitido, se resalta en el grafo. Además, este camino se muestra en el panel de salida de texto.</td></tr>
</table>

Tabla 7: Narrativa "Camino mínimo más fuerte"

- *S4.4: Posibles caminos.*

Título	**Posibles caminos**	
Actores	Usuario.	
Precondiciones	El usuario debe haber cargado algún grafo.	
Escenario principal		**Escenarios alternativos**
1. El usuario selecciona la opción “Posibles caminos”.		
2. El sistema solicita el *inicio* y el *fin*, el número *máximo de conexiones* a tener en cuenta y el criterio de ordenación de los posibles caminos.		
3. El usuario introduce en *inicio* y *fin* los identificadores de los usuarios para los cuales desea calcular los posibles caminos que los unen e indica el número máximo de conexiones a tener en cuenta y el criterio de ordenación.		
4a. El sistema comprueba que los identificadores introducidos se corresponden con miembros de la red social y que el número máximo de conexiones indicado es correcto y muestra los posibles caminos.		4b. Si alguno de los identificadores introducidos no es correcto o el número máximo de conexiones no es válido, el sistema informa de ello al usuario y vuelve al paso 2.
Postcondiciones	Los posibles caminos que unen a los usuarios indicados se muestran en el panel de salida de texto, ordenados según el criterio establecido.	

Tabla 8: Narrativa "Posibles caminos"

- **Caso de uso 5: Obtener información usuario.**

Título	**Obtener información usuario**	
Actores	Usuario.	
Precondiciones	El usuario debe haber cargado algún grafo.	
Escenario principal		**Escenarios alternativos**
1. El usuario posiciona el cursor del ratón sobre uno de los nodos.		
2. El sistema muestra la información relativa al usuario representado por dicho nodo.		
Postcondiciones	El sistema muestra información sobre el usuario cuyo identificador se corresponde con el del nodo sobre el que se encuentra el cursor del ratón.	

Tabla 9: Narrativa "Obtener información usuario"

4.3.3 Diagramas de secuencia del sistema

Los diagramas de secuencia del sistema permiten realizar una descripción gráfica de los distintos casos de uso. Un diagrama de secuencia del sistema es un dibujo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas.

Para detallar los casos de uso más complejos del sistema se van a utilizar estos diagramas, ya que muestran de forma clara la secuencia de intercambio de mensaje en el tiempo.

Para poder visualizar los diagramas de secuencia del sistema, se han eliminado de los mismos las operaciones y los mensajes de retorno, ya que con ellos alcanzaban una gran dimensión y resultaban prácticamente ilegibles. Para solucionar este problema, debajo de cada diagrama se mostrarán, siguiendo la secuencia de intercambio, las operaciones en cursiva y los valores de retorno en color gris.

DSS Cargar grafo

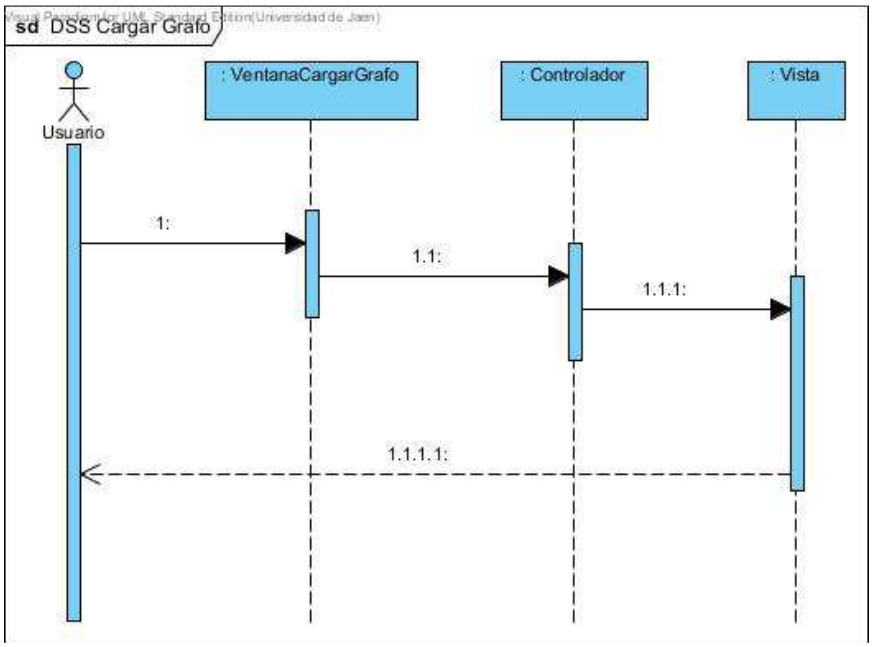

Ilustración 16: DSS "Cargar grafo"

1: cargarGrafo(grafo)

1.1: peticionCargarGrafo(grafo)

1.1.1: mostrarGrafo(grafo)

1.1.1.1: ok()

DSS Definición término fuerte

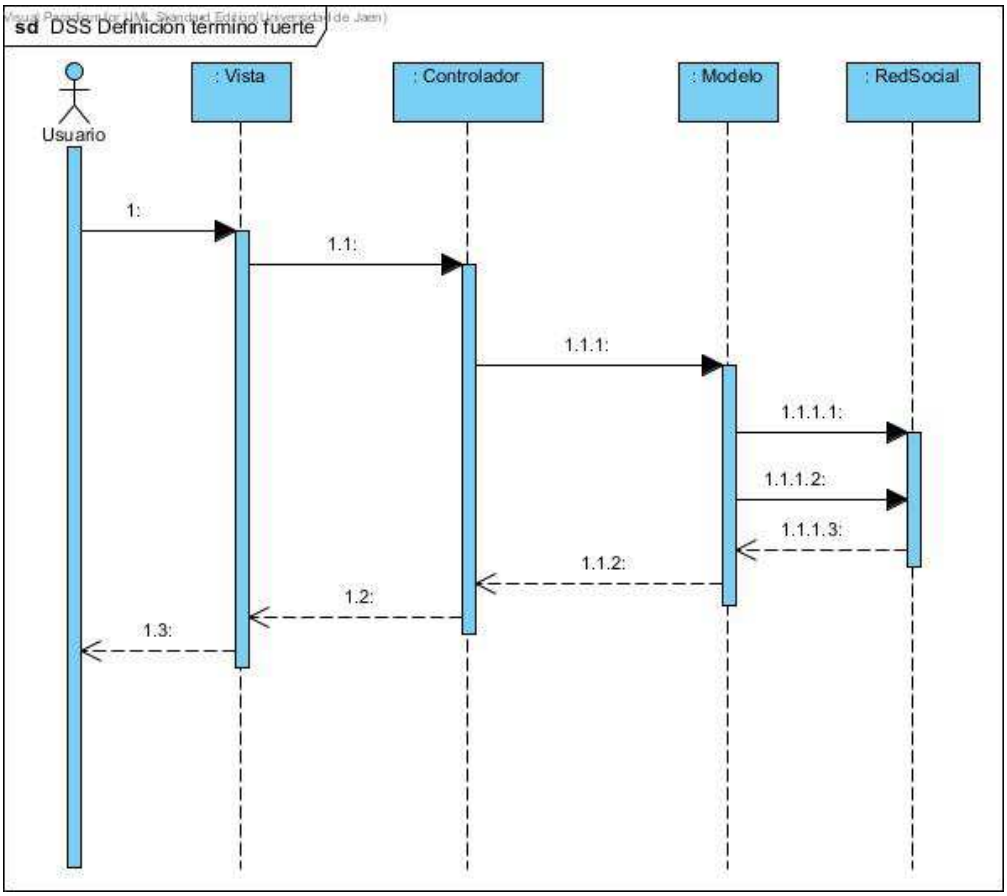

Ilustración 17: DSS "Definición término fuerte"

1: definirTerminoFuerte(alfa, beta)

1.1: peticionDefinirTerminoFuerte(alfa, beta)

1.1.1: definirTerminoFuerte(alfa, beta)

1.1.1.1: setAlfa(alfa)

1.1.1.2: setBeta(beta)

1.1.1.3: ok()

1.1.2: ok()

1.2: ok()

1.3: ok()

DSS Centralidad

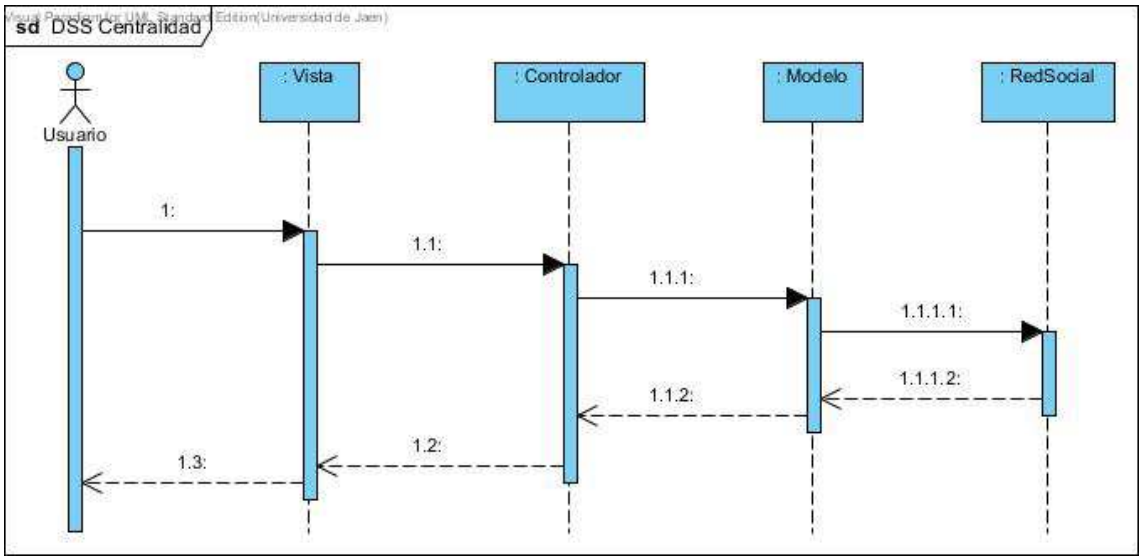

Ilustración 18: DSS "Centralidad"

1: obtenerLideres(maxConexiones, minFortaleza, numLideres)

1.1: peticionObtenerLideres(maxConexiones, minFortaleza, numLideres)

1.1.1: obtenerLideres(maxConexiones, minFortaleza, numLideres)

1.1.1.1: calcularCentralidadNodos()

1.1.1.2: centralidad nodos

1.1.2: líderes

1.2: líderes

1.3: líderes

DSS Camino mínimo

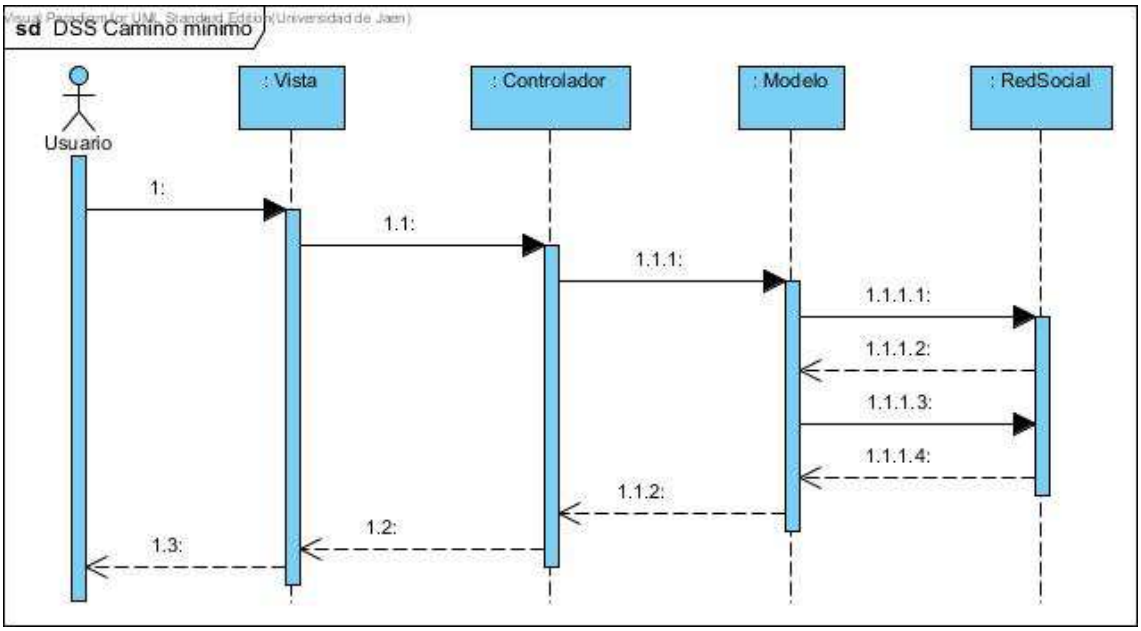

Ilustración 19: DSS "Camino mínimo"

1: calcularCaminoMin(inicio, fin)

1.1: peticionCalcularCaminoMin(inicio, fin)

1.1.1: calcularCaminoMin(inicio, fin)

1.1.1.1: calcularPosiblesCaminos(inicio, fin)

1.1.1.2: posiblesCaminos

*1.1.1.3: *[para cada posibleCamino] calcularLongitud()*

1.1.1.4: longitudposiblesCaminos

1.1.2: caminoMinimo

1.2: caminoMinimo

1.3: caminoMinimo

DSS Camino más fuerte

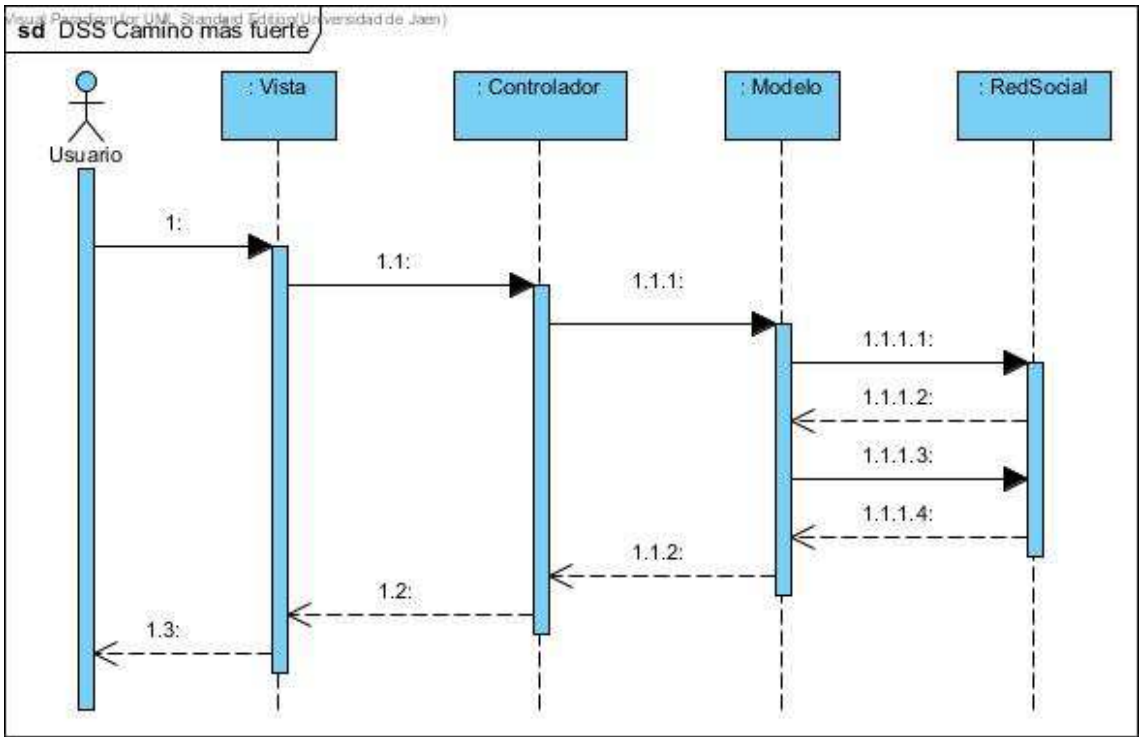

Ilustración 20: DSS "Camino más fuerte"

1: calcularCaminoMasFuerte(inicio, fin)

1.1: peticionCalcualrCaminoMasFuerte(inicio, fin)

1.1.1: calcularCaminoMasFuerte(inicio, fin)

1.1.1.1: calcularPosiblesCaminos(inicio, fin)

1.1.1.2: posiblesCaminos

*1.1.1.3: *[para cada posibleCamino] calcularFortaleza()*

1.1.1.4: fortalezaPosiblesCaminos

1.1.2: caminoMasFuerte

1.2: caminoMasFuerte

1.3. caminoMasFuerte

DSS Camino mínimo más fuerte

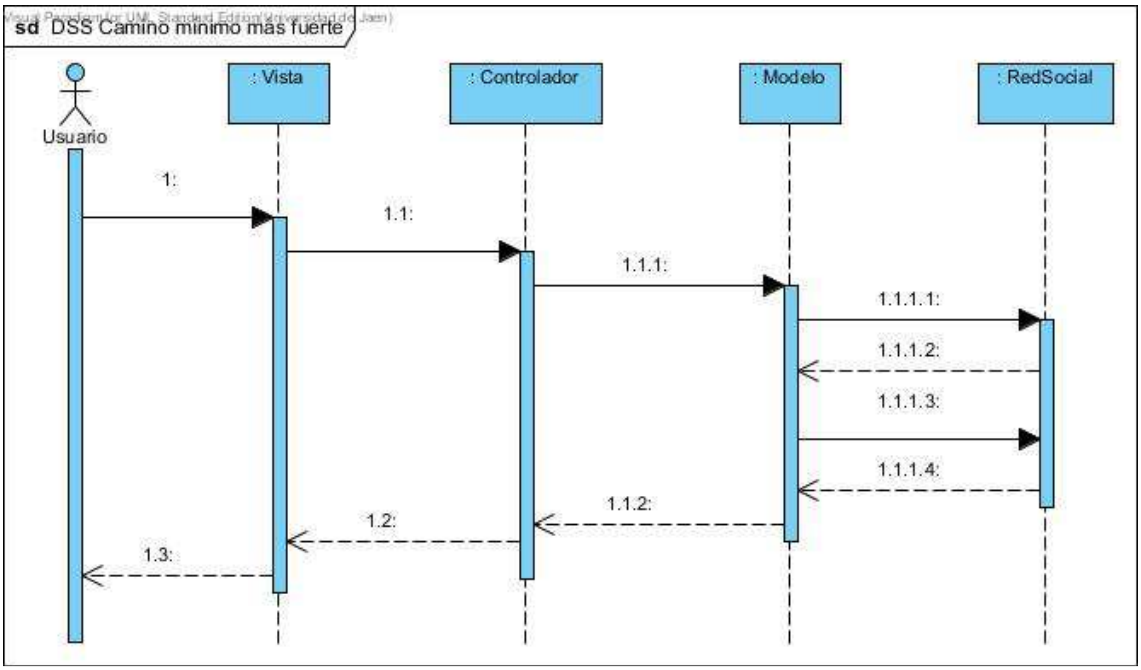

Ilustración 21: DSS "Camino mínimo más fuerte"

1: calcularCaminoMinMasFuerte(inicio, fin, maxConexiones)

1.1: peticionCalcularCaminoMinMasFuerte(inicio, fin, maxConexiones)

1.1.1: calcularCaminoMinMasFuerte(inicio, fin, maxConexiones)

1.1.1.1: posiblesCaminosKLinks(inicio, fin, maxConexiones)

1.1.1.2: posiblesCaminos

*1.1.1.3: *[para cada posibleCamino] calcularGradoMinimoYFuerte()*

1.1.1.4: gradoPosiblesCaminos

1.1.2. caminoMinimoMasFuerte

1.2: caminoMinimoMasFuerte

1.3. caminoMinimoMasFuerte

DSS Posibles caminos

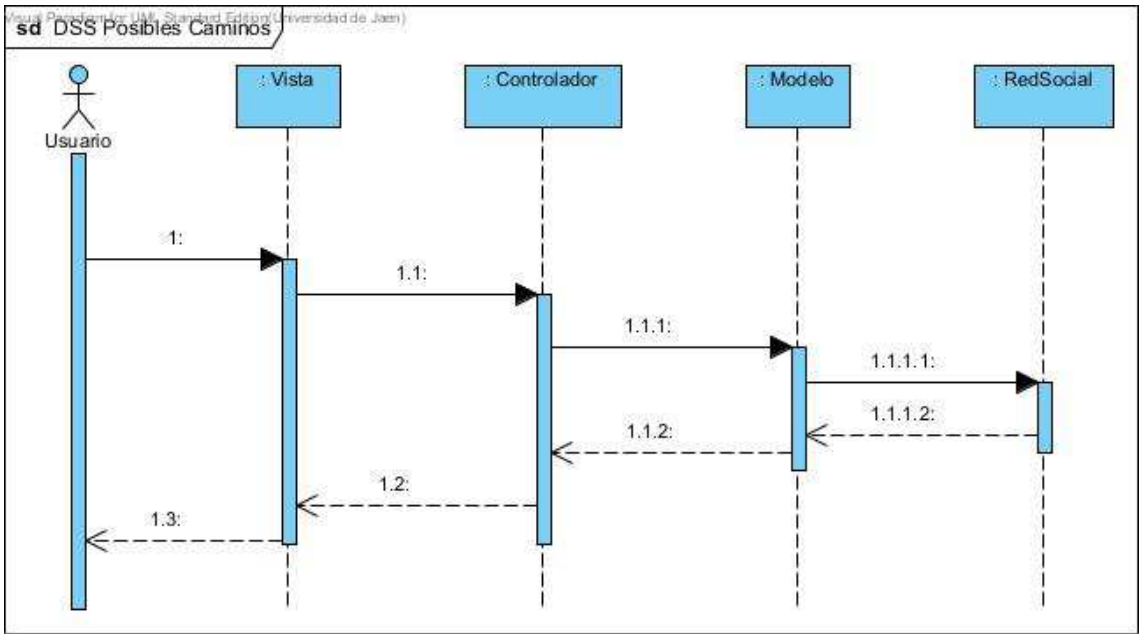

Ilustración 22: DSS "Posibles caminos"

1: calcularPosiblesCaminos(inicio, fin, maxConexiones, criterio)

1.1: peticionCalcularPosiblesCaminos(inicio, fin, maxConexiones, criterio)

1.1.1: calcularPosiblesCaminos(inicio, fin, maxConexiones, criterio)

1.1.1.1: posiblesCaminosKLinks(inicio, fin, maxConexiones)

1.1.1.2: posiblesCaminos

1.1.2: posiblesCaminos ordenados según criterio

1.2: posiblesCaminos ordenados según criterio

1.3: posiblesCaminos ordenados según criterio

DSS Obtener información usuario

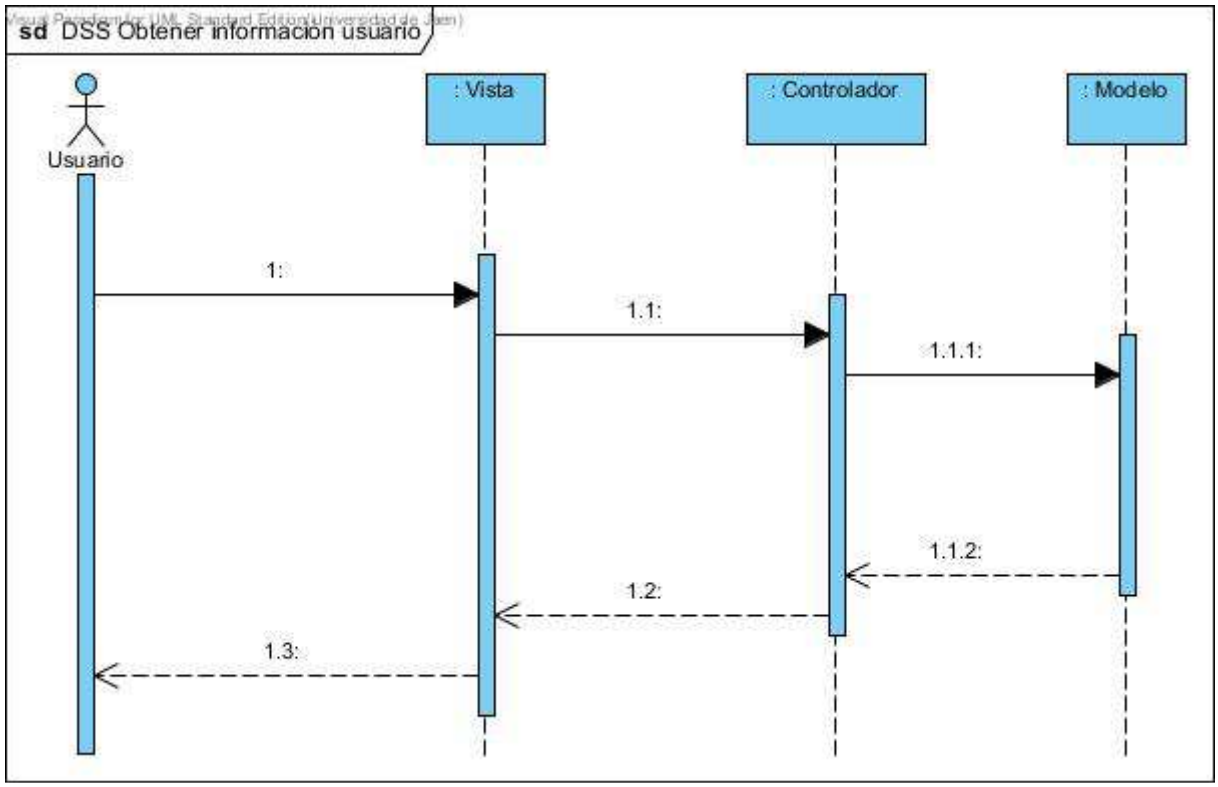

Ilustración 23: DSS "Obtener información usuario"

1: obtenerInformacionUsuario(idUsuario)

1.1: peticionObtenerInformacionUsuario(idUsuario)

1.1.1: obtenerInformacionUsuario(idUsuario)

1.1.2. informacionUsuario

1.2: informacionUsuario

1.3. informacionUsuario

4.4 Diseño del sistema

Tras analizar y comprender las características que debe poseer el sistema, es el momento de diseñarlo. Mientras que en la etapa de análisis se trata de entender la problemática a resolver mediante el análisis de los requerimientos del sistema, en la etapa de diseño se determina la estrategia que se va a utilizar para implementar el sistema de manera efectiva.

Todas las fases del proceso de Ingeniería del Software son importantes para el desarrollo y consecución de un proyecto software de calidad. Sin embargo, podemos decir que la fase de diseño es una de las más delicadas y complejas ya que si no se lleva a cabo correctamente puede resultar complicado codificar de forma correcta el modelo obtenido en el análisis del sistema.

Por tanto, se puede decir que el diseño del sistema es la actividad de la Ingeniería del Software en la que se identifican los objetivos finales del sistema y se plantean las diversas estrategias para alcanzarlos en la actividad de implementación [20].

El sistema no se suele diseñar de una sola vez, sino que hay que diferenciar entre su estructura, el diseño y estructura de los datos que va a manejar, y el diseño de la interfaz entre la aplicación y el usuario. Por ello, las fases de diseño que se van a llevar a cabo en este proyecto son:

- **Diagrama de clases de diseño:** muestra la estructura del sistema.
- **Diseño de los datos:** determina la estructura de los elementos de información del sistema.
- **Diseño de la interfaz:** define la apariencia visual de la aplicación.

4.4.1 Diagrama de clases de diseño

El diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Es una herramienta de diseño potente, ya que ayuda a los desarrolladores a planificar y establecer la estructura del sistema antes de escribir el código. Los elementos que pueden aparecer en este diagrama son:

- **Clases:** son los componentes fundamentales de estos diagramas. Una clase es unidad básica que encapsula toda la información de un conjunto de objetos que tienen el mismo comportamiento. Se representa mediante un rectángulo dividido en tres secciones, mostrando en el primero el nombre de la clase, en el segundo sus atributos y en el último las operaciones, tal y como se muestra en la Ilustración 24.

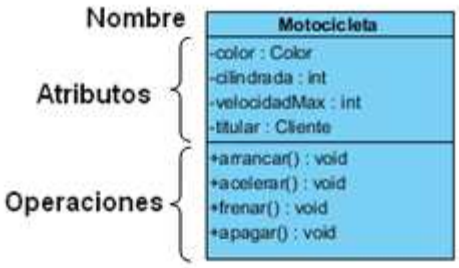

Ilustración 24: Ejemplo de clase

- **Atributos:** son los elementos que caracterizan a una clase. Pueden ser de tres tipos:

  - *Public (+):* son accesibles tanto dentro como fuera de la clase.

  - *Private (-):* sólo se puede acceder desde la clase a la que pertenecen.

  - *Protected (#):* se puede acceder desde la clase a la que pertenecen o desde las clases que heredan de ésta.

- **Operaciones:** indican la forma en que una clase interactúa con el entorno. Al igual que los atributos pueden ser de tres tipos:

  - *Public (+):* si el método es accesible tanto dentro como fuera de la clase.

  - *Private (-):* si el método sólo es accesible desde dentro de la clase.

  - *Protected (#):* si el método es accesible desde la propia clase y desde las clases que heredan de ella.

- **Relaciones:** muestran cómo se interrelacionan dos o más clases. Existen cinco tipos fundamentales de relaciones:

  - *Asociación:* es la relación más importante y común. Refleja una relación entre dos clases independientes que se mantiene durante la vida de los objetos de dichas clases o al menos durante un tiempo prolongado. Una clase A tiene una relación de asociación con una clase B, si en la clase A existe un atributo de la clase B. En la Ilustración 25 se puede observar que la clase Cuenta tiene una relación de asociación con la clase Cliente ya que contiene un atributo que es del tipo Cliente. La navegabilidad muestra dónde está ubicado el atributo y la multiplicidad indica si se trata de una colección o de un sólo elemento.

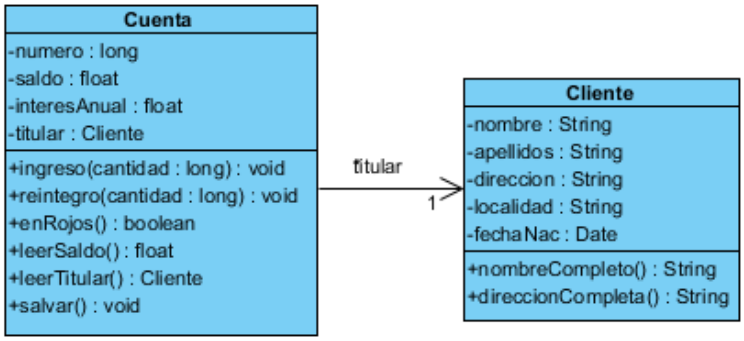

Ilustración 25: Asociación

- *Agregación:* es un tipo especial de asociación donde se añade el matiz semántico de que la clase de donde parte la relación representa el "todo" y las clases relacionadas las "partes". Ambas clases tienen sentido por sí solas y su tiempo de vida es independiente.

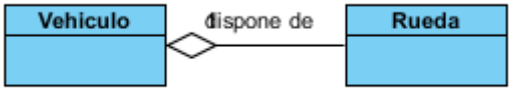

Ilustración 26: Agregación

Entre la clase Vehículo y la clase Rueda existe una relación de agregación ya que aunque el vehículo y la rueda son elementos independientes, la funcionalidad de uno depende del otro (ver Ilustración 26).

- *Composición:* es un tipo de agregación que añade el matiz de que la clase "todo" controla la existencia de las clases "parte". Es decir, la clase "todo" se encargará de la destrucción de las clases "parte". Por ejemplo, entre la clase Libro y la clase Capítulo existe una relación de composición ya que un capítulo sin libro no tiene ningún sentido (ver Ilustración 27). En este caso el Libro sería el "todo" y los capítulos las "partes".

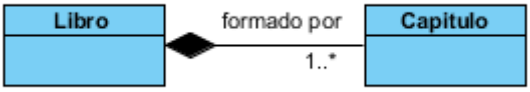

Ilustración 27: Composición

- *Generalización:* también conocida como herencia, es una relación de especialización o extensión en la que una clase adquiere las propiedades y métodos de una clase padre.

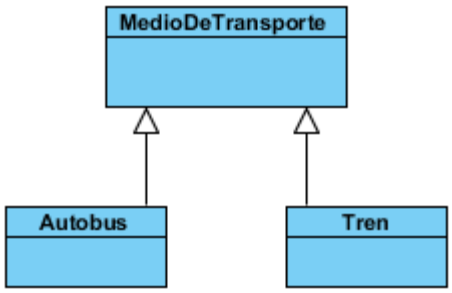

Ilustración 28: Generalización

Un Autobús y un Tren son medios de transporte por lo que heredarán los atributos y operaciones de la clase MedioDeTransporte (ver Ilustración 28).

- *Dependencia:* es una relación de uso. Refleja que la implementación de una clase depende de otra, es decir, puede indicar que un objeto de una clase se utiliza como argumento de una operación en otra clase o en su implementación. Como se puede ver en la Ilustración 29, la clase Cuenta requiere las clases FileOutputStream y ObjectOutputStream de la librería de clases Java para la implementación de la operación salvar.

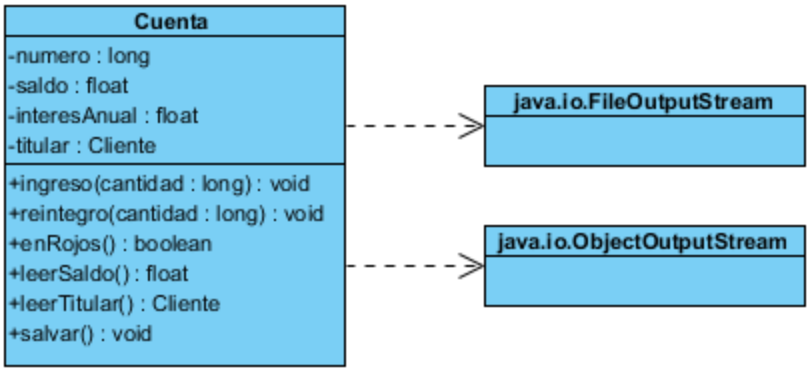

Ilustración 29: Dependencia

Una vez descritos los elementos de un diagrama de clases es el momento de presentar el de nuestro sistema. La comunicación entre las distintas partes del sistema se va a realizar siguiendo el patrón Modelo-Vista-Controlador (ver Ilustración 30). Es un patrón para el desarrollo de software que se basa en separar los datos, la interfaz de usuario y la lógica interna [18]. A continuación se detalla cada uno de estos elementos:

- **Modelo:** implementa la lógica de la aplicación. Su responsabilidad es almacenar los datos que maneja la aplicación (datos persistentes) y procesarlos.

- **Vista:** su función es mostrar la interfaz de usuario y recoger las acciones que el usuario realiza sobre la interfaz.

- **Controlador:** actúa de intermediario entre la vista y el modelo. Toma las acciones del usuario y las traduce a órdenes para el modelo. Además, maneja los datos temporales de la aplicación.

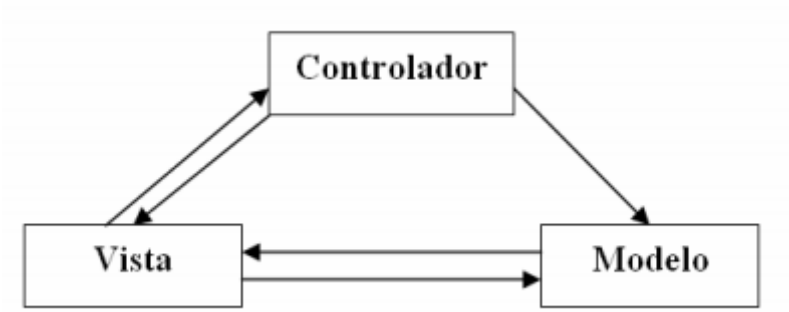

Ilustración 30: Esquema de la arquitectura MVC

El objetivo de esta arquitectura es mantener al modelo lo más independiente posible de la vista y del controlador, de manera que un cambio en uno de ellos no implique la necesidad de realizar una modificación en los demás componentes. Es decir, se pretende que las responsabilidades de cada uno de estos elementos estén perfectamente definidas y separadas para que el acoplamiento sea mínimo.

4.4.1.1 Diagrama completo

A continuación, se muestra el diagrama de clases de diseño del sistema desarrollado (ver Ilustración 31).

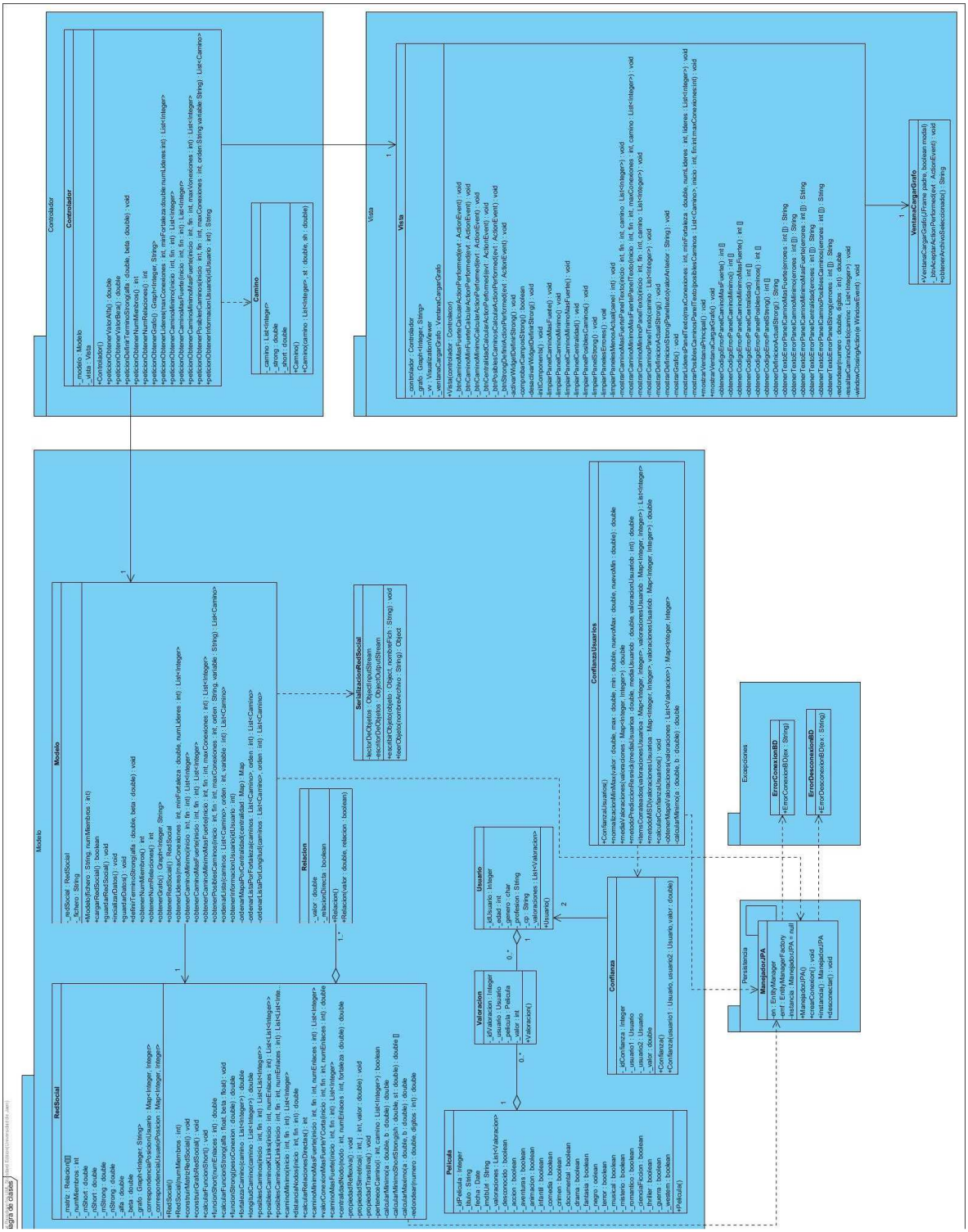

Ilustración 31: Diagrama de clases

En la Ilustración 31 se puede observar que el diagrama de clases del sistema se divide en cinco paquetes:

- **Paquete Modelo:** implementa la lógica de la aplicación, es decir, almacena los datos y el estado de la aplicación, y tiene los métodos necesarios para manipular esos datos.

- **Paquete Vista:** se encarga de mostrar la interfaz de usuario y de recoger las acciones que éste realiza sobre la misma.

- **Paquete Controlador:** actúa de intermediario entre el modelo y la vista. Posee los métodos necesarios para comunicarse con el modelo con el fin de obtener y guardar información tras la interacción del usuario con la interfaz.

- **Paquete Persistencia:** contiene los métodos y atributos necesarios para trabajar con JPA (Java Persistence API), de manera que se pueda consultar, insertar y modificar información de la base de datos.

- **Paquete Excepciones:** utilizado para controlar las excepciones producidas en la conexión/desconexión de la base de datos.

4.4.1.2 Diagramas por paquetes

Para poder visualizar mejor las clases que forman parte de cada uno de estos elementos, se muestran los paquetes concretos.

4.4.1.2.1 Paquete modelo

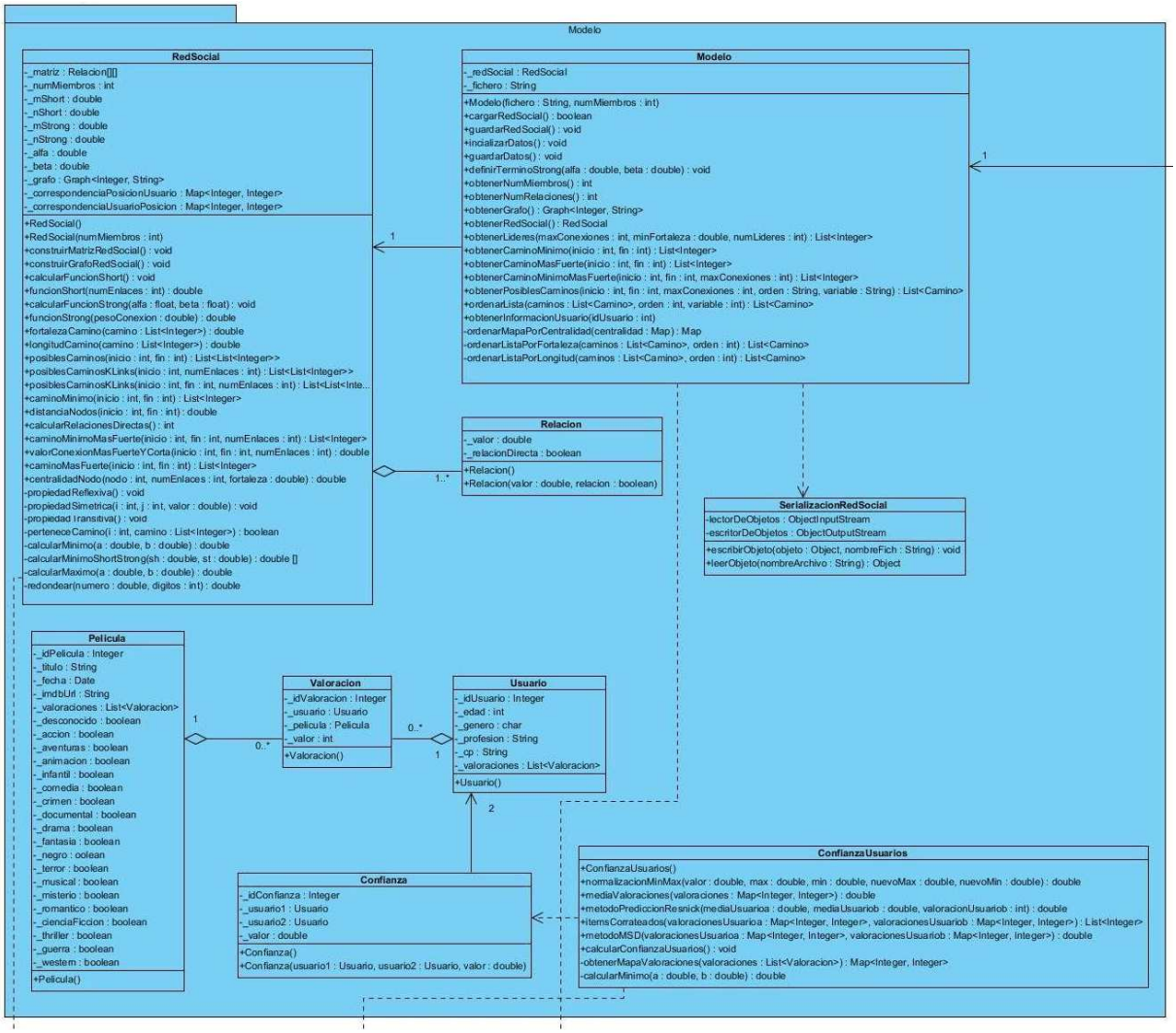

Ilustración 32: Paquete modelo

4.4.1.2.2 Paquete vista

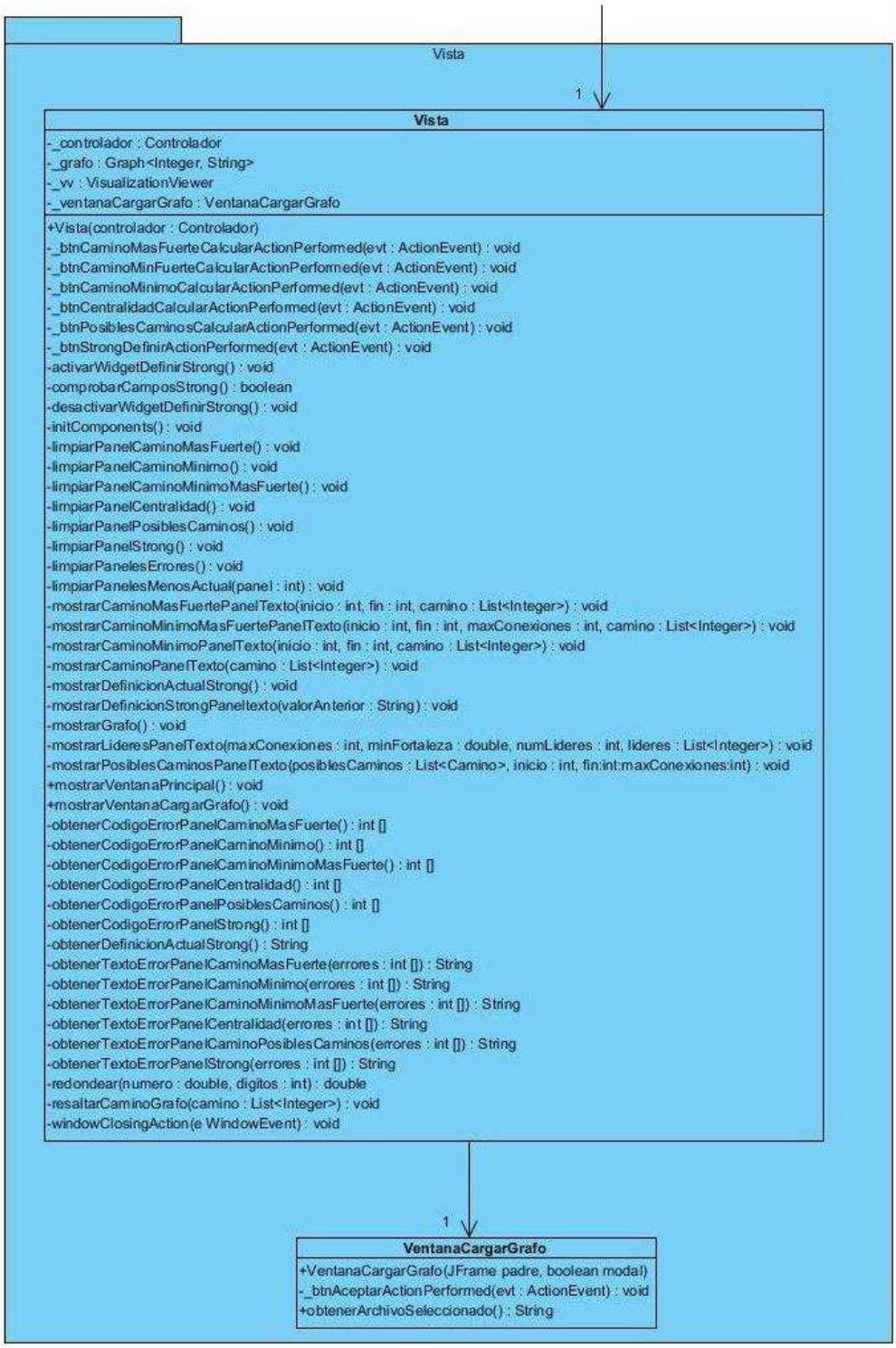

Ilustración 33: Paquete vista

4.4.1.2.3 Paquete controlador

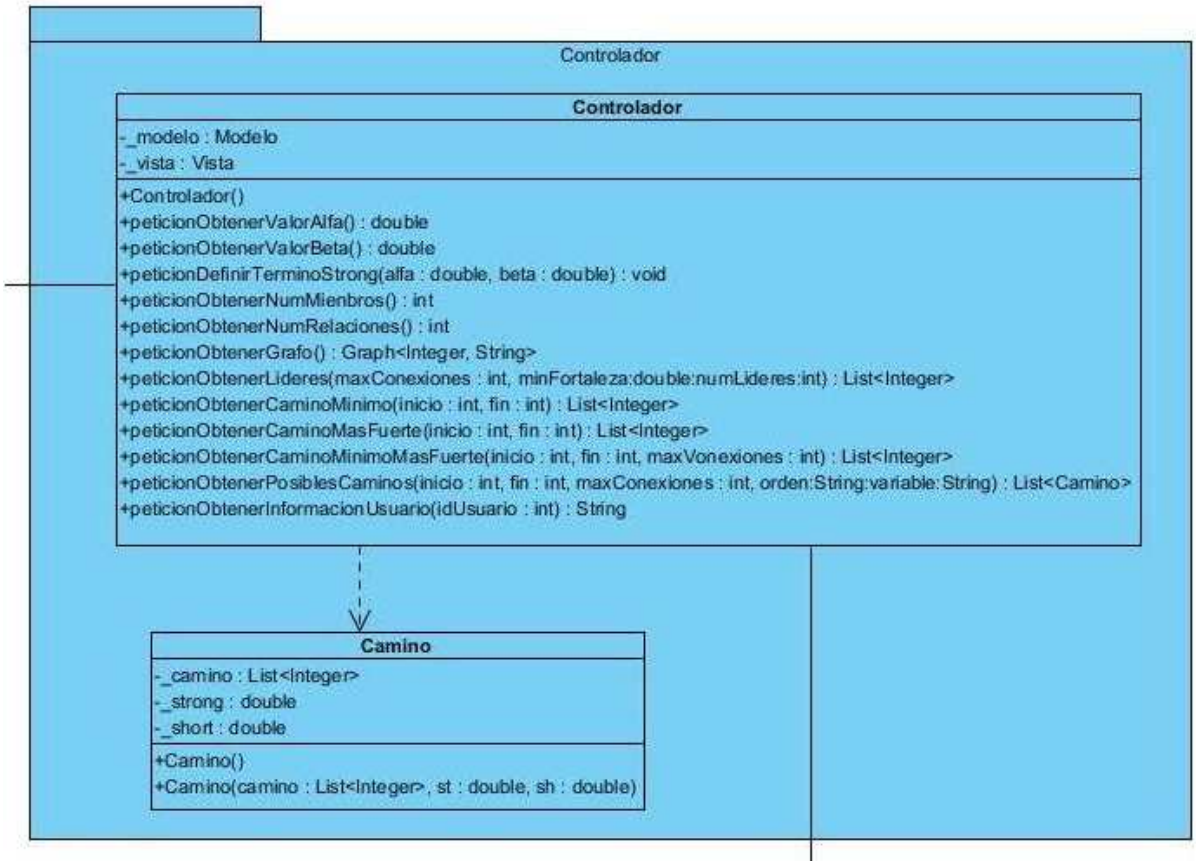

Ilustración 34: Paquete controlador

4.4.1.2.4 Paquetes persistencia y excepciones

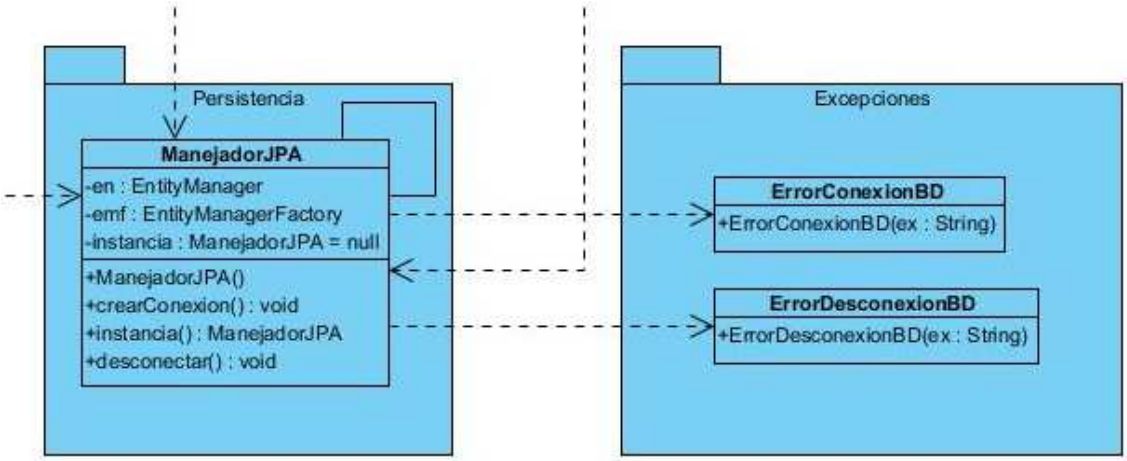

Ilustración 35: Paquetes persistencia y excepciones

4.4.2 Diseño de los datos

El objetivo de esta fase del diseño software es determinar la estructura de los elementos de información del sistema, es decir, la estructura de los datos sobre los que se va a trabajar. Estos elementos son:

- Los **usuarios**, de los que conocemos su identificador, edad, profesión, código postal y si es hombre o mujer.

- Las **películas**, sobre las cuales tenemos información acerca de su título, fecha de estreno, género/s al/a los que pertenece y la URL de su entrada en IMDB.

- Las **valoraciones**, de las que conocemos el usuario que las ha realizado, las películas valoradas y la puntuación que han recibido.

- El valor de **confianza** existente entre cada par de usuarios en base a las valoraciones que han realizado.

Tras haber especificado cuáles son los elementos de información del sistema, es el momento de obtener su representación en forma de tablas de una base de datos. En primer lugar, es necesario realizar un diseño conceptual de la base de datos. Para ello, se utilizará el modelo Entidad-Relación (E-R), el cual se explica a continuación.

Modelo Entidad-Relación

El modelo Entidad-Relación, también conocido por sus iniciales E-R, es un modelo de datos que se basa en la percepción del mundo real como una colección de objetos básicos, denominados entidades, y de relaciones entre dichos objetos [21]. Es la técnica de modelado de datos más extendida y se representa de forma gráfica a través del diagrama Entidad-Relación.

Los elementos fundamentales de un diagrama Entidad-Relación son:

- **Entidad:** objeto del mundo real con existencia propia y distinguible del resto, sobre el cual queremos almacenar información. Una entidad puede ser un objeto con existencia física como una persona, un libro, un empleado, etc. (entidad concreta) o un objeto con existencia conceptual como un puesto de trabajo, una asignatura, etc. (entidad abstracta). Se representa mediante un rectángulo etiquetado en su interior con un nombre, tal y como se puede ver en la Ilustración 36. Cualquier ejemplar completo de una entidad se conoce como instancia.

Ilustración 36: Entidad

- **Atributos:** características que definen o identifican a una entidad. Por ejemplo, los atributos que describen a la entidad alumno pueden ser: DNI, nombre y edad. Se representan mediante elipses etiquetadas con un nombre en su interior. (Ver Ilustración 37).

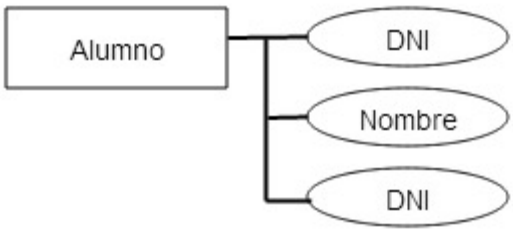

Ilustración 37: Entidad con atributos

- **Relación:** asociación, vínculo o correspondencia entre entidades relacionadas de alguna manera en el mundo real. Se representa mediante un rombo etiquetado en su interior con un verbo. En función del número de entidades que participan en una relación puede ser reflexiva (si es de una entidad consigo misma), binaria (si es entre dos entidades), ternaria (si es entre tres entidades) o múltiple (si es entre cuatro o más entidades). Según el número de instancias de cada entidad que están involucradas en la relación puede ser:

    - Uno a uno: una instancia de la entidad A se relaciona únicamente con una instancia de la entidad B. Se representa como 1:1.

    - Uno a muchos: cada instancia de la entidad A se relaciona con varias instancias de la entidad B. Se representa como 1: *.

    - Muchos a muchos: cada instancia de la entidad A puede relacionarse con varias instancias de la entidad B y viceversa. Se representa como *:*. En la Ilustración 38 se muestra un ejemplo de este tipo de relación.

Ilustración 38: Relación muchos a muchos

Para construir un diagrama Entidad-Relación también hay que tener en cuenta los siguientes aspectos:

- **Clave primaria:** toda entidad debe tener una clave que identifique de forma unívoca cada instancia del resto. Puede estar formada por uno o varios atributos. En este último caso se conoce como clave primaria compuesta y debe ser mínima, es decir, debe estar formada por el menor número posible de atributos que identifiquen a la entidad.

- **Clave foránea:** se trata de un atributo o conjunto de atributos de una entidad que son clave primaria en otra entidad con la cual se relaciona. También se conoce como clave externa o secundaria.

- **Entidad débil:** es aquella que no puede ser identificada unívocamente por sus atributos, es decir, necesita estar relacionada con una entidad fuerte para existir. Por ejemplo, si tenemos una entidad Libro y otra entidad Edición entre las que existe una relación, para poder identificar una edición es necesario conocer el identificador del libro.

Una vez conocidos los elementos que forman parte de un diagrama Entidad-Relación, podemos desarrollar el modelo Entidad-Relación. Los pasos a seguir son los siguientes:

1. Convertir los elementos del sistema software en un Esquema Conceptual del mismo.

2. Convertir este Esquema Conceptual en uno más refinado, conocido como Esquema Conceptual Modificado.

3. Obtener las tablas de la base de datos a partir del Esquema Conceptual Modificado y normalizarlas.

4.4.2.1 Esquema Conceptual

Para crear el Esquema Conceptual asociado a nuestro sistema necesitamos convertir sus elementos en entidades o relaciones. Observando detenidamente los elementos del sistema llegamos a la conclusión de que usuarios y películas se convierten en las entidades Usuario y Película, y valoraciones y confianza en las relaciones Valora y Confía. La relación Valora se produce entre la entidad Usuario y la entidad Película, y es una relación de muchos a muchos, ya que un usuario valora un conjunto de películas y una película puede ser valorada por más de un usuario. Por otro lado, Confía es una relación reflexiva de muchos a muchos ya que usuario puede tener una relación de confianza con muchos usuarios. El esquema conceptual resultante se puede observar en la Ilustración 39.

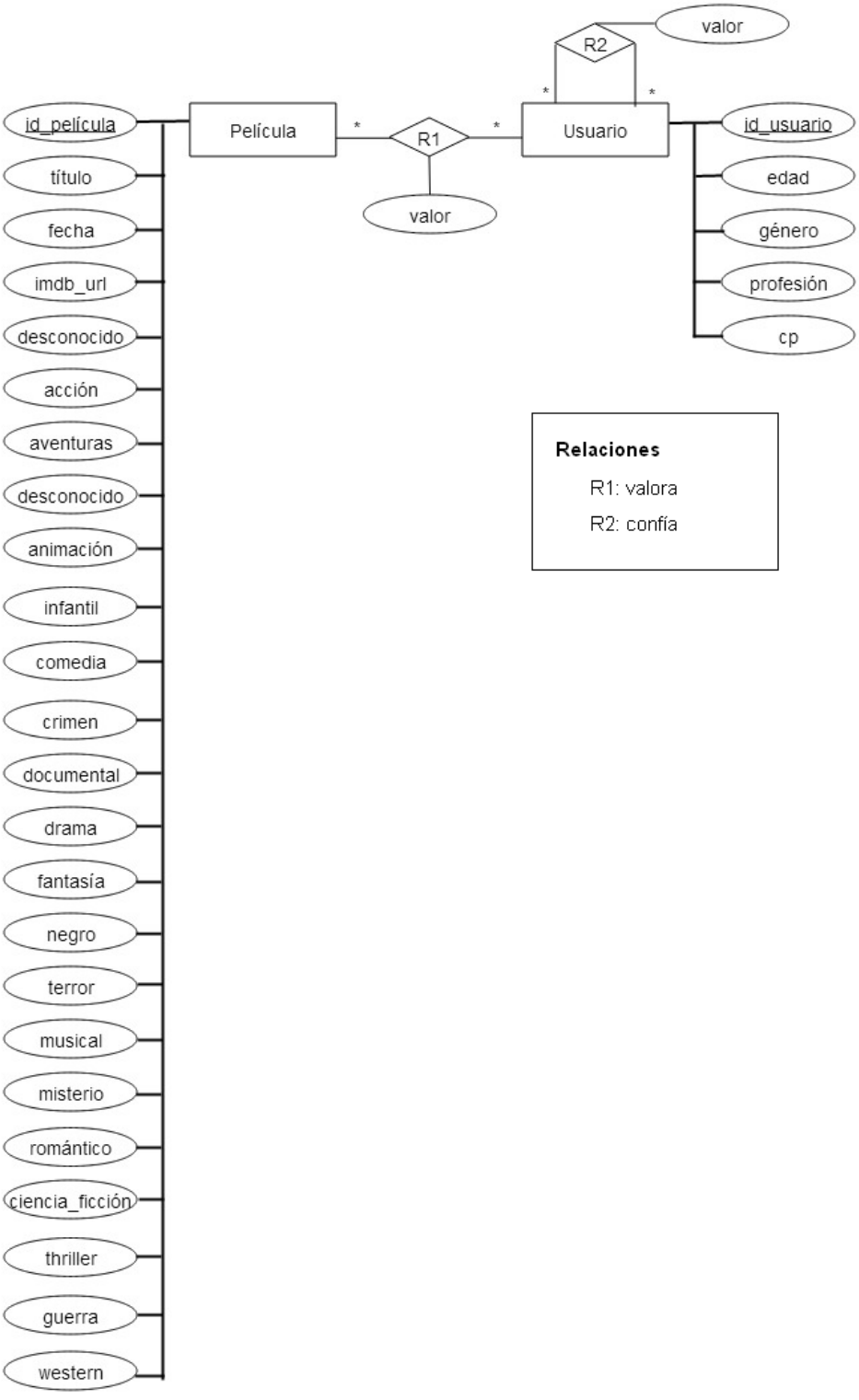

Ilustración 39: Esquema conceptual

Como hemos podido comprobar en el Esquema Conceptual, los grafos que representan a las redes sociales de estudio no se almacenan en la base de datos, sino que se guardarán serializados en ficheros para agilizar su proceso de carga.

4.4.2.2 Esquema Conceptual Modificado

A partir del Esquema Conceptual se puede obtener el Esquema Conceptual Modificado, pero para ello se deben realizar los cambios enunciados a continuación:

- Eliminar todas las entidades débiles.

- Eliminar las relaciones de muchos a muchos.

- Eliminar las relaciones con atributos.

En nuestro caso, las entidades débiles no son un problema, ya que en el Esquema Conceptual desarrollado no existe ninguna. Sin embargo, sí que se observan dos relaciones de muchos a muchos con atributos, lo que da lugar a la aparición de dos nuevas entidades:

- **Valoración:** entidad resultante de la relación de muchos a muchos entre Película y Usuario cuyos atributos serán las claves primarias de estas entidades (id_película, id_usuario) y el atributo de la relación (valor).

- **Confianza:** entidad que aparece como consecuencia de la relación reflexiva de muchos a muchos de la entidad Usuario. Sus atributos serán los identificadores de los dos usuarios entre los que existe una relación de confianza (id_usuario1, id_usuario2) y el atributo de dicha relación (valor).

Por tanto, el Esquema Conceptual Modificado quedaría como se muestra en la Ilustración 40.

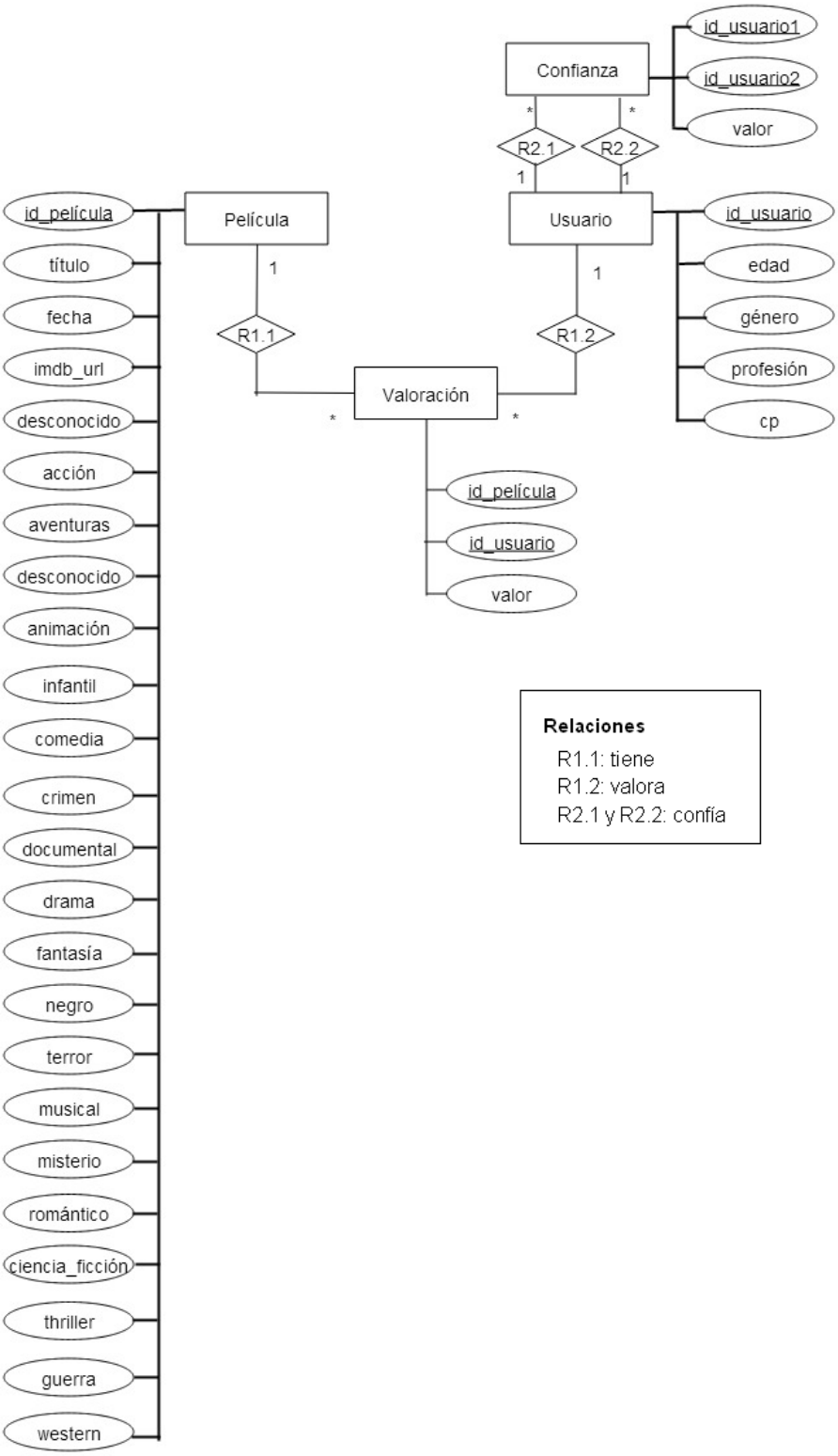

Ilustración 40: Esquema conceptual modificado

4.4.2.3 Tablas de la aplicación

Basándonos en el Esquema Conceptual Modificado obtenido previamente podemos determinar las tablas de la base de datos, teniendo en cuenta que:

- Cada entidad del ECM se transforma en una tabla.
- Los atributos de una entidad se convierten en los campos de la tabla respectiva.

Por tanto, obtenemos cuatro tablas (Usuario, Película, Valoración y Confianza), las cuales se detallan a continuación.

Usuario

Tabla que contiene información sobre los usuarios del Sistema de Recomendación de películas. Tiene un total de 943 registros y está formada por los siguientes campos.

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
Id_usuario	Int	Identificador unívoco del usuario	Clave primaria
Edad	Int	Edad del usuario	
Género	Char	M si el usuario es hombre y S si el usuario es mujer	
Profesión	Varchar	Profesión del usuario	
Cp	Varchar	Código Postal del usuario	

Tabla 10: Campos de la tabla Usuario

Película

Tabla que almacena los datos de las películas registradas en el Sistema de Recomendación que vamos a analizar. Tiene 1682 registros y está formada por los siguientes campos.

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
Id_película	Int	Identificador unívoco de la película	Clave primaria
Título	Varchar	Título de la película	
Fecha	Date	Fecha de estreno de la película	
Imdb_url	Varchar	Dirección URL de IMDB en la que aparece información sobre la película	
Acción	Int	1 si la película es de acción, 0 si no lo es	
Animación	Int	1 si la película es de animación, 0 si no lo es	
Aventuras	Int	1 si la película es de aventuras, 0 si no lo es	
Ciencia_ficción	Int	1 si la película es de ciencia ficción, 0 si no lo es	
Comedia	Int	1 si la película es una comedia, 0 si no lo es	
Crimen	Int	1 si la película es de crimen, 0 si no lo es	

Desconocido	Int	1 si el género de la película es desconocido, 0 si no lo es	
Documental	Int	1 si la película es un documental, 0 si no lo es	
Drama	Int	1 si la película es un drama, 0 si no lo es	
Fantasía	Int	1 si la película es de fantasía, 0 si no lo es	
Guerra	Int	1 si la película es de guerra, 0 si no lo es	
Infantil	Int	1 si la película es infantil, 0 si no lo es	
Misterio	Int	1 si la película es de misterio, 0 si no lo es	
Musical	Int	1 si la película es un musical, 0 si no lo es	
Negro	Int	1 si la película es de cine negro, 0 si no lo es	
Romántico	Int	1 si la película es romántica, 0 si no lo es	
Terror	Int	1 si la película de terror, 0 si no lo es	
Thriller	Int	1 si la película es un thriller, 0 si no lo es	

Western	Int	1 si la película es un western, 0 si no lo es	

Tabla 11: Campos de la tabla Película

Valoración

Tabla que contiene información sobre las valoraciones que los distintos usuarios han realizado de las películas. Tiene un total de 100000 registros y está formada por los siguientes campos.

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
Id_valoración	Int	Identificador unívoco de la valoración	Clave primaria
Id_usuario	Int	Identificador unívoco del usuario que realiza la valoración	Clave foránea
Id_película	Int	Identificador unívoco de la película valorada	Clave foránea
Valor	Int	Valor de la puntuación realizada. Debe ser 1, 2, 3, 4 ó 5	

Tabla 12: Campos de la tabla Valoración

Confianza

Tabla que guarda información sobre la confianza existente entre los distintos usuarios de la aplicación. Esta tabla se utilizará para construir el grafo de la red social formada por los miembros del Sistema de Recomendación. Tiene un total de 444153 registros y está formada por los siguientes campos.

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
Id_confianza	Int	Identificador unívoco de la confianza	Clave primaria
Id_usuario1	Int	Identificador unívoco del usuario 1	Clave foránea
Id_usuario2	Int	Identificador unívoco del usuario 2	Clave foránea
Valor	Int	Valor de la confianza existente entre el usuario 1 y el 2	

Tabla 13: Campos de la tabla Confianza

4.4.3 Diseño de la interfaz

En esta etapa del diseño del sistema software se define la apariencia visual de la interfaz de usuario y su comportamiento. Teniendo en cuenta las conclusiones de usabilidad, los casos de uso y los objetivos definidos, se procede a dar forma a las pantallas y secuencias de interacción que mejor se adaptan a las funcionalidades del sistema.

Sin duda, realizar un buen diseño de la interfaz es algo esencial en el desarrollo de un sistema. Ésta debe ser atractiva para el usuario de la aplicación, pero a la vez debe ser fácil de entender para que el usuario pueda trabajar sobre ella sin problemas.

La ingeniería de usabilidad divide esta fase en las siguientes etapas [18]:

- Definir el estilo.

- Metáforas.
- Pantallas.
- Caminos de navegación.
- Mensajes de error.

4.4.4 Definir el estilo

Antes de definir la apariencia de la interfaz de usuario, se debe definir el estilo de la misma. El estilo de la interfaz es el conjunto de reglas comunes que definen la apariencia visual que va a tener la interfaz de usuario de una aplicación, y que todos los diseñadores de la misma deben seguir [18].

Definir un estilo claro es muy importante para mantener la coherencia entre las diferentes partes de la interfaz, sobre todo cuando intervienen varios diseñadores. Para su definición por escrito se utiliza lo que se conoce como guía de estilo, que es un documento de ingeniería en el que se especifica con detalle el estilo de la interfaz.

Tras comprender la importancia del uso de guías de estilo pasamos a definir las reglas y normas de la guía de nuestra interfaz.

Fuentes

Las etiquetas, campos de texto y botones que muestran las funcionalidades del sistema deben tener las siguientes propiedades:

- Tipo de letra: “Tahoma”.
- Estilo: plano.
- Tamaño: 11 px.
- Color: 000000# (negro).

Los mensajes de error aparecerán en un panel con el borde rojo y tendrán las siguientes características:

- Tipo de letra: “Monospaced”.
- Estilo: plano.

- Tamaño: 13 px.
- Color: FF0000# (rojo).

Las salidas en el panel de texto resultantes de la interacción del usuario con la aplicación tendrán las siguientes propiedades:

- Tipo de letra: “Monospaced”.
- Estilo: plano.
- Tamaño: 13 px.
- Color: 000000# (negro).

Los nodos del grafo aparecerán por defecto de color 00FF00# (verde) y cambiarán al color FFFF00# (amarillo) cuando se resalten para marcar algún camino, los líderes de la red social, etc.

Las aristas del grafo se representarán con el color 000000# (negro) y por defecto su grosor será de 1.25f. Cuando se resalten para marcar algún camino su grosor pasará a ser de 2.5f.

Color de fondo

Para que se distingan claramente los distintos elementos de la interfaz el color de fondo que se utilizará en la misma es ECE9D8# (gris claro).

4.4.5 Metáforas

Las metáforas suponen una parte muy importante en el éxito o fracaso de una interfaz, por lo que deben diseñarse con mucho cuidado. Una metáfora se puede definir como una palabra o expresión que representa a un objeto o concepto, al cual no denota literalmente, con el fin de sugerir una comparación con otro objeto o concepto y facilitar así su comprensión. Por ejemplo, en la frase “El tiempo es oro” se transfieren las características del oro (valioso, escaso, preciado) al tiempo, para dar a entender su valor.

En el diseño de una interfaz resultan de gran utilidad las metáforas visuales ya que permiten al usuario, por comparación con otro objeto o concepto, comprender de forma más intuitiva las tareas que la interfaz le permite llevar a cabo. Las metáforas visuales funcionan de manera similar a las metáforas literarias. Permiten sustituir un concepto o expresión por una imagen más simple que lo representa. Por ejemplo, en una aplicación de visualización de imágenes, para representar el concepto

de acercar la imagen se suele utilizar una lupa con un símbolo más y para reflejar la idea de alejarla se utiliza una lupa con un símbolo menos.

Para que una metáfora cumpla con su cometido, es muy importante que el desarrollador de la aplicación y el usuario final de la misma tengan una base cultural similar. Es posible que una metáfora tenga un significado para un usuario occidental y otro bien distinto para un usuario oriental. Por ejemplo, un icono de una mano con el dedo pulgar levantado, para un usuario occidental significa una muestra de acuerdo. Sin embargo, un usuario en Turquía considera este gesto como un insulto parecido a levantar el dedo corazón en Europa. Por tanto, hay que intentar que las metáforas utilizadas sean lo más universales posibles, con el fin de que sean comprendidas por el mayor número posible de usuarios potenciales.

En nuestra aplicación se han utilizado las siguientes metáforas:

-  Minimizar aplicación.
-  Maximizar aplicación.
-  Cerrar aplicación.
-  Desplazar hacia arriba.
-  Desplazar hacia abajo.
-  Desplazar a la izquierda.
-  Desplazar a la derecha.

4.4.6 Pantallas

La mejor herramienta para diseñar las pantallas, es decir, para definir la estructura visual de la aplicación, es utilizar dibujos hechos a mano con lápiz y papel (bocetos). Un boceto es un dibujo a mano alzada que expresa visualmente la esencia de una idea sin todos los detalles complejos de esa idea.

Hay que tener en cuenta que los diseños de pantalla son un vehículo para poder analizar la calidad de nuestro diseño, por ello no es adecuado perder tiempo ni dinero realizando estos diseños previos con alguna herramienta especializada. Por esta razón, los bocetos son una herramienta de gran utilidad, ya que permiten crear de forma rápida y elástica una imagen de lo que será la futura aplicación.

Estos prototipos no expresan el diseño final, simplemente una idea de lo que será el sistema, por lo que serán susceptibles de cambio durante el proceso de implementación.

En la Ilustración 41 se muestra el diseño de la pantalla a la que accederá el usuario al iniciar la aplicación.

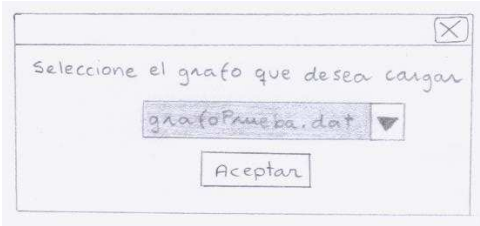

Ilustración 41: Cargar grafo

Esta pantalla permite seleccionar alguno de los grafos disponibles en la aplicación para realizar un análisis de la red social a la que representa. Tras elegir el grafo a visualizar, aparecerá la pantalla principal de la aplicación (ver Ilustración 42). Esta pantalla contendrá todas las funcionalidades del sistema e irá cambiando en función de las interacciones del usuario.

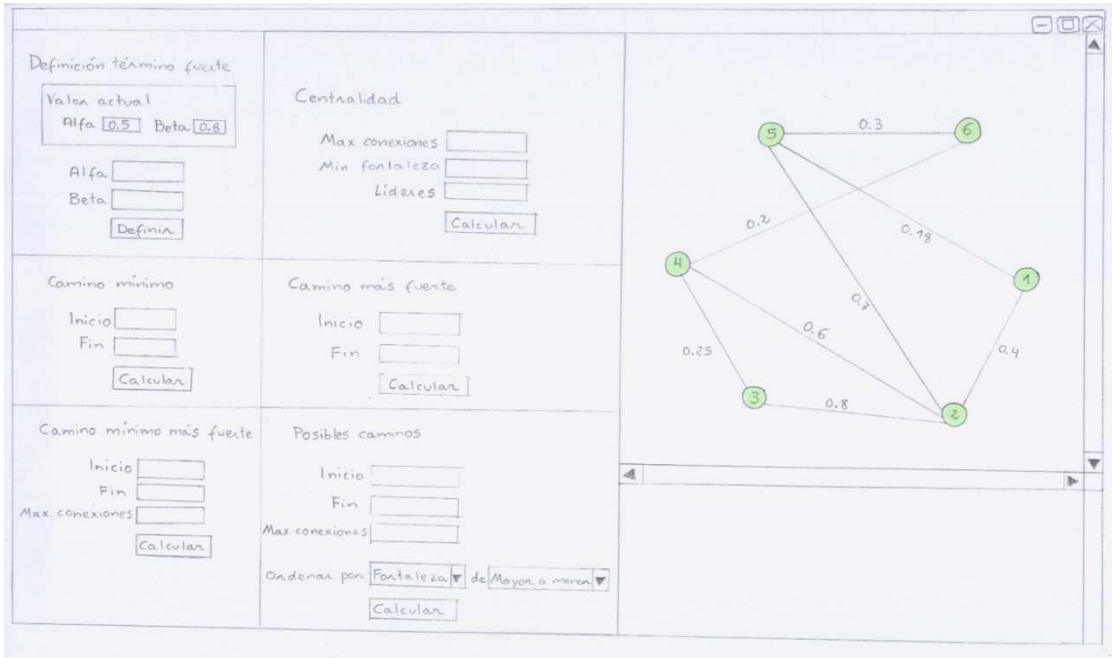

Ilustración 42: Pantalla aplicación

4.4.7 Caminos de navegación

Hasta ahora, tenemos un diseño visual de la interfaz estático, es decir, conocemos la pantalla de la aplicación con la que el usuario llevará a cabo la interacción. Pero no sabemos si esta interacción va a transcurrir de forma fluida ni si va a ser comprensible para el usuario. Por ello, vamos a diseñar la interfaz en movimiento y comprobar que es usable.

Para estudiar los caminos de navegación se utilizará una herramienta conocida como storyboard, que consiste en un conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender el flujo de la aplicación.

Para realizar un storyboard es necesario situar capturas de pantalla de la interfaz unidas mediante flechas para indicar el camino que sigue la interacción. El lugar del que procede una flecha debe indicar el componente de interacción que ha disparado la acción.

Esta técnica permite analizar las pantallas de los distintos escenarios para ver si los pasos que se dan son los correctos y si la acción se entiende bien, de manera que si es necesario se introduzcan correcciones en la interfaz antes de implementarla.

A continuación se muestran los storyboards de las acciones que se pueden llevar a cabo en el sistema:

- Storyboard 1: Cargar grafo.
- Storyboard 2: Definir el significado del término fuerte.
- Storyboard 3: Obtener líderes.
- Storyboard 4: Calcular el camino mínimo entre dos miembros.
- Storyboard 5: Obtener el camino más fuerte que une a dos miembros de la red.
- Storyboard 6: Calcular el camino mínimo más fuerte existente entre dos miembros.
- Storyboard 7: Obtener los posibles caminos existentes entre dos miembros de la red social.
- Storyboard 8: Obtener información sobre un usuario.
- Storyboard 9: Salir del sistema.

Storyboard 1: Cargar grafo.

Al iniciar la aplicación aparecerá una ventana que permitirá seleccionar el grafo a cargar. Tras elegir el grafo representativo de la red social a analizar y pulsar el botón ´Aceptar´ se mostrará la pantalla principal de la aplicación, la cual contiene todas las funcionalidades que ofrece el sistema. (Ver Ilustración 43).

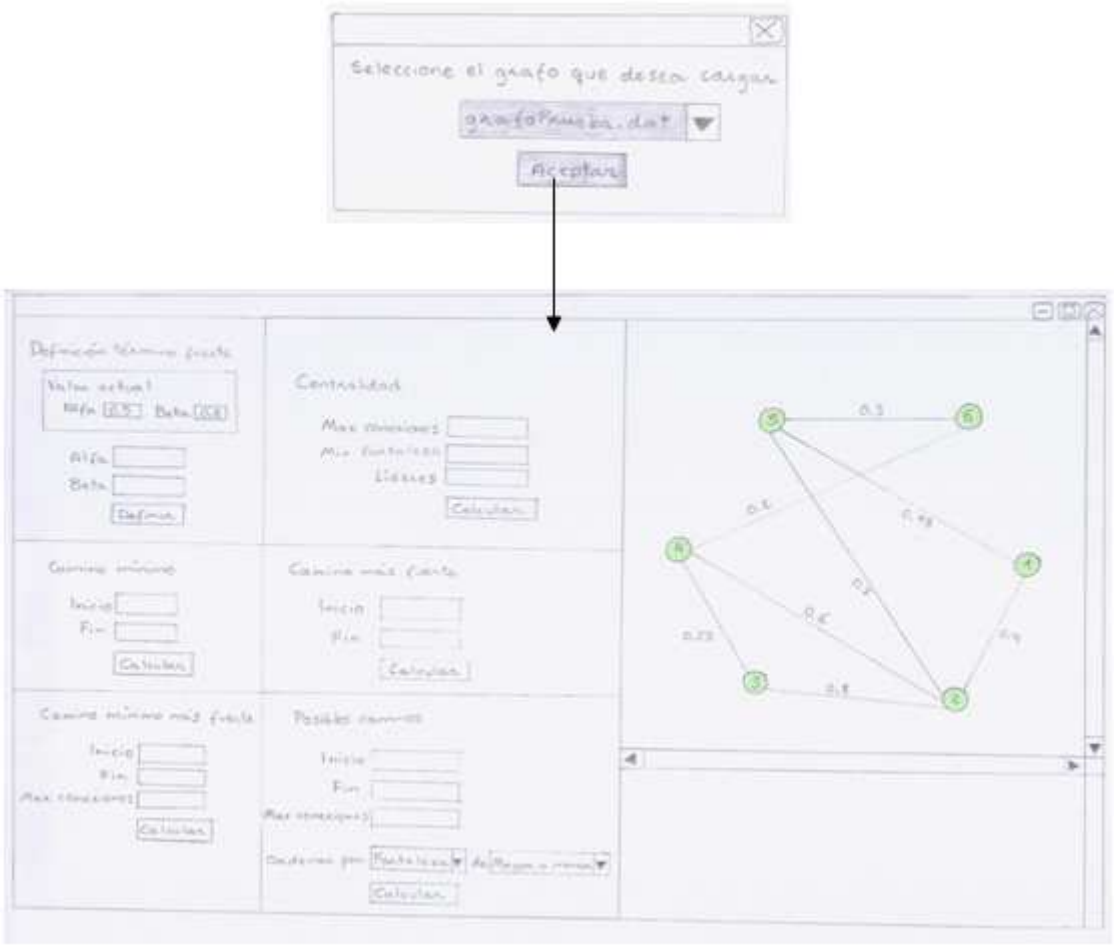

Ilustración 43: Storyboard 1 - Cargar grafo

Storyboard 2: Definir el significado del término fuerte.

Tras introducir los valores de alfa y beta y pulsar el botón ´Definir´, si estos son correctos, se modificará el significado del término fuerte. Además, en el panel de texto se mostrará el valor actual del término y el que tenía antes de realizar el cambio. (Ver Ilustración 44).

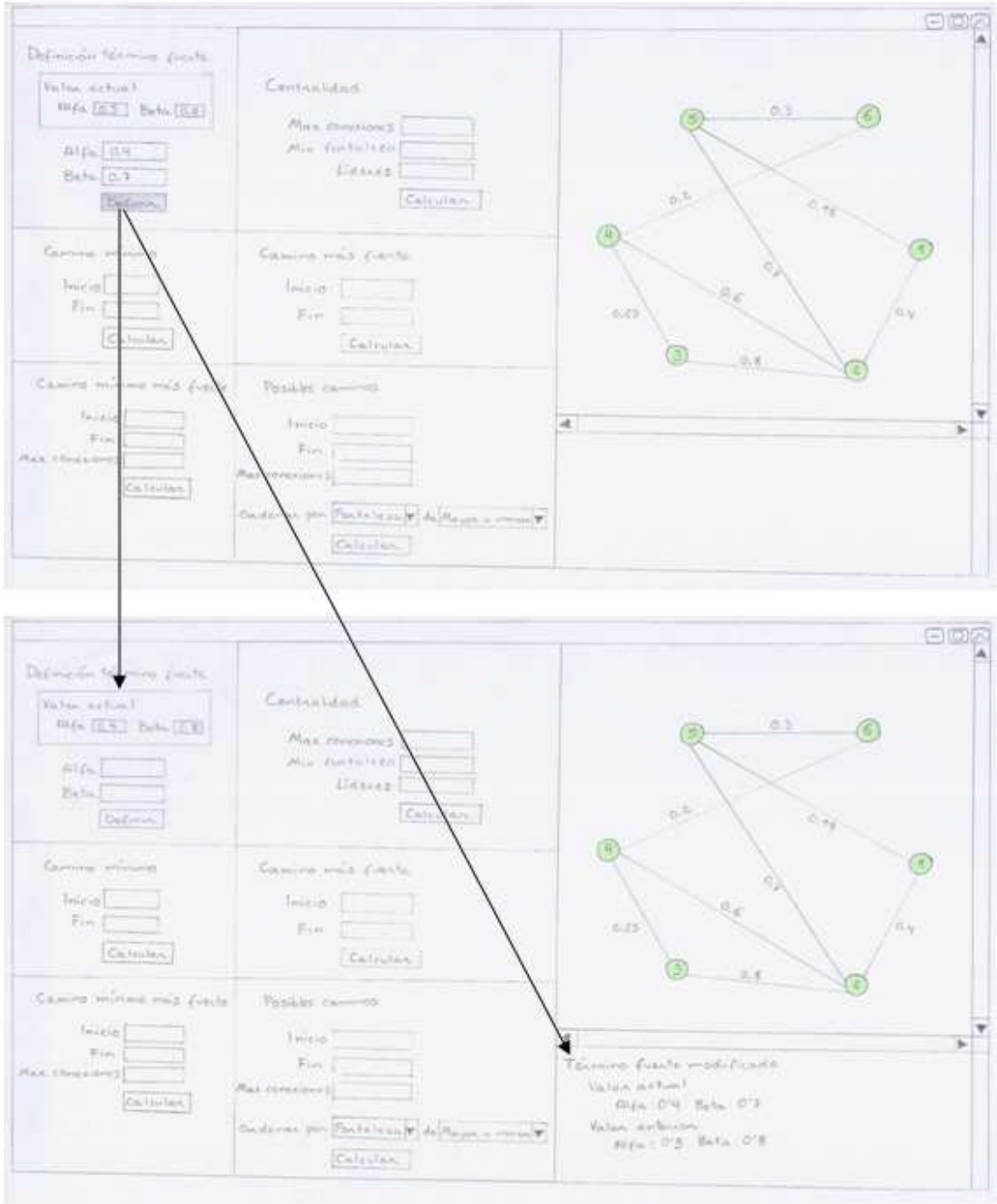

Ilustración 44: Storyboard 2 - Definir el significado del término fuerte

Storyboard 3: Obtener líderes.

Tras introducir los valores necesarios y pulsar el botón ´Calcular´ en el panel ´Centralidad´, se resaltarán los líderes con color amarillo en el grafo, para diferenciarlos del resto de individuos. También, se reflejarán en el panel de texto los criterios utilizados y el valor de centralidad de cada uno de los líderes. (Ver Ilustración 45).

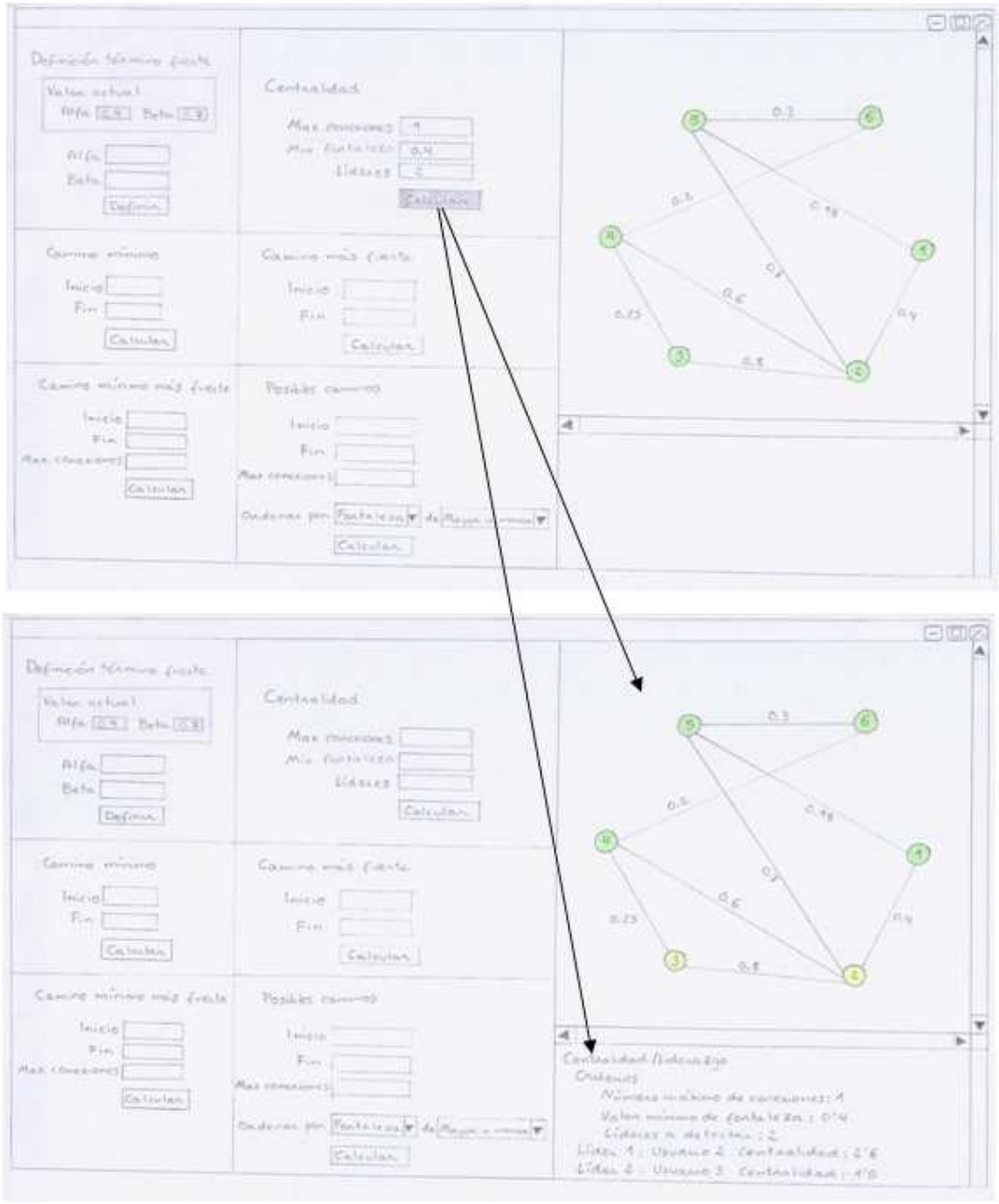

Ilustración 45: Storyboard 3 - Obtener líderes

Storyboard 4: Calcular el camino mínimo entre dos miembros.

Cuando se indique el valor de los nodos inicio y fin y se pulse el botón ´Calcular´ se resaltará en el grafo el camino mínimo. Para ello, se dibujarán con color amarillo los nodos que forman parte del camino y con un mayor grosor las aristas que unen dichos nodos. Además, se indicará en el panel de texto el camino obtenido. (Ver Ilustración 46).

Ilustración 46: Storyboard 4 - Calcular el camino mínimo entre dos miembros

Storyboard 5: Obtener el camino más fuerte que une a dos miembros de la red.

Una vez introducidos los valores de los nodos inicio y fin y pulsado el botón ´Calcular´, se destacará en el grafo el camino más fuerte que une a dichos nodos y se mostrará en el panel de texto el camino obtenido. (Ver Ilustración 47).

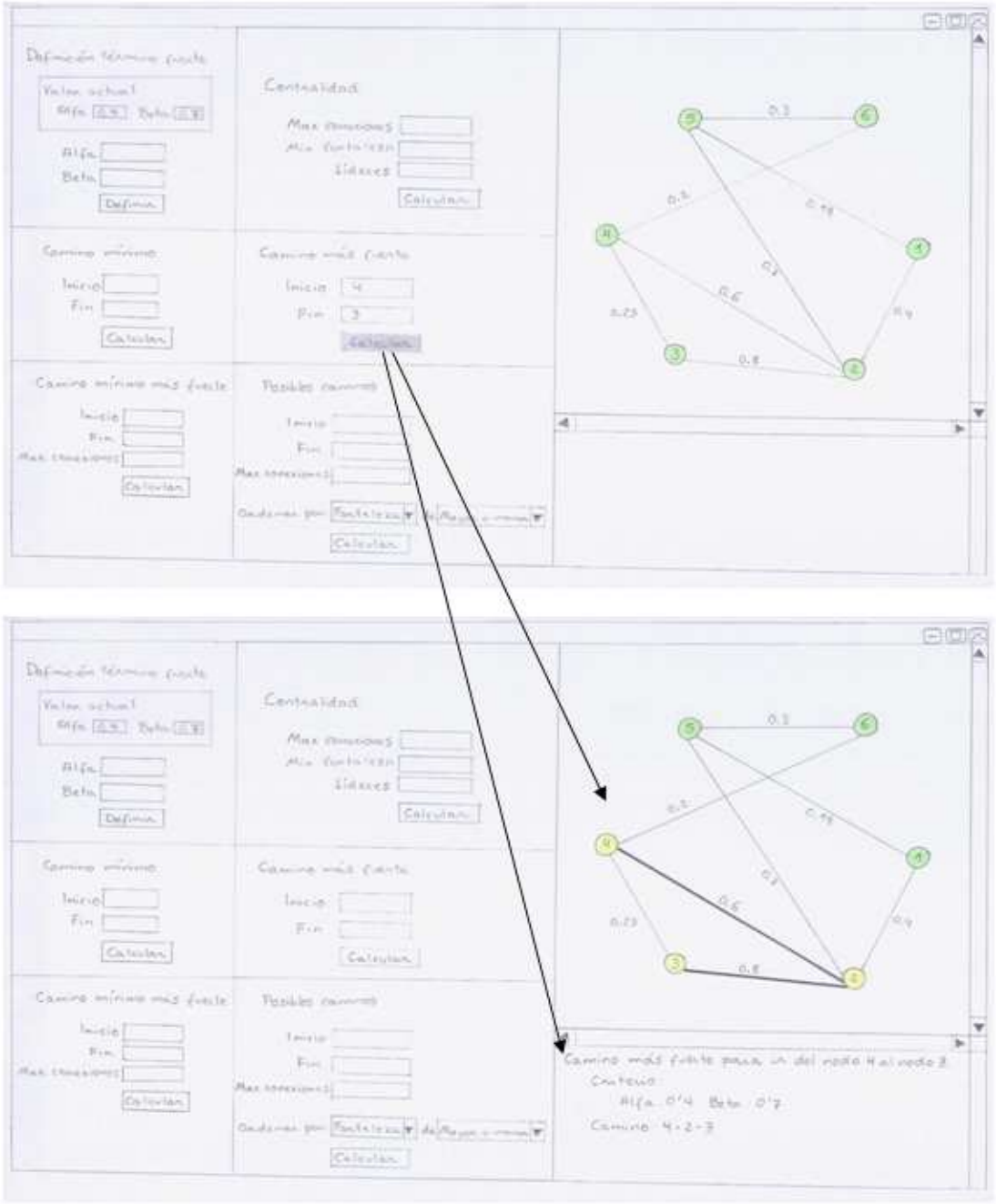

Ilustración 47: Storyboard 5 - Obtener el camino más fuerte que une a dos miembros

Storyboard 6: Calcular el camino mínimo más fuerte existente entre dos miembros.

Después de indicar los valores de los nodos inicio y fin, el número máximo de conexiones a tener en cuenta y pulsar el botón ´Calcular´ se resaltará en el grafo el camino mínimo más fuerte obtenido según los criterios especificados. Además, se reflejará en el panel de texto cuáles han sido dichos criterios. (Ver Ilustración 48).

Ilustración 48: Storyboard 6 - Calcular el camino mínimo más fuerte entre dos miembros

Storyboard 7: Obtener los posibles caminos existentes entre dos miembros de la red social.

Tras introducir los valores necesarios y pulsar el botón ´Calcular´ en el panel *´Posibles caminos´*, se mostrarán en el panel de texto todos los posibles caminos que unen al nodo inicio y al nodo fin según los criterios indicados. (Ver Ilustración 49).

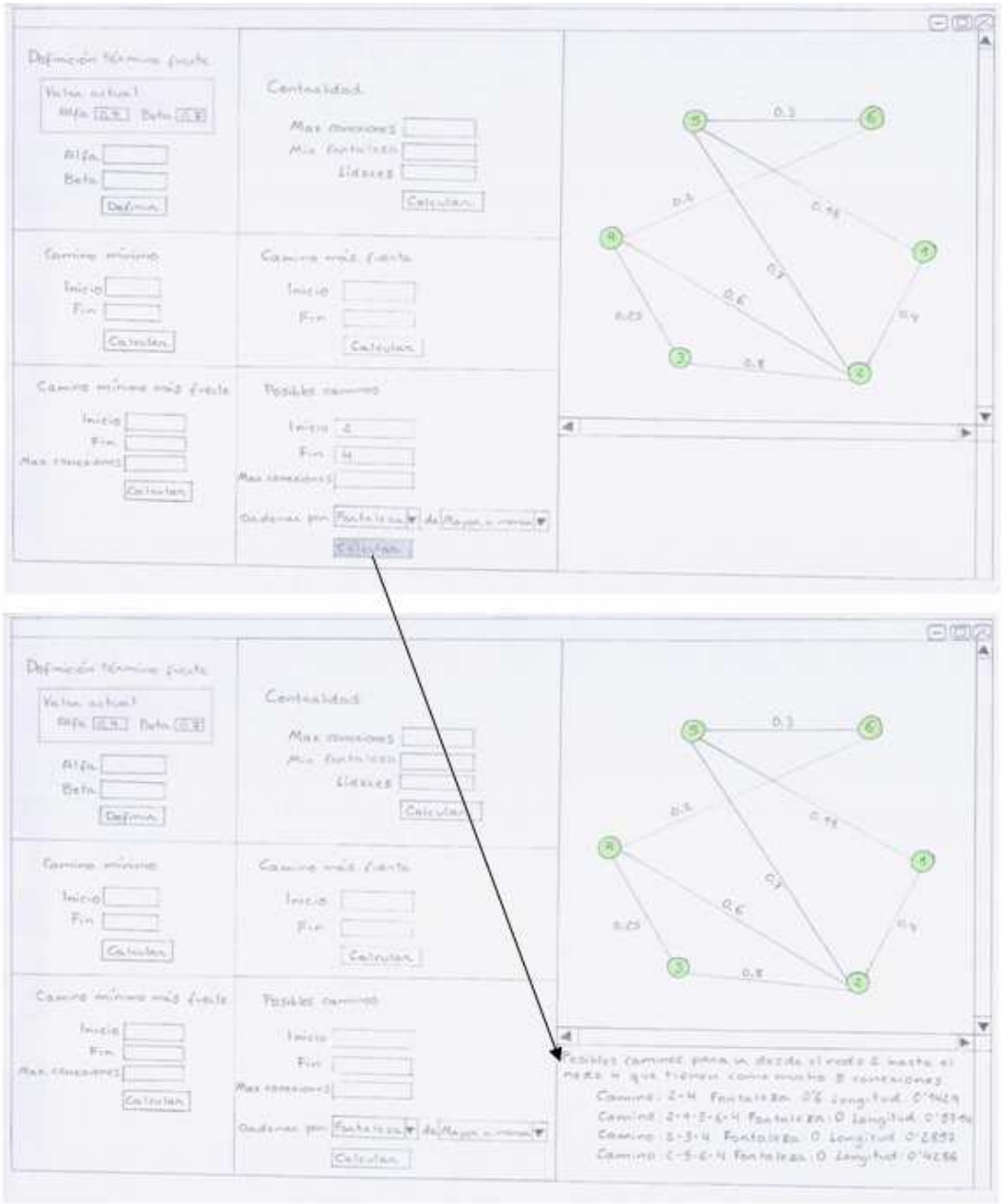

Ilustración 49: Storyboard 7 - Obtener los posibles caminos entre dos miembros

Storyboard 8: Obtener información sobre un usuario.

Al posicionar el cursor del ratón sobre alguno de los nodos del grafo se mostrará la información del usuario que representa, tal y como se puede observar en la ilustración. (Ver Ilustración 50).

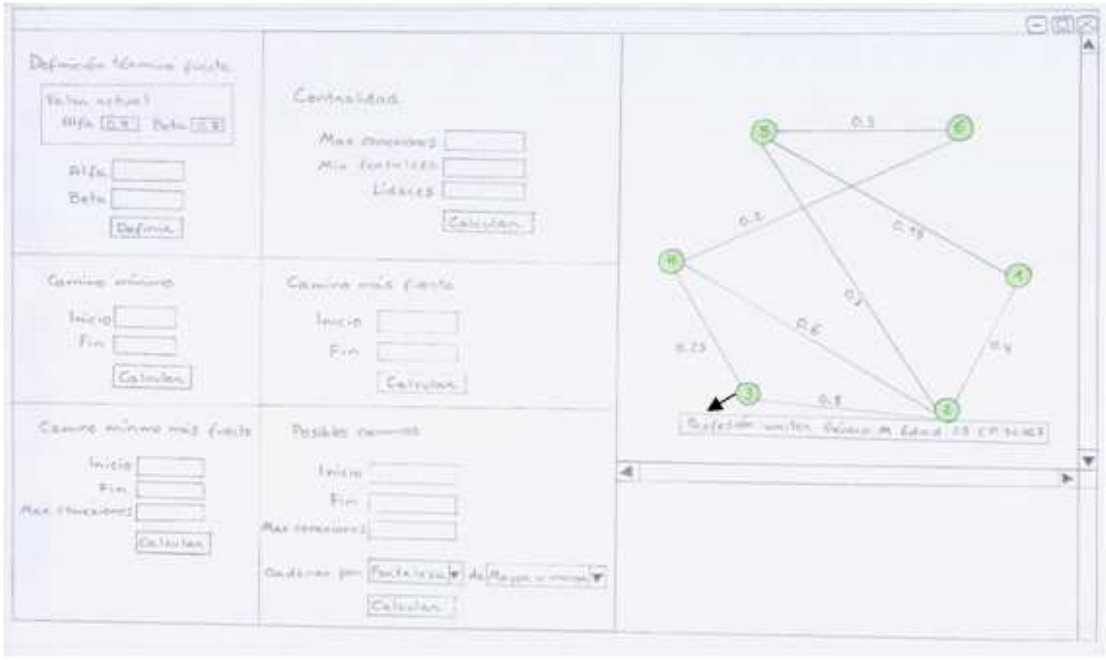

Ilustración 50: Storyboard 8 - Obtener información sobre un usuario

Storyboard 9: Salir del sistema.

Al pulsar la cruz señalada con la flecha negra, se cerrará la aplicación. (Ver Ilustración 51).

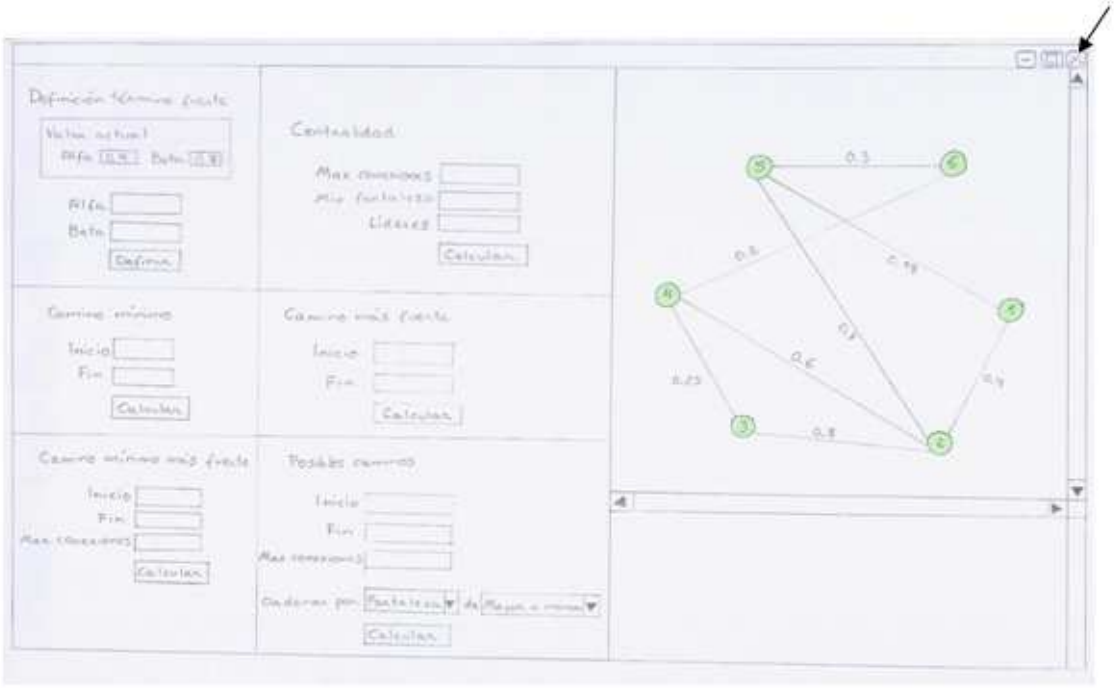

Ilustración 51: Storyboard 9 - Salir del sistema

4.4.8 Mensajes de error

El diseño de las pantallas y de los caminos de navegación proporciona una idea muy clara de cómo se va a desarrollar físicamente la interacción entre el usuario y la aplicación, y por tanto de las posibles situaciones de error que pueden darse en esa interacción. Por ello, ahora es el momento de diseñar los mensajes de error.

Los mensajes de error son el medio por el que el sistema comunica al usuario que se ha producido un error en la interacción.

Los errores se producen por falta de conocimiento sobre la interfaz, porque no se ha entendido correctamente el sistema o bien inadvertidamente. Cuando un usuario comete un error se siente confuso y aumenta su ansiedad por lo que hay que pensar detenidamente cómo se van a mostrar los mensajes de error y el texto explicativo que se va a utilizar para que quede claro el error cometido.

Los mensajes de error son muy importantes de cara a la usabilidad, ya que bien diseñados, permiten aumentar la confianza del usuario en el uso de la interfaz y que pueda seguir con su propósito. Por ello, a la hora de diseñar mensajes de error se deberían seguir las reglas mostradas a continuación:

- **Ser breves.** Hay que diseñar mensajes de error claros pero breves ya que el usuario es una persona ocupada tratando de llevar a cabo una tarea, por lo que si se le presenta un mensaje de error muy largo probablemente no lo lea.

- **Ser específicos.** Para que el usuario entienda qué debe hacer para impedir que se produzca de nuevo el error cometido, los mensajes deben indicar claramente qué ha ido mal y no ser demasiado generales.

- **Utilizar un tono positivo y guía constructiva.** Siempre que sea posible se debe indicar al usuario qué debe hacer para eliminar el error, sin recriminarle lo que ha hecho mal.

- **Usar un formato físico apropiado.** Es preferible utilizar mensajes en los que se mezclan letras mayúsculas y minúsculas ya que para la mayoría de usuarios resulta más fácil leer un mensaje con este formato. Los mensajes escritos únicamente en mayúsculas deberían reservarse para avisos breves y graves. Además, los mensajes de error deberían aparecer siempre en una misma posición en la pantalla para que el usuario pueda localizarlos fácilmente.

Teniendo en cuenta las funcionalidades que ofrece el sistema, los mensajes de error que pueden aparecer tras la interacción del usuario con la aplicación son:

- **Definición término fuerte.**

  - "Alfa debe ser un valor real entre 0 y 1". Si el campo correspondiente al valor Alfa está vacío, contiene un tipo de dato erróneo (carácter, cadena,…) o un valor incorrecto (valor menor que 0 o mayor que 1).

  - "Beta debe ser un valor real entre 0 y 1". Si el campo correspondiente al valor Beta está vacío, contiene un tipo de dato erróneo (carácter, cadena,…) o un valor incorrecto (valor menor que 0 o mayor que 1).

  - "Alfa y beta deben ser valores reales entre 0 y 1". Si los campos Alfa y Beta están vacíos, contienen un tipo de dato erróneo (carácter, cadena,…) o valores incorrectos (menores que 0 o mayores que 1).

  - "Alfa debe ser menor que beta". Si el campo correspondiente al valor Alfa contiene un valor mayor o igual que el del campo Beta.

- **Centralidad.**

  - "Max. conexiones debe ser un valor entero entre 1 y numRelaciones", donde numRelaciones es el número de conexiones existentes en la red social. Este mensaje aparece si el campo Max. conexiones contiene un tipo de dato erróneo (carácter, cadena, real,…) o un valor incorrecto (valor menor que 1 o mayor que numRelaciones).

  - "Min. fortaleza debe ser un valor real entre 0 y 1". Si el campo Min. fortaleza contiene un tipo de dato erróneo (carácter, cadena,…) o un valor incorrecto (valor menor que 0 o mayor que 1).

  - "Líderes debe ser un valor entero entre 1 y numMiembros", donde numMiembros es el número de miembros de la red social. Si el campo Líderes contiene un tipo de dato erróneo (carácter, cadena, real,…) o un valor incorrecto (valor menor que 1 o mayor que numMiembros).

- **Camino mínimo.**

  - "Inicio debe ser el valor de un nodo". Si el campo Inicio está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

- "Fin debe ser el valor de un nodo". Si el campo Fin está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

- "Inicio y fin deben ser valores de nodos". Si los campos Inicio y Fin están vacíos, contienen un tipo de dato erróneo (cadena, carácter, real,…) o sus valores no se corresponden con el id de algún usuario.

- "Inicio y fin deben ser distintos". Si los campos Inicio y Fin contienen el mismo valor.

- **Camino más fuerte.**

  - "Inicio debe ser el valor de un nodo". Si el campo Inicio está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

  - "Fin debe ser el valor de un nodo". Si el campo Fin está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

  - "Inicio y fin deben ser valores de nodos". Si los campos Inicio y Fin están vacíos, contienen un tipo de dato erróneo (cadena, carácter, real,…) o sus valores no se corresponden con el id de algún usuario.

  - "Inicio y fin deben ser distintos". Si los campos Inicio y Fin contienen el mismo valor.

- **Camino mínimo más fuerte.**

  - "Inicio debe ser el valor de un nodo". Si el campo Inicio está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

  - "Fin debe ser el valor de un nodo". Si el campo Fin está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

  - "Inicio y fin deben ser valores de nodos". Si los campos Inicio y Fin están vacíos, contienen un tipo de dato erróneo (cadena, carácter, real,…) o sus valores no se corresponden con el id de algún usuario.

- "Inicio y fin deben ser distintos". Si el nodo Inicio y Fin son iguales.

- "Max. conexiones debe ser un valor entero entre 1 y numRelaciones", donde numRelaciones es el número de conexiones existentes en la red social. Este mensaje aparece si el campo Max. conexiones contiene un tipo de dato erróneo (cadena, carácter, real,…) o un valor incorrecto (valor menor que 1 o mayor que numRelaciones).

- **Posibles caminos.**

  - "Inicio debe ser el valor de un nodo". Si el campo Inicio está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

  - "Fin debe ser el valor de un nodo". Si el campo Fin está vacío, contiene un tipo de dato erróneo (cadena, carácter, real,…) o si su valor no se corresponde con el id de algún usuario.

  - "Inicio y fin deben ser valores de nodos". Si los campos Inicio y Fin están vacíos, contienen un tipo de dato erróneo (cadena, carácter, real,…) o sus valores no se corresponden con el id de algún usuario.

  - "Inicio y fin deben ser distintos". Si el nodo Inicio y Fin son iguales.

  - "Max. conexiones debe ser un valor entero entre 1 y numRelaciones", donde numRelaciones es el número de conexiones existentes en la red social. Este mensaje aparece si el campo Max. conexiones contiene un tipo de dato erróneo (cadena, carácter, real,…) o un valor incorrecto (valor menor que 1 o mayor que numRelaciones).

4.5 Implementación

La implementación es la etapa del proceso de Ingeniería del Software en el que el modelo obtenido en las anteriores fases se transforma en código fuente. Para ello, es necesario elegir el lenguaje de programación que se va a emplear para la codificación y el entorno que se va a utilizar para desarrollar la aplicación.

4.5.1 Tipo de arquitectura de la aplicación

En este proyecto, se va a desarrollar una aplicación de escritorio con tecnología JAVA y con acceso a una base de datos local. Una aplicación de escritorio es un programa informático, que se ejecuta en el ordenador del usuario, diseñado para ayudarle a realizar determinadas tareas. En nuestro caso, el funcionamiento de la aplicación es muy sencillo, ya que se ejecutará en el ordenador del usuario sin necesidad de disponer de conexión a Internet ni de acceder a un servidor externo, ya que la base de datos utilizada por la aplicación se encontrará en el propio ordenador del usuario por medio del sistema de gestión de bases de datos MySQL. Podría haberse utilizado un servidor externo para alojar la base de datos pero debido a que el tamaño de la misma no es desmesurado, al hecho de que esto supondría unos costes adicionales de alojamiento y contratación de un dominio, y a que de esta manera sería necesario que el usuario tuviera conexión a Internet, se ha decidido utilizar un servidor de bases de datos local.

4.5.2 Lenguajes de programación utilizados

Para la implementación de la aplicación se ha utilizado el lenguaje de programación Java y, el lenguaje de consulta orientado a objetos JPQL mediante el empleo de la API de persistencia de Java JPA.

Java es un lenguaje de programación orientado a objetos que fue desarrollado por Sun MicroSystems a principios de los 90. Su sintaxis deriva de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java normalmente se compilan a bytecode, que es un código intermedio utilizado por la máquina virtual de Java más abstracto que el código máquina, de manera que puedan ejecutarse en cualquier máquina virtual Java sin importar la arquitectura del computador. El objetivo de este lenguaje es permitir que los desarrolladores de aplicaciones escriban el programa una vez y puedan ejecutarlo en cualquier dispositivo.

Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java. Es un framework que realiza una abstracción de las bases de datos y proporciona un estándar para la persistencia de datos en Java. Su objetivo es no perder las ventajas de la orientación a objetos al interactuar con una base de datos y permitir utilizar objetos regulares. El mapeo objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML.

JPQL (Java Persistence Query Language) es un lenguaje de consulta orientado a objetos definido como parte de la especificación de la API de persistencia de Java (JPA). Es utilizado para hacer consultas sobre entidades almacenadas en una base de datos relacional. Está inspirado en SQL y la sintaxis de sus consultas se asemeja a las de este lenguaje, pero opera con objetos entidad de JPA en lugar de hacerlo directamente con las tablas de la base de datos. Permite tanto consultas de recuperación de objetos (SELECT), como de actualización (UPDATE) y borrado (DELETE).

4.5.3 Herramientas de desarrollo

Tras describir los lenguajes de programación empleados, se especifican las herramientas de desarrollo utilizadas.

Para desarrollar la aplicación de escritorio en el lenguaje de programación Java se ha utilizado la plataforma Netbeans en su versión 7.2.1 (ver Ilustración 52). Netbeans es un entorno de desarrollo integrado diseñado para escribir, compilar, depurar y ejecutar programas. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene varias clases de Java escritas para interactuar con las API´s de Netbeans y un archivo especial (manifest file) que lo identifica como módulo. De esta manera, las aplicaciones construidas a partir de módulos pueden ser extendidas mediante la agregación de nuevos módulos. El hecho de que los módulos puedan ser construidos de forma independiente, facilita la extensión por parte de otros desarrolladores de software.

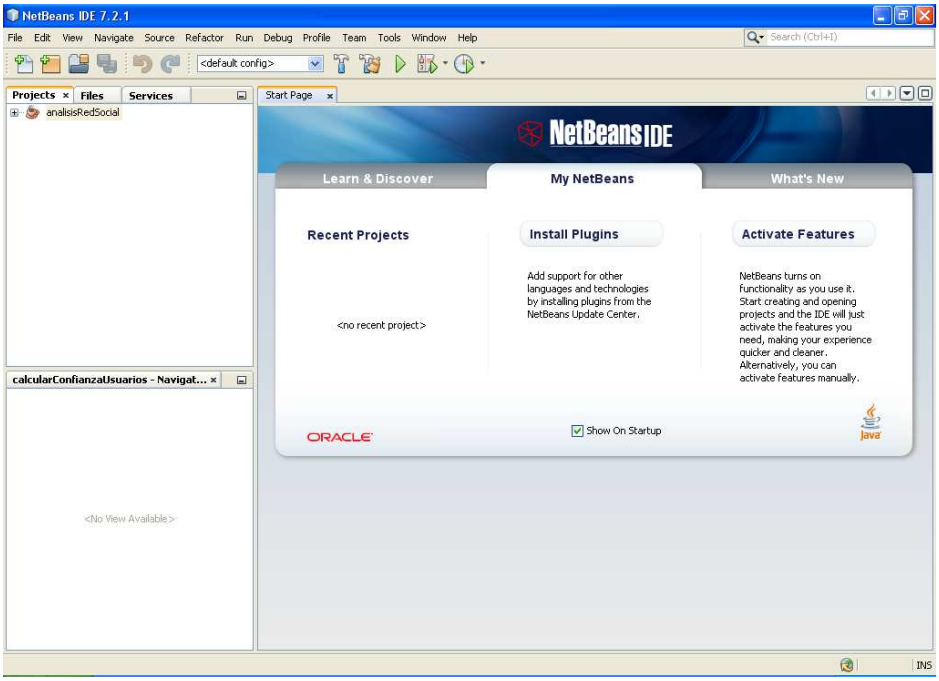

Ilustración 52: Netbeans

Para el manejo de la base de datos se ha utilizado la herramienta phpMyAdmin. Se trata de un software de código abierto diseñado para la administración y gestión de bases de datos MySQL a través de una interfaz gráfica de usuario (ver Ilustración 53).

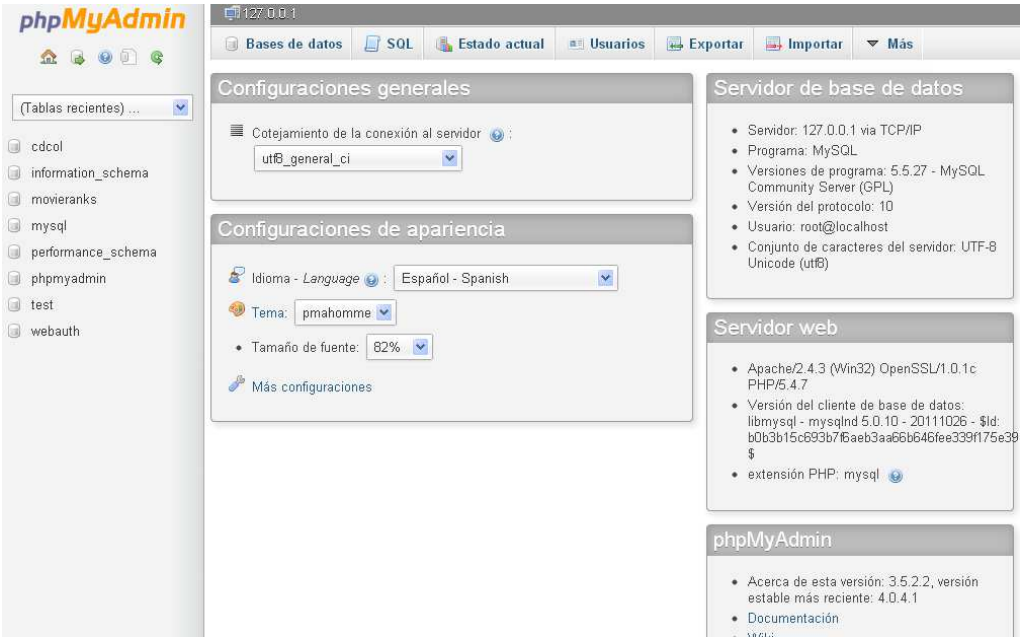

Ilustración 53: phpMyAdmin

4.6 Pruebas

El objetivo de esta fase es realizar la verificación y validación del sistema desarrollado, es decir, comprobar que el software realiza correctamente las tareas especificadas en los requisitos.

El desarrollo de sistemas software implica la realización de una serie de actividades en las que se pueden cometer errores debido a la naturaleza del ser humano. Por ello, es necesario incorporar una actividad que garantice la calidad del software. Aquí es donde adquiere un papel relevante la etapa de pruebas.

En este contexto, definimos como prueba el proceso de ejecución de un programa con la intención de descubrir errores potenciales. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

Habitualmente, las pruebas utilizadas para comprobar el funcionamiento de un sistema software son:

- **Pruebas de caja blanca**. Son pruebas estructurales que se desarrollan de forma que se asegure que la operación interna se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada. En estas pruebas se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos. Conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo

nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones.

- **Pruebas de caja negra**. Son pruebas funcionales. Se parte de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. La metodología a seguir consiste en observar la funcionalidad y contrastar con la especificación.

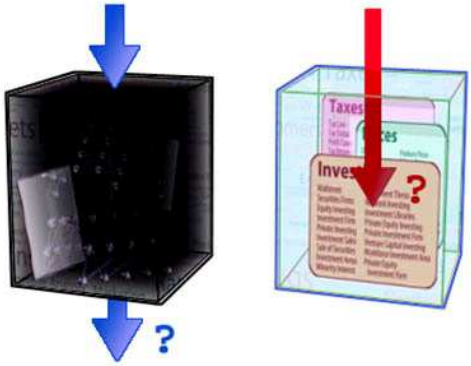

Ilustración 54: Pruebas de caja negra y de caja blanca

4.6.1 Pruebas de caja negra

Las pruebas de validación que se van a llevar a cabo sobre nuestro sistema son las de caja negra. Se ha elegido este tipo de pruebas porque resultan idóneas para realizarlas sobre la interfaz, obviando el comportamiento interno y la estructura del programa.

Ilustración 55: Pruebas de caja negra

Los casos de prueba de caja negra pretenden demostrar que:

- Las funcionalidades del sistema son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida adecuada.
- La integridad de la información externa se mantiene.

Por ello, en esta etapa se van a realizar una serie de pruebas de caja negra, con el objetivo de estudiar el comportamiento del sistema desde el punto de vista de las entradas que recibe y las salidas que produce, sin tener en cuenta su funcionamiento interno.

Los casos de prueba de caja negra describen la interacción del usuario con el sistema, especificando las acciones realizadas por el usuario en el sistema (entradas) y las salidas o respuestas esperadas, sin definir ni conocer los detalles internos que se producen. A cada una de las respuestas esperadas del sistema se les llama Checkpoint o puntos de comprobación.

A continuación se muestran los casos de prueba diseñados y los resultados obtenidos para cada uno de ellos.

4.6.2 Casos de prueba

Prueba 1: Cargar un grafo

Requisito testeado	RF-01
Acción	Seleccionar el grafo a cargar y pulsar sobre el botón *Aceptar*.
Checkpoint 1	El sistema debe mostrar la pantalla principal de la aplicación donde se podrá visualizar el grafo elegido.

Tabla 14: Prueba 1

Prueba 2: Definición correcta del término fuerte

Requisito testeado	RF-02
Acción	Introducir valores de *alfa* y *beta* pertenecientes al intervalo [0,1] (siendo alfa < beta) y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe actualizar la definición del término fuerte y mostrar en el panel de texto el cambio producido.

Tabla 15: Prueba 2

Prueba 3: Definición del término fuerte con un valor incorrecto de alfa

Requisito testeado	RF-02
Acción	Introducir un valor de *alfa* que no pertenezca al intervalo [0,1] y un valor de *beta* correcto y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que alfa debe ser un valor real entre 0 y 1.

Tabla 16: Prueba 3

Prueba 4: Definición del término fuerte con el valor de alfa vacío

Requisito testeado	RF-02
Acción	Dejar el campo de *alfa* vacío, introducir un valor de *beta* correcto y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que alfa debe ser un valor real entre 0 y 1.

Tabla 17: Prueba 4

Prueba 5: Definición del término fuerte con un tipo de dato erróneo en alfa

Requisito testeado	RF-02
Acción	Introducir un tipo de dato erróneo en *alfa* (carácter, cadena,…) y un valor de *beta* correcto y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que alfa debe ser un valor real entre 0 y 1.

Tabla 18: Prueba 5

Prueba 6: Definición del término fuerte con un valor incorrecto de beta

Requisito testeado	RF-02
Acción	Introducir un valor de *alfa* correcto y un valor de *beta* que no pertenezca al intervalo [0,1] y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que beta debe ser un valor real entre 0 y 1.

Tabla 19: Prueba 6

Prueba 7: Definición del término fuerte con el valor de beta vacío

Requisito testeado	RF-02
Acción	Introducir un valor de *alfa* correcto, dejar el campo de *beta* vacío y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que beta debe ser un valor real entre 0 y 1.

Tabla 20: Prueba 7

Prueba 8: Definición del término fuerte con un tipo de dato erróneo en beta

Requisito testeado	RF-02
Acción	Introducir un valor de *alfa* correcto, un tipo de dato erróneo en *beta* (carácter, cadena,…) y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que beta debe ser un valor real entre 0 y 1.

Tabla 21: Prueba 8

Prueba 9: Definición del término fuerte con alfa mayor o igual que beta

Requisito testeado	RF-02
Acción	Introducir valores de *alfa* y *beta* pertenecientes al intervalo [0,1], siendo alfa ≥ beta, y pulsar sobre el botón *Definir*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que alfa debe ser menor que beta.

Tabla 22: Prueba 9

Prueba 10: Obtener líderes

Requisito testeado	RF-03
Acción	Introducir el número *máximo de conexiones*, el valor *mínimo de fortaleza* y el número de *líderes* y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe resaltar en el grafo x líderes (según el número indicado) y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 23: Prueba 10

Prueba 11: Obtener líderes con todos los campos vacíos

Requisito testeado	RF-03
Acción	Pulsar sobre *Calcular* con los campos *Max. conexiones*, *Min. fortaleza* y Líderes vacíos.
Checkpoint 1	El sistema debe resaltar en el grafo el líder (sólo uno) y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 24: Prueba 11

Prueba 12: Obtener líderes con un valor incorrecto del número máximo de conexiones

Requisito testeado	RF-03
Acción	Introducir un número *máximo de conexiones* menor que 1 o mayor que el número de relaciones, el valor *mínimo de fortaleza* y el número de *líderes* y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el número máximo de conexiones debe ser un valor entero entre 1 y el número de relaciones.

Tabla 25: Prueba 12

Prueba 13: Obtener líderes con un tipo de dato erróneo en el campo número máximo de conexiones

Requisito testeado	RF-03
Acción	Introducir un tipo de dato erróneo en el campo número *máximo de conexiones* (carácter, cadena, real,…), el valor *mínimo de fortaleza* y el número de *líderes* y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el número máximo de conexiones debe ser un valor entero entre 1 y el número de relaciones.

Tabla 26: Prueba 13

Prueba 14: Obtener líderes con un valor incorrecto del valor mínimo de fortaleza

Requisito testeado	RF-03
Acción	Introducir el número *máximo de conexiones*, un valor en el campo *mínimo de* fortaleza que no pertenezca al intervalo [0,1] y el número de *líderes* y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el mínimo de fortaleza debe ser un valor real entre 0 y 1.

Tabla 27: Prueba 14

Prueba 15: Obtener líderes con un tipo de dato erróneo en el campo valor mínimo de fortaleza

Requisito testeado	RF-03
Acción	Introducir el número *máximo de conexiones*, un tipo de dato erróneo en el campo *mínimo de fortaleza* (carácter, cadena,…) y el número de *líderes* y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el mínimo de fortaleza debe ser un valor real entre 0 y 1.

Tabla 28: Prueba 15

Prueba 16: Obtener líderes con un valor incorrecto del campo número de líderes

Requisito testeado	RF-03
Acción	Introducir el número *máximo de conexiones*, el *mínimo de fortaleza* y un valor en el campo número de *líderes* que sea menor que 1 o mayor que el número de miembros y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el campo número de líderes debe ser un valor entero entre 1 y el número de miembros.

Tabla 29: Prueba 16

Prueba 17: Obtener líderes con un tipo de dato erróneo en el campo número de líderes

Requisito testeado	RF-03
Acción	Introducir el número *máximo de conexiones*, el *mínimo de fortaleza* y un tipo de dato erróneo en el campo número de *líderes* (carácter, cadena, real,…) y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el campo número de líderes debe ser un valor entero entre 1 y el número de miembros.

Tabla 30: Prueba 17

Prueba 18: Calcular camino mínimo

Requisito testeado	RF-04
Acción	Introducir en los campos *inicio* y *fin, respectivamente,* el valor de alguno de los nodos del grafo, y pulsar sobre *Calcular.*
Checkpoint 1	El sistema debe resaltar en el grafo el camino mínimo entre el nodo inicio y el nodo fin, y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 31: Prueba 18

Prueba 19: Calcular camino mínimo con el campo inicio vacío

Requisito testeado	RF-04
Acción	Dejar el campo *inicio* vacío e introducir en el campo *fin* el valor de alguno de los nodos del grafo, y pulsar sobre el botón *Calcular.*
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 32: Prueba 19

Prueba 20: Calcular camino mínimo con un tipo de dato erróneo en el campo inicio

Requisito testeado	RF-04
Acción	Introducir en el campo *inicio* un tipo de dato erróneo y en el campo *fin* el valor de alguno de los nodos del grafo, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 33: Prueba 20

Prueba 21: Calcular camino mínimo con un valor incorrecto en el campo inicio

Requisito testeado	RF-04
Acción	Introducir en el campo *inicio* un valor que no se corresponda con ninguno de los nodos y en el campo *fin* el valor de alguno de los nodos del grafo, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 34: Prueba 21

Prueba 22: Calcular camino mínimo con el campo fin vacío

Requisito testeado	RF-04
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo, dejar el campo *fin* vacío y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 35: Prueba 22

Prueba 23: Calcular camino mínimo con un tipo de dato erróneo en el campo fin

Requisito testeado	RF-04
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un tipo de dato erróneo, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 36: Prueba 23

Prueba 24: Calcular camino mínimo con un valor incorrecto en el campo fin

Requisito testeado	RF-04
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un valor que no se corresponda con ninguno de los nodos, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 37: Prueba 24

Prueba 25: Calcular camino mínimo con el mismo valor en los campos inicio y fin

Requisito testeado	RF-04
Acción	Introducir en el campo *inicio* y en el campo *fin* el mismo valor, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio y fin deben ser distintos.

Tabla 38: Prueba 25

Prueba 26: Calcular camino más fuerte

Requisito testeado	RF-05
Acción	Introducir en los campos *inicio* y *fin, respectivamente,* el valor de alguno de los nodos del grafo, y pulsar sobre *Calcular.*
Checkpoint 1	El sistema debe resaltar en el grafo el camino más fuerte entre el nodo inicio y el nodo fin, y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 39: Prueba 26

Prueba 27: Calcular camino más fuerte con el campo inicio vacío

Requisito testeado	RF-05
Acción	Dejar el campo *inicio* vacío e introducir en el campo *fin* el valor de alguno de los nodos del grafo, y pulsar sobre el botón *Calcular.*
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 40: Prueba 27

Prueba 28: Calcular camino más fuerte con un tipo de dato erróneo en el campo inicio

Requisito testeado	RF-05
Acción	Introducir en el campo *inicio* un tipo de dato erróneo y en el campo *fin* el valor de alguno de los nodos del grafo, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 41: Prueba 28

Prueba 29: Calcular camino más fuerte con un valor incorrecto en el campo inicio

Requisito testeado	RF-05
Acción	Introducir en el campo *inicio* un valor que no se corresponda con ninguno de los nodos y en el campo *fin* el valor de alguno de los nodos del grafo, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 42: Prueba 29

Prueba 30: Calcular camino más fuerte con el campo fin vacío

Requisito testeado	RF-05
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo, dejar el campo *fin* vacío y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 43: Prueba 30

Prueba 31: Calcular camino más fuerte con un tipo de dato erróneo en el campo fin

Requisito testeado	RF-05
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un tipo de dato erróneo, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 44: Prueba 31

Prueba 32: Calcular camino más fuerte con un valor incorrecto en el campo fin

Requisito testeado	RF-05
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un valor que no se corresponda con ninguno de los nodos, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 45: Prueba 32

Prueba 33: Calcular camino más fuerte con el mismo valor en los campos inicio y fin

Requisito testeado	RF-05
Acción	Introducir en el campo *inicio* y en el campo *fin* el mismo valor, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio y fin deben ser distintos.

Tabla 46: Prueba 33

Prueba 34: Calcular camino mínimo más fuerte

Requisito testeado	RF-06
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe resaltar en el grafo el camino mínimo más fuerte entre el nodo inicio y el nodo fin, y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 47: Prueba 34

Prueba 35: Calcular camino mínimo más fuerte con el campo inicio vacío

Requisito testeado	RF-06
Acción	Dejar el campo *inicio* vacío, introducir en el campo *fin* el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 48: Prueba 35

Prueba 36: Calcular camino mínimo más fuerte con un tipo de dato erróneo en el campo inicio

Requisito testeado	RF-06
Acción	Introducir en el campo *inicio* un tipo de dato erróneo y en el campo *fin* el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 49: Prueba 36

Prueba 37: Calcular camino mínimo más fuerte con un valor incorrecto en el campo inicio

Requisito testeado	RF-06
Acción	Introducir en el campo *inicio* un valor que no se corresponda con ninguno de los nodos y en el campo *fin* el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* a tener en cuenta y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 50: Prueba 37

Prueba 38: Calcular camino mínimo más fuerte con el campo fin vacío

Requisito testeado	RF-06
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo, dejar el campo *fin* vacío, indicar el número *máximo de conexiones* y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 51: Prueba 38

Prueba 39: Calcular camino mínimo más fuerte con un tipo de dato erróneo en el campo fin

Requisito testeado	RF-06
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un tipo de dato erróneo, indicar el número *máximo de conexiones* a tener en cuenta y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 52: Prueba 39

Prueba 40: Calcular camino mínimo más fuerte con un valor incorrecto en el campo fin

Requisito testeado	RF-06
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un valor que no se corresponda con ninguno de los nodos, indicar el número máximo de conexiones y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 53: Prueba 40

Prueba 41: Calcular camino mínimo más fuerte con el mismo valor en los campos inicio y fin

Requisito testeado	RF-06
Acción	Introducir en el campo *inicio* y en el campo *fin* el mismo valor, indicar el número *máximo de conexiones* a tener en cuenta y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio y fin deben ser distintos.

Tabla 54: Prueba 41

Prueba 42: Calcular camino mínimo más fuerte con el campo número máximo de conexiones vacío

Requisito testeado	RF-06
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, dejar el campo número *máximo de conexiones* vacío y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe resaltar en el grafo el camino mínimo más fuerte entre el nodo inicio y el nodo fin, y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 55: Prueba 42

Prueba 43: Calcular camino mínimo más fuerte con un tipo de dato erróneo en el campo número máximo de conexiones

Requisito testeado	RF-06
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, un tipo de dato erróneo en el campo número *máximo de conexiones* (cadena, carácter, real,…) y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el número máximo de conexiones debe ser un valor entero entre 1 y el número de relaciones.

Tabla 56: Prueba 43

Prueba 44: Calcular camino mínimo más fuerte con un valor incorrecto en el campo número máximo de conexiones

Requisito testeado	RF-06
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, un valor incorrecto en el campo número *máximo de conexiones* (menor que 1 o mayor que el número de relaciones) y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el número máximo de conexiones debe ser un valor entero entre 1 y el número de relaciones.

Tabla 57: Prueba 44

Prueba 45: Calcular posibles caminos

Requisito testeado	RF-07
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* y el criterio de ordenación, y pulsar sobre *Calcular*.
Checkpoint 1	El sistema mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 58: Prueba 45

Prueba 46: Calcular posibles caminos con el campo inicio vacío

Requisito testeado	RF-07
Acción	Dejar el campo *inicio* vacío, introducir en el campo *fin* el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* y el criterio de ordenación, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 59: Prueba 46

Prueba 47: Calcular posibles caminos con un tipo de dato erróneo en el campo inicio

Requisito testeado	RF-07
Acción	Introducir en el campo *inicio* un tipo de dato erróneo y en el campo *fin* el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* y el criterio de ordenación, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 60: Prueba 47

Prueba 48: Calcular posibles caminos con un valor incorrecto en el campo inicio

Requisito testeado	RF-07
Acción	Introducir en el campo *inicio* un valor que no se corresponda con ninguno de los nodos y en el campo *fin* el valor de alguno de los nodos del grafo, indicar el número *máximo de conexiones* a tener en cuenta y el criterio de ordenación a seguir, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio debe ser el valor de un nodo.

Tabla 61: Prueba 48

Prueba 49: Calcular posibles caminos con el campo fin vacío

Requisito testeado	RF-07
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo, dejar el campo *fin* vacío, indicar el número *máximo de conexiones* y el criterio de ordenación, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 62: Prueba 49

Prueba 50: Calcular camino posibles caminos con un tipo de dato erróneo en el campo fin

Requisito testeado	RF-07
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un tipo de dato erróneo, indicar el número *máximo de conexiones* a tener en cuenta y el criterio de ordenación a seguir, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 63: Prueba 50

Prueba 51: Calcular posibles caminos con un valor incorrecto en el campo fin

Requisito testeado	RF-07
Acción	Introducir en el campo *inicio* el valor de alguno de los nodos del grafo y en el campo *fin* un valor que no se corresponda con ninguno de los nodos, indicar el número máximo de conexiones y el criterio de ordenación, y pulsar sobre el botón *Calcular.*
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que fin debe ser el valor de un nodo.

Tabla 64: Prueba 51

Prueba 52: Calcular posibles caminos con el mismo valor en los campos inicio y fin

Requisito testeado	RF-07
Acción	Introducir en el campo *inicio* y en el campo *fin* el mismo valor, indicar el número *máximo de conexiones* a tener en cuenta y el criterio de ordenación, y pulsar sobre el botón *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que inicio y fin deben ser distintos.

Tabla 65: Prueba 52

Prueba 53: Calcular posibles caminos con el campo número máximo de conexiones vacío

Requisito testeado	RF-07
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, dejar el campo número *máximo de conexiones* vacío, indicar el criterio de ordenación y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe resaltar en el grafo el camino mínimo más fuerte entre el nodo inicio y el nodo fin, y mostrar los criterios utilizados y el resultado obtenido en el panel de texto.

Tabla 66: Prueba 53

Prueba 54: Calcular posibles caminos con un tipo de dato erróneo en el campo número máximo de conexiones

Requisito testeado	RF-07
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, un tipo de dato erróneo en el campo número *máximo de conexiones* (cadena, carácter, real,…), indicar el criterio de ordenación y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el número máximo de conexiones debe ser un valor entero entre 1 y el número de relaciones.

Tabla 67: Prueba 54

Prueba 55: Calcular posibles caminos con un valor incorrecto en el campo número máximo de conexiones

Requisito testeado	RF-07
Acción	Introducir en los campos *inicio* y *fin*, respectivamente, el valor de alguno de los nodos del grafo, un valor incorrecto en el campo número *máximo de conexiones* (menor que 1 o mayor que el número de relaciones), indicar el criterio de ordenación a seguir y pulsar sobre *Calcular*.
Checkpoint 1	El sistema debe mostrar un mensaje de error indicando que el número máximo de conexiones debe ser un valor entero entre 1 y el número de relaciones.

Tabla 68: Prueba 55

Prueba 56: Obtener información sobre un miembro de la red social

Requisito testeado	RF-08
Acción	Posicionar el cursor del ratón encima de un nodo.
Checkpoint 1	El sistema debe mostrar la información del usuario representado por dicho nodo.

Tabla 69: Prueba 56

Prueba 57: Salir de la aplicación

Requisito testeado	RF-09
Acción	Pulsar cerrar aplicación.
Checkpoint 1	El sistema debe cerrar la aplicación.

Tabla 70: Prueba 57

4.6.3 Resultados obtenidos

Una vez diseñadas las pruebas, es necesario comprobar cada uno de los checkpoints existentes para verificar que las respuestas obtenidas se ajustan con las esperadas.

A continuación se presentan los resultados obtenidos tras realizar sobre la aplicación cada una de las pruebas especificadas en el apartado anterior.

Prueba 1	**Resultado**
Checkpoint 1	OK
Prueba 2	**Resultado**
Checkpoint 1	OK
Prueba 3	**Resultado**
Checkpoint 1	OK
Prueba 4	**Resultado**
Checkpoint 1	OK
Prueba 5	**Resultado**
Checkpoint 1	OK
Prueba 6	**Resultado**
Checkpoint 1	OK
Prueba 7	**Resultado**
Checkpoint 1	OK
Prueba 8	**Resultado**
Checkpoint 1	OK
Prueba 9	**Resultado**
Checkpoint 1	OK

Prueba 10	**Resultado**
Checkpoint 1	OK
Prueba 11	**Resultado**
Checkpoint 1	OK
Prueba 12	**Resultado**
Checkpoint 1	OK
Prueba 13	**Resultado**
Checkpoint 1	OK
Prueba 14	**Resultado**
Checkpoint 1	OK
Prueba 15	**Resultado**
Checkpoint 1	OK
Prueba 16	**Resultado**
Checkpoint 1	OK
Prueba 17	**Resultado**
Checkpoint 1	OK
Prueba 18	**Resultado**
Checkpoint 1	OK
Prueba 19	**Resultado**
Checkpoint 1	OK
Prueba 20	**Resultado**
Checkpoint 1	OK

Prueba 21	**Resultado**
Checkpoint 1	OK
Prueba 22	**Resultado**
Checkpoint 1	OK
Prueba 23	**Resultado**
Checkpoint 1	OK
Prueba 24	**Resultado**
Checkpoint 1	OK
Prueba 25	**Resultado**
Checkpoint 1	OK
Prueba 26	**Resultado**
Checkpoint 1	OK
Prueba 27	**Resultado**
Checkpoint 1	OK
Prueba 28	**Resultado**
Checkpoint 1	OK
Prueba 29	**Resultado**
Checkpoint 1	OK
Prueba 30	**Resultado**
Checkpoint 1	OK
Prueba 31	**Resultado**
Checkpoint 1	OK

Prueba 32	**Resultado**
Checkpoint 1	OK
Prueba 33	**Resultado**
Checkpoint 1	OK
Prueba 34	**Resultado**
Checkpoint 1	OK
Prueba 35	**Resultado**
Checkpoint 1	OK
Prueba 36	**Resultado**
Checkpoint 1	OK
Prueba 37	**Resultado**
Checkpoint 1	OK
Prueba 38	**Resultado**
Checkpoint 1	OK
Prueba 39	**Resultado**
Checkpoint 1	OK
Prueba 40	**Resultado**
Checkpoint 1	OK
Prueba 41	**Resultado**
Checkpoint 1	OK
Prueba 42	**Resultado**
Checkpoint 1	OK

Prueba 43	**Resultado**
Checkpoint 1	OK
Prueba 44	**Resultado**
Checkpoint 1	OK
Prueba 45	**Resultado**
Checkpoint 1	OK
Prueba 46	**Resultado**
Checkpoint 1	OK
Prueba 47	**Resultado**
Checkpoint 1	OK
Prueba 48	**Resultado**
Checkpoint 1	OK
Prueba 49	**Resultado**
Checkpoint 1	OK
Prueba 50	**Resultado**
Checkpoint 1	OK
Prueba 51	**Resultado**
Checkpoint 1	OK
Prueba 52	**Resultado**
Checkpoint 1	OK
Prueba 53	**Resultado**
Checkpoint 1	OK

Prueba 54	**Resultado**
Checkpoint 1	OK
Prueba 55	**Resultado**
Checkpoint 1	OK
Prueba 56	**Resultado**
Checkpoint 1	OK
Prueba 57	**Resultado**
Checkpoint 1	OK

Tabla 71: Resultados de las pruebas

Capítulo 5. Conclusiones.

5.1 Conclusiones

El ser humano se comunica a cada instante, ya sea de forma verbal o no verbal, por lo que continuamente busca nuevos medios que le permitan comunicarse de manera efectiva para cubrir sus necesidades. Este hecho unido al desarrollo de Internet y a los continuos avances tecnológicos lleva a desarrollar las redes sociales.

Las redes sociales son estructuras sociales formadas por un conjunto de actores, tales como individuos u organizaciones, entre los que existe algún tipo de relación. Surgen como una importante herramienta de comunicación entre las personas y su éxito radica en la facilidad para intercambiar opiniones y contenido entre los individuos que forman parte de ellas.

El papel fundamental que tienen las redes sociales en la actualidad unido a su influencia, hacen que resulte interesante desarrollar herramientas que permitan llevar a cabo un análisis de las mismas.

El análisis de redes sociales estudia esta estructura social aplicando la teoría de grafos, identificando las entidades como nodos o vértices y las relaciones como enlaces o aristas. Pero los conceptos formales relacionados con esta teoría a menudo resultan complejos. Por ello, con este proyecto se ha pretendido avanzar un poco más, utilizando términos lingüísticos que ayuden al ser humano a comprender estos conceptos. Es aquí donde juega un papel fundamental el paradigma de computación con palabras de Zadeh.

El uso de la computación con palabras resulta de gran utilidad en el análisis de redes sociales, ya que facilita la compresión de conceptos matemáticos complejos, estableciendo una correspondencia entre los valores numéricos utilizados por las máquinas para llevar a cabo los distintos cálculos, y la forma de comunicación de los seres humanos mediante términos lingüísticos.

Por ello, la herramienta desarrollada puede ser de gran utilidad, ya que permite determinar cuáles son los líderes de una estructura social, qué camino seguir para difundir un mensaje lo más rápido posible, cómo comunicarnos con una persona utilizando intermediarios fiables, etc., todo ello sin necesidad de comprender los conceptos formales en los que se fundamenta.

Finalmente, comentar que, la realización de este proyecto me ha permitido poner en práctica muchas de las metodologías y habilidades adquiridas durante los años de formación académica. Además, me ha ayudado a profundizar en temas que hasta entonces desconocía, como el paradigma de computación con palabras de Zadeh y la representación de grafos mediante la herramienta JUNG de Java.

5.2 Trabajos futuros

En el futuro se podrían abordar diferentes mejoras en el sistema desarrollado, tales como:

- Ofrecer soporte para distintos idiomas (inglés, alemán, japonés, francés,…).
- Permitir descargar un PDF con todo el análisis llevado a cabo.
- Incorporar nuevos métodos de análisis.

Bibliografía

Bibliografía

[1] "Concept Representation and Database Structures in Fuzzy Social Relational Networks", Ronald R. Yager, Fellow, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 40, NO. 2, MARCH 2010.

[2] "Fuzzy Logic = Computing with Words", Lotfi A. Zadeh, Life Fellow, IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 4, NO. 2, MAY 1996.

[3] "Intelligent Social Network Analysis Using Granular Computing", Ronald R. Yager∗, Machine Intelligence Institute, Iona College, New Rochelle, NY 10801.

[4] "Fuzzy graphs", A. Rosenfeld in Fuzzy Sets and Their Applications to Cognitive and Decision Processes, L. A. Zadeh, K. S. Fu, K. Tanaka and M. Shimura, Eds. New York: Academic, 1975, pp. 77-97.

[5] "Fuzzy graph in the evaluation and optimization of networks", L. T. Koczy, Fuzzy Sets Syst., vol. 46, no. 3, pp. 307-319, Mar. 1992.

[6] "An Effective Recommender System by Unifying User and Item Trust Information for B2B Applications", Qusai Shambour, Jie Lu.

[7] "Using trust in collaborative filtering recommendation", C.- S. Hwang, Y. – P. Chen, in H. Okuno, M. Ali Eds. New Trends in Applied Artificial Intelligence, Springer, Heidelberg, 2007, pp 1052-1060.

[8] "GroupLens: an open architecture for collaborative filtering of netnews", P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Procedings of the 1994 ACM Conference on Computer Supported Cooperative Work, 1994, pp. 175-186.

[9] "Social information filtering: algorithm for automating 'word of mouth'", U. Shardanand, P. Maes, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1995, pp. 210-217.

[10] "Data Mining: Concepts and Techniques", J. Han, M. Kamber, 2nd edition, Morgan Kaufmann, San Francisco, 2006.

[11] "The concept of a linguistic variable and its application to approximate reasoning: Part 1", L. Zadeh, Inf. Sci., vol. 8, no. 3, pp. 199-249, Jan. 1975.

[12] "Fuzzy measures and fuzzy integrals", T. Murofushi and M. Sugeno, in Fuzzy Measures and Integrals, M. Grabisch, T. Murofushi and M. Sugeno, Eds. Berlin, Germany: Physica – Verlag, 2000, pp. 3-41.

[13] "Aggregation Functions: A Guide for Practitioners", G. Beliakov, A. Pradera and T. Calvo, Eds. Berlin, Germany: Springer – Verlag, 2007.

[14] "Social Network Analysis", J. Scott, Los Angeles, CA: SAGE, 2000.

[15] "Head First Object-Oriented Analysis and Design", Brett D. McLaughli, Pollice, G., West, D., Ed. O'Reilly.

[16] "Ingeniería del software. Un enfoque práctico", 6ª Edición. McGraw-Hill, 2005.

[17] "No me hagas pensar: una aproximación a la Usabilidad en la Web". Krug, S., Ed. Prentice Hall

[18] "Cómo hacer una buena interfaz de usuario y no morir en el intento". Conde, Francisco. Apuntes de la asignatura.

[19] "Ingeniería del software". Benet Campderrich Falgueras, Ed.UOC.

[20] "Head First Design Patterns". Freeaman, E., Sierra, K., Bert Bates. Ed. O´Reilly.

[21] "Análisis y Gestión de Datos". Feito Higueruela, F., Ruiz de Miras, J., Molina Aguilar, A., Universidad de Jaén, 1996.

[22] Lenguaje R. http://www.r-project.org/

[23] Web IGraph. http://igraph.sourceforge.net/introduction.html

[24] Web JUNG. http://jung.sourceforge.net/index.html

[25] "Analysis and visualization of network data using JUNG". Joshua O'Madadhain, Danyel Fisher, Padhraic Smyth, White, and Yan-Biao Boey.Journal of Statistical Soft-ware, VV(2), 2005.

[26] Web Gephi. http://gephi.org/

[27] Toolkit Gephi. http://gephi.org/toolkit/

Anexo A. Manual de instalación del servidor de bases de datos.

A.1 Introducción

En este manual se refleja el proceso de instalación que hay que llevar a cabo en el equipo del usuario para que la aplicación funcione correctamente.

Antes de explicar los pasos a seguir para instalar cada una de las herramientas necesarias, han de tenerse en cuenta dos consideraciones previas. Durante todo el manual se ha supuesto que la unidad principal de disco duro es C: y que la unidad principal de disco óptico es D: . Además, toda la instalación y configuración descrita en este manual se refiere a un servidor montado sobre un Sistema Operativo Windows.

Todo el material necesario para poner en funcionamiento la aplicación se encuentra disponible en el CD que acompaña a esta memoria.

A continuación se muestran los pasos a seguir para instalar cada una de las herramientas necesarias. En primer lugar, hay que introducir el CD con todo el material, situarse en la unidad D: \ AnalisisRedSocial y comprobar que se encuentran los siguientes archivos:

- **xampp-win32-1.8.1-VC9-installer.exe**
- **movieranks.sql**
- **crearBaseDeDatos.sql**
- **importarBaseDeDatos.bat**
- **analisisRedSocial.jar**

Si es así, puede comenzar con la instalación. En caso de que falte algún archivo o esté dañado puede ponerse en contacto con el responsable de la aplicación para solucionar el problema.

A.2 XAMPP

Para que el funcionamiento de la aplicación sea correcto, el equipo debe disponer del gestor de bases de datos MySQL y de Apache. Es posible realizar la instalación de cada uno de estos elementos por separado pero luego deberíamos proceder a configurarlos. Para evitar tener que realizar el tedioso proceso de configuración vamos a utilizar una herramienta llamada Xampp que integra a ambos componentes y nos permite una instalación y configuración transparente para el usuario.

Xampp es una herramienta formada por un servidor web Apache, una base de datos MySQLy los intérpretes para los lenguajes PHP y Perl. Es independiente de la plataforma y tiene licencia GNU. Su nombre procede del acrónimo X (para cualquier Sistema Operativo), A (Apache), P (PHP) y P (Perl). Actualmente está disponible para Windows, Linux, Solaris y MacOS X.

Ilustración 56: Xampp

Para proceder con la instalación de Xampp podemos descargar su última versión de la página http://www.apachefriends.org/es/xampp.html o bien utilizar el archivo ejecutable que se ha facilitado en el CD (xampp-win32-1.8.1-VC9-installer.exe).

Tras hacer doble click sobre el archivo ejecutable se iniciará el asistente para la instalación y nos pedirá que seleccionemos el idioma de instalación (ver Ilustración 57).

Ilustración 57: Selección idioma instalación

Al elegir el idioma y pulsar el botón ´*OK*´ aparecerá una ventana de bienvenida (ver Ilustración 58).

Ilustración 58: Pantalla bienvenida Xampp

Tras pulsar el botón *´Next >´* el asistente nos pedirá que seleccionemos los componentes que deseamos instalar en nuestro equipo. Marcamos todos los componentes y pulsamos de nuevo el botón *´Next >´*. (Ver Ilustración 59).

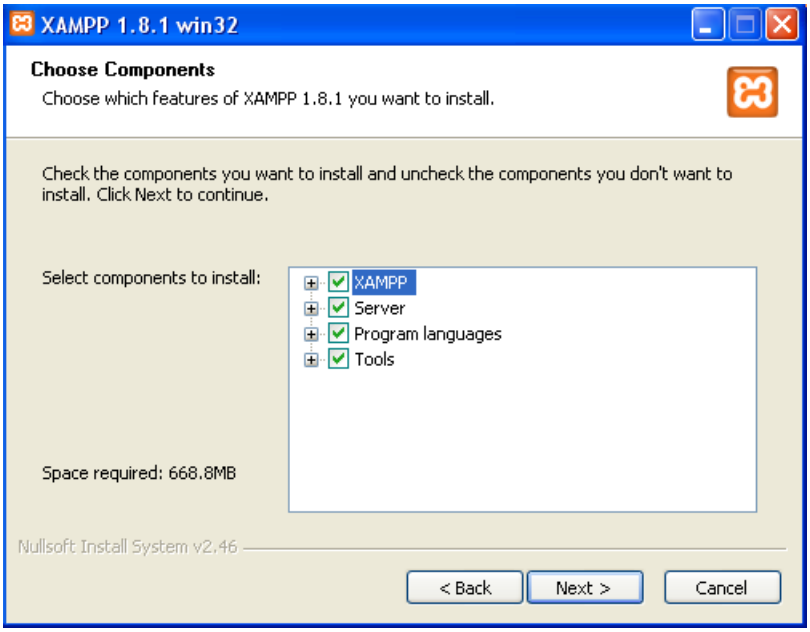

Ilustración 59: Componentes Xampp

A continuación, el asistente nos indicará en qué directorio se va a instalar la aplicación. Podemos especificar cuál es el directorio en el que deseamos instalarla pero es preferible dejar el directorio por defecto (ver Ilustración 60).

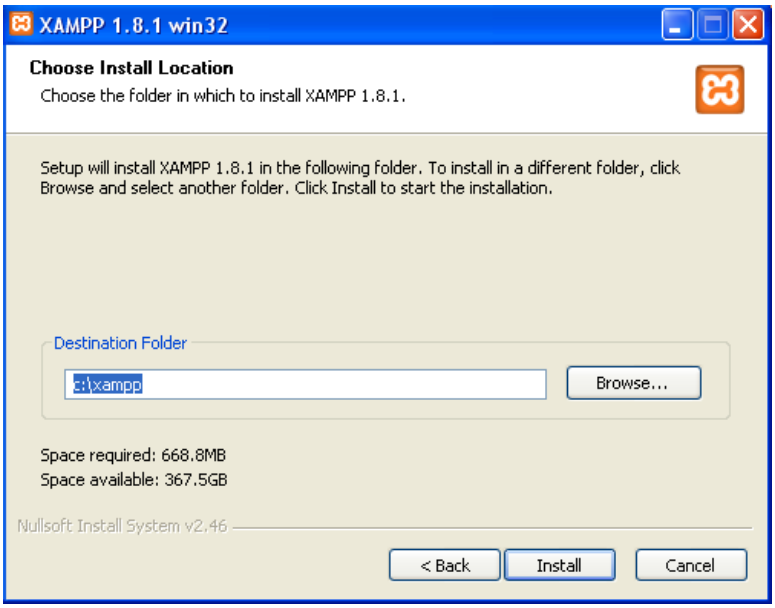

Ilustración 60: Directorio instalación Xampp

Una vez seleccionado el directorio pulsamos el botón *´Install´* para comenzar la instalación.

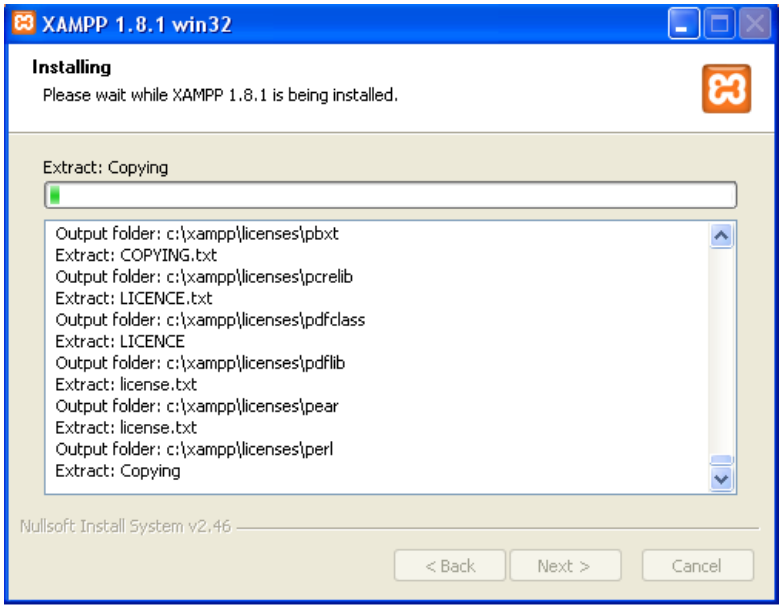

Ilustración 61: Progreso de instalación Xampp

Por último, aparecerá una pantalla informándonos de que la herramienta ha sido instalada con éxito en nuestro equipo. En ella debemos pulsar el botón *´Finish´*. (Ver Ilustración 62).

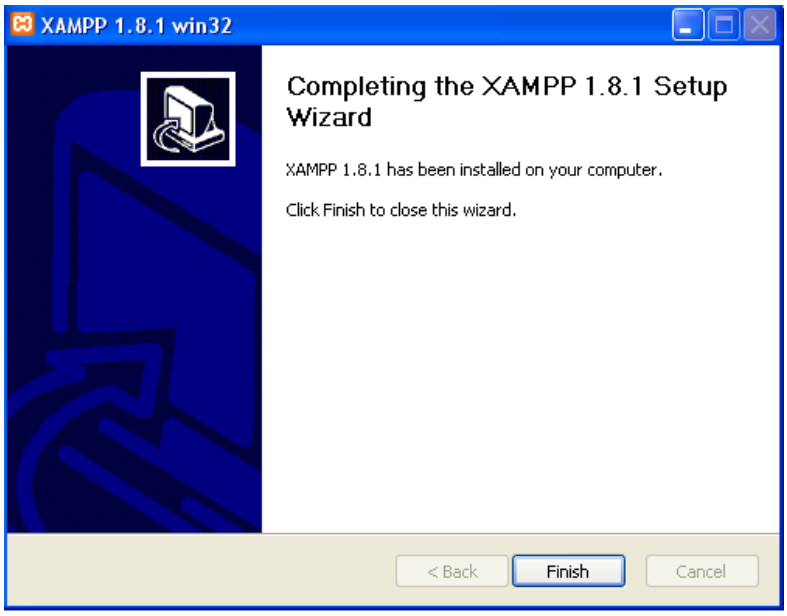

Ilustración 62: Instalación Xampp finalizada

Seguidamente, la aplicación nos preguntará si queremos iniciar el Panel de Control de Xampp. Elegimos la opción *´Sí´* para poder visualizar las funcionalidades que nos ofrece (ver Ilustración 63).

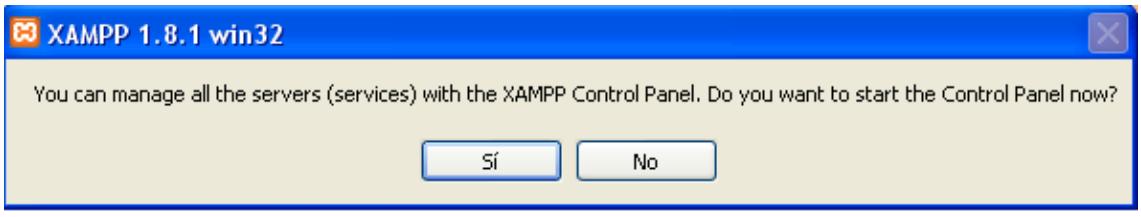

Ilustración 63: Iniciar panel Xampp después de la instlación

Si observamos el panel de control de Xampp, mostrado en la Ilustración 64, podemos comprobar que cuenta con el gestor de bases de datos MySQL y con el servidor web Apache, que son las herramientas necesarias para poner en marcha nuestra aplicación.

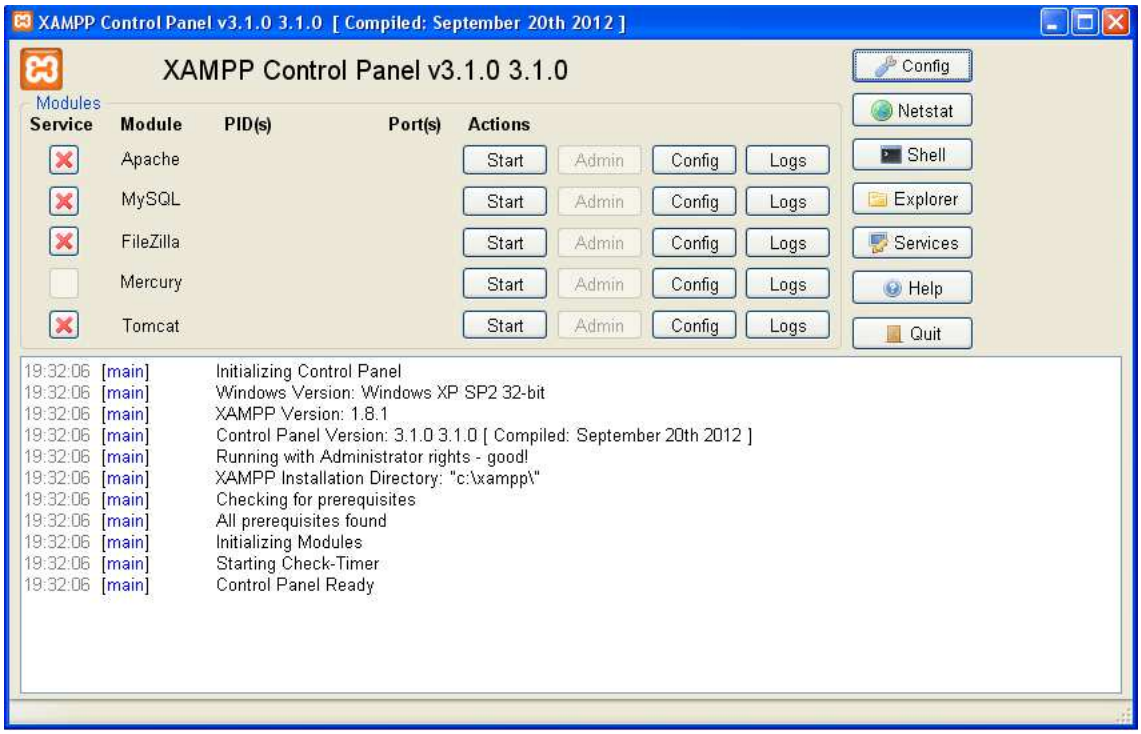

Ilustración 64: Panel Xampp

A continuación se explica cómo poner en marcha cada uno de estos componentes.

Apache

Para activar el servicio de Apache basta con pulsar sobre el botón *´Start´* correspondiente a dicho servicio (ver Ilustración 65).

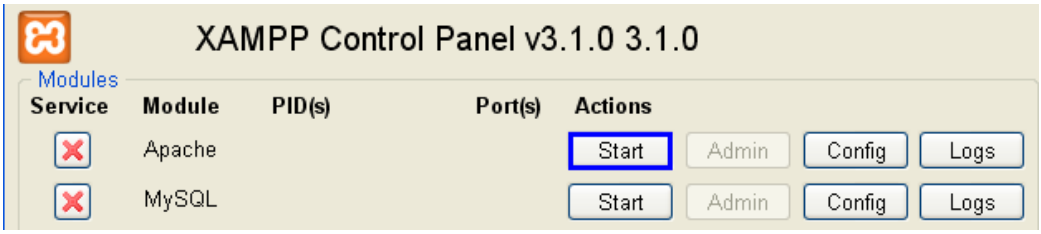

Ilustración 65: Iniciar Apache

Si se ha iniciado correctamente aparecerá la frase´ [Apache] *Status change detected running´* en el panel de salida de texto (ver Ilustración 66).

Ilustración 66: Iniciación correcta de Apache

MySQL

Para manejar el gestor de bases de datos MySQL es necesario pulsar sobre el botón ´*Start*´ correspondiente a dicho servicio (ver Ilustración 67).

Ilustración 67: Iniciar MySQL

Si se ha iniciado correctamente aparecerá en el panel de salida de texto la siguiente frase: ´ [mysql] *Status change detected running´* (ver Ilustración 68).

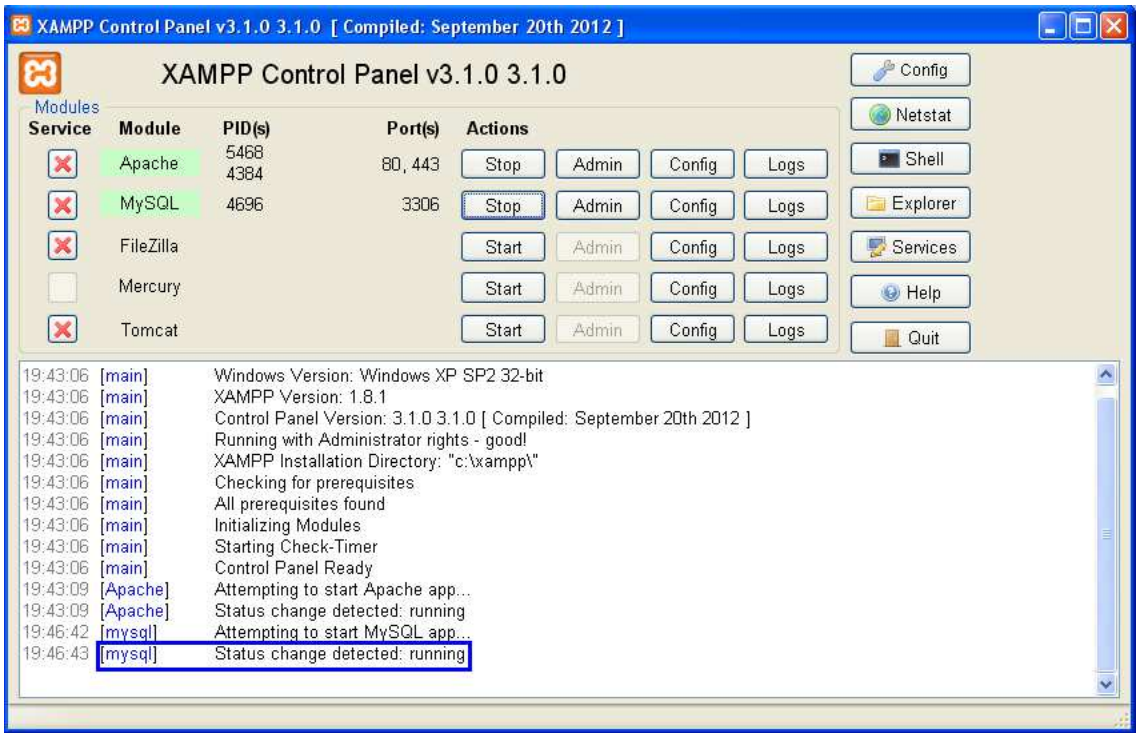

Ilustración 68: Iniciación correcta MySQL

A.3 Importar la base de datos

Una vez instalada la aplicación XAMPP es el momento de importar la base de datos al servidor. Para ello, es necesario situar los archivos movieranks.sql, crearBaseDeDatos.sql e importarBaseDeDatos.bat en el directorio C:\xampp (ver Ilustración 69).

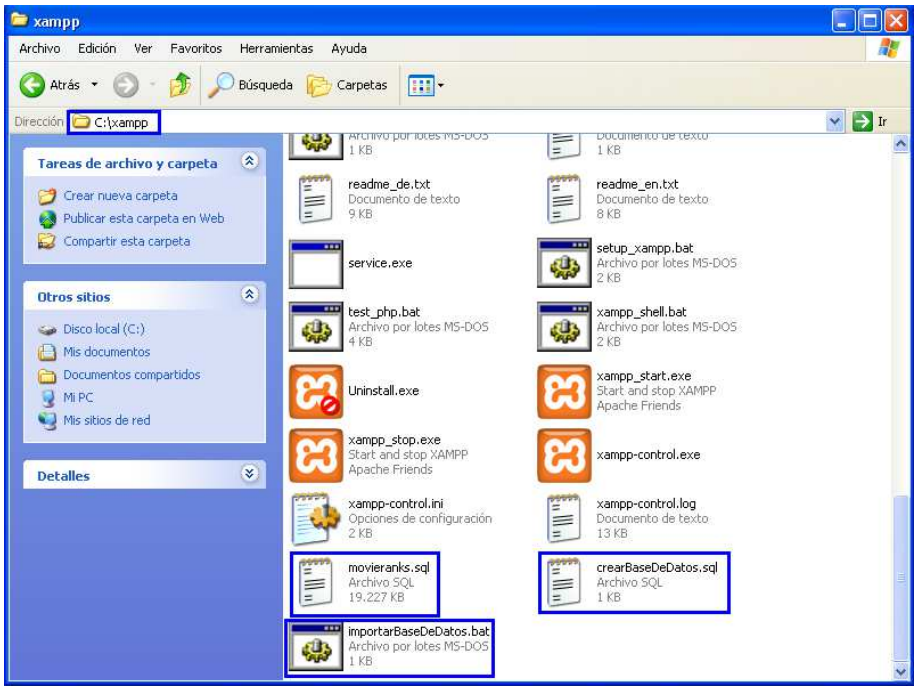

Ilustración 69: Situar archivos directorio Xampp

Tras situar los archivos en el directorio indicado haremos doble click sobre el archivo importarBaseDeDatos.dat (ver Ilustración 70), que contiene los comandos necesarios para crear la base de datos e importar la información.

Ilustración 70: Importar base de datos

Tras hacer doble click sobre este archivo, aparecerá una ventana que se mostrará activa durante el proceso de importación (ver Ilustración 71). Es muy importante no cerrar esta ventana, ya que si lo hacemos la base de datos podría quedar corrupta. Una vez finalizada la importación, la ventana se cerrará automáticamente.

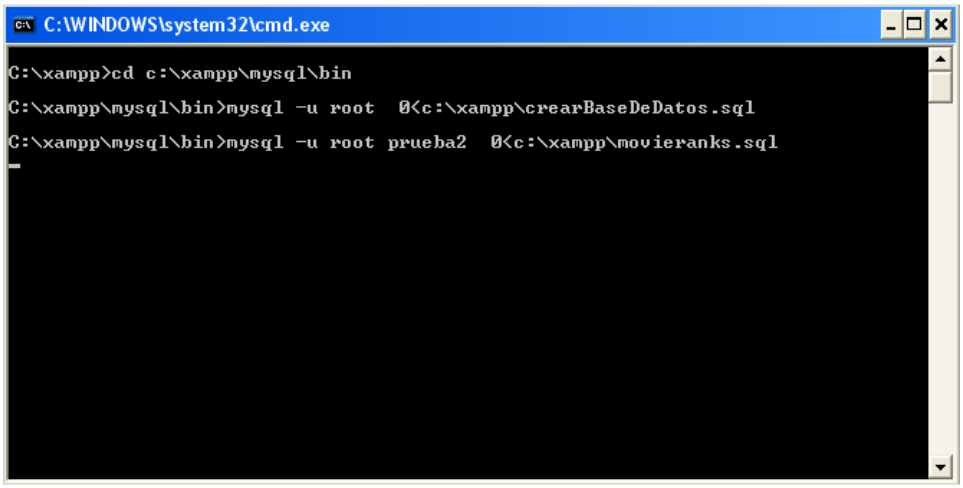

Ilustración 71: Proceso de importación

Anexo B. Manual de usuario.

B.1 Introducción

En este anexo se muestra, a modo de visita guiada, un manual de la aplicación que puede ser de gran utilidad para el usuario.

Este manual pretende dar una visión amplia de los contenidos de la aplicación desarrollada para el análisis de redes sociales. En él, se van a explicar todos los detalles de la aplicación con el objetivo de resolver cualquier duda que el usuario pueda tener a la hora de utilizarla.

Para poder usar la aplicación es necesario tener instalada la Máquina Virtual de Java (Java Virtual machine, JVM). En caso de no disponer de ella puede descargarla en la página http://www.oracle.com/technetwork/es/java/javase/downloads/index.html. Para ello tendrá que descargar el JRE (Java Runtime Environment) que simplemente es el conjunto de aplicaciones y librerías necesarias para poder utilizar aplicaciones Java (contiene la JVM). Además, para poder utilizar todas las funcionalidades que ofrece la aplicación hay que instalar el servidor de bases de datos, para ello ver el Anexo A, "Manual de instalación del servidor de bases de datos".

B.2 Abrir aplicación

El primer paso para utilizar la aplicación es permitir su conexión a la base de datos. Para ello es necesario abrir la herramienta Xampp e iniciar los servicios de Apache y MySQL pulsando el botón *´Start´* correspondiente a cada uno de estos servicios (ver Ilustración 72).

Ilustración 72: Iniciar Apache y MySQL

El segundo paso es abrir el archivo analisisRedSocial.jar. Para ello basta con hacer doble click sobre el icono (ver Ilustración 73).

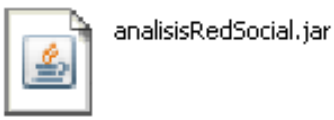

Ilustración 73: Archivo aplicación

También es posible ejecutar la aplicación a través de la línea de comandos, escribiendo la orden java –jar analisisRedSocial.jar una vez que nos hayamos desplazado al directorio en el que se encuentra la aplicación (ver Ilustración 74).

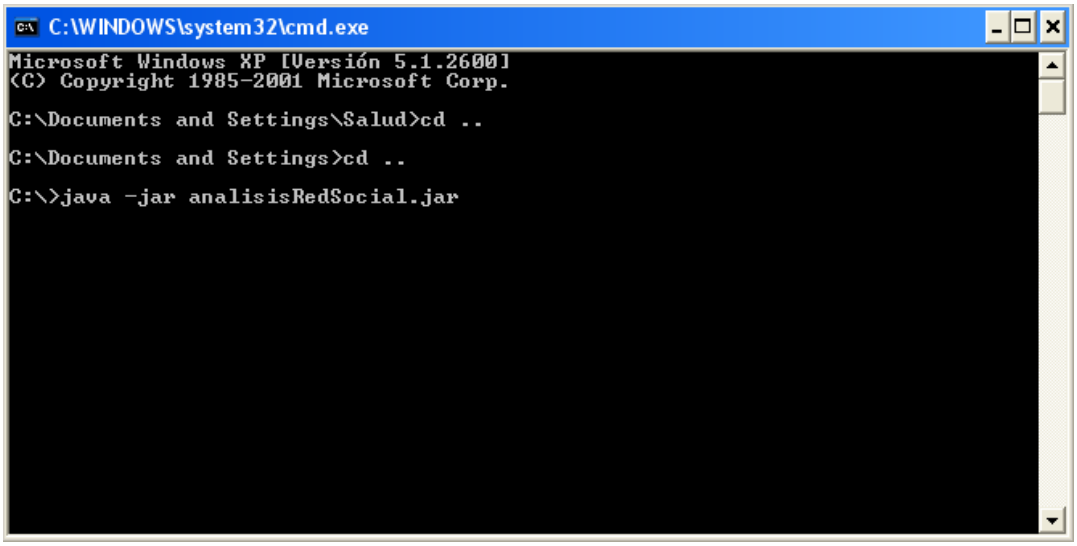

Ilustración 74: Ejecutar aplicación desde la línea de comandos

Al abrir la aplicación, se mostrará una ventana que permitirá elegir el grafo representativo de la red social a analizar (ver Ilustración 75).

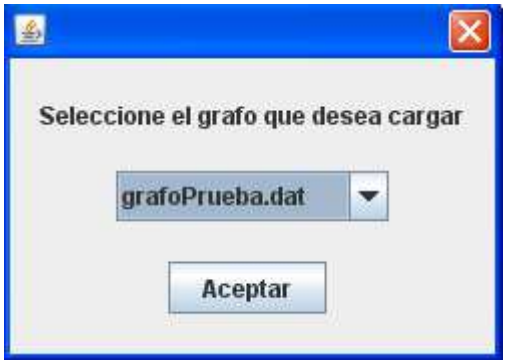

Ilustración 75: Cargar grafo

Una vez elegido el grafo a cargar se mostrará la pantalla principal de la aplicación. En la parte izquierda de esta pantalla se mostrarán las funcionalidades que ofrece el sistema, y en la zona de la derecha aparecerá la red social a analizar representada mediante un grafo, así como un panel de texto

en el que se indicarán los resultados obtenidos tras cada interacción del usuario (ver Ilustración 76). Cada individuo se representa mediante un nodo con su identificador, y las relaciones entre los distintos miembros se reflejan a través de la unión de los distintos nodos mediante aristas.

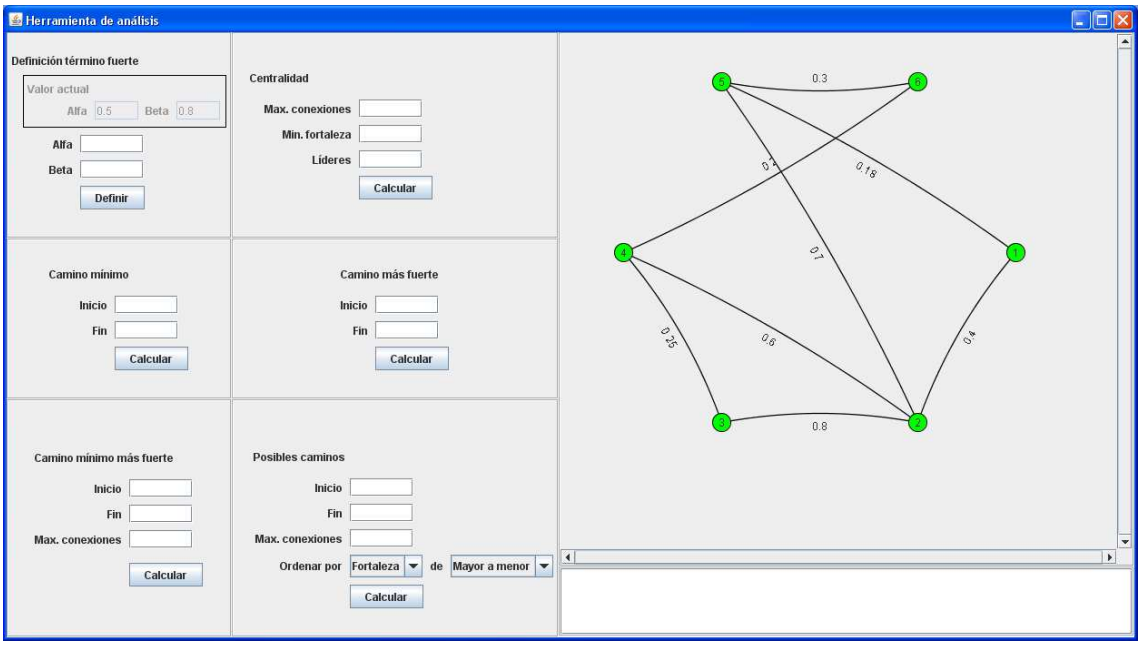

Ilustración 76: Inicio aplicación

Una vez iniciada la aplicación, el usuario puede utilizar las distintas funcionalidades, las cuales se explicarán con detalle a continuación:

B.3 Definición del término fuerte

La aplicación ofrece la posibilidad de indicar qué significado tiene el término fuerte, de manera que en el resto de interacciones que se produzcan se tenga en cuenta esta definición.

Así, el significado de este término se definirá en el intervalo [0, 1], donde 1 representa el máximo valor de fortaleza. Para ello, el usuario tiene que establecer dos valores, alfa y beta. Alfa y beta deben ser valores reales pertenecientes al intervalo [0, 1], donde alfa representa el valor hasta el cual se considera que una conexión no es fuerte y beta el valor a partir del cual se considera que la conexión es fuerte. De esto se puede deducir que el valor de alfa debe ser menor que el de beta.

Por defecto, la aplicación establece una definición de este término, que aparece en un recuadro no editable (ver Ilustración 77).

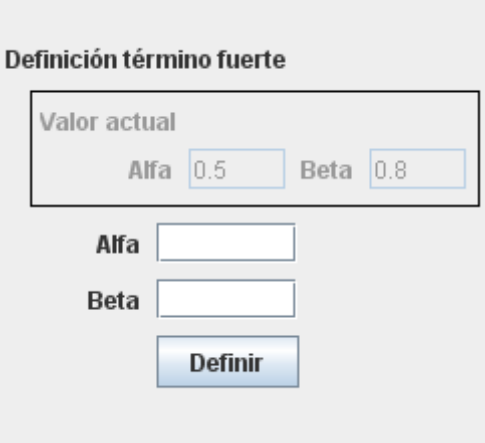

Ilustración 77: Definición por defecto del término fuerte

Si el usuario no indica su propia definición del término fuerte, todas las operaciones se realizarán teniendo en cuenta la definición por defecto. En la ilustración 78 podemos ver de forma más clara su significado. En el eje de abscisas (eje x) se representa el peso de la conexión y en el de ordenadas (eje y) el grado de pertenecía al término fuerte. De esta manera, según la definición actual, toda conexión cuyo peso sea menor o igual a 0.5 no se considerará fuerte. A partir de este valor y hasta 0.8 la fortaleza se irá incrementando de forma lineal. Y por último, a partir de 0.8 se considerará que la conexión es fuerte.

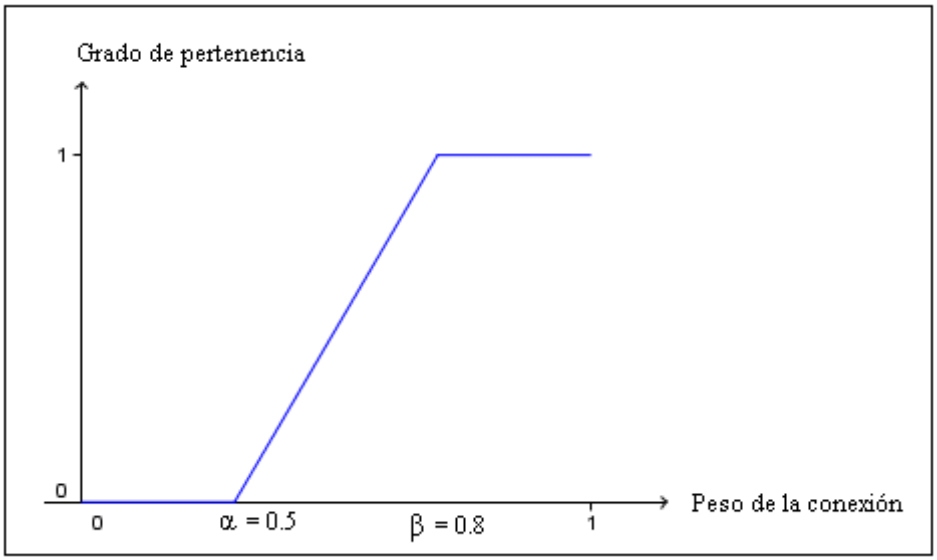

Ilustración 78: Representación del término fuerte definido por defecto en la aplicación

Si el usuario desea establecer su propia definición, tiene que introducir los valores de alfa y beta en los respectivos campos y pulsar el botón *´Definir´*.

Si ha dejado alguno de los campos en blanco aparecerá un mensaje en un recuadro rojo indicando el error cometido (ver Ilustración 79).

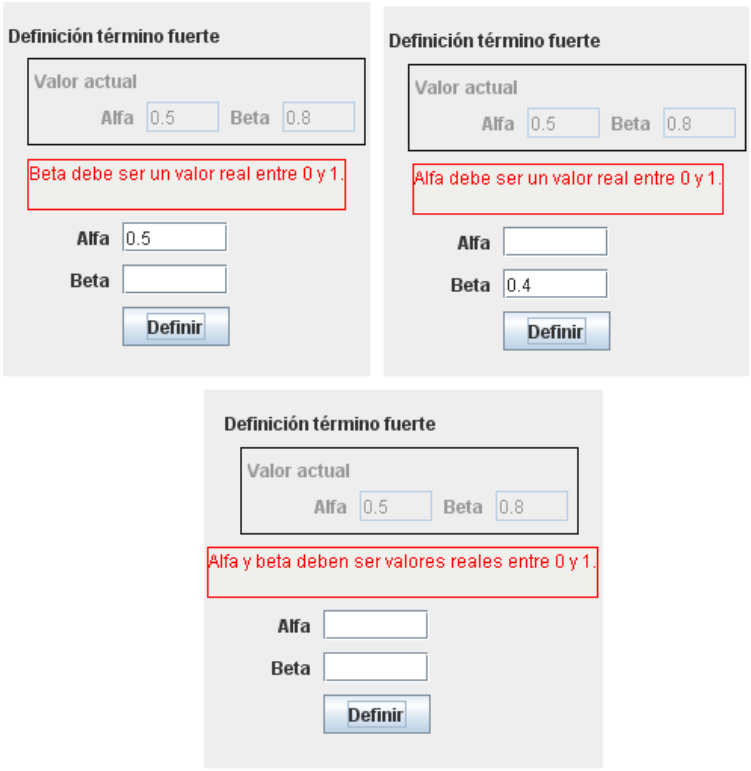

Ilustración 79: Definición término fuerte - Error campo en blanco

Si alguno de los campos contiene un tipo de dato erróneo (cadena, carácter,..), el sistema mostrará un mensaje en color rojo para informar del error cometido (ver Ilustración 80).

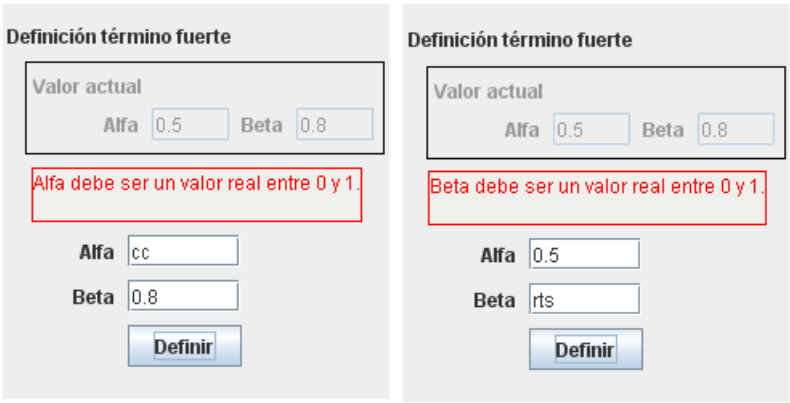

Ilustración 80: Definición término fuerte - Error tipo de dato

Si alguno de los valores introducidos se encuentra fuera del intervalo [0, 1] o no se verifica la restricción de que alfa sea menor que beta, aparecerá un mensaje indicando el error que se ha producido para que el usuario pueda solucionarlo (ver Ilustración 81).

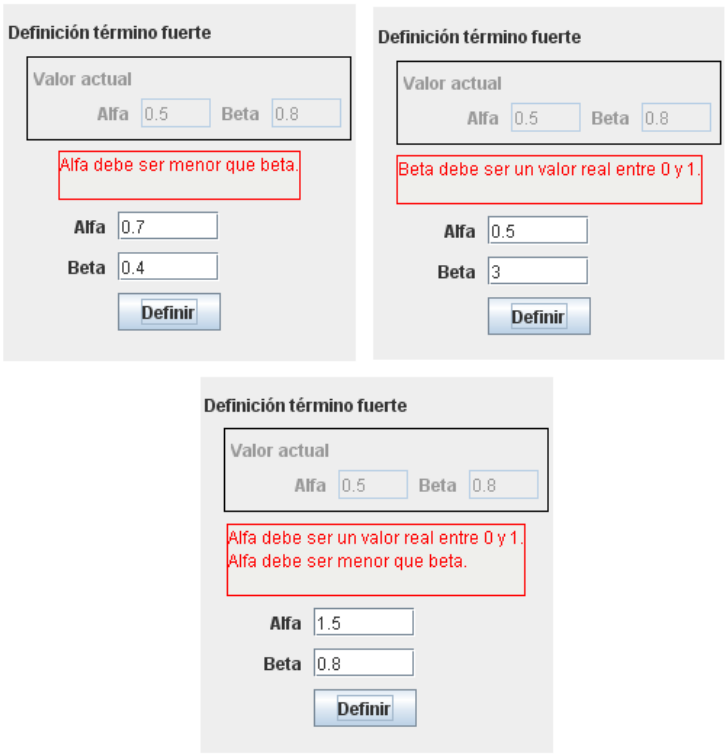

Ilustración 81: Definición término fuerte - Error valor incorrecto

Por el contrario, si los valores introducidos son correctos (ver Ilustración 82), el sistema actualizará la definición del término fuerte, la cual se puede observar en el recuadro negro, y mostrará en el panel de texto el cambio que se ha producido (ver Ilustración 83).

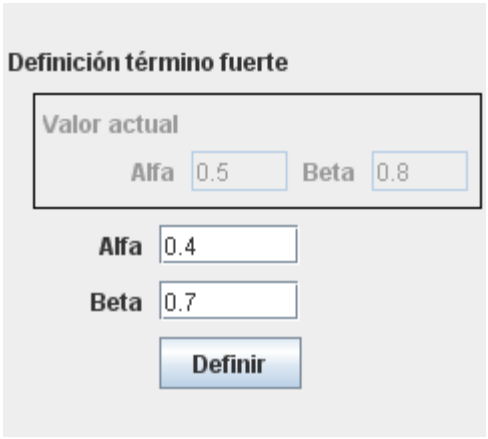

Ilustración 82: Definición término fuerte - Valores correctos

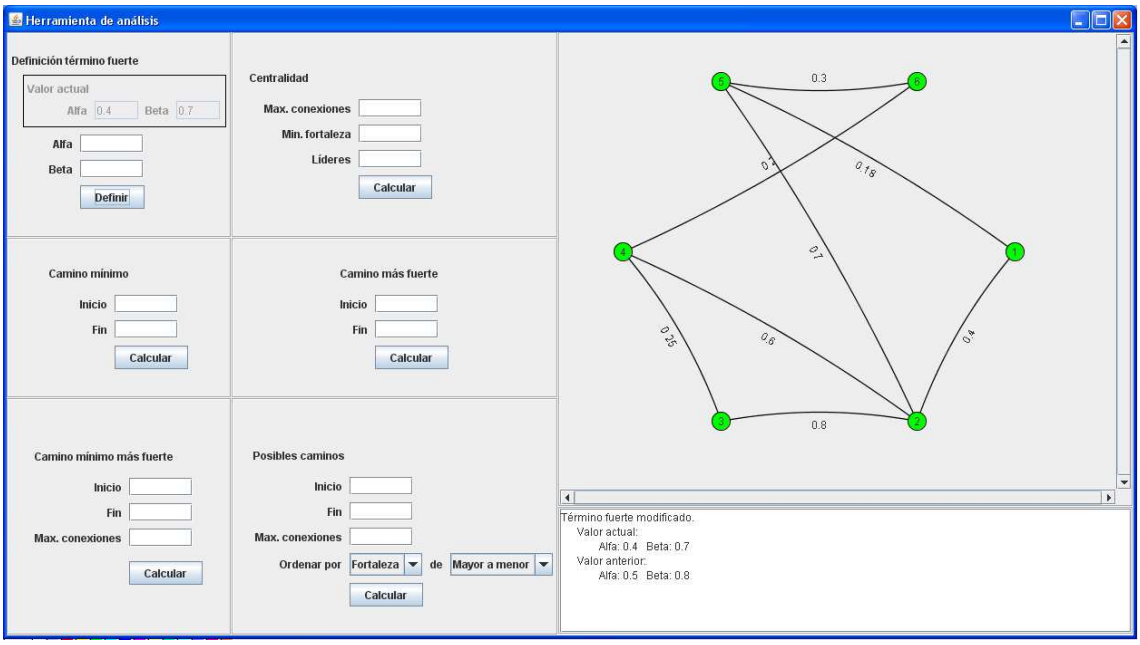

Ilustración 83: Término fuerte definido

B.4 Centralidad

Esta opción permite obtener los líderes de la red social. Para ello, es necesario indicar el número máximo de conexiones que se van a tener en cuenta a la hora calcular la centralidad de cada uno de los miembros de la red, el valor mínimo de fortaleza que deben tener esas conexiones y el número de líderes a calcular, y por último pulsar el botón *´Calcular´* (ver Ilustración 84).

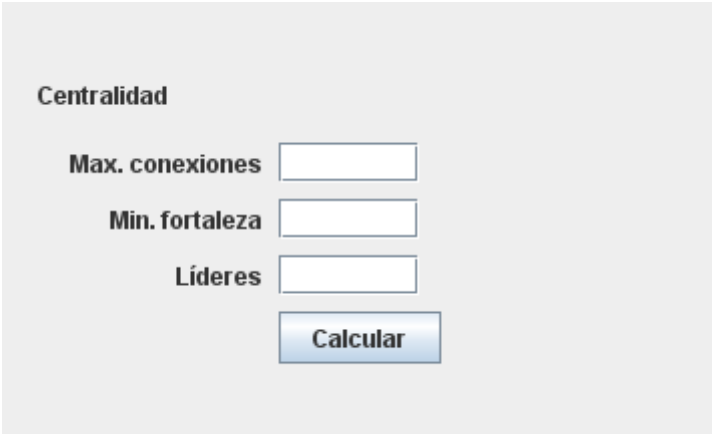

Ilustración 84: Panel Centralidad

Si el usuario deja vacío alguno de los campos y pulsa el botón *´Calcular´*, el sistema utilizará unos criterios por defecto para la búsqueda de líderes que son los siguientes:

- Si deja vacío el campo ´*Max. conexiones*´ se tendrán en cuenta las conexiones de todos los niveles.

- Si no introduce ningún valor en el campo ´Min. fortaleza´ se considerarán todas las conexiones cuya fortaleza sea superior a 0.

- Si no indica el número de líderes que desea obtener se mostrará un líder.

En la Ilustración 85 podemos ver un ejemplo en el que no se introduce ningún valor en los tres campos y la respuesta que produce el sistema.

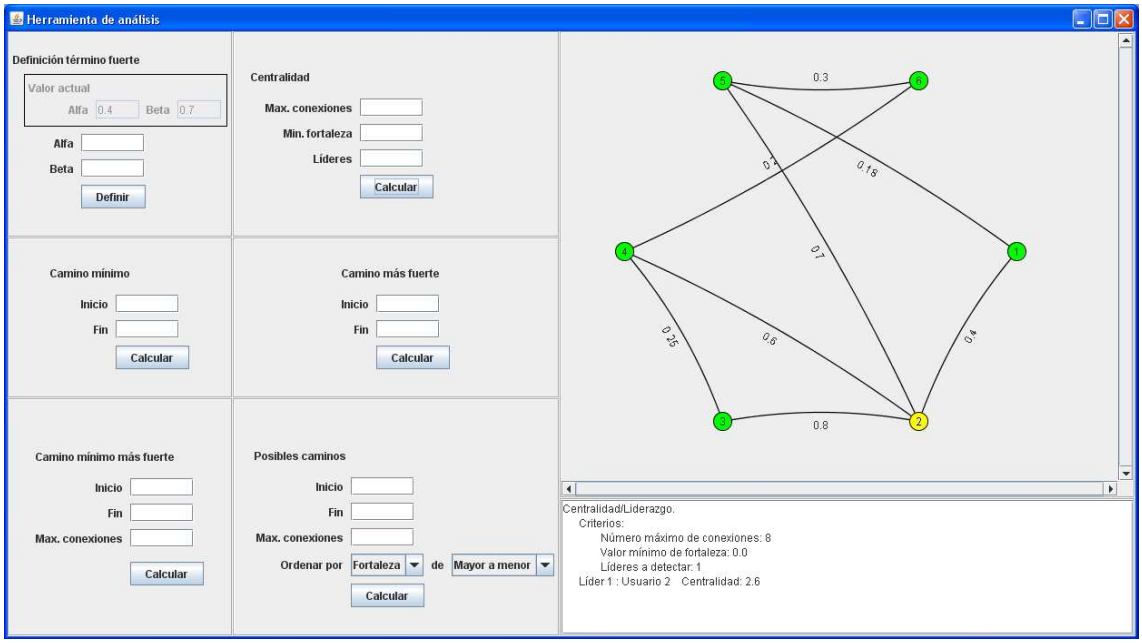

Ilustración 85: Centralidad - Campos en blanco

En caso de que se haya introducido un tipo de dato erróneo en alguno de los campos, se mostrará un mensaje de error para ayudar al usuario, indicando el tipo de dato requerido (ver Ilustración 86).

Ilustración 86: Centralidad - Error tipo de dato

Por otro lado, si se ha introducido un valor incorrecto en alguno de los campos, es decir, fuera del rango establecido, aparecerá un mensaje de error en color rojo indicando el tipo de dato y el rango de valores que pueden ir en ese campo (ver Ilustración 87).

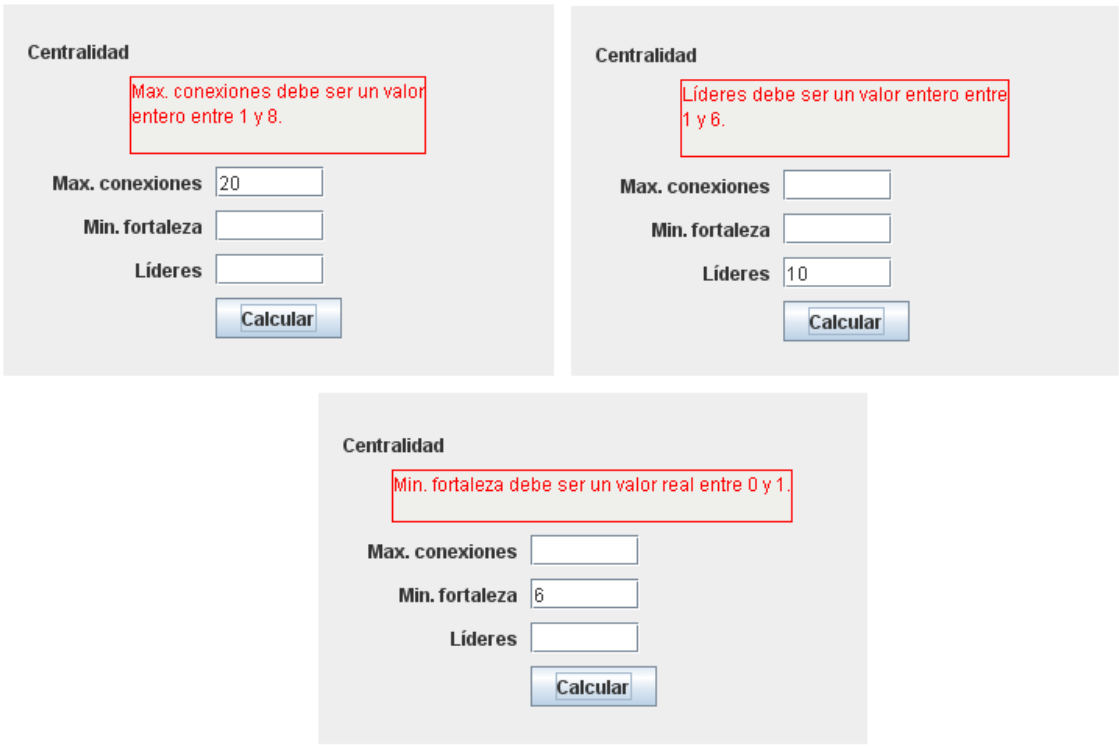

Ilustración 87: Centralidad - Error valor incorrecto

Si los valores introducidos son correctos (ver Ilustración 88) se resaltarán con color amarillo los nodos correspondientes a los líderes obtenidos y, en el panel de texto, se mostrarán los criterios utilizados y el valor de centralidad correspondiente a cada uno de los líderes (ver Ilustración 89).

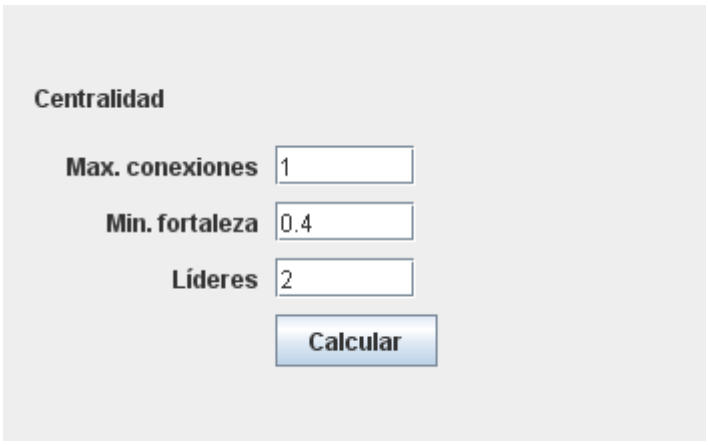

Ilustración 88: Centralidad - Valores correctos

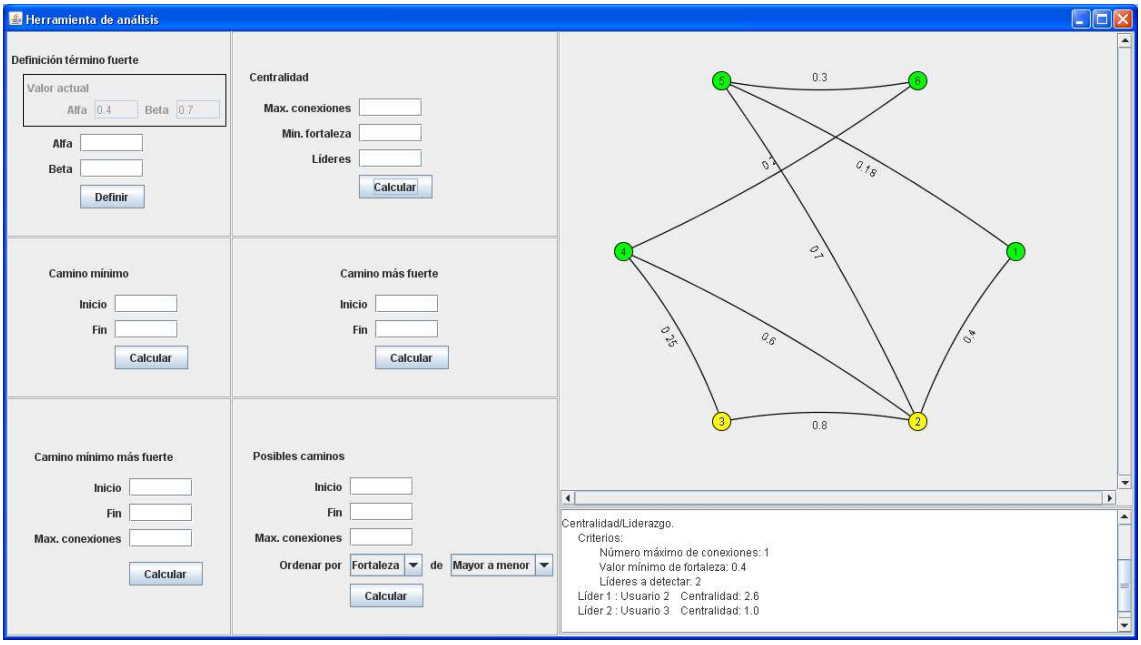

Ilustración 89: Centralidad - Salida producida tras introducir valores correctos

B.5 Camino mínimo

Para obtener el camino mínimo existente entre dos individuos, el usuario debe introducir en los campos *´Inicio´* y *´Fin´* los identificadores de dichos individuos y pulsar el botón *´Calcular´* (ver Ilustración 90).

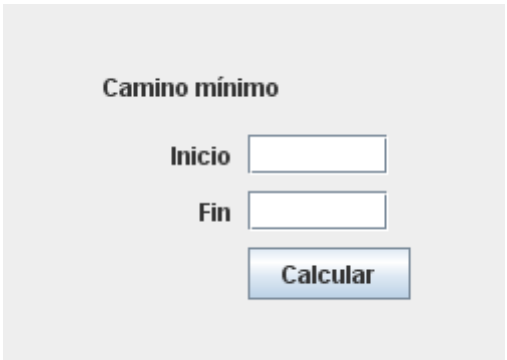

Ilustración 90: Panel Camino mínimo

Si deja alguno de los campos en blanco, el sistema le indicará con un mensaje de error que debe introducir el identificador de algún miembro de la red en los campos (ver Ilustración 91).

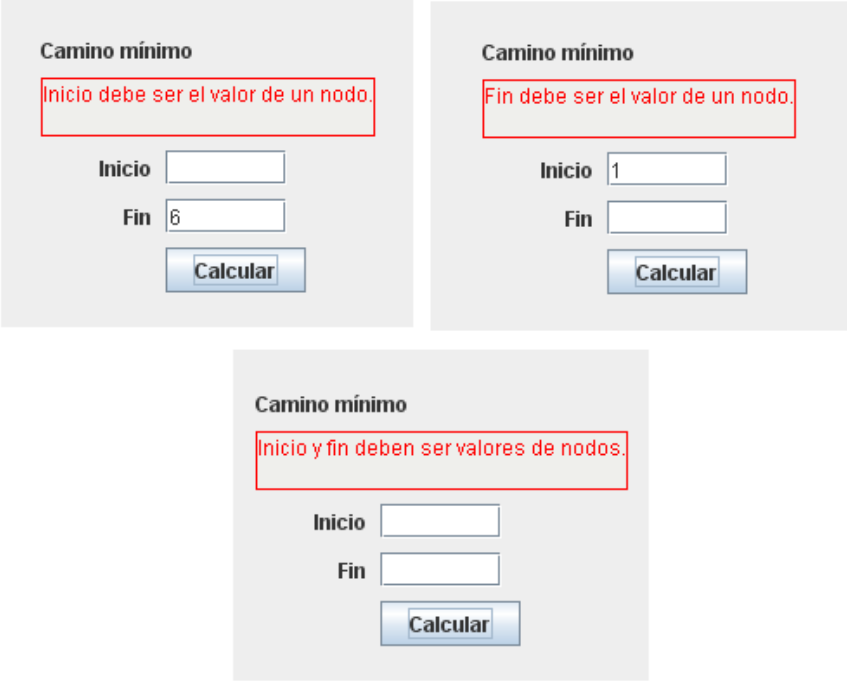

Ilustración 91: Camino mínimo - Error campo en blanco

Si ha introducido un tipo de dato erróneo (cadena, carácter, real,...) en alguno de los campos, aparecerá un mensaje indicando el error cometido (ver Ilustración 92).

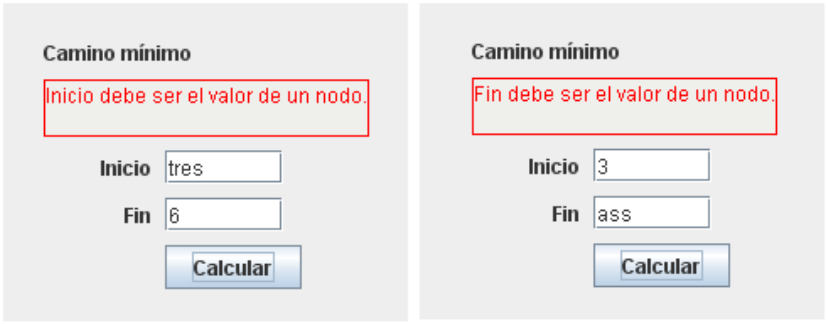

Ilustración 92: Camino mínimo - Error tipo de dato

Si alguno de los campos contiene un valor que no se corresponde con el identificador de algún usuario o bien el valor indicado en *´Inicio´* y en *´Fin´* es el mismo, se mostrará un mensaje para informar al usuario del error que se ha producido para que pueda solucionarlo (ver Ilustración 93).

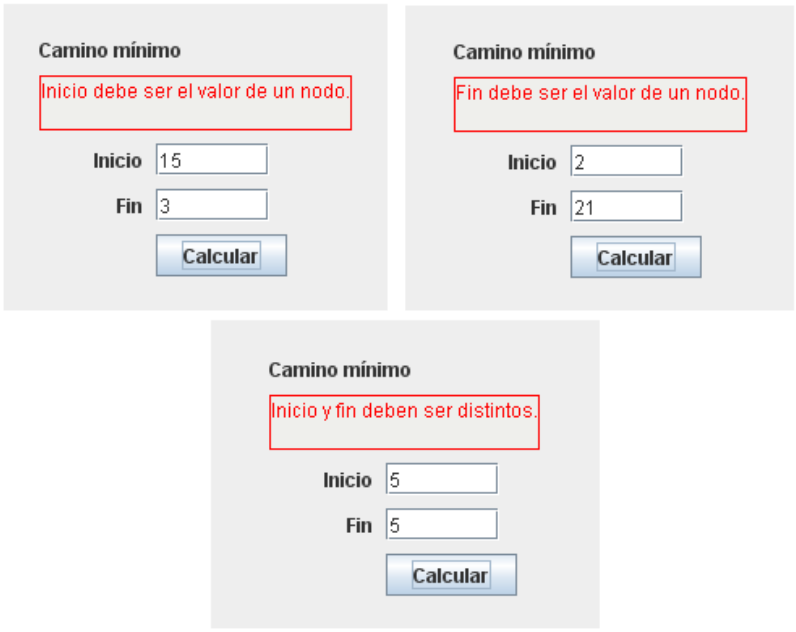

Ilustración 93: Camino mínimo - Errores valor incorrecto e inicio y fin iguales

Por el contrario, si todos los valores son correctos, como en el caso de la Ilustración 94, se resaltará en el grafo el camino mínimo para ir desde el nodo inicio hasta el nodo fin, señalando con color amarillo los nodos que forman parte de dicho camino y con un mayor grosor las aristas que unen dichos nodos. Además, se reflejará el camino obtenido en el panel de texto (ver Ilustración 95).

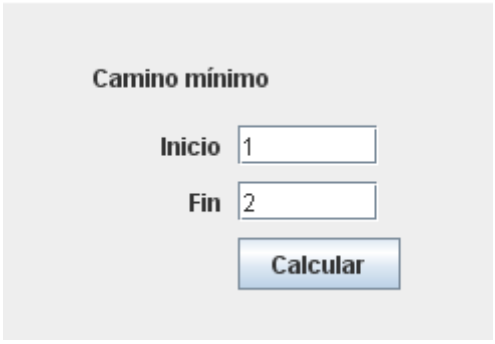

Ilustración 94: Camino mínimo - Valores correctos

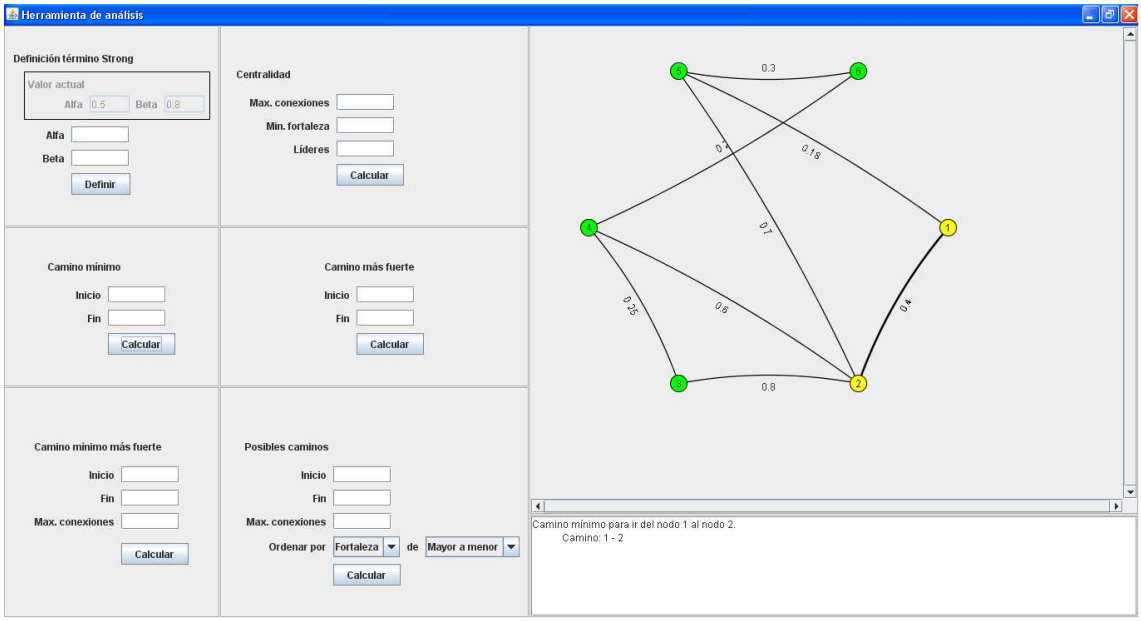

Ilustración 95: Camino mínimo - Salida producida tras introducir valores correctos

B.6 Camino más fuerte

Para calcular el camino más fuerte que une a dos miembros es necesario indicar en los campos *´Inicio´* y *´Fin´* sus identificadores y pulsar el botón *´Calcular´* (ver Ilustración 96).

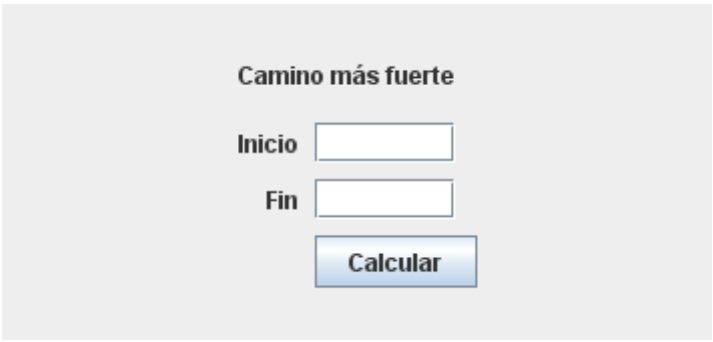

Ilustración 96: Panel Camino más fuerte

Si el usuario ha dejado en blanco alguno de los campos, aparecerá un mensaje de error indicando que debe introducir el identificador de algún miembro de la red (ver Ilustración 97).

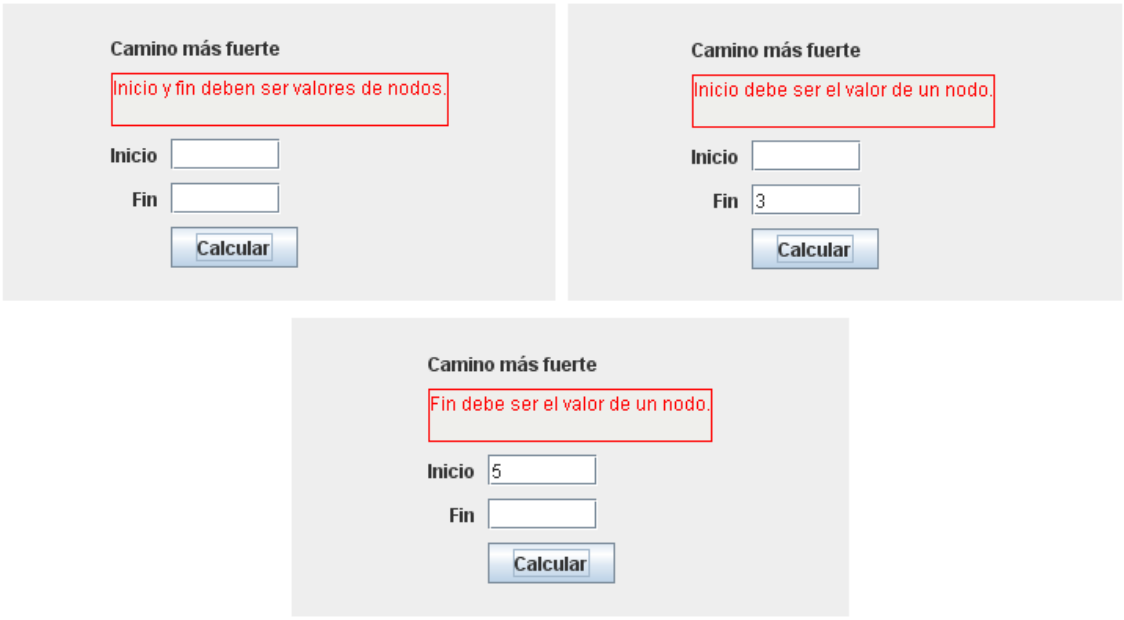

Ilustración 97: Camino más fuerte - Error campo en blanco

Si alguno de los campos contiene un tipo de dato erróneo (cadena, carácter, real,...), el sistema mostrará un mensaje indicando el error cometido para ayudar al usuario a solucionarlo (ver Ilustración 98).

Ilustración 98: Camino más fuerte - Error tipo de dato

Si ha introducido en alguno de los campos un valor que no se corresponde con el identificador de algún usuario o bien el valor indicado en *´Inicio´* y en *´Fin´* es el mismo, se mostrará un mensaje para informar del error que se ha producido (ver Ilustración 99).

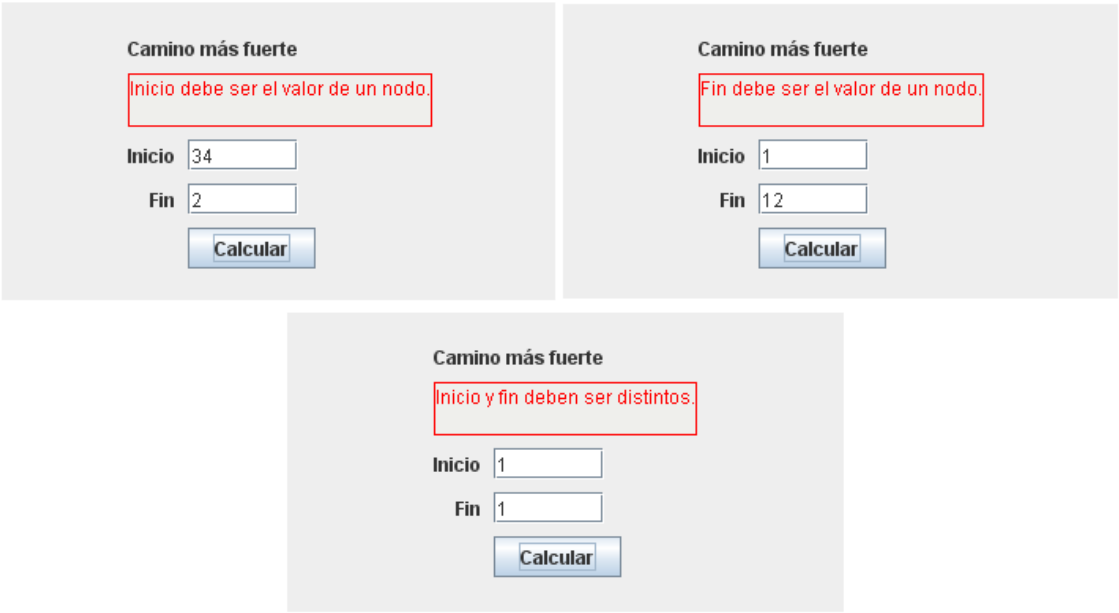

Ilustración 99: Camino más fuerte – Errores valor incorrecto e inicio y fin iguales

Si todos los valores son correctos, como se muestra en la Ilustración 100, se resaltará en el grafo el camino más fuerte que une al nodo inicio y al nodo fin, señalando con color amarillo los nodos que forman parte de dicho camino y con un mayor grosor las aristas que unen dichos nodos. Además, se mostrará el camino obtenido en el panel de texto (ver Ilustración 101).

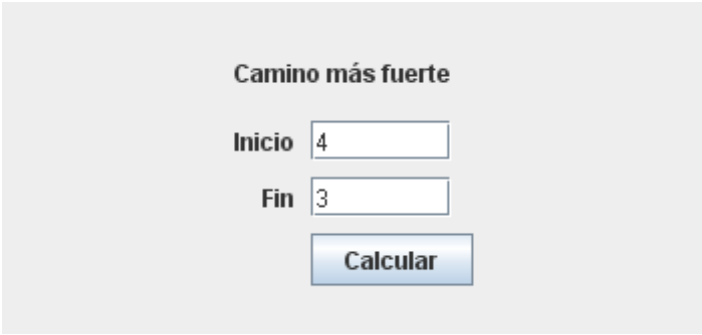

Ilustración 100: Camino más fuerte - Valores correctos

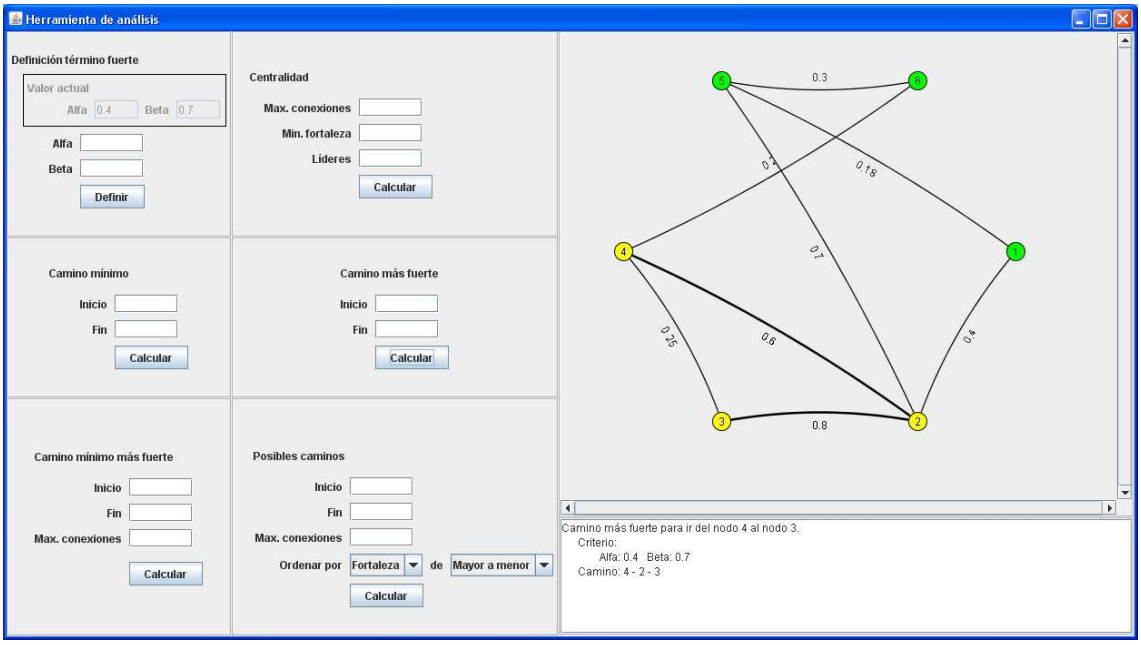

Ilustración 101: Camino más fuerte - Salida producida tras introducir valores correctos

B.7 Camino mínimo más fuerte

La aplicación ofrece la posibilidad de calcular el camino mínimo más fuerte existente entre dos individuos. Para ello, hay que indicar en los campos *´Inicio´* y *´Fin´* los identificadores de dichos individuos, en el campo *´Max. conexiones´* el número máximo de conexiones que puede tener el camino y, por último, pulsar el botón *´Calcular´* (ver Ilustración 102).

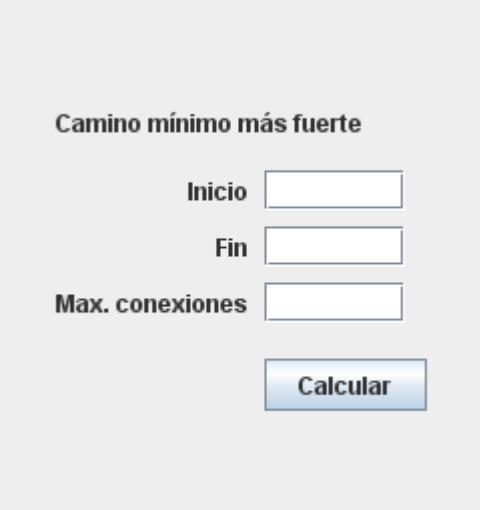

Ilustración 102: Panel Camino mínimo más fuerte

Si el usuario deja vacío el campo *´Inicio´* o el campo *´Fin´*, aparecerá un mensaje de error en el que se le informará de que debe introducir el identificador de alguno de los miembros de la red (ver Ilustración 103). Por el contrario, si no introduce ningún valor en el campo *´Max. conexiones´*, el sistema buscará entre todos los caminos, sin aplicar restricción en el número máximo de conexiones.

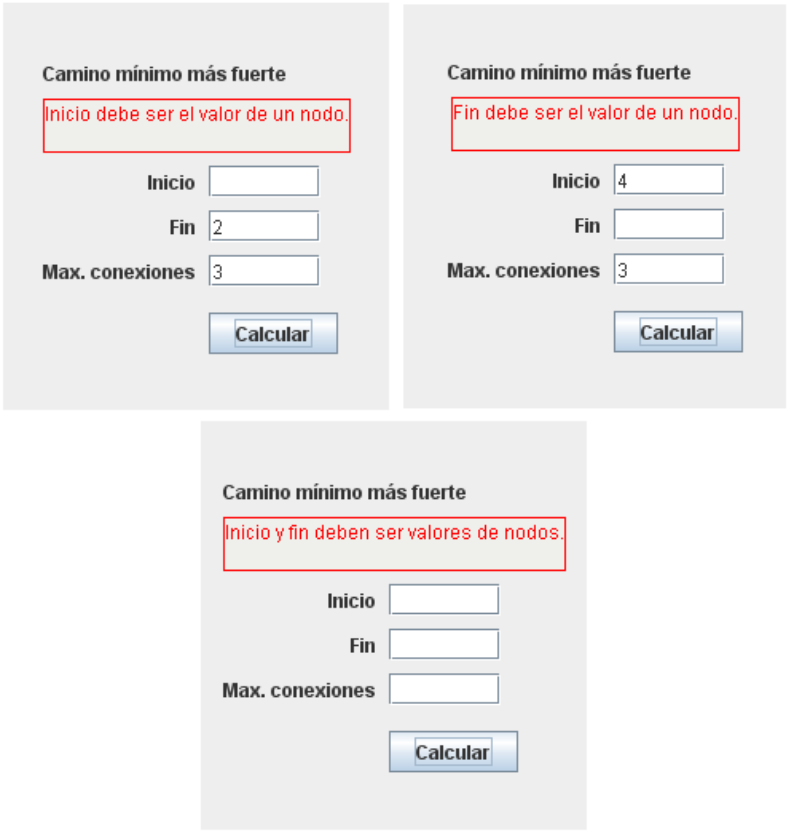

Ilustración 103: Camino mínimo más fuerte - Error campo vacío

En caso de que se haya introducido un tipo de dato erróneo en alguno de los campos, se mostrará un mensaje de error para ayudar al usuario, indicando el tipo de dato requerido (ver Ilustración 104).

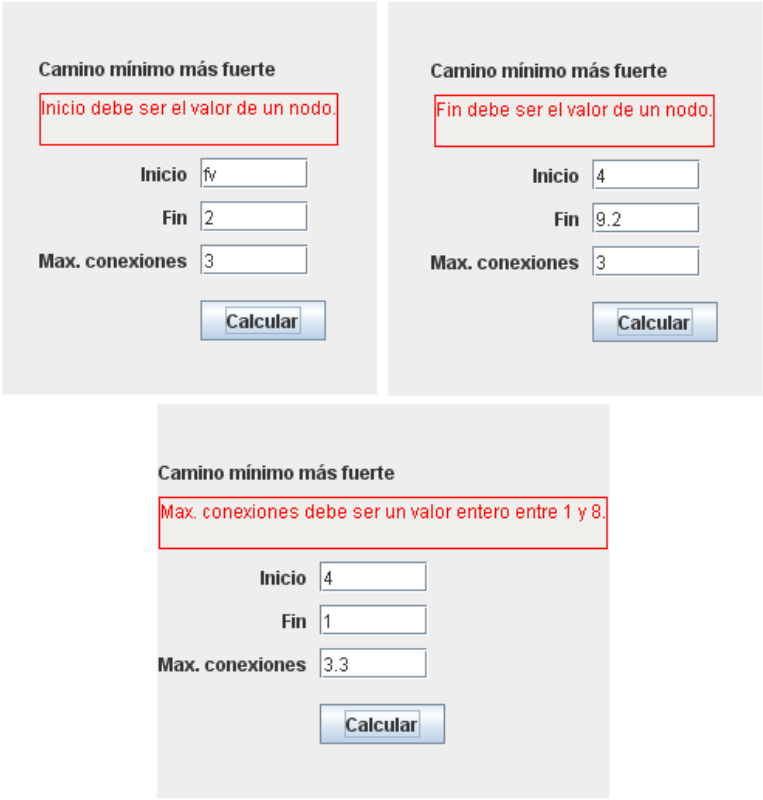

Ilustración 104: Camino mínimo más fuerte - Error tipo de dato

Por otro lado, si se ha introducido un valor incorrecto en alguno de los campos, es decir, fuera del rango establecido, aparecerá un mensaje de error indicando el tipo de dato y el rango de valores que pueden ir en ese campo, con el objetivo de ayudar al usuario a solucionar el error cometido (ver Ilustración 105).

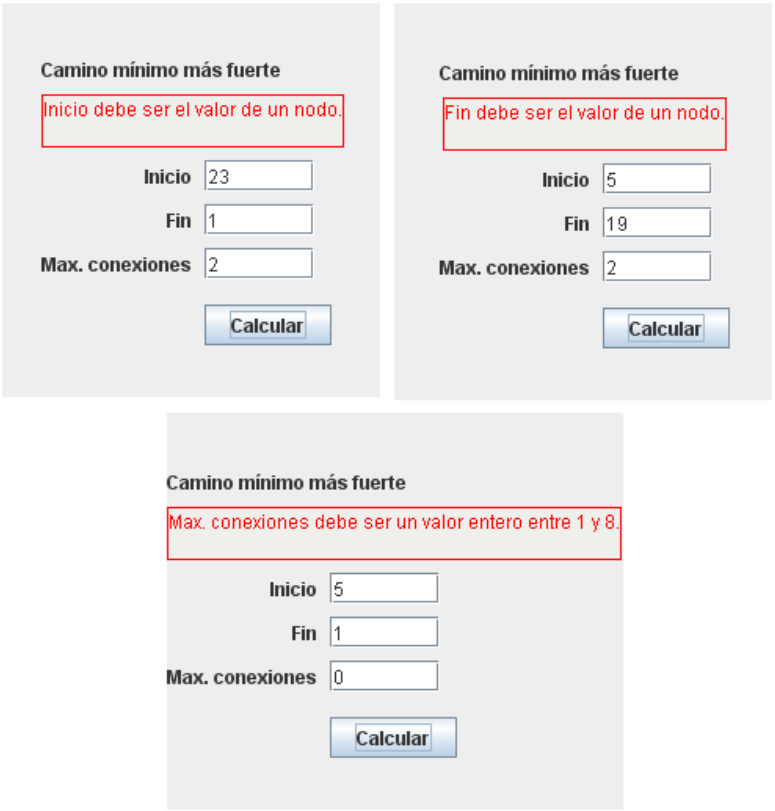

Ilustración 105: Camino mínimo más fuerte - Error valor incorrecto

Además, si el valor indicado en ´*Inicio*´ y en ´*Fin*´ es el mismo, se mostrará un mensaje para informar del error que se ha producido (ver Ilustración 106).

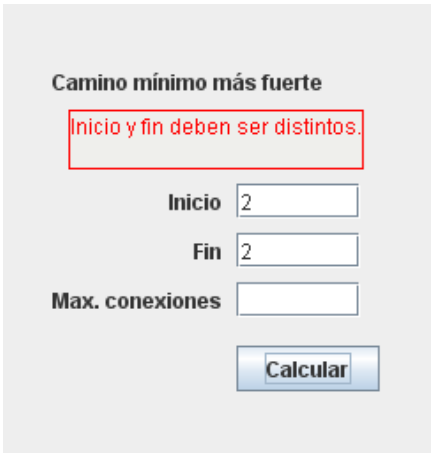

Ilustración 106: Camino mínimo más fuerte - Error inicio y fin iguales

Si por el contrario, todos los datos introducidos son correctos, como en la Ilustración 107, se resaltará en el grafo el camino mínimo más fuerte que une al nodo inicio y al nodo fin, señalando con color amarillo los nodos que forman parte de dicho camino y con un mayor grosor las aristas que unen dichos nodos. Además, se mostrará el camino obtenido en el panel de texto (ver Ilustración 108).

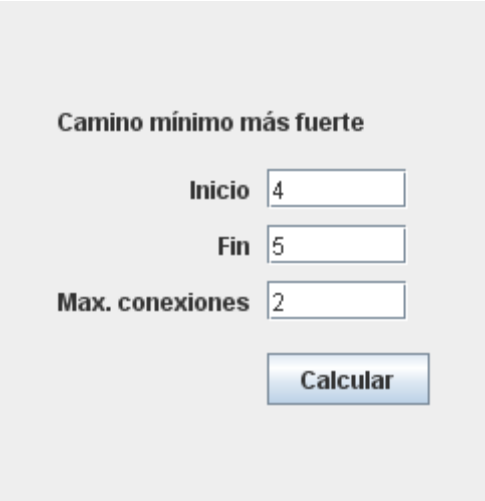

Ilustración 107: Camino mínimo más fuerte - Valores correctos

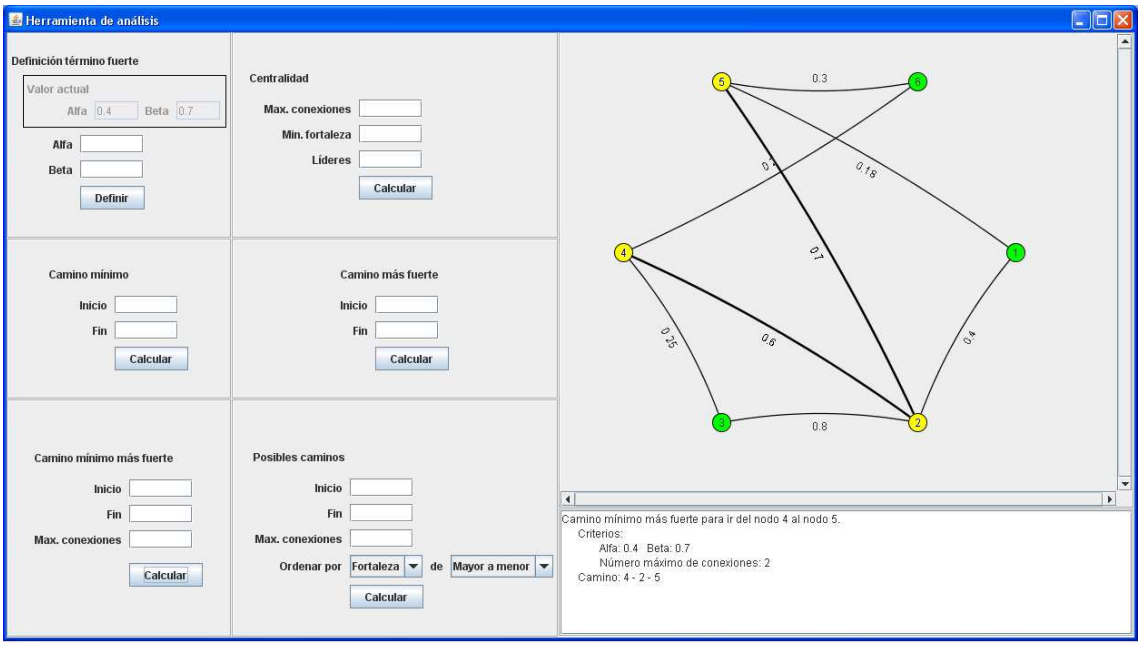

Ilustración 108: Camino mínimo más fuerte - Salida tras introducir valores correctos

B.8 Posibles caminos

Esta opción permite obtener todos los caminos que unen a dos individuos. Para ello, es necesario indicar en los campos *´Inicio´* y *´Fin´* cuáles son los identificadores de dichos individuos. Además, en el campo *´Max. conexiones´* se puede establecer el número máximo de conexiones que pueden tener los caminos a calcular y si el usuario lo desea puede especificar el criterio de ordenación a seguir a la hora de mostrar los caminos. Por último, tras definir los valores de los campos, hay que pulsar el botón *´Calcular´* (ver Ilustración 109).

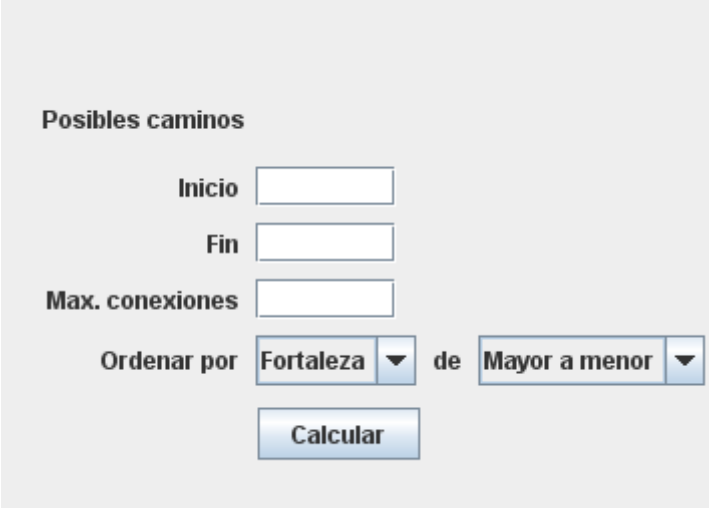

Ilustración 109: Panel Posibles caminos

Si se deja vacío el campo *´Inicio´* o el campo *´Fin´*, aparecerá un mensaje de error en el que se informará de que es necesario introducir el identificador de alguno de los miembros de la red (ver Ilustración 110). Sin embargo, dejar el campo *´Max. conexiones´* en blanco no supondrá un error, ya que el sistema buscará todos los posibles caminos para ir desde el nodo inicio hasta el nodo fin, sin aplicar ninguna restricción en el número máximo de conexiones.

Ilustración 110: Posibles caminos - Error campo en blanco

En caso de que alguno de los campos contenga un tipo de dato erróneo, aparecerá un mensaje de error en rojo indicando el tipo de dato requerido (ver Ilustración 111).

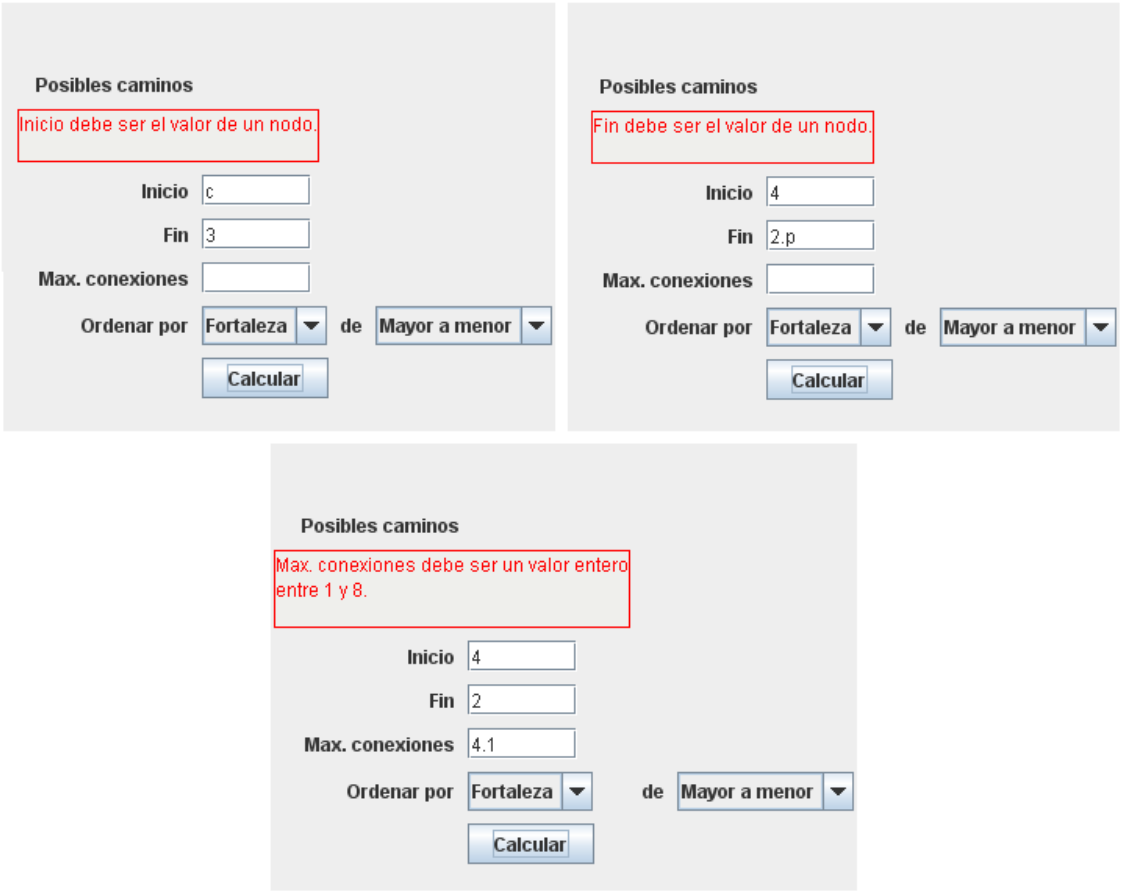

Ilustración 111: Posibles caminos - Error tipo de dato

Si se ha introducido un valor fuera del rango establecido en alguno de los campos, se mostrará un mensaje de error indicando el tipo de dato y el rango de valores que pueden ir en ese campo (ver Ilustración 112).

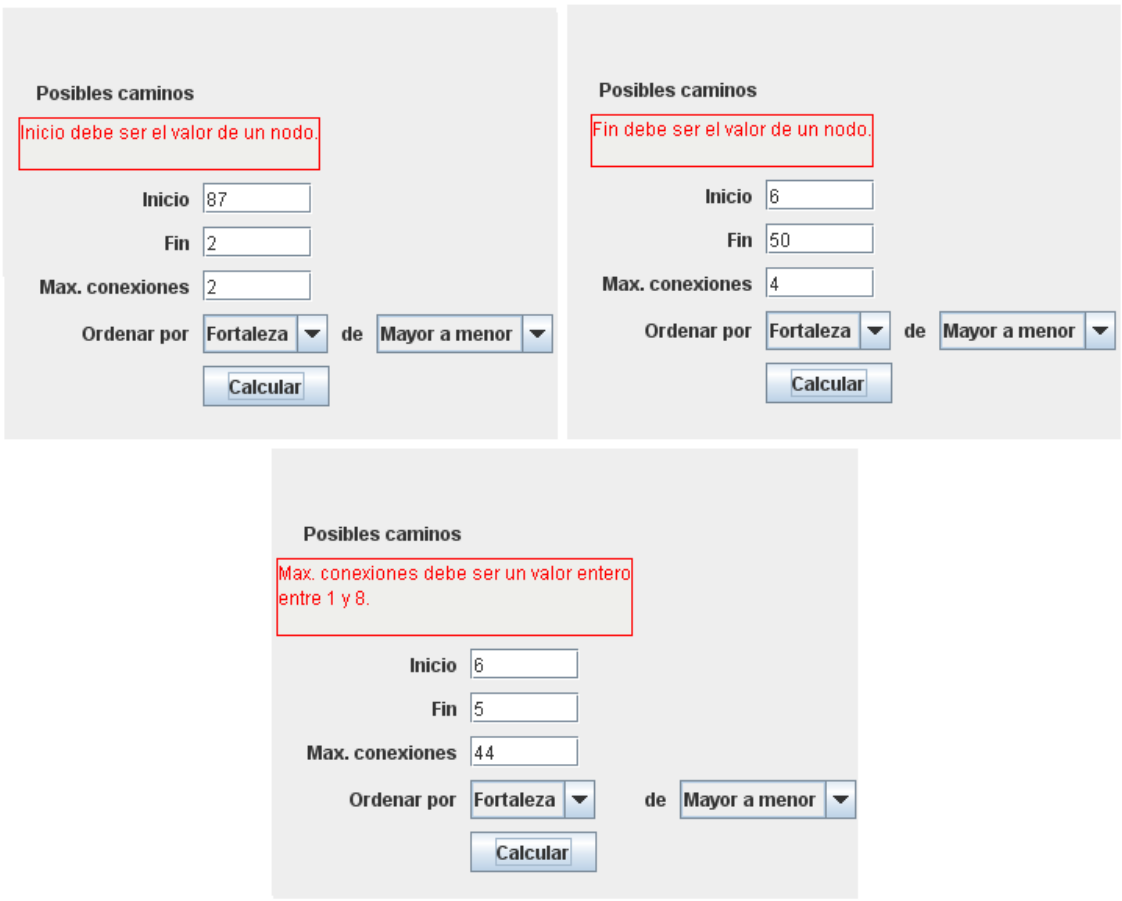

Ilustración 112: Posibles caminos - Error valor incorrecto

Por otro lado, si el valor indicado en *´Inicio´* y en *´Fin´* es el mismo, se mostrará un mensaje para informar del error que se ha producido, con el objetivo de ayudar al usuario a solucionar el error cometido (ver Ilustración 113).

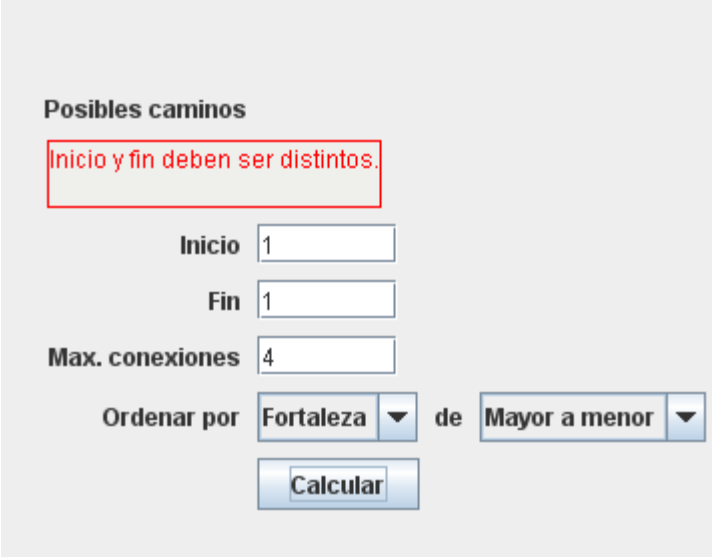

Ilustración 113: Posibles caminos - Error inicio y fin iguales

Si todos los valores introducidos son correctos, como en el caso de las ilustraciones 114 y 116, se mostrarán en el panel de texto todos los posibles caminos para ir desde el nodo inicio hasta el nodo fin, ordenados, según los criterios especificados, por su fortaleza (ver Ilustración 115) o longitud (ver Ilustración 117).

Posibles caminos ordenados por fortaleza

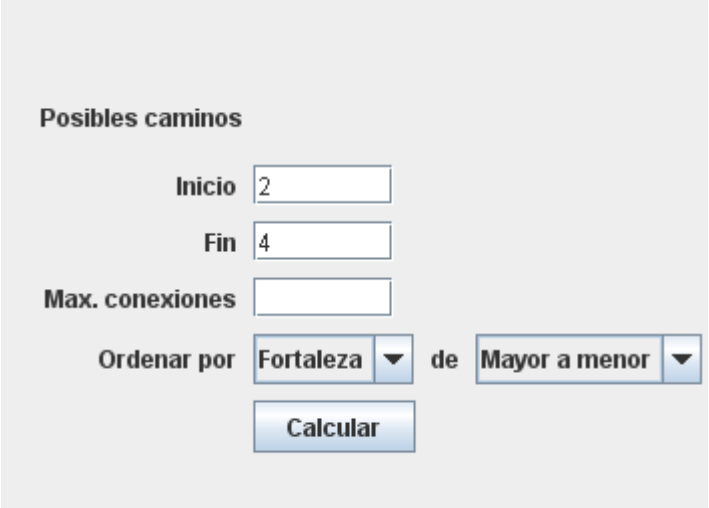

Ilustración 114: Posibles caminos - Valores correctos - Ordenar por fortaleza

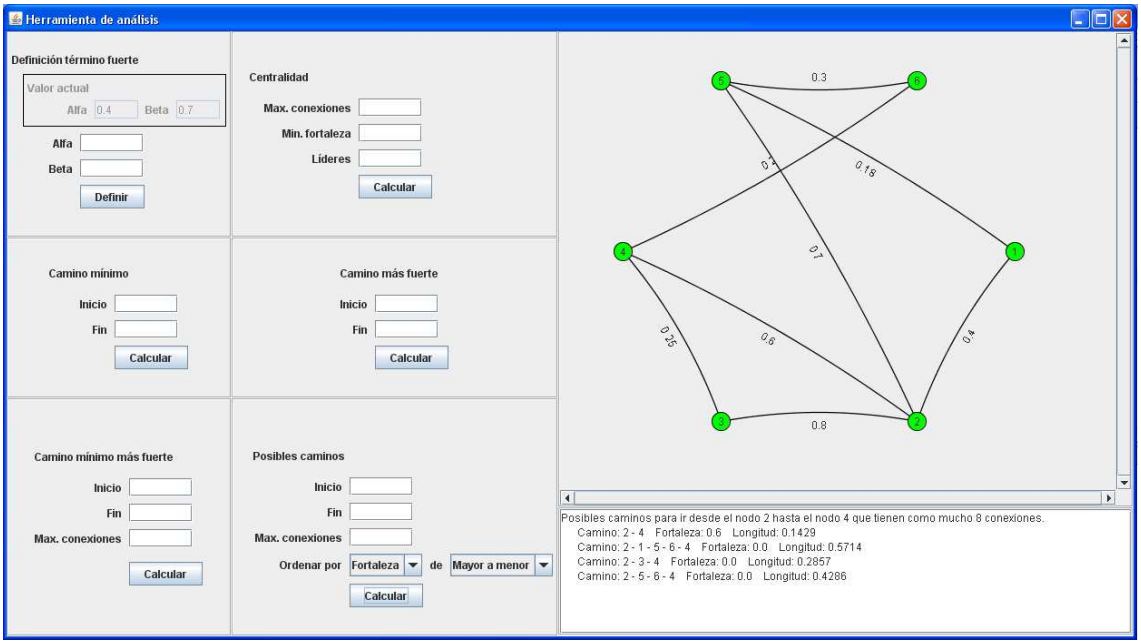

Ilustración 115: Posibles caminos - Salida producida tras introducir valores correctos – Ordenación realizada por fortaleza

Posibles caminos ordenados por longitud

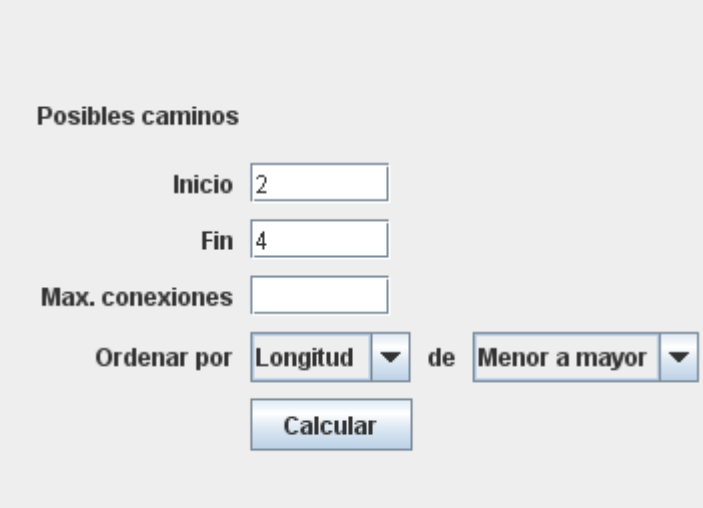

Ilustración 116: Posibles caminos - Valores correctos - Ordenar por longitud

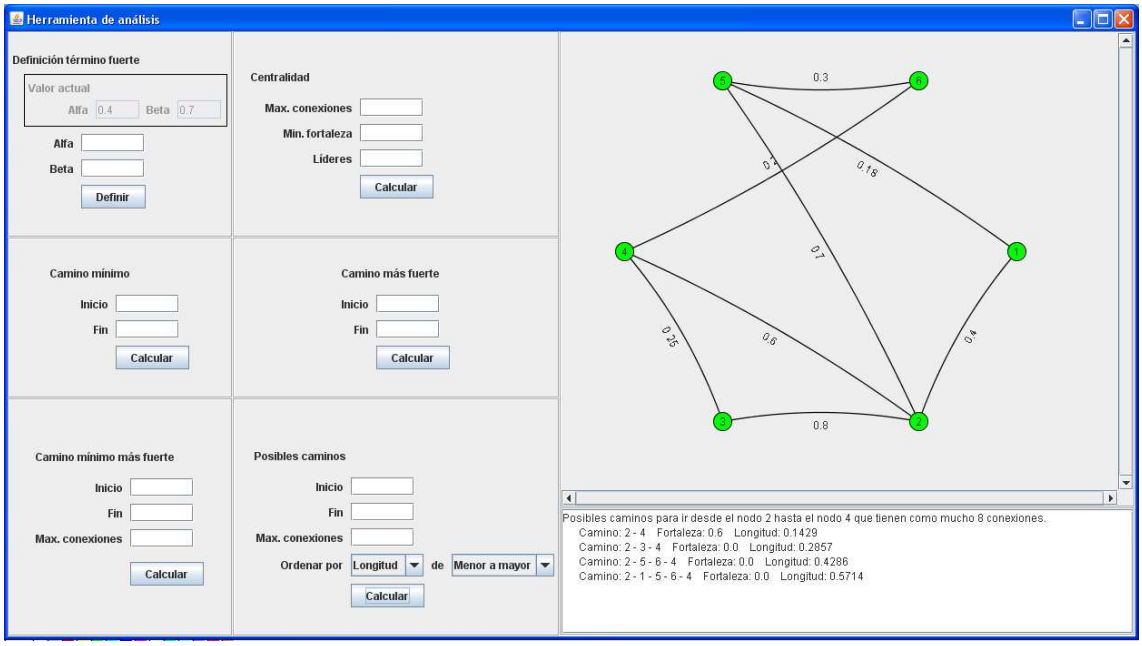

Ilustración 117: Posibles caminos - Salida producida tras introducir valores correctos – Ordenación realizada por longitud

B.9 Obtener información usuarios

Para obtener información sobre los miembros de la red basta con posicionar el curso del ratón sobre el nodo cuyo identificador se corresponde con el usuario del que se pretende conocer la información (ver Ilustración 118).

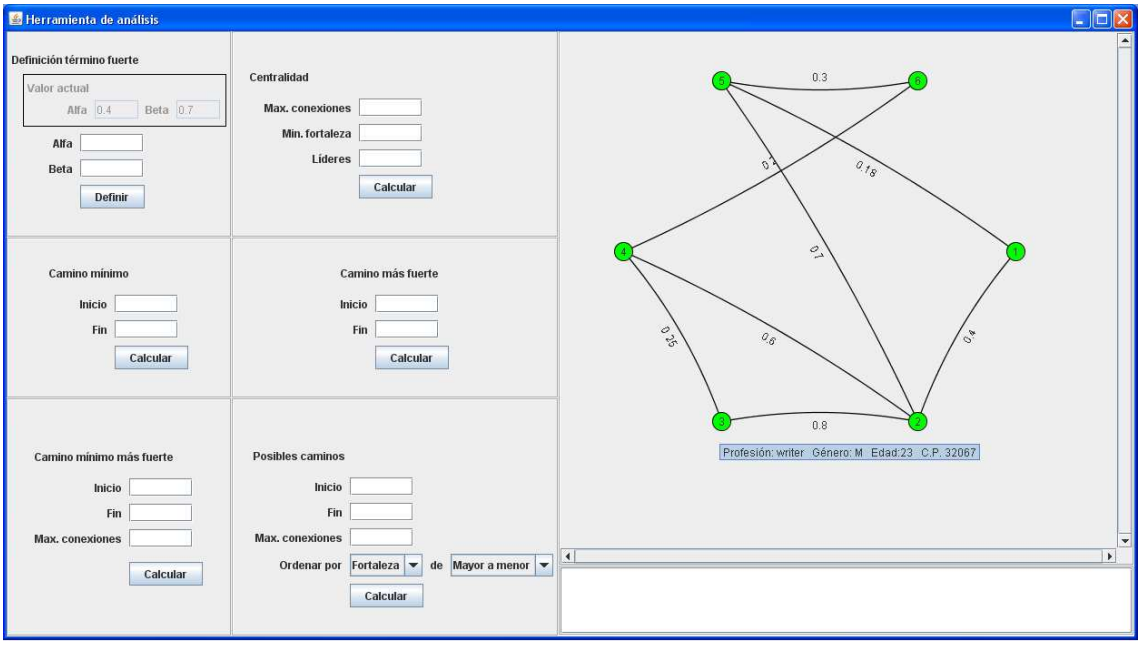

Ilustración 118: Obtener información usuario

B.10 Zoom

La aplicación también ofrece la posibilidad de acercar el grafo para observarlo con mayor claridad (ver Ilustración 119), o de alejarlo para ver su estructura completa (ver Ilustración 120). Para ello sólo hay que pinchar con el ratón sobre el grafo y mover la rueda del mismo.

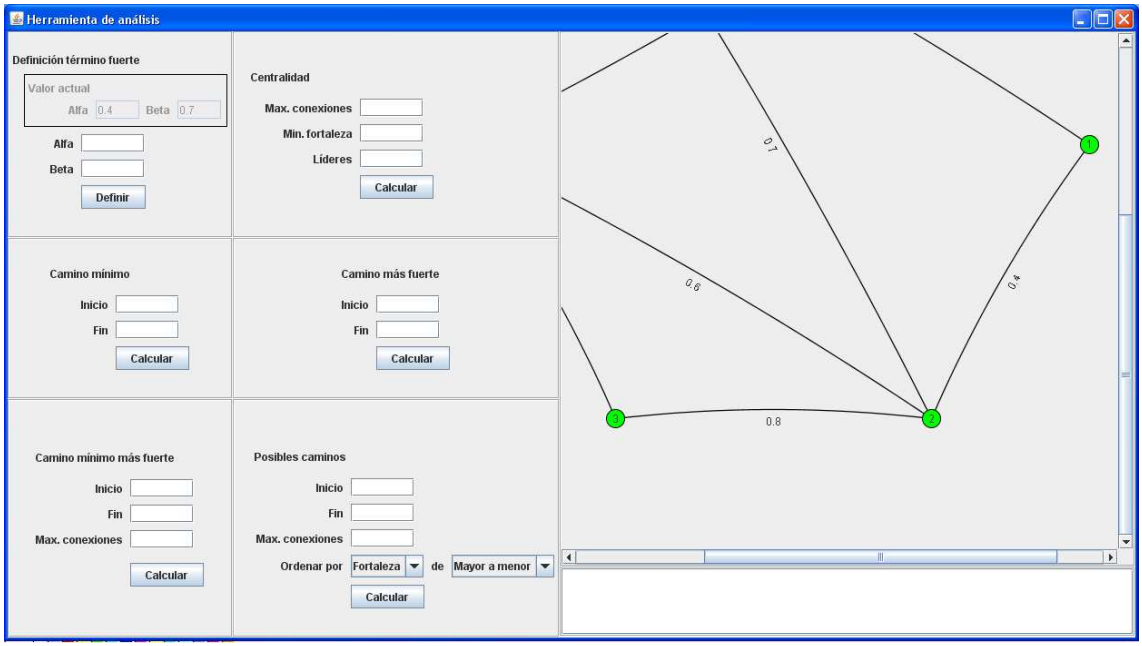

Ilustración 119: Acercar (Zoom +)

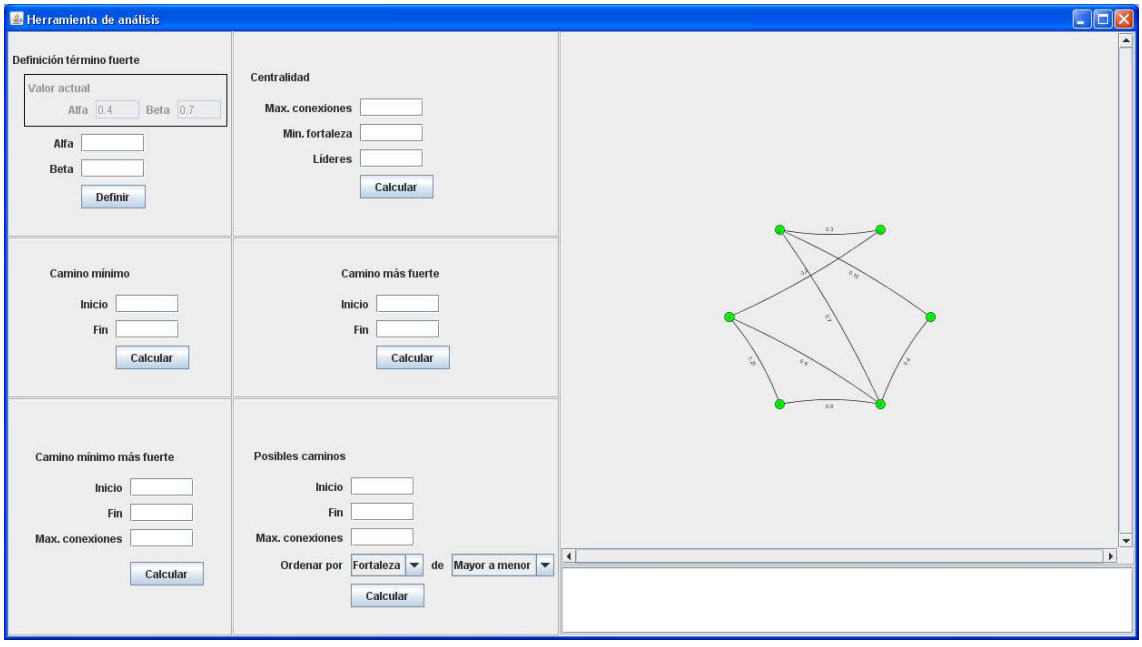

Ilustración 120: Alejar (Zoom -)

B.11 Cerrar aplicación

Cuando el usuario desee salir de la aplicación tendrá que pulsar el botón rojo situado en la esquina superior derecha (ver Ilustración 121).

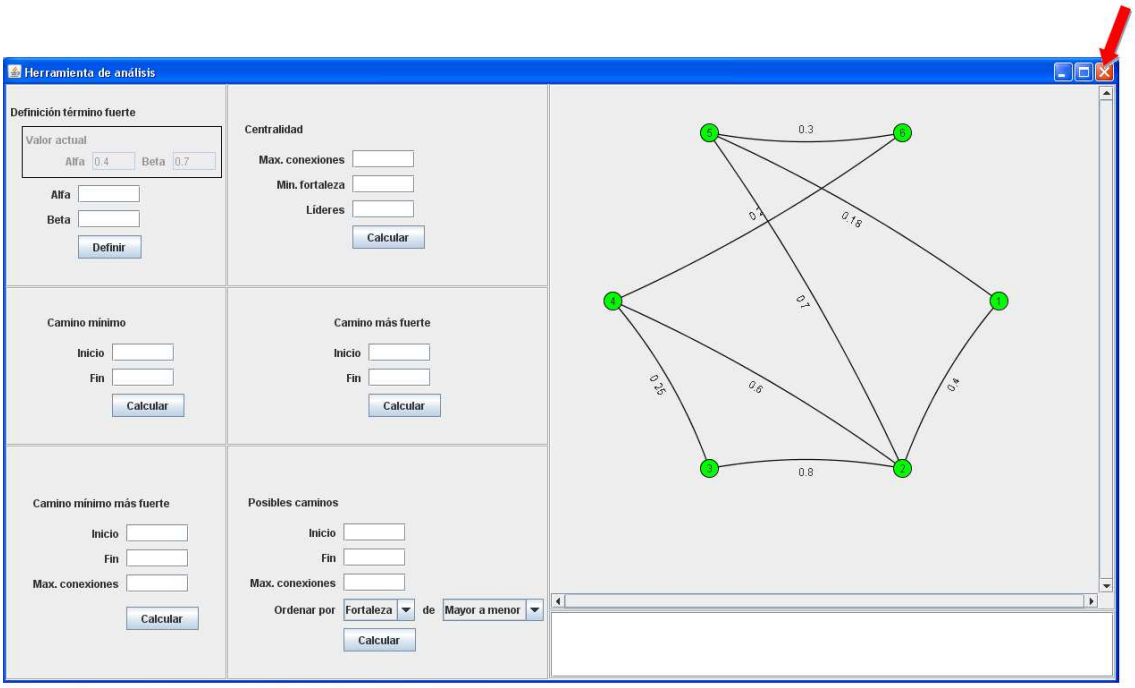

Ilustración 121: Cerrar aplicación

Anexo C. Índice de Ilustraciones.

Índice de ilustraciones

Anexo D. Índice de Tablas.

Índice de tablas