UNIVERSIDAD COMPLUTENSE DE MADRID
Facultad de Informática

Título: Krowdix, simulador de redes sociales
Autores: Daniel Alonso Fernández, Mario Arnaldos Navarro y César Montenegro Portillo
Director del proyecto: Diego Blanco Moreno
Curso académico: 2008-2009
Proyecto de Sistemas Informáticos, Facultad de Informática, Universidad Complutense de Madrid

Resumen

Krowdix es un programa para el análisis de redes sociales. Estas redes se pueden crear y modificar de un modo sencillo, y están formadas por nodos (los usuarios de la red) y relaciones (conexiones entre esos nodos), pero además tienen otros elementos como aficiones (los intereses de los nodos), contenidos (cualquier cosa creada por un nodo en la red) y comentarios (de un nodo a un contenido).

El programa es capaz de mostrar estas redes por pantalla de distintos modos, así como de generar informes que recojan sus propiedades. Pero además, gracias al uso de agentes inteligentes, Krowdix puede actuar como un simulador, mostrando la evolución de la red a partir de un instante dado.

Abstract

Krowdix is a Social Network analyzer. Those networks can be created and modified easily, and they consist on nodes (network users) and relations (connections among nodes). They have other elements also, like likings (nodes interests), contents (anything created by a node) and comments (from nodes to contents).

The program can show these networks on screen in different ways and generate reports with its properties. Furthermore, using intelligent agents, Krowdix can simulate the evolution of a network from a certain moment.

Palabras clave

Redes sociales, análisis, simulación, inteligencia artificial.

Agradecimientos

Los autores de Krowdix queremos dedicar unas líneas de agradecimiento a todos los que han hecho posible que el proyecto saliera adelante.

A **Julio Galarón**, por todas las ideas que aportó y por involucrarse como *betatester*.

Índice

Resumen 2

Abstract 2

Palabras clave 2

Agradecimientos 3

1. Introducción 8
2. Estado actual 10
   - 2.1. Programas existentes 10
     - 2.1.1. KrackPlot 10
     - 2.1.2. NetDraw 10
     - 2.1.3. SocNetV 11
   - 2.2. Conclusiones 12
3. Objetivos 13
   - 3.1. Creación y modificación de redes sociales 13
   - 3.2. Análisis de la red 13
   - 3.3. Interfaz intuitiva y agradable 13
   - 3.4. Inteligencia artificial 13
   - 3.5. Persistencia 13
   - 3.6. Soporte multiplataforma 13
   - 3.7. Posibilidad de ampliación 13
4. Memoria del proyecto 14
   - 4.1. Octubre de 2008 14
   - 4.2. Noviembre de 2008 14
   - 4.3. Diciembre de 2008 15
   - 4.4. Enero de 2009 15
   - 4.5. Febrero de 2009 15
   - 4.6. Marzo de 2009 16
   - 4.7. Abril de 2009 18
   - 4.8. Mayo de 2009 18
   - 4.9. Junio de 2009 19
5. Arquitectura de la aplicación 20
   - 5.1. Vista general 20
   - 5.2. Módulos de la aplicación 21
     - 5.2.1. Red social 21
     - 5.2.2. Interfaz 21
   - 5.3. Agentes inteligentes 27
     - 5.3.1. Introducción 27
     - 5.3.2. Arquitectura 27

5.3.3. Ampliar, mejorar o reemplazar los Agentes Inteligentes 30
5.4. Cuestiones de diseño 31
5.4.1. Introducción 31
5.4.2. Patrón modelo-vista-controlador 31
5.4.3. Patrón *singleton* 31
5.4.4. Patrón fábrica (*factory*) 31
5.4.5. Patrón constructor (*builder*) 32
5.4.6. Patrón estrategia 32
5.4.7. Patrón decorador 33
5.4.8. Patrón cadena de responsabilidad 33
5.5. Cómo hacer modificaciones en el proyecto 34
5.5.1. Introducción 34
5.5.2. Capa de presentación 34
5.5.3. Capa de negocio 34
5.5.4. Capa de datos 35
5.6. Herramientas y detalles de implementación 36
5.6.1. Lenguaje de programación 36
5.6.2. Herramientas CASE 36
5.6.3. Entorno de desarrollo 36
5.6.4. Desarrollo de la interfaz 37
5.6.5. Base de datos 38
5.6.6. Librerías externas usadas 38
6. Conclusiones 40
6.1. Conclusiones del proyecto 40
6.2. Trabajo futuro y ampliaciones 40
6.2.1. Degradación de redes 40
6.2.2. Mejoras en la comunicación entre agentes 40
6.2.3. Modo Dios 40
6.2.4. Cambios en la conexión con la base de datos 40
6.2.5. Importar y exportar archivos en otros formatos 41
6.2.6. Conexión con redes sociales reales 41
6.2.7. Internacionalización 41
6.2.8. Nuevas estrategias para la creación de redes 41
6.2.9. Persistencia de la interfaz de usuario 41
6.2.10. Simulación constante 41
7. Anexos 42
7.1. Glosario 42
7.2. Requisitos del sistema 42

7.3. Manual de referencia....42
7.4. Especificación de requisitos de software (noviembre 2008)....43
7.4.1. Introducción....43
7.4.2. Glosario....43
7.4.3. Requisitos específicos....43
7.5. Casos de uso (actualizados junio 2009)....45
CU-001: Crear una nueva red social....45
CU-002: Guardar red social....45
CU-003: Cargar red social....46
CU-004: Crear nodo....46
CU-005: Seleccionar nodos....47
CU-006: Modificar nodo o relación....47
CU-007: Borrar nodos....48
CU-008: Crear contenidos y comentarios....48
CU-009: Crear relación....49
CU-010: Eliminar relación....49
CU-011: Visualizar subredes....49
CU-012: Poner afición a un nodo....50
CU-013: Quitar afición a un nodo....50
CU-014: Crear informe....50
CU-015: Realizar simulación....51
CU-016: Desplazar selección....51
CU-017: Ver más información de la red....52
CU-018: Elegir visualizaciones alternativas de la red....52
CU-019: Ampliar el mapa de la red....52
7.6. Manual de usuario....54
7.6.1. Operaciones básicas de nodos y relaciones....54
7.6.2. Simulaciones....54
7.6.3. Crear una red social nueva....54
7.6.4. Cargar y guardar redes sociales....55
7.6.5. Opciones de visualización....55
7.6.6. Barra de métricas....55
7.6.7. Ayuda interactiva....55
7.7. Información de copyright....56
7.7.1. Krowdix, simulador de redes sociales....56
7.7.2. Iconos de Tango Icon Theme....56
7.7.3. JUNG....56
7.7.4. JFreeChart....56

7.7.5. JNA....56
7.8. Contenido del CD....57
7.8.1. Versión electrónica de la memoria....57
7.8.2. Versión ejecutable de la aplicación....57
7.8.3. Código fuente....57
7.8.4. Prototipos y versiones antiguas....57
Bibliografía....58
Licencia de este documento....59
Usted es libre de:....59
Bajo las condiciones siguientes:....59
Autorización....60

1. Introducción

Desde su creación, las redes de ordenadores se han utilizado para intercambiar opiniones.

En los años 60 nació el correo electrónico (1). Durante mucho tiempo, este fue el modo habitual de enviar información de un ordenador a otro. De hecho, hoy en día el correo electrónico sigue usándose y es de hecho uno de los medios de comunicación más usados en todo el mundo: algunos proveedores como Hotmail tienen más de 200 millones de clientes (2) y en 2009 se concedió el premio Príncipe de Asturias al inventor del correo electrónico por su contribución a la comunicación global. Sin embargo, el correo electrónico como medio de comunicación tiene ciertas limitaciones: los mensajes tienen un grupo fijo de destinatarios y el autor debe conocer la dirección de todos ellos.

A medida que el número de usuarios de Internet crecía, a finales de los años 70 surgieron los BBS y los grupos de noticias (3). El funcionamiento de estos grupos era similar al de un tablón de anuncios real: un usuario accede al tablón, cuelga en él los mensajes que ha escrito y mira lo que han colgado otras personas. Muchos de estos BBS ni siquiera estaban en Internet, para participar en ellos había que conectarse directamente al número de teléfono del servidor en el que estaban.

En los años 90, Internet experimenta un gran crecimiento. Los BBS que no están en Internet se trasladan a la autopista de la información, pero aún así la gran cantidad de mensajes generados hace que los grupos de noticias empiecen a quedarse pequeños y surgen las comunidades virtuales. Estas comunidades son una evolución de los grupos de noticias, en las que se apuesta por la especialización: los grandes grupos se dividen en pequeñas particiones sobre temas concretos para que cada persona participe en aquellos temas en los que estuviera interesado. Las comunidades virtuales más habituales son los foros, basados en el web, y las listas de correo, basadas en los mensajes de correo electrónico.

Aunque las comunidades virtuales siguen siendo hoy en día un medio importante de comunicación, tienen algunos problemas: mucha gente se siente coartada, ya que los administradores de los foros pueden decidir que mensajes aparecen y que mensajes no. Otro tanto pasa en las listas de correo, donde un usuario puede ser expulsado si no cumple las normas de netiqueta o se dedica a escribir mensajes fuera del tema. Esto hace que a principios del siglo XXI los *blogs* y *fotologs* experimenten un crecimiento exponencial: los *blogs* facilitan que la gente pueda tener una página propia en Internet, en la que dedicarse a escribir sobre lo que se quiera, sin tener que preocuparse por ceñirse a un tema ni por si un administrador borrará sus mensajes.

A finales de la primera década de siglo, los blogs sufren una crisis. Los posibles lectores de blogs tienen una cantidad ingente de material, pero no tienen un modo sencillo de acceder únicamente a aquel en el que están interesados; al sentirse saturados, dedican su atención a otros temas. Millones de blogs "mueren" (dejan de actualizarse), o directamente se cierran cuando sus autores se frustran por no tener lectores. La gente busca un nuevo modo de comunicarse y aparecen las redes sociales.

Las redes sociales surgen originalmente como comunidades virtuales de estudiantes universitarios, se orientan en torno a gente con intereses comunes. Sin embargo, pese a que en una red pueden participar miles de personas, en realidad está fragmentada en subredes más pequeñas de "amigos" o "contactos". Estas subredes son una especie de híbrido entre las comunidades virtuales, al ser grupos más o menos pequeños, y los blogs, en los que la gente puede escribir libremente.

El atractivo de las redes sociales hace que pronto abandonen su orientación original y pueda participar en ellas cualquier persona. El crecimiento de las redes sociales es imparable, tanto que tras las elecciones legislativas de Irán en junio 2009 la red Twitter se convirtió en el principal medio (4) que usaron los opositores para comunicarse entre sí y para informar al mundo de lo que estaba pasando en el país.

En el momento actual, las redes sociales siguen siendo nuevas. Son un buen negocio para la publicidad en línea, algunas empresas se están dedicando a crear sus propias redes para atraer esos ingresos y los partidos políticos utilizan las redes más importantes para difundir su programa electoral. Pero resulta difícil saber cómo evolucionarán estas redes en el futuro.

¿Sería interesante realizar nuevas inversiones para ampliar las redes?, ¿es factible crear nuevas redes o el mercado está saturado?, ¿el número de usuarios de las principales redes seguirá creciendo?, o incluso ¿cómo se propagan las noticias a través de las redes existentes?

Krowdix surge para intentar resolver algunas de estas preguntas.

2. Estado actual

2.1. Programas existentes

Las redes son estructuras con una larga historia, tanto en el campo de la informática (por ejemplo, redes de computadores) como en otros campos (por ejemplo, redes de distribución en cadenas de supermercados).

El análisis de redes es, por tanto, un tema sobre el que se ha trabajado mucho. Por este motivo, en el momento de empezar el proyecto ya existía una vasta colección de herramientas para el análisis de redes en general. Las redes sociales, sin embargo, son un modelo de red relativamente reciente, por lo que las herramientas en este campo son menos numerosas.

Antes de decidir los objetivos que debería cumplir el proyecto, se estudiaron algunas de estas herramientas para ver lo que se había hecho hasta el momento.

2.1.1. KrackPlot

El primer programa que se examinó fue KrackPlot (5). Pertenece a la categoría de programas de análisis de redes en general.

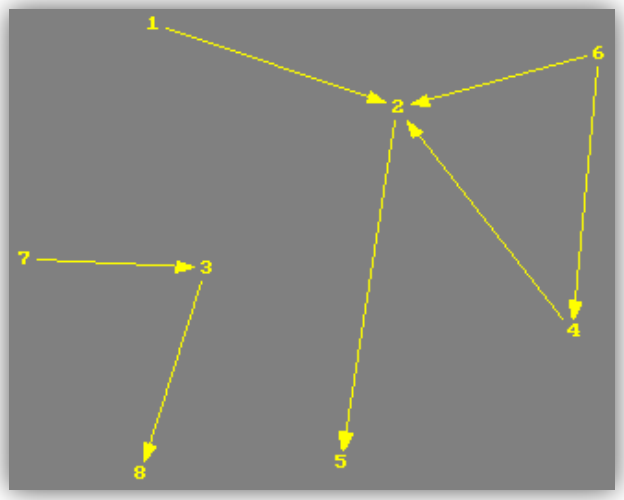

Ilustración 1: red creada usando KrackPlot

El programa tiene una interfaz muy simple y se ejecuta en el modo gráfico de MS-DOS.

Permite calcular distintas métricas, tales como el índice de dispersión de los elementos del grafo, o la "alcanzabilidad" de la red (el promedio de nodos que pueden ser alcanzados desde otro nodo del grafo).

2.1.2. NetDraw

Otro programa que se estudió fue NetDraw (6). Al igual que el anterior, pertenece al grupo de programas para analizar redes en general.

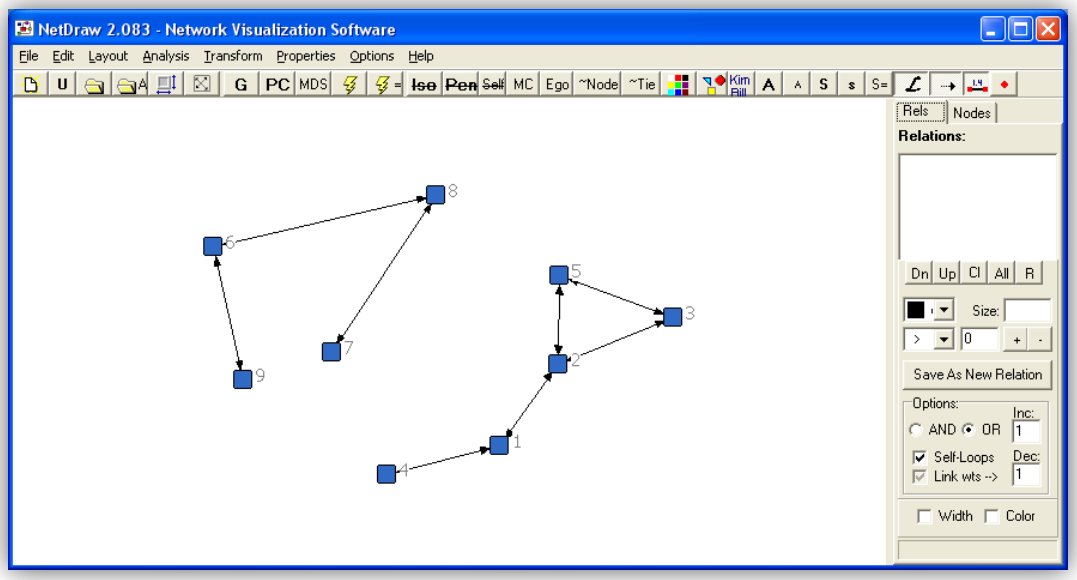

Ilustración 2: captura de pantalla de NetDraw

Este programa tiene una interfaz gráfica de ventana, bastante más agradable que la del programa anterior. Sin embargo, resulta poco intuitivo.

Ofrece mayores posibilidades de edición al permitir asignar atributos a los nodos y dar valor a las relaciones. En lo que respecta al análisis, ofrece muchas más métricas que KrackPlot. También permite crear una red aleatoria indicando el número de nodos y relaciones que se desean.

2.1.3. SocNetV

El último programa analizado fue SocNetV (7). Al contrario que los anteriores, este programa se descubrió cuando el trabajo en el proyecto ya había empezado y se encontraba en un estado avanzado. Es el único programa examinado específico para el análisis de redes sociales.

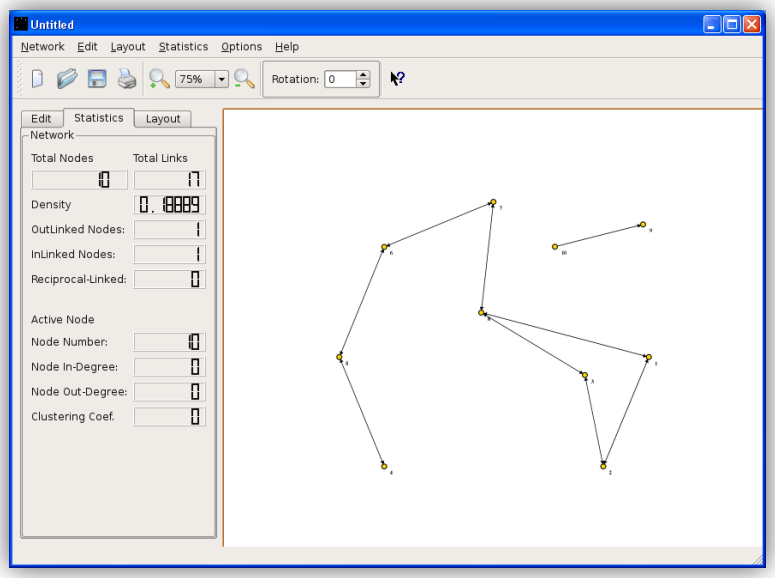

Ilustración 3: captura de pantalla de SocNetV

Este programa también presenta una interfaz de ventana visualmente agradable y bastante intuitiva. En cambio, no permite añadir atributos a los nodos.

2.2. Conclusiones

Las herramientas existentes permiten dibujar redes y obtener cierta información sobre ellas (centralidad, etc.). El problema es que estas herramientas sólo analizan las conexiones entre los nodos de la red, dejando de lado algo tan importante como el contenido creado por esos nodos.

Además, ninguna de las herramientas analizadas ofrece ningún tipo de inteligencia artificial para ver cuál puede ser la evolución de la red a partir de un estado dado.

3. Objetivos

Después de analizar el estado actual se llegó a la conclusión de que ninguno de los programas existentes servía para responder las preguntas que planteaban las redes sociales. De este modo, se decide crear una nueva aplicación.

Algo que queda claro desde un principio es que el desarrollo de una aplicación de este calibre es una tarea demasiado grande para realizarla con el tiempo disponible, por lo que la creación del programa debe dividirse en dos años. Para el primer año del trabajo, los objetivos serán:

3.1. Creación y modificación de redes sociales

El programa tratará con redes sociales. El primer objetivo es, por tanto, que las pueda crear y modificar.

3.2. Análisis de la red

Todos los programas existentes permiten un análisis más o menos detallado de la red. El programa a desarrollar debe incluir también esta facilidad.

3.3. Interfaz intuitiva y agradable

Algunos de los programas existentes resultaban excesivamente cargados o difíciles de manejar. Se hace imprescindible que la creación de redes y las modificaciones que se hagan en esas redes se puedan hacer usando una interfaz sencilla y a la vez fácil de manejar.

3.4. Inteligencia artificial.

Algo de lo que carecían todos los programas existentes era la simulación de redes existentes para ver cómo una red puede evolucionar a partir de un estado inicial. El programa debe poder realizar estas simulaciones sobre la red que se esté manejando.

3.5. Persistencia

Las redes se deben poder guardarse en disco y recuperarse después, para poder trabajar con una misma red entre diferentes sesiones.

3.6. Soporte multiplataforma

Durante las pruebas se descubrió que las herramientas existentes estaban diseñadas en su mayor parte para sistemas Windows. Debido al auge de otros sistemas en los últimos años, es un requisito que la aplicación pueda ejecutarse en el mayor número posible de plataformas.

3.7. Posibilidad de ampliación

Por último, si en un proyecto cualquiera hay que tener presente en todo momento que la aplicación *puede* ser ampliada en un futuro, en este proyecto hay que tener en cuenta que de hecho la aplicación *va a ser* ampliada.

4. Memoria del proyecto

En este apartado se hace un pequeño resumen de las fases de la aplicación.

4.1. Octubre de 2008

Éste fue el primer mes de trabajo en el proyecto. Sirvió principalmente para que los miembros del grupo nos conociéramos.

También se aprovechó para estudiar qué se había hecho en el campo de las redes sociales y que quedaba que hacer. Se escribió el primer borrador de la especificación de requisitos de software.

4.2. Noviembre de 2008

En la parte de análisis se decide una versión final de la especificación de los requisitos de software. Estos requisitos se pormenorizan en un nuevo documento que recoge los distintos casos de uso de la aplicación.

Una vez terminado el análisis de la aplicación se pasa a la fase de diseño. Se decide una organización básica del trabajo de implementación a partir de la que empezar trabajar y se empiezan a dibujar los primeros prototipos de interfaz.

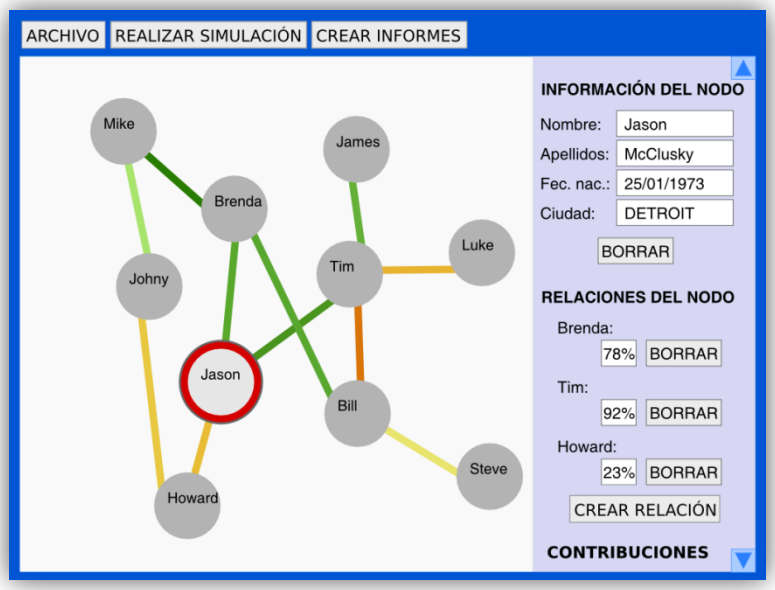

Ilustración 4: primer borrador de la interfaz

También se deciden las herramientas que se usarán durante el desarrollo. Estas herramientas se detallan en la sección 5.6, herramientas y detalles de implementación.

Ilustración 5: primer logotipo para la aplicación

4.3. Diciembre de 2008

Se sigue trabajando en el diseño de la aplicación, creando una jerarquía de objetos para las funcionalidades básicas.

Comienza la implementación, creando un prototipo de interfaz con una funcionalidad básica, básicamente un editor de grafos.

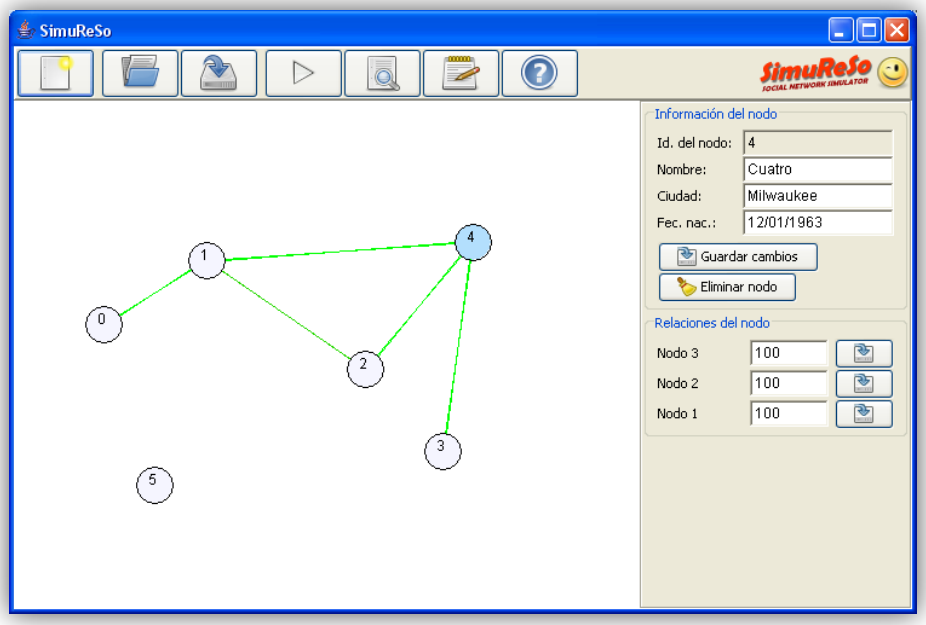

Ilustración 6: prototipo de interfaz

4.4. Enero de 2009

Debido a una falta de coordinación, el prototipo realizado en diciembre presenta muchos fallos. Se corrigen algunos de estos errores y se comienza a utilizar un repositorio, para facilitar que varias personas puedan trabajar a la vez en el mismo código y los cambios sean más sencillos de integrar.

No obstante, el fracaso de diciembre hace mella en los componentes del grupo y reina un desánimo general. A mediados de mes, el desarrollo se interrumpe para preparar los exámenes de febrero.

4.5. Febrero de 2009

Una vez terminados los exámenes, en una reunión extraordinaria del grupo se decide no abandonar el proyecto y redoblar los esfuerzos por sacarlo adelante. La decisión principal es la de realizar reuniones semanales y preparar un plan de entregas, dividiendo el trabajo a realizar en tareas más o menos independientes.

En la parte de diseño, se estudian los problemas surgidos en el prototipo y se realizan cambios en la arquitectura. Se tiene una versión prácticamente definitiva del diseño de la aplicación.

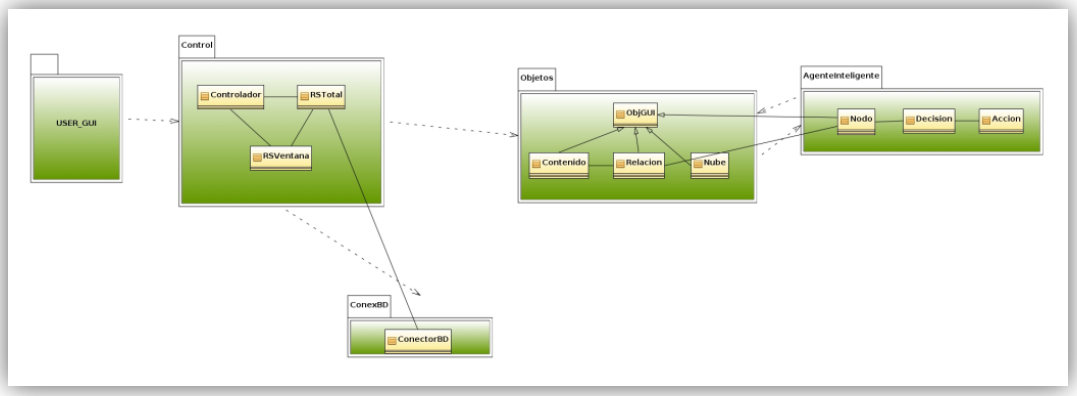

Ilustración 7: UML simplificado del diseño decidido

Comienza el desarrollo real de la aplicación a partir de los documentos del diseño. Se modifica la interfaz para que pueda trabajar con datos reales. Además, se crea la primera versión de la inteligencia artificial.

Todo lo anterior hace que a finales de febrero se tenga un proyecto maduro. Se decide cambiar el nombre del proyecto para que se refleje que se está haciendo algo distinto a lo que se había hecho anteriormente.

Ilustración 8: nueva imagen del proyecto

4.6. Marzo de 2009

En marzo, se amplía la funcionalidad y se corrigen todos los errores encontrados en la parte de inteligencia artificial, con lo que termina el desarrollo de los agentes inteligentes. Se integra esta inteligencia con la interfaz, permitiendo realizar simulaciones en la red.

Se realizan muchos cambios en la interfaz: se añade una opción para crear redes a partir de ciertos parámetros, así como nuevos paneles para permitir al usuario editar más atributos de los nodos. El cambio más importante es que para la representación gráfica, cambiando la representación de los nodos por otra más agradable, añadiendo funcionalidades como el zoom e introduciendo mejoras de rendimiento.

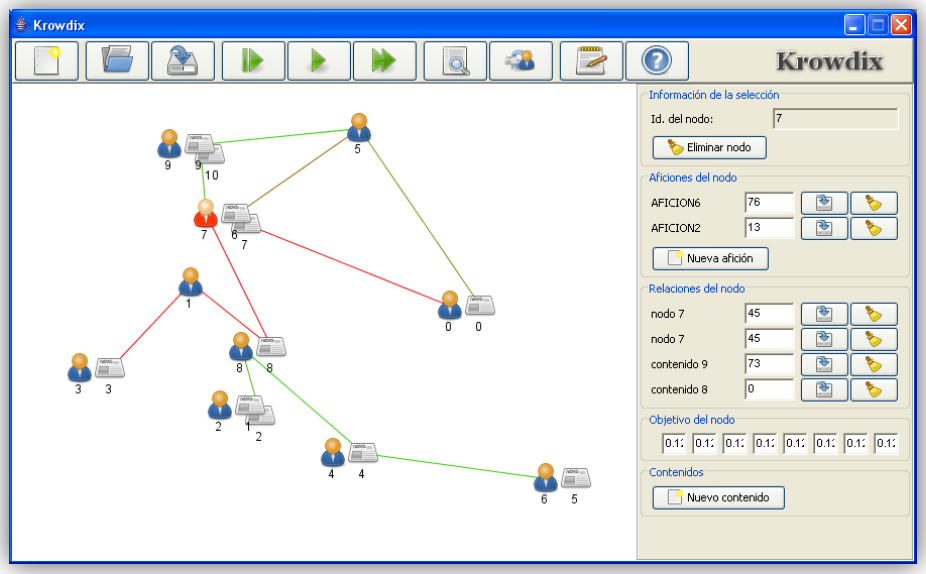

Ilustración 9: captura de la aplicación tras las mejoras en la interfaz

A finales de marzo y a pesar del trabajo realizado, se decide que la realización de un motor gráfico no forma parte de los objetivos originales de la aplicación, por lo que se cambia la representación de la red para utilizar una librería gráfica de representación de grafos. Esto provoca algunos cambios en la jerarquía de clases, eliminando las partes que dejan de ser necesarias.

Una vez terminada la migración al nuevo motor gráfico, se añaden nuevas funcionalidades a la interfaz: menús contextuales, nuevos gestos de ratón, se mejoran las ventanas existentes, etc.

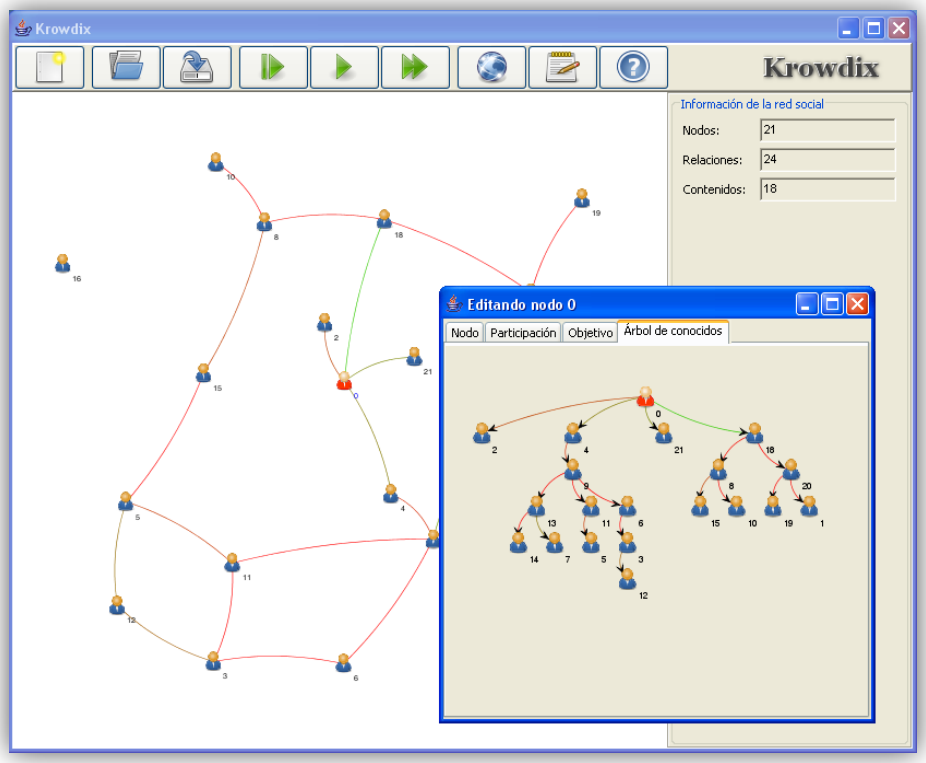

Ilustración 10: captura de la aplicación con la nueva librería gráfica

4.7. Abril de 2009

Se realizan nuevas mejoras en la inteligencia artificial, permitiendo que los agentes inteligentes utilicen mejor la información obtenida del entorno para alcanzar antes su objetivo.

Se realiza un gran trabajo en la parte de análisis de red, mejorando los informes creados y añadiendo gráficos para reflejar el estado de la red.

También continúa el trabajo en la parte de la interfaz, añadiendo efectos gráficos para mejorar la apariencia. Se reordenan las opciones para que resulten lo más intuitivas posibles y se crean asistentes para realizar las tareas más complejas.

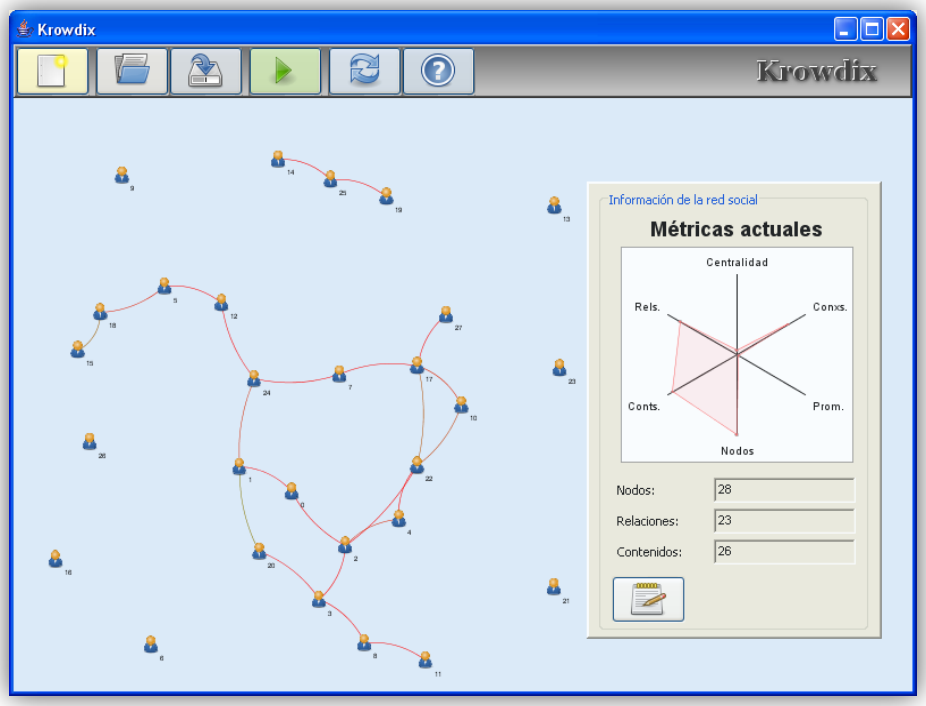

Ilustración 11: captura de la aplicación tras los últimos cambios

4.8. Mayo de 2009

Se amplía la interfaz, añadiendo una agrupación de aficiones (usando diagramas de Voronoi). Se modifican los asistentes, siguiendo los consejos de la gente que se ofreció a probarlos. Se añaden atajos de teclado para algunas de las acciones.

Se añaden nuevos gráficos, tanto para ver cómo cambia la red tras una acción cualquiera como para mostrar la evolución de la red desde su creación.

Se analiza todo el código, comentando todo lo necesario para preparar las futuras ampliaciones de la aplicación. Se corrigen errores de diseño encontrados. Se realizan pruebas intensivas para detectar posibles fallos de la aplicación, y se corrigen todos los encontrados. Se considera como "final" la versión resultante tras este proceso.

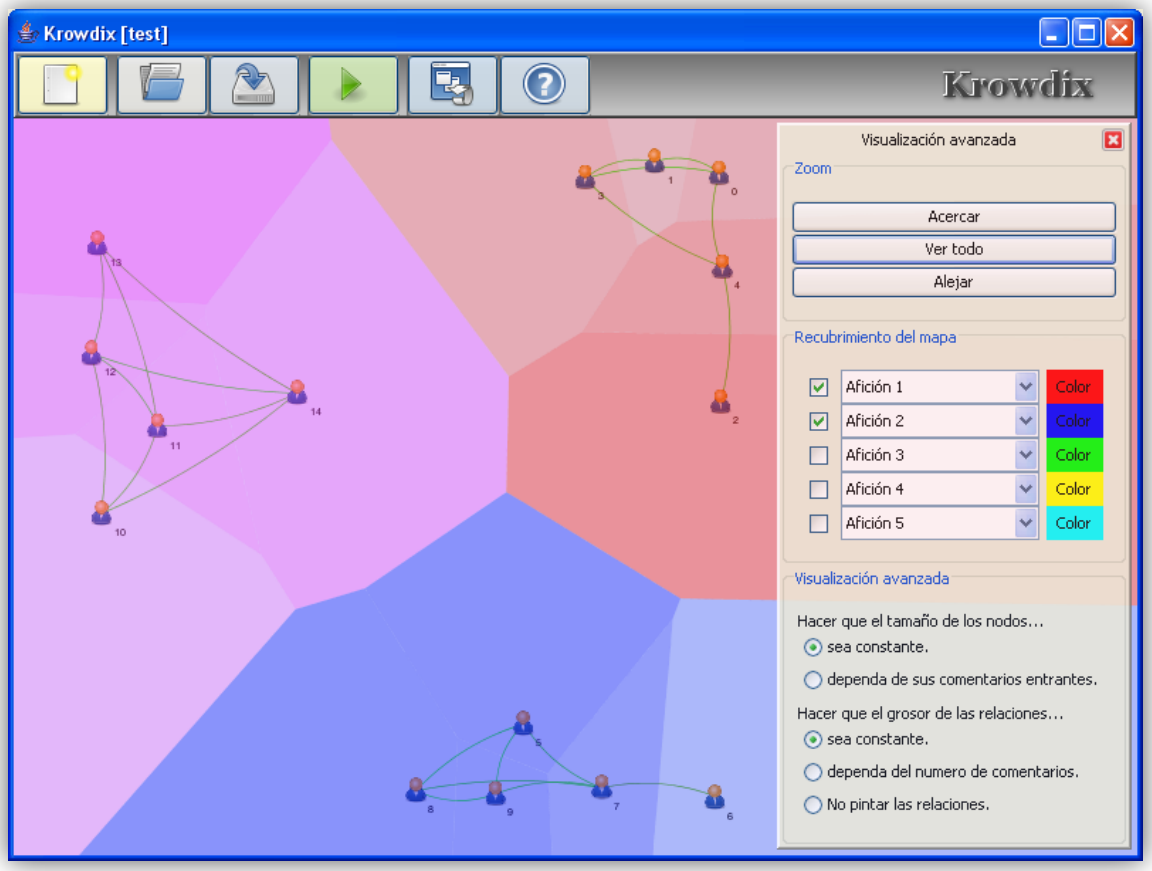

Ilustración 12: captura de la versión final de la aplicación

4.9. Junio de 2009

Una vez terminada la parte de implementación, los esfuerzos del grupo se centran en preparar la documentación del mismo (esta memoria). Termina el desarrollo de la primera parte del proyecto.

5. Arquitectura de la aplicación

5.1. Vista general

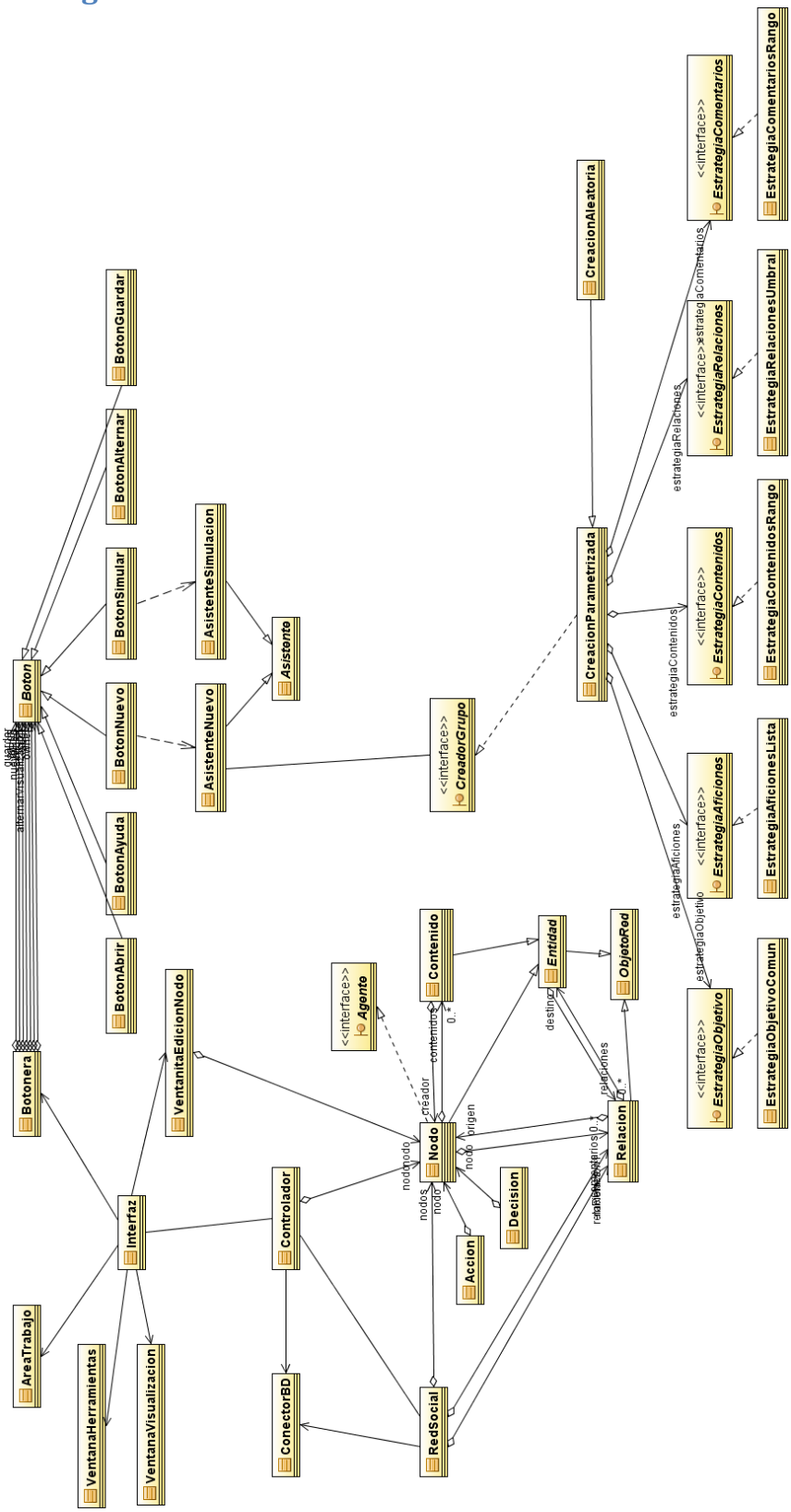

Ilustración 13: vista general de la arquitectura de la aplicación

5.2. Módulos de la aplicación

5.2.1. Red social

Nuestra red se basará en la clásica representación de redes sociales en forma de grafo. Es decir, tendremos:

- una serie de **nodos**: los usuarios de la red;
- y unas aristas que los unen: las conexiones entre los nodos.

Esta estructura, bastante común en los programas de análisis de redes, nos resultaba insuficiente para cumplir nuestros objetivos. Por ello, se añadieron algunos atributos adicionales a los nodos:

- una lista de **aficiones**, cada una con un grado: representan en que temas están interesados los nodos, y como de interesados están en esos temas;
- un conjunto de **objetivos**: lo que el nodo pretende conseguir en la red;
- una lista de **contenidos**: la información que el nodo publica en la red.

Además, dividimos las conexiones en dos tipos:

- **relaciones**: conexiones no dirigidas entre dos nodos, con un valor que indica la afinidad o fortaleza de la conexión;
- **comentarios**: conexiones dirigidas entre un nodo y un contenido creado por otro nodo, también con un valor.

Resumiendo, el diseño del modelo de red social es el siguiente: la parte principal del modelo será la RedSocial; la RedSocial tendrá Nodos y Relaciones; los Nodos tendrán una lista de Contenidos creados por ellos, un vector de objetivos y la lista de las Relaciones en las que participan, mientras que las Relaciones sólo sabrán cuáles son sus extremos y cuál es la afinidad entre ellos. Los Contenidos también cuentan con una lista de Relaciones, por lo que por sus características comunes, tanto los Nodos como los Contenidos se consideran Entidades. Por último, las Entidades y las Relaciones también tienen rasgos comunes, como la presencia de un identificador único; se consideran, por tanto, como ObjetosRed.

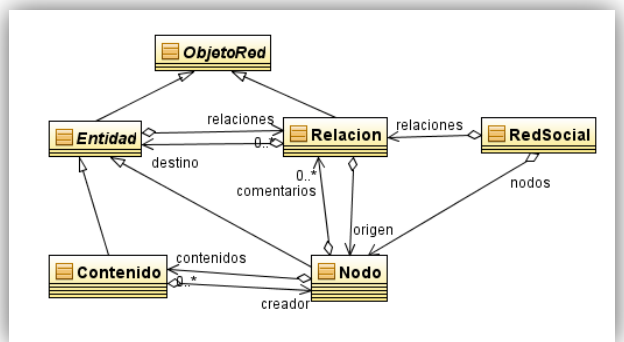

Ilustración 14: modelo de red social

5.2.2. Interfaz

Desde el primer momento se tuvo en cuenta que la parte visual de la aplicación iba a ser muy importante. Se creó una interfaz visualmente agradable, intuitiva y fácil de manejar.

La ventana principal se redujo a la mínima expresión, ocupando la mayor parte de ella un panel que muestra la red: los nodos de la red se muestran con un dibujo que representa una

persona, mientras que las relaciones entre ellos se muestran como líneas cuyo color depende de su valor. En la parte superior se prescindió de la clásica barra de menús, que se sustituyó por una botonera más "visual".

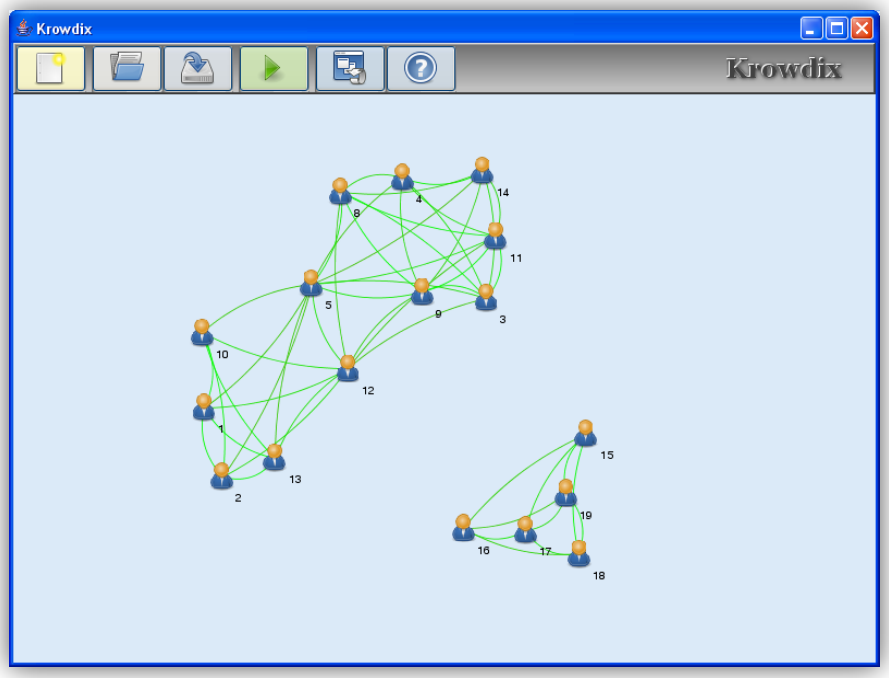

Ilustración 15: interfaz de la aplicación

Los nodos del panel se pueden editar haciendo doble clic sobre ellos, o seleccionando la opción correspondiente en el menú contextual. Al iniciar la edición de un nodo se muestra una pequeña ventana emergente que muestra todos los datos que se pueden modificar.

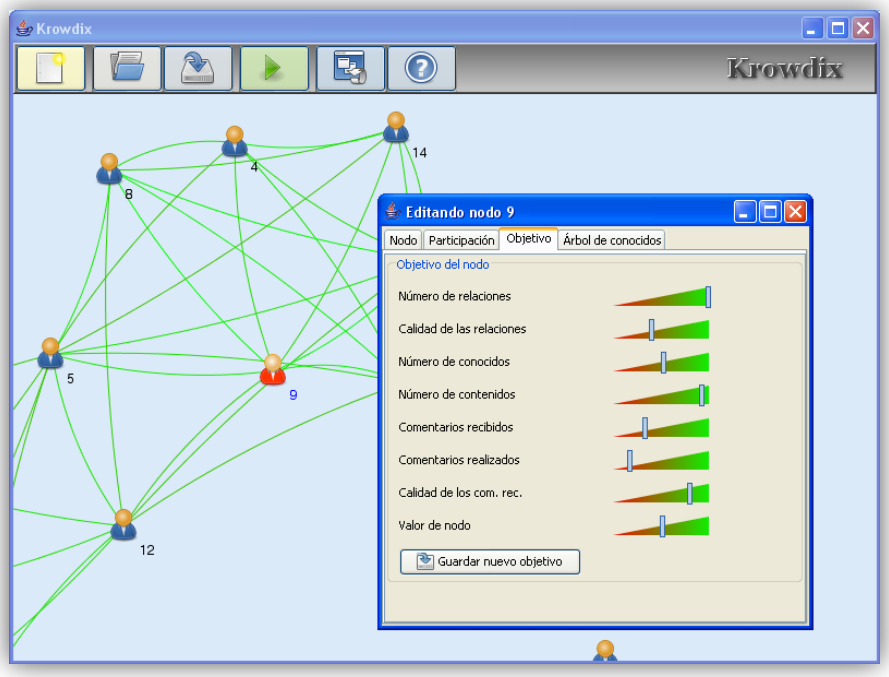

Ilustración 16: ventana emergente para editar atributos de los nodos

Por otra parte, los botones de la botonera superior sirven para realizar acciones sobre la red (crear, guardar, simular, etc.). Al pulsarse despliegan, a su vez, una botonera auxiliar que da opciones adicionales para la acción que queremos realizar. Esta botonera auxiliar toma el color del botón que la creó, de modo que resulta mucho más intuitivo saber que se está haciendo en cada momento.

Ilustración 17: distintas botoneras para distintas acciones

Otro punto a destacar es que la botonera secundaria no despliega más paneles, por lo que las acciones principales están a dos clics de distancia del estado "normal". Para realizar acciones especiales se dispone de asistentes que guían al usuario mediante pasos y ofrecen en todo momento ayuda contextual para mayor facilidad.

Ilustración 18: ejemplo de asistente, asistente de simulación

Por muy "visual" que sea la pantalla que muestra la red, hay cosas que resultan difíciles de ver (cuántos nodos hay en total, cuál es el promedio de relaciones por nodo, etc.). Este tipo de datos, que nosotros llamamos métricas, tiene que reflejarse de algún modo. En la interfaz hay una ventana auxiliar con gráficos que muestran los valores actuales y la evolución a lo largo del tiempo de esas métricas.

Ilustración 19: ventana auxiliar con gráficos de la red

A partir de la interfaz también se puede obtener un informe, en modo texto, con todos estos datos. Este informe se puede guardar para futuras referencias.

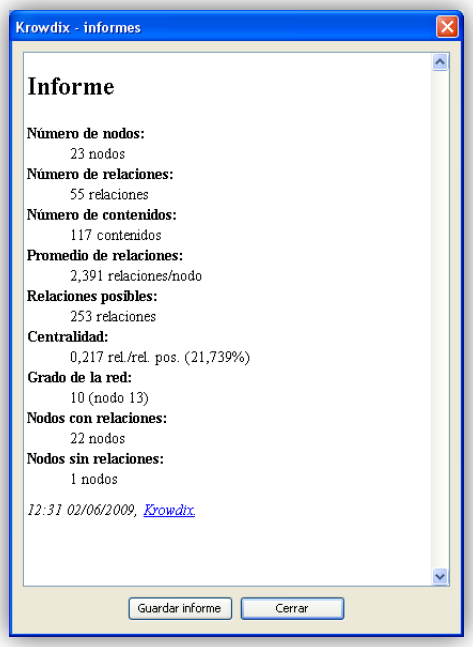

Ilustración 20: informe de una red

Además de los gráficos y los informes, la interfaz también tiene opciones para ver algunas características de la red en el propio panel de edición. En primer lugar, la representación de los nodos y las relaciones puede cambiarse: en lugar de pintar todos los nodos del mismo tamaño, pueden pintarse con un tamaño diferente en función de alguna característica como el número de sus comentarios entrantes; las relaciones, por su parte, pueden también pintarse de distinto modo, por ejemplo variando su grosor en función de la cantidad de comentarios intercambiados entre sus extremos.

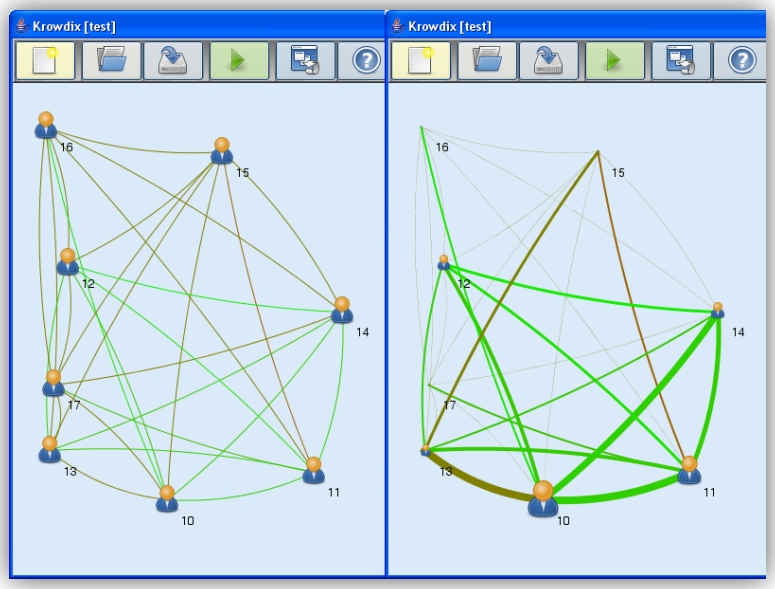

Ilustración 21: dos modos de ver una misma red

Por último, las aficiones de los nodos juegan un papel muy importante. Por ello, se añade la opción de ver en el mapa de nodos una representación de qué nodos tienen qué aficiones, usando diagramas de Voronoi.

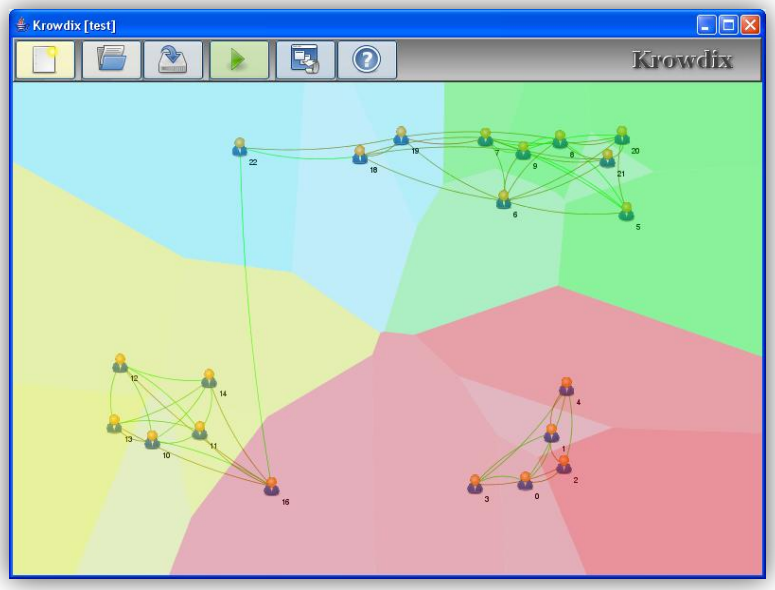

Ilustración 22: recubrimiento de la red usando diagramas de Voronoi

Teniendo en cuenta todo lo anterior, la estructura de la interfaz es la siguiente. Una Interfaz tiene un AreaTrabajo y una Botonera. La Botonera, por su parte, contiene Botones que pueden generar Asistentes. Por último, con mediación del Controlador, algunas acciones en la Interfaz pueden abrir ventanas adicionales, como la VentanitaEdicionNodo, la VentanaHerramientas y la VentanaVisualizacion.

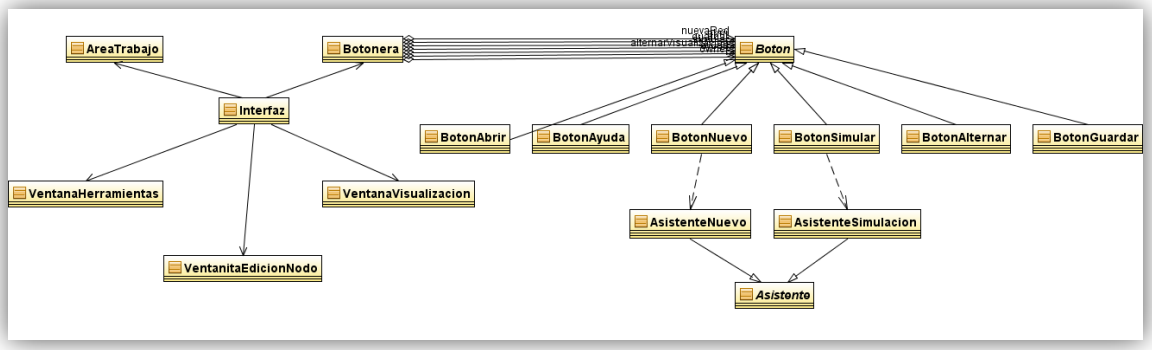

Ilustración 23: modelo de la interfaz

5.3. Agentes inteligentes

5.3.1. Introducción

"Un agente inteligente, es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado" (8)

Krowdix es una herramienta de estudio de Redes Sociales que se destaca del resto por ser capaz de simular el comportamiento de cada individuo además de analizar la red como un todo. En este apartado nos centraremos en las técnicas usadas para lograr un comportamiento racional por parte de los elementos que componen nuestras Redes Sociales.

Para conseguir unos Agentes Inteligentes fieles a su definición, expondremos el desarrollo de nuestros agentes siguiendo paso a paso la definición anteriormente citada:

*"Un **agente inteligente**, es una entidad capaz de percibir su entorno, procesar tales percepciones..."*

Es de esperar que un agente que quiere tomar acciones adecuadas para conseguir un fin ha de conocer el entorno que le rodea ya que su progreso depende de su interacción con el mismo.

En Krowdix parametrizamos el entorno desde el punto de vista de cada individuo ya que en un sistema tan variado es improbable que desde dos puntos diferentes se observe el mismo entorno.

"...y responder o actuar en su entorno de manera racional, es decir, de manera correcta..."

Razonar es discurrir, ordenando ideas en la mente para llegar a una conclusión, por ello hemos de diseñar los Agentes inteligentes teniendo en cuenta que han de tener un objetivo, que será lo que impulse al individuo a tomar una decisión u otra para alcanzarlo.

"...y tendiendo a maximizar un resultado esperado."

Es necesario aprender de nuestras acciones para maximizar el resultado, si una acción no produce el efecto esperado, es necesario aprender y corregir nuestro conocimiento para no caer en el error más veces, de esta forma maximizaremos el resultado de nuestras acciones hacia nuestro objetivo.

5.3.2. Arquitectura

Conocido el problema podemos diferenciar tres fases que componen el comportamiento de los agentes de Krowdix, por ello nos hemos basado en el modelo de 3 niveles para agentes inteligentes propuesto por (9) que divide el comportamiento de un Agente Inteligente en tres fases, Percepción, Decisión y Acción. Entraremos ahora a fondo en el funcionamiento de cada una de estas fases.

5.3.2.1. Percepción

El ser humano es curioso por naturaleza y eso le lleva a observar el entorno para conocerlo, este conocimiento lo extrae de conclusiones que obtiene de un análisis de la situación de la cual no necesita conocer hasta el más mínimo detalle, pero sí los que considera más importantes y representan toda la información que define dicha situación.

En una Red Social creada con Krowdix, tenemos Nodos, Aficiones, Contenidos, Relaciones y Comentarios, por lo tanto todo lo que un Nodo pueda percibir del entorno se obtendrá de la información que podamos extraer de estos parámetros. Cambiar o añadir cualquier parámetro

puede cambiar completamente el comportamiento de los Agentes Inteligentes, y el equipo de Krowdix hemos definido los siguientes:

- **Cantidad de Relaciones:** Cualquier persona partícipe de una Red Social tiene una idea sobre cuántos amigos desearía tener, no un número fijo sino más o menos de lo normal, habrá gente que desee tener muchos amigos, pocos o indiferente. Gracias a este parámetro podemos conocer si nuestra situación es la que deseábamos.
- **Media de calidad de las Relaciones:** Es de esperar que haya gente más preocupada sobre la calidad de sus relaciones que sobre la cantidad, calcularemos la media de las relaciones del nodo para conocer su situación.
- **Número de Conocidos Rango:** Si tenemos en cuenta que tenemos más probabilidad de entablar nuevas relaciones cuanta más gente nueva podamos conocer a través de nuestras relaciones, nos preocuparemos por aumentar este índice que nos dice cuántas relaciones de los nodos con los que estamos relacionados no conocemos todavía.
- **Índice de Contenidos no útiles:** Está bien ofrecer Contenidos a los demás, pero publicar Contenidos que no interesan a nadie frustra al autor como ocurre con los blogs que nadie lee, terminan por morir. Gracias a este índice el nodo puede saber cuántos Contenidos suyos que caen en desuso o que no interesan a nadie.
- **Comentarios en mis Contenidos:** Tener comentarios en los contenidos está bien, pero hace falta tener una referencia sobre la cantidad de comentarios de la red social que acaparamos en nuestros Contenidos. Obtenemos pues el porcentaje de comentarios de la red social que van a parar a nuestros Contenidos.
- **Número de Comentarios realizados:** Darse a conocer en los contenidos de otros nodos es útil también para conseguir futuras relaciones, puede ser una carta de presentación hacia otros nodos.
- **Media calidad en mis Contenidos:** Puede preocuparnos también el hecho de que se comente bien o mal sobre nuestros contenidos, ya que no siempre tienen por qué ser buenos. Esta media nos dará una idea sobre lo que opina la gente sobre nosotros.
- **Valor del Nodo:** Podríamos valorar a un nodo de muchas formas, nosotros hemos decidido que los nodos más importantes son aquellos que tienen más información en Contenidos con buenos comentarios, es decir, aquellos que ofrecen a la Red Social más comunicación, son nodos que difunden información de manera más eficiente por la Red Social.

5.3.2.2. Decisión

Para decidir qué acción realizar el nodo ha de tener un objetivo, ya que de otra forma, todas las acciones serían igual de válidas.

La decisión también es un aspecto importante en el comportamiento, y hay diferentes formas de evaluarlo; nosotros hemos decidido que el nodo sea capaz de predecir lo que una acción va a provocar en su entorno, de esta forma, si deseamos llegar a un objetivo concreto simplemente tenemos que realizar aquella acción que a partir de nuestro estado nos acerque lo más posible a nuestro objetivo.

Representamos por lo tanto el objetivo como los valores a los que han de llegar los parámetros que definen nuestro entorno por medio de nuestras acciones, tendremos en cuenta todos y cada uno de los parámetros en cada objetivo.

Cada acción que podemos realizar tendrá por lo tanto asociada tantos valores como parámetros definen el entorno, que multiplicándolos por el parámetro correspondiente, predicen el valor que tendrá una vez realizada la acción. Este conjunto de valores son los elementos de predicción de cada acción.

En el momento de la decisión analizaremos nuestro entorno y a continuación probaremos todas y cada una de las acciones y compararemos la predicción con el objetivo, cuanto más se parezcan, mejor será la acción y decidiremos en consecuencia.

5.3.2.3. Acción

Este módulo termina de darle sentido a todo el sistema, ya que además de realizar las acciones del nodo, realiza todo el esfuerzo del aprendizaje, haciendo que el sistema actúe siempre de una manera racional.

Basándonos en el funcionamiento del Perceptrón (10), elemento básico de las redes neuronales, haremos que un nodo sea capaz de aprender, ajustando los valores de la predicción, de los resultados obtenidos por sus propias acciones.

El aprendizaje se consigue a partir de un método tan natural como el de ensayo y error, método por el que aprenden todos los animales, incluidos los seres humanos.

En un momento inicial el nodo, como si de un bebé se tratara, no sabe qué provocan las acciones que puede realizar, sólo probándolas y examinando los cambios provocados en el entorno extraerá conclusiones sobre su efecto. Pero las acciones no siempre tienen el mismo resultado, ya que en una Red Social que tiende a evolucionar, una acción que en una situación podría tener mucho éxito, en otras condiciones podría no serlo; por esto es necesario aprender a medida que vamos realizando las acciones, a lo largo de toda la vida del nodo.

De la misma forma que una acción exitosa en un momento determinado puede no ser tan efectiva en un futuro, una acción que no es exitosa en otro momento, podría tener más éxito en otro futuro, por lo tanto aquellas acciones que una vez fueron desechadas por no conducir al objetivo, deberán ser probadas de nuevo periódicamente, para confirmar o corregir los resultados.

El aprendizaje consiste por lo tanto en corregir los elementos de predicción de las acciones para ser cada vez más fiables.

Las acciones que hemos considerado convenientes para que evolucionen las Redes Sociales en Krowdix son las siguientes:

Iniciar Relación

Ninguna Red Social podría prosperar sin relaciones entre sus nodos, sin embargo, esta acción básica entraña más trabajo del estimado inicialmente, puesto que el inicio de una relación depende de varios aspectos.

Para comenzar, un nodo que desee iniciar una nueva relación, ha de buscar un candidato, este candidato ha de conocerlo de de alguna forma, ya sea por haber comentado en contenidos comunes o propios, o por tener una relación con un nodo común. De todos los nodos que se puedan conocer por este método, intentaremos establecer relación con aquel que sea más afín a nosotros, es decir, con el que tengamos más probabilidades de iniciar una relación, esto lo determinaremos por lo parecidas que sean nuestras aficiones, ya que cuando hagamos una petición de amistad, será el otro nodo el que determine si realmente le interesa iniciar una nueva relación o no, y esto depende en parte de la afinidad, ya que una afinidad demasiado baja será desechada inmediatamente.

Crear Contenido

Los contenidos representan la información que un nodo comparte con el resto de la Red Social, y éste representa alguna de nuestras aficiones; es la mejor forma de conocer a otros nodos con la misma afición, ya que como veremos en la siguiente acción los contenidos nos

permiten llegar a conocer a nodos que no sean muy propensos a crear nuevas relaciones y por lo tanto haga falta recorrer varios enlaces para llegar a él.

Hacer Comentario

Hacer un comentario en un contenido da la oportunidad al nodo de dar a conocerse, los contenidos son una de las formas de conocer a todos aquellos nodos que no están interesados en iniciar relaciones a la ligera y sólo buscan relaciones de cierta afinidad, por lo tanto comentar un contenido que corresponde con las afinidades propias de un nodo, da la posibilidad de conocer otros nodos con las mismas aficiones.

Mejorar Comentario

Un nodo podría estar interesado en tener solamente comentarios buenos sobre contenidos y no conocer así a gente con aficiones que no comparte, por lo tanto antes que hacer un nuevo comentario, se dedicaría a mejorar los comentarios ya realizados.

Fortalecer Relación

Del mismo modo, podemos estar más interesados en tener buenas relaciones que muchas, si es este nuestro objetivo, y en vez de explorar nuevas relaciones que podrían empeorar nuestra media, conservar las que se tienen y mejorarlas.

Hemos conseguido por lo tanto un Agente Inteligente capaz de aprender y actuar de forma conveniente para llegar a un objetivo, con capacidad de adaptarse a los cambios que pudiera haber en el entorno para seguir prosperando.

5.3.3. Ampliar, mejorar o reemplazar los Agentes Inteligentes

Gracias a tener las distintos áreas de conciencia del Agente separados en diferentes estructuras, es posible cambiar una parte completa sin necesidad de cambiar las demás. Si quisiéramos cambiar los Agentes por completo, o añadir otro tipo de Agente más, bastaría con añadirlo a la Factoría de Agentes siempre y cuando implemente correctamente la `<<interface Agente>>`. No es necesario que los nuevos agentes sigan ninguna estructura predefinida, nosotros hemos elegido la estructura de tres niveles por que cumple con creces con lo esperado de nuestros agentes, dejándonos todavía muchísimas posibilidades en lo que a desarrollo de agentes se refiere, y aunque recomendamos usar esta estructura por lo clara que es, el sistema permitiría cualquier otra estructura.

Tenemos que tener en cuenta el funcionamiento del Controlador de la red social para añadir nuevos agentes inteligentes a la aplicación, es importante saber que la simulación, que es la parte del Control que influye directamente en los nodos, se realiza secuencialmente en cada uno de los nodos, por lo que las llamadas a la simulación de los nodos se repetirá a lo largo de la vida del programa. Podría parecer evidente que esto sea así, pero la simulación podría haberse diseñado de otras muchas formas que nos planteamos durante el diseño del sistema, la simulación podría ser fraccionada, y llamada secuencialmente en distintos ordenes, o una simulación constante; no las elegimos finalmente por que la primera complica en exceso una simulación de cara al usuario, y la segunda, aunque se ajusta más a la realidad, es menos clara cuando queremos analizar la evolución ya que es complicado inducir cambios en una subred.

5.4. Cuestiones de diseño

5.4.1. Introducción

En el diseño del proyecto se han utilizado varios patrones de diseño. Conviene destacar que en ningún momento se ha intentado forzar el diseño para usar patrones cuando no era necesario, sólo se han utilizado para resolver problemas típicos de la programación para los que existía una solución satisfactoria en forma de patrón.

5.4.2. Patrón modelo-vista-controlador

Como esquema general, la arquitectura de la aplicación se basa en el patrón modelo-vista-controlador o MVC. Es decir, está dividida en tres partes diferentes:

- El **modelo**, que especifica cuáles son los datos que usa la aplicación. En este caso, RedSocial, Nodo, Relación, etc.
- La **vista**, una interfaz de usuario para manejar esos datos. En este caso, Interfaz, VentanaHerramientas, etc.
- Y el **controlador**, que sirve de puente entre ambos. En este caso, Controlador, ConectorBD, etc.

Ilustración 24: elementos principales del patrón MVC

5.4.3. Patrón *singleton*

La organización de la aplicación hace que haya cierto número de clases de las que no es preciso tener más de una instancia. Por ejemplo, todos los nodos necesitan acceder a la red social para pedirle cierta información, cada vez que se está realizando una simulación. Para evitar que pueda haber más de una red social, se utiliza el patrón singleton.

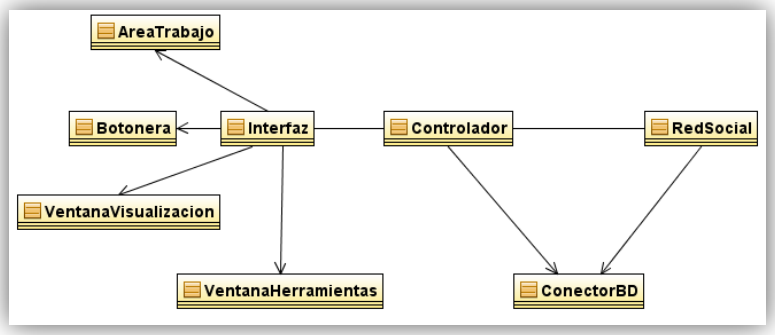

Ilustración 25: elementos que usan el patrón singleton

5.4.4. Patrón fábrica (*factory*)

Así como hay objetos de los que sólo tendremos una instancia, habrá otros que estemos construyendo continuamente. Por ejemplo, los nodos y los contenidos. Cada objeto de estas clases tendrá una serie de características concretas, como un identificador único.

Para facilitar la creación de estos objetos, se ha hecho que sus clases sean "fábricas", de modo que para obtener una instancia basta con pedirle a su clase que la fabrique.

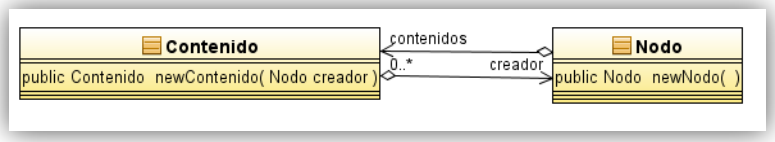

Ilustración 26: elementos que usan el patrón fábrica

5.4.5. Patrón constructor (*builder*)

Mencionábamos que de algunos objetos, como los nodos, crearemos varias instancias. Una de las funcionalidades de la aplicación es crear redes de modo automático, para poder trabajar con ellas en lugar de tener que crear una manualmente.

Para poder hacer estas creaciones, necesitaremos un "constructor", en este caso un objeto de la clase CreacionParametrizada. Este constructor realizará las llamadas necesarias para crear una nueva red.

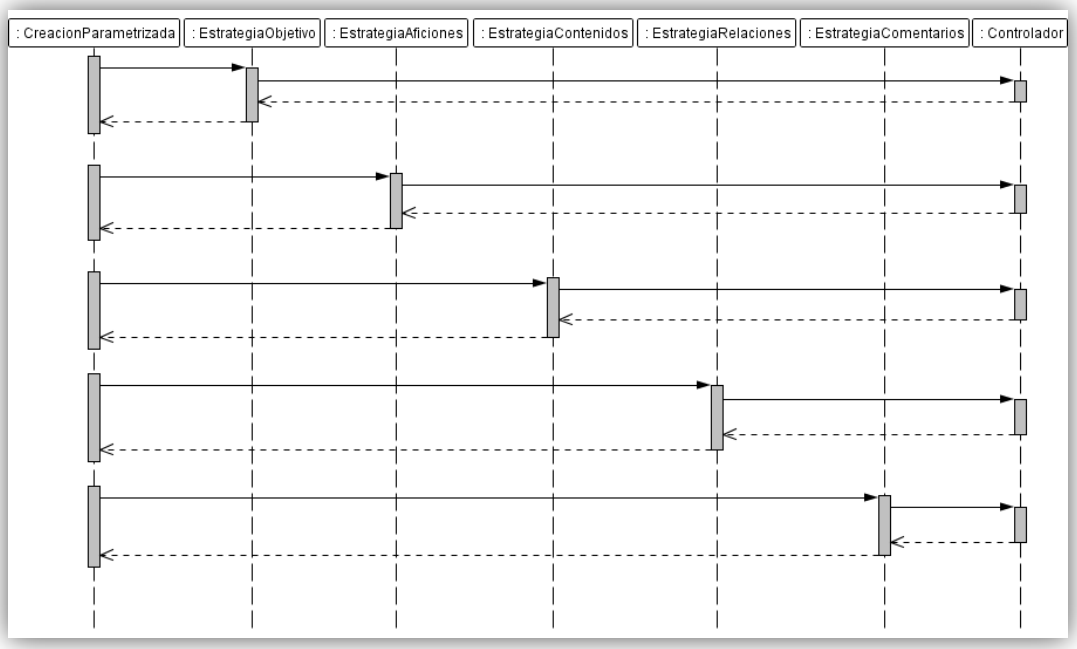

Ilustración 27: llamadas producidas durante la creación de una red

5.4.6. Patrón estrategia

Para aumentar la funcionalidad de CreacionParametrizada puede ser interesante el uso de varias estrategias. Por ejemplo, a la hora de asignar aficiones a los nodos de una red podemos querer que todos los nodos de la red tengan las mismas aficiones, que los nodos tengan aficiones elegidas aleatoriamente a partir de un conjunto dado, etc.

En la implementación actual sólo hay una estrategia para cada posible categoría. No obstante, la estructura de la aplicación permite añadir fácilmente más estrategias.

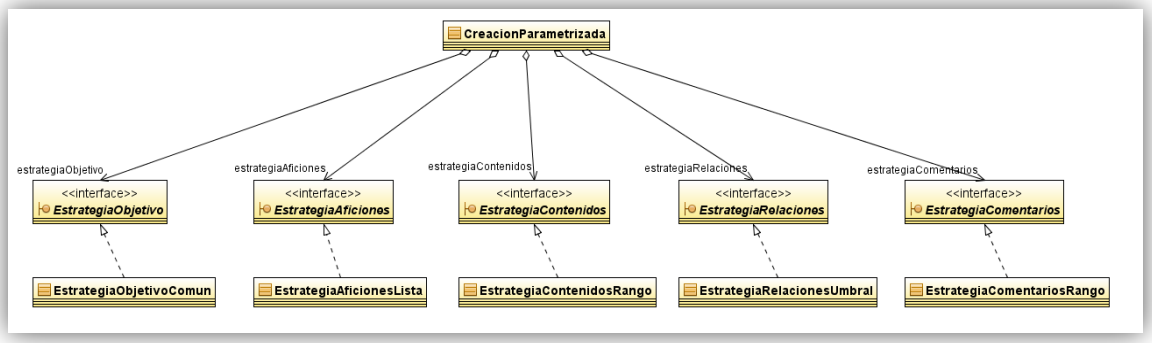

Ilustración 28: estrategias usadas por CreacionParametrizada

5.4.7. Patrón decorador

La parte principal de la interfaz de usuario es el área de trabajo. Allí se dibuja la red y se pueden realizar acciones como añadir nodos, crear relaciones, etc. La funcionalidad de esta área de trabajo se aumenta gracias a una botonera, que se añade a la misma ventana.

De este modo, la botonera actúa como un decorador del área de trabajo, ampliando su funcionalidad sin necesidad de modificar su código.

5.4.8. Patrón cadena de responsabilidad

Como se comentaba al hablar del patrón modelo-vista-controlador, en la aplicación tenemos un modelo de datos, una interfaz y un controlador que se encarga de comunicar ambos. En nuestra aplicación, además del controlador principal tenemos un conector de base de datos, que se encarga de guardar en disco los cambios que se produzcan en el modelo.

Por lo tanto, cuando se produce un evento, se realiza una llamada al controlador, el controlador informa a la red social y la red social, a su vez, puede reenviar la llamada al conector de la base de datos.

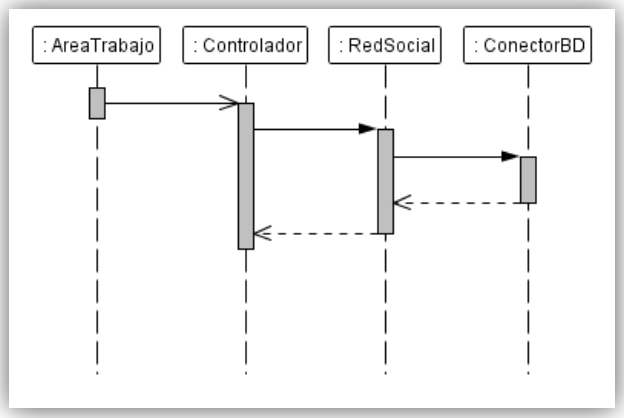

Ilustración 29: propagación de un evento producido en la interfaz

5.5. Cómo hacer modificaciones en el proyecto

5.5.1. Introducción

Como se ha podido ver en el punto anterior, la aplicación sigue un diseño modular, preparado para afrontar posibles ampliaciones. La principal división es la arquitectura de tres niveles: capa de presentación, capa de negocio y capa de datos (11).

A continuación, veremos cómo se pueden modificar cada una de estas capas. Para todo lo relativo a la ampliación de los agentes inteligentes, véase la sección 5.3.3. Para una lista de posibles ampliaciones, véase la sección 6.2.

5.5.2. Capa de presentación

En lo relativo a la capa de presentación, conviene tener en cuenta que la interfaz se encuentra en cierto modo dividida en varias partes.

La principal, la que muestra el grafo de la red, sería la clase AreaTrabajo. Esta clase es un componente que se puede usar directamente como cualquier objeto Swing. Para representar la red, utiliza todas las clases del paquete krowdix.control.util. Los eventos que recibe, tanto de teclado como de ratón, se tratan por el Controlador, que se registra cuando se crea una instancia del área de trabajo.

Otras clases importantes de esta capa son la Botonera (formada por objetos de distintas clases heredadas de Boton), que permite acceder a distintas funciones del Controlador, y la propia clase Interfaz, que une en una misma ventana una Botonera y un AreaTrabajo.

También se tienen distintas ventanas, cada una con una funcionalidad diferente: en lo tocante a la visualización de gráficas de la red, se tiene la clase VentanaHerramientas; para realizar cambios en el modo de representar el área de trabajo se tiene la VentanaVisualizacion y para editar los nodos se tiene la clase VentanitaEdicionNodo.

Dentro de la presentación también se cuenta con dos asistentes (AsistenteNuevo y AsistenteSimulacion), ambos heredan de Asistente y basándose en ellos se pueden crear fácilmente más asistentes.

Otros elemento que encontramos en la presentación es una clase Voronoi junto a otras clases auxiliares dentro del paquete krowdix.interfaz.voronoi, una implementación de código abierto del algoritmo de Fortune para el cálculo de diagramas de Voronoi.

Por último, en el paquete krowdix.interfaz.util se tienen distintos componentes visuales, como un Slider retocado para resultar más agradable, un Spinner diseñado para mostrar porcentajes, un panel con un color degradado de fondo, etc.

5.5.3. Capa de negocio

La capa de negocio especifica las operaciones que podemos realizar sobre los datos. La mayor parte de los elementos de esta capa se encuentran en el objeto Controlador.

Este Controlador se encargará de vigilar que no tengamos más de una relación entre dos nodos dados, que todos los valores de afinidad estén entre 0 y 100, etc. También se encarga de borrar los contenidos y aficiones de un nodo cuando éste deja de participar en una red.

Nótese que esto último podría realizarse utilizando la integridad referencial de la base de datos. Sin embargo, hemos decidido dejarlo aquí porque en algún momento puede ser deseable no borrar esos contenidos (algunas redes reales no los borrarían).

En la capa de negocio también se han incluido las clases de tipo CreadorGrupo. Este tipo es una interfaz que especifica lo que deben implementar los creadores de grupo, y es una de las partes más importantes de esta capa, ya que permite crear redes a partir de ciertos parámetros. Se han incluido dos creadores, completamente documentados, para que resulte sencillo añadir más creadores.

5.5.4. Capa de datos

La parte principal de esta capa es la RedSocial. Está implementada a partir de un grafo no dirigido de JUNG, por lo que si en un determinado momento se decidiese, por ejemplo, que las relaciones deben ser dirigidas, habría que cambiar la clase de la que hereda.

Puede parecer que en este caso hemos incumplido ligeramente la división en las capas de la arquitectura, ya que la capa de datos lleva implícita una regla de negocio: la relación A -> B es la misma que B -> A. En realidad esta comprobación se hace igualmente en el Controlador, y en esta capa básicamente no se utiliza la propiedad de que estamos trabajando con grafos no dirigidos.

Otros elementos de la capa son los nodos, los contenidos y las relaciones. Todos estos elementos heredan de la clase ObjetoRed, que especifica que deben tener un identificador. Además, tanto Nodo como Contenido heredan de Entidad, lo que además de un identificador les da aficiones y relaciones.

Nótese que en este nivel estamos llamando Relacion tanto a las relaciones como a los comentarios. A nivel lógico, se diferencian porque la primera tiene como destino a un nodo y la segunda a un contenido. Además, cuentan con un campo booleano que lo especifica. Aún así, puede ser interesante dividir la clase en dos, para ello habría que tener en cuenta que la clase Entidad debería dejar de tener relaciones, ya que los contenidos guardan los comentarios que han recibido en el vector de relaciones que heredan de Entidad.

5.6. Herramientas y detalles de implementación

5.6.1. Lenguaje de programación

Uno de los objetivos del proyecto era tener una aplicación multiplataforma. Este requisito llevo a pensar, casi de inmediato, en usar el lenguaje Java. Quedaba por decidir qué versión utilizar.

Algunos estudios informales (12) indican que, aunque la versión de Java más usada es Java 6, en muchos ordenadores todavía se sigue utilizando la versión 5.

Tras estudiar las novedades que incluía Java 6 se decidió que ninguna de ellas resultaba imprescindible para el desarrollo de la aplicación. Además, un programa desarrollado para la versión 5 funcionará sin problemas en un ordenador con Java 6 instalado, sin embargo un programa desarrollado para la versión 6 no llegará a abrirse en un ordenador con Java 5. Por tanto, para asegurar la máxima compatibilidad del programa se acordó usar Java 5.

5.6.2. Herramientas CASE

Para plasmar en código las estructuras que íbamos a usar se utilizaron distintos métodos aprendidos de Ingeniería del Software. Uno de estos métodos es el uso de herramientas CASE de ayuda al desarrollo.

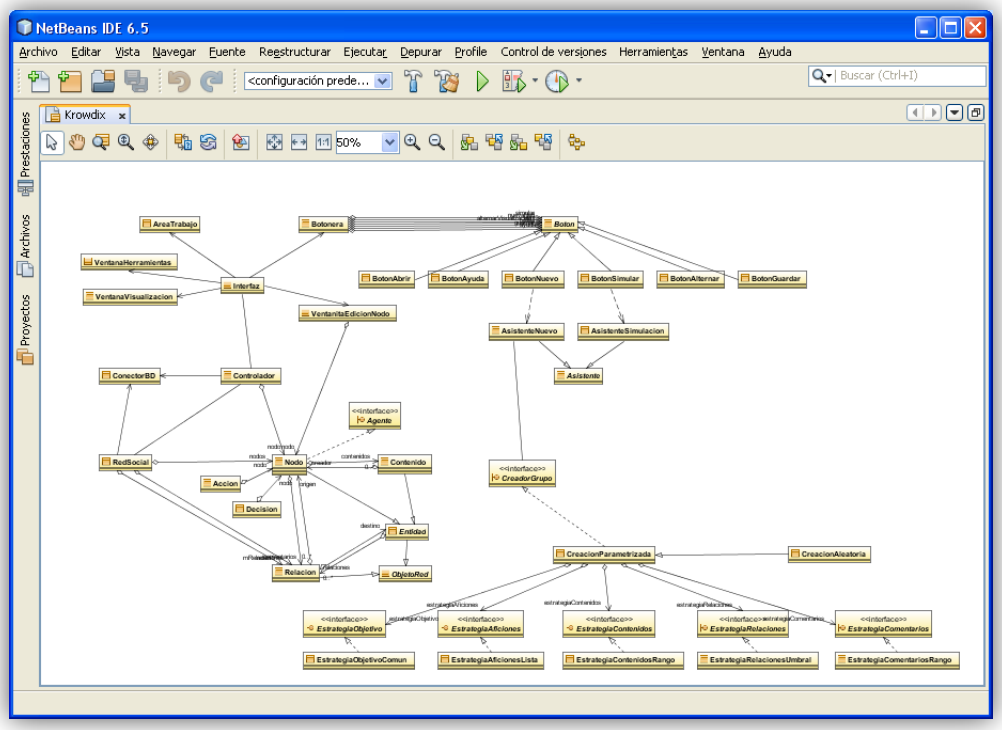

Ilustración 30: editando UML en el entorno de desarrollo NetBeans IDE 6.5

La estructura del programa, su división en distintas clases de objetos y el modo en el que estas clases se comunicaban entre sí se plasmó en un diagrama UML usando el editor de NetBeans IDE 6.5 (13). Una vez perfilado este diagrama, se utilizó el propio NetBeans para generar el esqueleto del código, que posteriormente se completó a mano.

5.6.3. Entorno de desarrollo

Pese a que el UML se diseñó con NetBeans, a la hora de escribir código se cambió el entorno de desarrollo a Eclipse Ganymede (14).

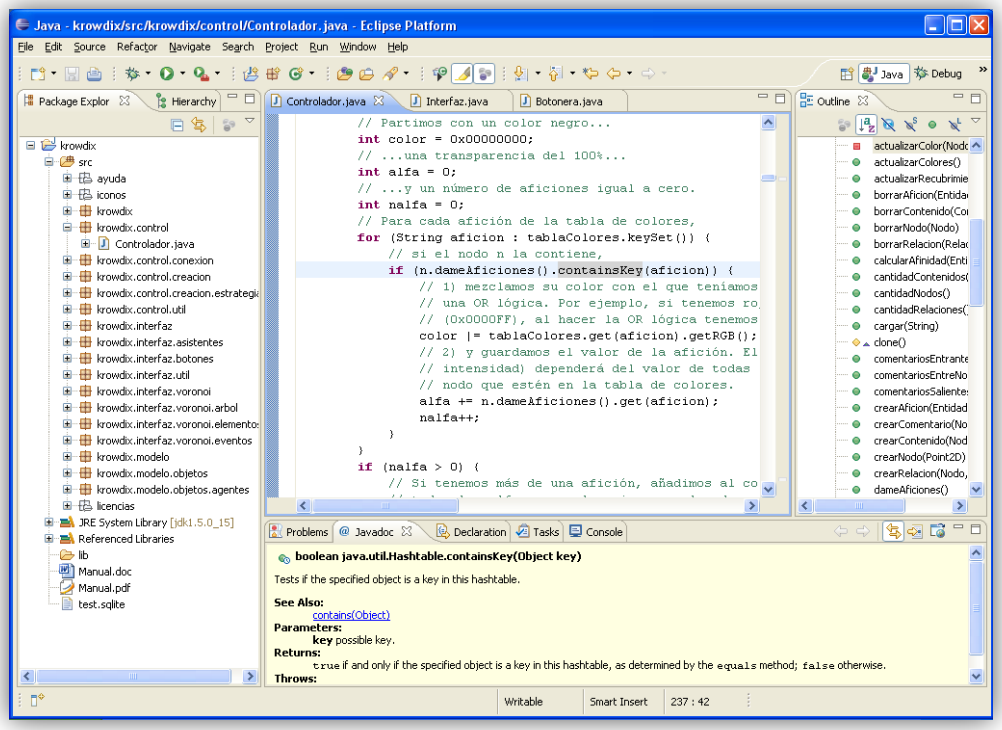

Ilustración 31: entorno de desarrollo Eclipse Ganymede

NetBeans es un IDE especialmente diseñado para el desarrollo de aplicaciones para Internet. Por el contrario, Eclipse es un entorno más orientado al desarrollo de aplicaciones de escritorio. Al ser precisamente un programa de escritorio lo que se iba a desarrollar, se optó por utilizar Eclipse.

5.6.4. Desarrollo de la interfaz

Los elementos principales de la interfaz (botones, diálogos, cuadros de texto) se han creado usando la librería estándar Swing de Java. Esto permite que en cada plataforma los componentes de muestren usando un estilo "nativo", lo que ayuda a la integración del programa en distintos entornos de escritorio.

Algunos componentes, no obstante, se han sobrecargado usando implementaciones propias. Esto nos ha permitido tener, por ejemplo, componentes con degradados de color como fondo.

Ilustración 32: *JSlider* original (izquierda) y modificado (derecha)

Para la creación de la interfaz no se ha hecho uso de ninguna herramienta de generación de interfaces: todo el código correspondiente se ha escrito a mano. Esto permite que se tenga un código más legible y, por tanto, más fácilmente modificable.

En la interfaz también se han usado algunas librerías, que se detallan más adelante. El uso de estas librerías ha permitido varias cosas: encapsular la visualización de la red y la creación de gráficos, así como tener ventanas traslúcidas para un mejor efecto visual.

5.6.5. Base de datos

En los programas de análisis de redes hay una serie de formatos habituales en los que guardar la representación de una red. Nuestra aplicación no podía guardar los datos en ninguno de estos formatos, ya que manejamos una serie de datos adicionales. Era necesario, por tanto, decidir de qué modo se iba a realizar la persistencia.

En principió se descartó XML por el tamaño de archivo que se obtenía para redes de tamaño medio. También se descartó usar la *serialización* de Java, ya que los cambios en el código provocaban que no se pudiera recuperar una red guardada con una versión vieja de la aplicación. Finalmente se decidió que para manejar las redes se utilizarían bases de datos. Dentro de los diferentes tipos de bases de datos se optó por SQLite, ya que no necesitaba la instalación de un servidor por parte del usuario.

El esquema EER de la base de datos es el siguiente:

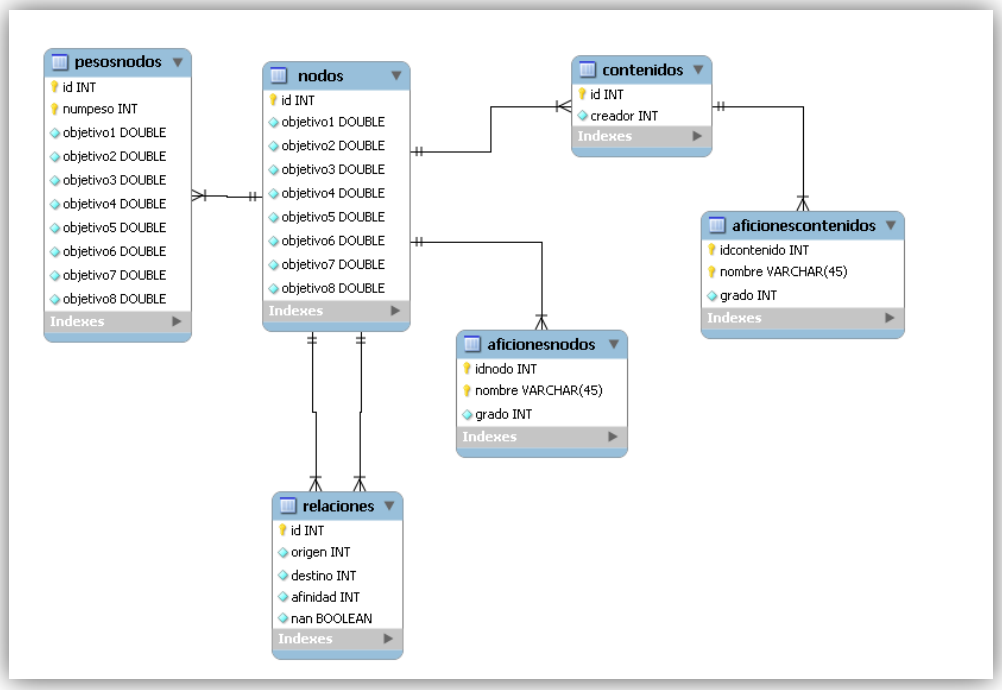

Ilustración 33: esquema EER de la base de datos

Mientras que el modelo relacional correspondiente es el siguiente:

nodos (id, objetivo[8])
pesosnodos (id, numpeso, objetivo[8])
relaciones (id, origen, destino, afinidad, nan)
contenidos (id, creador)
aficionesnodos (idnodo, nombre, grado)
aficionescontenidos (idcontenido, nombre, grado)

5.6.6. Librerías externas usadas

En el desarrollo de la aplicación se usaron varias librerías externas para conseguir funcionalidades adicionales cuya implementación quedaba fuera del ámbito del proyecto. Estas librerías son las siguientes:

- **SQLite JDBC 0.54:** permite el uso de bases de datos con formato SQLite sin necesidad de que el usuario instale software adicional.

- **JUNG 2.0 beta1:** es una librería de representación gráfica. Permite mostrar por pantalla distintos tipos de grafos, dando una gran flexibilidad al programador a la hora de decidir cómo pintar vértices y aristas, así como la opción de elegir distintos modos de reordenado. En la aplicación se usa para representar la red social.
- **JFreeChart 1.0.12:** es una librería para la creación de gráficas. Permite generar distintas gráficas para reflejar una serie de datos. En la aplicación se usa para mostrar gráficas con el estado de la red y su evolución a lo largo del tiempo.
- **JNA 3.0.9:** es una librería para acceder a funciones nativas del sistema operativo. Permite realizar acciones como mostrar notificaciones del sistema, crear ventanas con formas distintas al rectángulo clásico, etc. En la aplicación se usa para conseguir que las ventanas de herramientas tengan cierto grado de transparencia.

6. Conclusiones

6.1. Conclusiones del proyecto

El trabajo realizado a lo largo de los últimos meses ha dado sus frutos y los objetivos se han cumplido.

El proyecto actual ya permite empezar a trabajar en el análisis de redes sociales. Es posible crear redes, ya sea aleatoriamente o dando todos los datos que se quieran. Es posible añadir y eliminar grupos de nodos de las redes. Es posible modificar fácilmente todos los atributos de los nodos (relaciones, contenidos, comentarios). Es posible obtener en todo momento informes y gráficas para analizar la red. Por último, es posible realizar simulaciones en la red gracias al uso de agentes inteligentes.

Hemos pedido a personas ajenas al proyecto su opinión sobre la interfaz para valorar si es lo suficientemente intuitiva. El programa se ha probado en distintos ordenadores y funciona por igual en Windows XP, Ubuntu 9.04 y Mac OS X.

Podemos considerar que el proyecto está listo para que, el próximo curso, un nuevo grupo tome las riendas y termine de desarrollar este simulador de redes sociales.

6.2. Trabajo futuro y ampliaciones

Como se mencionaba en la sección de objetivos, el programa a desarrollar debía estar preparado para realizar ampliaciones. Las modificaciones que a medio plazo se pueden llevar a cabo son:

6.2.1. Degradación de redes

En el estado actual, las redes creadas por la aplicación sólo evolucionan: los usuarios cada vez tienen más relaciones, crean más contenido, invitan a otros usuarios, etc.

Este es el comportamiento de una red social en sus inicios, pero no el de una red madura. Una ampliación posible sería añadir a la aplicación otros comportamientos: enlaces que se rompen, nodos que se van de la red o que simplemente dejan de participar.

6.2.2. Mejoras en la comunicación entre agentes

Los agentes de la aplicación se comunican entre sí para ver cuáles son sus características comunes (aficiones comunes, amigos comunes) y usan esta información para crear relaciones y comentarios.

Esta comunicación podría mejorarse para que los agentes también informaran tanto de su objetivo como de aquello que han aprendido. Esto haría posible que los agentes se ayudaran entre sí a alcanzar antes su objetivo minimizando la distancia global al objetivo en lugar que cada nodo intente minimizar únicamente su propia distancia.

6.2.3. Modo Dios

El modo Dios, o de administrador de la red, serviría para realizar ciertas acciones especiales en la red, como por ejemplo prohibir crear nuevos contenidos. Esto permitiría ver cuáles serían los efectos en los usuarios de una red real provocados por una modificación de los servicios ofrecidos.

6.2.4. Cambios en la conexión con la base de datos

Actualmente se usan archivos temporales para ir guardando las modificaciones hechas sobre una red. El modo en el que estos archivos se manejan hace que no se pueda lanzar más de una instancia de la aplicación, por lo que sería interesante hacer los cambios que fueran necesarios para que se pudieran abrir varias ventanas, cada una con sus propios archivos temporales.

6.2.5. Importar y exportar archivos en otros formatos

Como hemos visto anteriormente, ya existen varios programas para el análisis de redes. Una ampliación muy deseable sería poder importar archivos creados con esos programas para ver la evolución de redes creadas con ellos.

La red ya evolucionada podría exportarse de nuevo al formato original para poder analizarla.

6.2.6. Conexión con redes sociales reales

Al igual que el manejo de archivos en otros formatos, también sería interesante poder conectarse a redes reales para analizarlas.

Esta ampliación requeriría más trabajo, pues se haría necesario crear un robot que recogiera esos datos, pero probablemente el esfuerzo merecería la pena ya que se tendrían datos mucho más reales que los que se pueden obtener analizando redes creadas por ordenador.

La conexión con una red real también serviría para mejorar la inteligencia artificial. Los agentes inteligentes de la aplicación pueden comprobar si la evolución que ellos decidieron se corresponde con lo que sucedió en la realidad

6.2.7. Internacionalización

Aunque muchas acciones se realizan a través de botones con iconos, tanto la interfaz del programa como los manuales de ayuda están escritos en español. Para preparar el uso de la aplicación por gente de otras culturas, sería conveniente añadir otros idiomas.

6.2.8. Nuevas estrategias para la creación de redes

Otra posible ampliación consistiría en añadir nuevas estrategias en el módulo de creación de redes.

6.2.9. Persistencia de la interfaz de usuario

De cara a mejorar la experiencia del usuario, podría ser interesante guardar algunas propiedades de la interfaz como las posiciones de las ventanas de herramientas, los colores escogidos para hacer recubrimientos, etc.

6.2.10. Simulación constante

Una posible mejora para los Agentes Inteligentes sería que la simulación fuera constante, es decir, que una vez lanzada la petición de simular, el nodo según unos parámetros a definir, fuera realizando acciones con más o menos frecuencia. Actualmente conseguimos algo parecido simulando todos los nodos cada cierto tiempo en una simulación continua, pero todos los nodos realizan acciones la misma cantidad de veces, cosa que no es del todo real, ya que hay gente que es más activa que otras independientemente de sus objetivos.

7. Anexos

7.1. Glosario

Este glosario recoge, por orden alfabético, todos los términos técnicos empleados en la memoria.

- **Afición**: cualquier interés o "gusto" que tenga un nodo.
- **Afinidad**: porcentaje de aficiones comunes entre dos nodos.
- **BBS**: *Bulletin Board System*, sistema de tablón de anuncios.
- **Comentario**: conexión entre un nodo y un contenido.
- **Contenido**: cualquier elemento creado por un nodo.
- **IDE**: *Integrated Development Enviroment*, entorno de desarrollo integrado.
- **Métricas**: conjunto de medidas o propiedades de una red.
- **Netiqueta**: reglas de buena conducta en Internet.
- **Nodo**: usuario de una red.
- **Relación**: conexión entre dos nodos de una red.

7.2. Requisitos del sistema

- Procesador Intel Celeron 2.0 GHz o superior.
- 1024 MiB o más de memoria RAM.
- 10 MiB o más de espacio libre en disco duro.
- Tarjeta gráfica con soporte 3D.
- Resolución de pantalla mínima de 1024 x 768 píxeles.
- Sistema operativo Windows XP, Ubuntu 8.10 o superior
- Java 5 (J2RE, SE 1.5) o superior instalado y en el PATH del sistema

7.3. Manual de referencia

- **Lenguaje:** Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_15-b04).
- **Diseño de UML:** NetBeans IDE 6.5.
- **Plataforma de desarrollo:** Eclipse Ganymede 3.4.1.
- **Base de datos:** SQLite con conector JDBC 0.54.
- **Representación gráfica:** JUNG 2 beta1.
- **Gráficas:** JFreeChart 1.0.12.
- **Efectos en ventanas:** JNA 3.0.9.

7.4. Especificación de requisitos de software (noviembre 2008)

7.4.1. Introducción

Nuestro simulador nos permitirá estudiar a fondo las redes sociales en todas aquellas situaciones que seamos capaces de crear, nos facilitará información de redes sociales como unidad, así como de cada individuo que integre dicha red.

En las redes sociales los individuos se relacionan entre sí de muchas formas diferentes, es nuestro objetivo plasmar esas formas diferentes en nuestro simulador para conseguir situaciones lo más reales posible.

7.4.2. Glosario

Introduciremos los términos no comunes que usamos a lo largo de este documento y su significado en el simulador:

- **Red social:** Conjunto de usuarios, las relaciones entre ellos y la información que intercambian.
- **Nodo:** Usuario de la red simulada. Cuenta con varios atributos: sus gustos, la información creada por ese nodo y las relaciones que tiene.
- **Relación:** Unión entre dos nodos. La relación tiene un origen, un destino y un nivel de vinculación. Representa que ambos nodos se consideran amigos.
- **Afinidad:** Es un valor calculado a partir de los atributos de dos nodos. Cuantos más atributos tengan en común estos dos nodos, mayor será la afinidad.
- **Contenido:** Representación virtual de textos, fotos, videos, eventos, música o cualquier tipo de objeto que cada nodo podrá crear para relacionarse con el resto de nodos.

7.4.3. Requisitos específicos

7.4.3.1. Requisitos de edición

- **REQ001**: Creación de una red.- La aplicación nos permitirá crear una nueva red social.
- **REQ002**: Creación de un nodo.- La aplicación nos permitirá añadir un nuevo nodo a la red social.
- **REQ003**: Modificación de un nodo.- La aplicación nos permitirá modificar un Nodo.
- **REQ004**: Creación de una relación.- La aplicación nos permitirá crear relaciones entre nodos
- **REQ005**: Creación de contenidos.- La aplicación nos permitirá crear contenidos virtuales asociados a un nodo, así como opiniones acerca de esos contenidos.
- **REQ006**: Eliminación de un nodo.- La aplicación nos permitirá eliminar los nodos de la red social con todas sus relaciones.
- **REQ007**: Modificación de una relación.- La aplicación nos permitirá redefinir las relaciones existentes.
- **REQ008**: Eliminación de una relación.- La aplicación nos permitirá eliminar relaciones de los nodos.

7.4.3.2. Requisitos de información

- **REQ009**: Cálculo de métricas.- La aplicación nos permitirá calcular diferentes métricas de la red tales como el número de nodos, la cantidad de conexiones de un nodo, la relación entre el número de relaciones y el de relaciones posibles (centralidad), etc.
- **REQ010**: Generación de informes.- El sistema deberá poder generar un resumen detallado de toda la información de la red social que podamos obtener. Esta información también se podrá mostrar gráficamente en la aplicación.

7.4.3.3. Requisitos de simulación de la red

Dado que queremos hacer una simulación automática, es necesario añadir los siguientes requisitos.

- **REQ011**: Generación automática de nodos.- El sistema se encargará de generar un número de nodos determinado, estos podrán estar relacionados con nodos pertenecientes previamente a la red, o bien, empezar sin relación con los nodos existentes.
- **REQ012**: Generación automática de relaciones.- El sistema deberá poder generar relaciones automáticamente entre los nodos miembros de la red social. Estas relaciones se crearán con más probabilidad entre nodos con alto nivel de afinidad.
- **REQ013**: Generación automática de contenidos.- El sistema se encargará de crear contenidos virtuales y asociarlos con un nodo. Asimismo creará comentarios y opiniones acerca de los contenidos de los nodos.
- **REQ014**: Simulación completa.- Uniendo los requisitos REQ011 a REQ013, el sistema debe poder permitir al usuario seleccionar un intervalo de tiempo y realizar automáticamente todas las tareas mencionadas en los requisitos anteriores.

7.4.3.4. Requisitos de guardado

Para facilitar simulaciones de redes que involucren más de una sesión por parte del usuario, se introducen las siguientes funcionalidades:

- **REQ017**: Cargar una red social.- Se deberá poder cargar una red social previamente guardada en disco.
- **REQ018**: Guardar una red social.- Se deberá poder guardar una red social completa, incluyendo todos sus nodos y relaciones, así como sus correspondientes atributos.

7.5. Casos de uso (actualizados junio 2009)

Este documento recoge los casos de uso del Simulador de Redes Sociales **Krowdix**. Los casos de uso aquí detallados han sido redactados intentando que reflejen las funcionalidades de la especificación de requisitos, aunque no siguen ningún orden en particular.

CU-001: Crear una nueva red social

Código:	**CU-001**
Nombre de caso:	Crear una nueva red social
Creado por:	Daniel (19/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	Para que el usuario pueda trabajar con redes sociales, en primer lugar es preciso crear una. El usuario selecciona la opción correspondiente en la interfaz y el sistema crea la red.
Precondiciones:	Ninguna.
Flujo normal:	**1.** El usuario pulsa el botón “nueva red social” en la interfaz. **2.1.** El usuario selecciona “red en blanco”. **2.2.** El usuario selecciona “red aleatoria”. **2.3.** El usuario selecciona “usar asistente”. **2.3.1.** El usuario rellena todos los parámetros del asistente. **3.** Se crea una red social de acuerdo a lo decidido por el usuario.
Flujos alternativos:	Ninguno.
Postcondiciones:	El sistema pasa a tener en la memoria de trabajo una red social con los parámetros asignados.
Actualización caso de uso:	El modo de crear una red cambia respecto a la versión anterior.

CU-002: Guardar red social

Código	**CU-002**
Nombre de caso:	Guardar red social
Creado por:	Daniel (19/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	Puede ser interesante para el usuario guardar toda la información relativa a la red social (es decir, los nodos, las relaciones y las subredes de la red). El usuario selecciona la opción correspondiente en la interfaz y el sistema guarda la red.
Precondiciones:	Hay una red social en la memoria de trabajo.
Flujo normal:	**1.** El usuario selecciona la opción "guardar la red social" en la interfaz. **2.** El sistema pregunta al usuario un nombre para guardar la red. **3.** El usuario introduce un nombre. **4.** El sistema guarda la red social en el archivo especificado.
Flujos alternativos:	**3.1.** Archivo no escribible. **caso no recuperable:** se informa al usuario de que el archivo no pudo ser guardado y del motivo (tamaño en disco insuficiente, medio de sólo lectura) para que pueda corregirlo. **3.2.** Ya existe una red con ese nombre. Se le pregunta al usuario si quiere sobrescribirla. **caso 3.2.1 (recuperable):** el usuario acepta, se sobrescribe el archivo. **caso 3.2.2 (recuperable):** el usuario no acepta, se vuelve a pedir un

	nombre para la red **3.3.** El usuario introduce un nombre en blanco. **caso recuperable:** como nombre se utiliza la fecha actual.
Postcondiciones:	La red social queda almacenada en con el nombre especificado.
Actualización caso de uso:	Nuevos flujos alternativos en el caso de uso.

CU-003: Cargar red social

Código	**CU-003**
Nombre de caso:	Cargar red social
Creado por:	Daniel (19/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	Una vez guardada la información (CU-002), ésta debe poder ser recuperada. El usuario selecciona la opción correspondiente en la interfaz y el sistema carga en memoria una red guardada anteriormente.
Precondiciones:	Ninguna.
Flujo normal:	**1.** El usuario selecciona la opción "cargar una red" en la interfaz. **2.** El sistema muestra al usuario todas las redes almacenadas. **3.** El usuario selecciona una de ellas. **4.** El sistema carga la información de la red en la memoria de trabajo.
Flujos alternativos:	**3.2.** El archivo no contiene información válida o no se puede leer. **caso no recuperable:** se informa al usuario de que el archivo no pudo ser leído y del motivo (archivo inexistente, corrupto, sin permisos de lectura, etc.) para que pueda corregirlo manualmente.
Postcondiciones:	El sistema pasa a tener en memoria la red elegida por el usuario.
Actualización caso de uso:	Cambio en el paso 2 del flujo normal.

CU-004: Crear nodo

Código	**CU-004**
Nombre de caso:	Crear nodo
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	Para simular una red social, necesitaremos usuarios ("nodos") en esa red. El usuario de la aplicación podrá crear esos nodos manualmente.
Precondiciones:	Ninguna.
Flujo normal:	**1.** El usuario pulsa con el botón secundario del ratón en una zona vacía de la pantalla. **2.** Se muestra un menú contextual con distintas opciones. **3.** El usuario selecciona la opción "crear un nodo" en el menú contextual. **4.** El sistema crea un nuevo nodo, sin aficiones, contenidos ni relaciones y con un objetivo aleatorio, y lo sitúa en la posición en la que se desplegó el menú contextual.
Flujos alternativos:	Ninguno.
Postcondiciones:	La red social en memoria pasa a tener el nuevo nodo. La pantalla principal mostrará el nodo con el icono de una persona.

Actualización caso de uso:	Caso de uso totalmente cambiado.

CU-005: Seleccionar nodos

Código	CU-005
Nombre de caso:	Seleccionar nodos
Creado por:	César (23/11/2008)
Actualizado por:	Daniel (28/11/2008), Daniel(18/06/2009)
Actores:	Usuario.
Descripción:	Para obtener información específica o aplicar posteriores acciones es necesario poder seleccionar un nodo concreto de entre los nodos de la red social
Precondiciones:	Hay nodos en la red social.
Flujo normal:	**1.** El usuario selecciona un nodo pinchando sobre él en la pantalla principal. **2.** Los nodos seleccionados se marcan de un color diferente en la interfaz.
Flujos alternativos:	**1.1.** El usuario pincha en un nodo ya seleccionado. **caso no recuperable:** el nodo se deselecciona. **1.2.** El usuario mantiene pulsada la teda Control al seleccionar nodos. **caso recuperable:** se cambia el estado de selección a los nodos que se pinchen **1.3.** El usuario mantiene pulsada la teda Mayúsculas mientras arrastra el ratón. **caso recuperable:** se muestra un rectángulo, cuando el usuario suelte el botón del ratón se seleccionarán todos los nodos que queden dentro. **1.2.** El usuario pulsa a la vez las teclas Control y A. **caso recuperable:** se seleccionan todos los nodos.
Postcondiciones:	El sistema pasa a tener como objetivo de las acciones seleccionadas posteriormente a los nodos seleccionados.
Actualización caso de uso:	Añadidos flujos alternativos.

CU-006: Modificar nodo o relación

Código	CU-006
Nombre de caso:	Modificar nodo o relación
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (28/11/2008), Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario quiere modificar los datos de algún nodo, o bien de alguna relación en la que ese nodo participara. Esto induye añadir, modificar y borrar aficiones, así como modificar y borrar relaciones.
Precondiciones:	Hay nodos en la red social.
Flujo normal:	**1.** El usuario hace doble clic en un nodo o selecciona “editar nodo” en el menú contextual. **2.** Se muestra una ventana con todos los datos del nodo. **3.** El usuario cambia en esta ventana aquellos datos que quiere modificar (sean del nodo o de alguna relación). **4.** El sistema guarda los cambios que ha realizado el usuario.

Flujos alternativos:	Ninguno.
Postcondiciones:	Se modifican el nodo o las relaciones editadas por el usuario.
Actualización caso de uso:	Caso de uso totalmente cambiado.

CU-007: Borrar nodos

Código	CU-007
Nombre de caso:	Borrar nodos
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (28/11/2008), Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario quiere quitar uno o más nodos de la red social. Para ello, selecciona la opción correspondiente en la interfaz.
Precondiciones:	Hay uno o más nodos seleccionados.
Flujo normal:	**1.** El usuario selecciona la opción "borrar nodo" o "borrar todos los nodos seleccionados" en el menú contextual, o pulsa el botón de suprimir en el tedado. **2.** Los nodos se eliminan de la interfaz. **3.** El sistema elimina los nodos seleccionados, las relaciones en las que éstos participaran (como origen o destino), sus contenidos y sus comentarios.
Flujos alternativos:	Ninguno.
Postcondiciones:	Los nodos seleccionados se eliminan de la red social. La pantalla deja de mostrar los nodos.
Actualización caso de uso:	Cambiado modo de borrar nodos.

CU-008: Crear contenidos y comentarios

Código	CU-008
Nombre de caso:	Crear contenidos y comentarios
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	Los nodos de la red social interactúan creando contenidos (representación virtual de textos, fotos, vídeos, eventos, música o cualquier tipo de objeto) y comentarios.
Precondiciones:	Hay nodos en la red social.
Flujo normal:	**1.** El usuario abre la pantalla de edición de nodos (CU-006) **2.** El usuario selecciona la opción "crear nuevo contenido" o "nuevo comentario". **2.1.** Si se elige crear un comentario, el sistema muestra todos los contenidos en los que se puede comentar. **2.2.** El usuario elige un contenido. **3.** Se crea el contenido o el comentario.
Flujos alternativos:	Ninguno.
Postcondiciones:	El nodo pasa a tener un nuevo contenido o comentario.
Actualización caso de uso:	Cambiado modo de generar contenidos y comentarios.

CU-009: Crear relación

Código	CU-009
Nombre de caso:	Crear relación
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (26/11/2008), Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	La simulación de una red implica que los nodos que en ella participan están conectados; a esas conexiones las llamamos "relaciones". El usuario de la aplicación puede crear estas relaciones a partir de la interfaz.
Precondiciones:	Hay dos o más nodos en la red social.
Flujo normal:	**1.** El usuario pulsa el ratón sobre un nodo no seleccionado. **2.** El usuario arrastra el ratón hacia otro nodo. **3.** El usuario suelta el ratón. **4.** El sistema crea una relación entre ambos nodos, calculando automáticamente la afinidad entre ambos.
Flujos alternativos:	**3.1.** Ya existe una relación entre ambos nodos, o bien el nodo destino es el mismo que el origen. **caso no recuperable:** no se crea ninguna relación.
Postcondiciones:	La red social pasa a tener la nueva relación. La pantalla principal mostrará la relación como una arco entre los dos nodos, su color dependerá de la afinidad entre sus extremos.
Actualización caso de uso:	Cambiado modo de trazar relaciones.

CU-010: Eliminar relación

Código	CU-010
Nombre de caso:	Eliminar relación
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (26/11/2008), Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario quiere eliminar una relación en la que participe un nodo.
Precondiciones:	Hay al menos una relación en la red.
Flujo normal:	Ver CU-006.
Flujos alternativos:	Ninguno.
Postcondiciones:	La relación seleccionada se elimina de la red social. La pantalla deja de mostrar la relación.
Actualización caso de uso:	Caso de uso movido.

CU-011: Visualizar subredes

Código	CU-011
Nombre de caso:	Visualizar subredes
Creado por:	Daniel (18/06/2009)
Actualizado por:	
Actores:	Usuario.
Descripción:	Entendemos por subredes grupos de nodos con una afición común. Para ver más fácilmente estas subredes se pueden dibujar recubrimientos sobre el mapa de la red.
Precondiciones:	Hay nodos con aficiones en la red.

Flujo normal:	**1.** El usuario elige en la ventana de visualización avanzada un color para resaltar los nodos que tengan una cierta afición. **2.** El mapa de la red se pinta con el nuevo color como fondo para los nodos que tengan esa afición.
Flujos alternativos:	**1.1.** La ventana de visualización avanzada no está visible. **caso recuperable:** el usuario pulsa el botón "seleccionar ventanas" y a continuación "mostrar vis. avanzada" para mostrar la ventana. **1.2.** Hay nodos con esa afición que ya tenían un color asignado **caso recuperable:** esos nodos se pintan con una mezcla de ambos colores.
Postcondiciones:	Se dibuja un recubrimiento en el mapa.

CU-012: Poner afición a un nodo

Código	CU-012
Nombre de caso:	Poner afición a un nodo
Creado por:	Daniel (28/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario quiere crear o modificar una subred. Para ello, asigna una afición a un nodo.
Precondiciones:	Hay al menos un nodo en la red.
Flujo normal:	Ver CU-006
Flujos alternativos:	Ninguno.
Postcondiciones:	El nodo pasa a tener la afición escogida.
Actualización caso de uso:	Caso de uso movido.

CU-013: Quitar afición a un nodo

Código	CU-013
Nombre de caso:	Quitar afición a un nodo
Creado por:	Daniel (28/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario quiere que un nodo deje de pertenecer a una subred en concreto. Para ello, le quita la afición correspondiente.
Precondiciones:	Hay al menos un nodo con una afición en la red.
Flujo normal:	Ver CU-006.
Flujos alternativos:	Ninguno.
Postcondiciones:	El nodo deja de tener la afición.
Actualización caso de uso:	Caso de uso movido.

CU-014: Crear informe

Código	CU-014
Nombre de caso:	Crear informe
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario de la aplicación querrá conocer estadísticas de la red. Para ello, selecciona la opción correspondiente en la interfaz
Precondiciones:	Hay una red social en memoria.

Flujo normal:	**1.** El usuario selecciona la opción “crear informe”. **2.** El sistema genera un informe con toda la información de la red social. Este informe incluirá: número de nodos de la red, número de relaciones, número de subredes, promedio de relaciones y grado de la red. El informe también incluirá métricas de la red, como relación entre nº de relaciones y relaciones posibles (centralidad), grado de clustering de la red e índice de cohesión entre grupos.
Flujos alternativos:	Ninguno.
Postcondiciones:	El sistema genera un informe sobre la red social.

CU-015: Realizar simulación

Código	CU-015
Nombre de caso:	Realizar simulación
Creado por:	Daniel (24/11/2008)
Actualizado por:	Daniel (18/06/2009)
Actores:	Usuario, agentes inteligentes.
Descripción:	Para analizar redes sociales con más comodidad, el usuario podrá automatizar las tareas especificadas en CU-004, CU-006, CU-007, CU-008, CU-009, CU-010, CU-012 y CU-013, y usar agentes inteligentes para que las realice él.
Precondiciones:	Ninguna.
Flujo normal:	**1.** El usuario selecciona la opción “realizar simulación”. **2.** El sistema muestra al usuario los diferentes tipos de simulación que se pueden hacer. **3.** El usuario selecciona una de ellas. **4.** El sistema, con la ayuda de los agentes inteligentes, realiza la simulación.
Flujos alternativos:	Ninguno.
Postcondiciones:	Se realiza la simulación que seleccione el usuario.
Actualización caso de uso:	Cambios menores.

CU-016: Desplazar selección

Código	CU-016
Nombre de caso:	Desplazar selección
Creado por:	César (23/11/2008)
Actualizado por:	Daniel (28/11/2008), Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	Para hacer más clara la visualización de la red, el usuario podrá mover los nodos que tenga seleccionados.
Precondiciones:	Hay uno o más nodos seleccionados.
Flujo normal:	**1.** El usuario arrastra la selección a otra zona de la pantalla. **2.** Se establece la selección en su nuevo emplazamiento.
Flujos alternativos:	Ninguno.
Postcondiciones:	La interfaz se actualiza, reflejando los cambios de posición de los elementos seleccionados.
Actualización caso de uso:	Eliminado flujo alternativo.

CU-017: Ver más información de la red

Código	CU-017
Nombre de caso:	Ver más información de la red
Creado por:	Mario (10/3/2009)
Actualizado por:	Mario (20/5/2009), Daniel (18/06/2009)
Actores:	Usuario.
Descripción:	El usuario puede querer ver más información sobre la red. Para ello se dispone de una ventana con 3 gráficas: 1. Refleja el estado actual de la red. Es una gráfica con varios ejes que tienen la información de diferentes métricas. 2. Refleja el crecimiento de la red a lo largo de la simulación. Es una gráfica de 2 ejes con tiempo y valor en el tiempo de las mismas métricas que se reflejan en la gráfica 1. 3. Refleja los cambios de las métricas a los largo del tiempo, similar a la gráfica 2.
Precondiciones:	Ninguna.
Flujo normal:	**1.** El usuario pulsa el botón "seleccionar ventanas". **2.** El usuario selecciona "mostrar info. red". **3.** Se muestra la ventana antes descrita.
Flujos alternativos:	Ninguno.
Postcondiciones:	La ventana con gráficos pasa a estar visible, actualizándose cada vez que se produzca un cambio en la red.
Actualización caso de uso:	Caso de uso anteriormente nombrado "Lateral GUI", reescrito para mostrar su flujo.

CU-018: Elegir visualizaciones alternativas de la red

Código	CU-018
Nombre de caso:	Elegir visualizaciones alternativas de la red
Creado por:	Daniel (18/06/2009)
Actualizado por:	
Actores:	Usuario.
Descripción:	El usuario puede querer ver la red de diferentes maneras. Una visualización alternativa permitirá dibujar los nodos de modo que su tamaño sea proporcional al número de comentarios que han recibido; también se pueden dibujar las relaciones de modo que su grosor dependa del número de comentarios intercambiados.
Precondiciones:	Hay nodos en la red.
Flujo normal:	**1.** El usuario elige en la ventana de visualización avanzada una visualización diferente para los nodos o las relaciones. **2.** El mapa de la red se actualiza, usando la nueva visualización.
Flujos alternativos:	**1.1.** La ventana de visualización avanzada no está visible. **caso recuperable:** el usuario pulsa el botón "seleccionar ventanas" y a continuación "mostrar vis. avanzada" para mostrar la ventana.
Postcondiciones:	Se utiliza una visualización alternativa para mostrar la red.

CU-019: Ampliar el mapa de la red

Código	CU-019
Nombre de caso:	Ampliar el mapa de la red
Creado por:	Daniel (18/06/2009)
Actualizado por:	

Actores:	Usuario.
Descripción:	Para trabajar con mayor comodidad, el usuario puede querer acercar o alejar la imagen. Para ello utilizará las características de zoom.
Precondiciones:	Hay nodos con aficiones en la red.
Flujo normal:	**1.** El usuario pulsa, en la ventana de visualización avanzada, el botón "acercar", "alejar" o "ver todo". **2.** La ampliación del mapa de la red cambia.
Flujos alternativos:	**1.1.** La ventana de visualización avanzada no está visible. **caso 1.1.1 (recuperable):** el usuario pulsa el botón "seleccionar ventanas" y a continuación "mostrar vis. avanzada" para mostrar la ventana. **caso 1.2.1 (recuperable):** el usuario también puede cambiar la ampliación en el menú contextual o usando la rueda del ratón.
Postcondiciones:	Se cambia la ampliación con la que se dibuja el mapa.

7.6. Manual de usuario

7.6.1. Operaciones básicas de nodos y relaciones

- Para añadir un nodo, pulsa con el botón derecho en el sitio en el que quieres ponerlo y selecciona la opción "Crear nodo aquí". Se creará un nodo con un objetivo aleatorio y sin relaciones. Los nodos se muestran con un dibujo que representa a una persona.
- Para seleccionar un nodo, pulsa sobre él. Si mantienes pulsado Control, puedes modificar la selección añadiendo o quitando nodos. Para seleccionar varios nodos, pulsa en una zona vacía mientras presionas Mayúsculas (Shift) y arrastra el ratón; se irá dibujando un rectángulo para indicar los nodos que se añadirán a la selección.
- Para editar un nodo, haz doble clic sobre él.
- Para crear una relación entre dos nodos, pulsa sobre uno nodo y arrastra hacia el otro. Mientras arrastras el ratón, se irá mostrando una línea para ayudarte. Cuando hayas llegado al otro nodo, suelta el botón del ratón y la relación se creará con el valor de afinidad calculado entre los dos nodos. Las relaciones se muestran como líneas entre dos nodos; su color depende de su afinidad, siendo rojo para afinidades bajas y verde para afinidades altas.
- Algunas operaciones sobre los nodos se pueden realizar a través del menú contextual. Para acceder a él, pulsa con el botón secundario sobre un nodo.

7.6.2. Simulaciones

Hay tres tipos diferentes de simulación: de un paso, continua y usando un asistente.

- La simulación de un paso realiza una simulación y devuelve el control al usuario.
- La simulación continua realiza simulaciones de un paso a intervalos regulares.
- La simulación usando un asistente permite elegir algunas características de la simulación. Por ejemplo, puede realizar simulaciones continuas más rápidas o simular los nodos que tengan una afición en concreto.

7.6.3. Crear una red social nueva

Al pulsar el botón de nueva red social, tendremos 3 opciones:

7.6.3.1. Red en blanco

Crea una red vacía que nos permite ir creando la red y los grupos de la red a nuestro antojo.

7.6.3.2. Red aleatoria

Genera una red aleatoria, tanto en número de usuarios, contenidos, grupos, etc. Como en aficiones y objetivos de los usuarios.

7.6.3.3. Usar asistente

El asistente nos ayudará a lo largo de la creación de una red compleja completamente configurada. Los pasos del asistente son:

- Seleccionamos el número de nodos que entrarán en el nuevo grupo.
- Introducimos en cada linea las aficiones que queremos que tengan los nodos del grupo.
- Seleccionamos el valor mínimo de gustos comunes que deben tener los nodos para crear relaciones. También los contenidos y comentarios que tendrá cada nodo.
- Seleccionamos gráficamente los objetivos que seguirán los nodos en su evolución. Si queremos continuar creando grupos, marcamos la opción "Crear otro grupo".

7.6.4. Cargar y guardar redes sociales

La aplicación permite cargar y guardar redes sociales directamente, sólo escribiendo el nombre que le queremos dar la a red.

Estos archivos quedan almacenados como bases de datos SQLite en la carpeta donde se encuentra el archivo jar que estamos ejecutando, de modo que se pueden hacer respaldos de las bases de datos, o salvarlas para posterior análisis.

7.6.5. Opciones de visualización

La aplicación permite modificar la visualización mediante un panel flotante que permite cambiar el zoom, el recubrimiento de los usuarios en función de sus diferentes aficiones y otras opciones que ayudan a entender mejor la red. Esta barra se puede desactivar y reactivar en el menú de barras de herramientas integrado en la aplicación.

7.6.6. Barra de métricas

Esta barra flotante permite observar la evolución de las métricas y contenidos de la red social conforme evoluciona. Además permite obtener un informe del estado actual. Esta barra se puede desactivar y reactivar en el menú de barras de herramientas integrado en la aplicación.

7.6.7. Ayuda interactiva

Además de este manual, Krowdix tiene ayuda integrada en cualquier punto donde puedan surgir dudas, haz click en un interrogante y aparecerá una ayuda en referencia a la sección en la que te encuentras.

7.7. Información de copyright

7.7.1. Krowdix, simulador de redes sociales

7.7.2. Iconos de Tango Icon Theme

7.7.3. JUNG

7.7.4. JFreeChart

7.7.5. JNA

7.8. Contenido del CD

El CD entregado junto a esta memoria contiene:

7.8.1. Versión electrónica de la memoria

Este mismo documento, en versión PDF. Se encuentra en el directorio raíz del CD.

7.8.2. Versión ejecutable de la aplicación

En formato JAR de Java. Se encuentra en el directorio “ejecutable”, junto con una red de ejemplo y el manual de usuario.

7.8.3. Código fuente

En lenguaje Java. Se encuentra en el directorio “fuente”. El código se encuentra documentado e incluye el proyecto de Eclipse, todas las librerías necesarias y el Javadoc de las clases.

7.8.4. Prototipos y versiones antiguas

Como referencia, se han incluido los primeros prototipos de la aplicación. Se encuentran en el directorio “historia”.

Bibliografía

1. **Contribuidores de Wikipedia.** Correo electrónico - Wikipedia, la enciclopedia libre. *Wikipedia.* [En línea] http://es.wikipedia.org/wiki/Correo_electrónico.

2. —. Hotmail - Wikipedia, la enciclopedia libre. *Wikipedia.* [En línea] http://es.wikipedia.org/wiki/Hotmail.

3. —. Bulletin Board System - Wikipedia, la enciclopedia libre. *Wikipedia.* [En línea] http://es.wikipedia.org/wiki/Bulletin_Board_System.

4. *Noticias Cuatro.* s.l. : Cuatro, 17 de junio de 2009.

5. **Krackhardt, David, Blythe, Jim y McGrath, Cathleen.** KrackPlot 3.0: An Impoved Network Drawing Program. *Connections, Vol. 17(2):53-55.* Diciembre de 1994.

6. **Borgatti, S.P.** NetDraw: Graph Visualization Software. Harvard: Analytic Technologies. 2002.

7. **Kalamaras, Dimitris V.** Social Network Visualiser (SocNetV). 2009.

8. **Contribuidores de Wikipedia.** Agente inteligente (Inteligencia Artificial) - Wikipedia, la enciclopedia libre. *Wikipedia.* [En línea] http://es.wikipedia.org/wiki/Agente_inteligente_(Inteligencia_Artificial).

9. **Nilson, Nils J.** *IA, una nueva síntesis.*

10. **Palma, J. y Marin, T.** *IA: Técnicas, métodos y aplicaciones.*

11. **Contribuidores de Wikipedia.** Programación por capas - Wikipedia, la enciclopedia libre. *Wikipedia.* [En línea] 2009. http://es.wikipedia.org/wiki/Programación_por_capas.

12. **Varios autores.** Tracking Java Versions using Google. *java.net Forums.* [En línea] http://forums.java.net/jive/message.jspa?messageID=315629#315629.

13. **netbeans.org.** NetBeans IDE 6.5. *NetBeans IDE.* 2008.

14. **Eclipse contributors and others.** Eclipse Ganymede 3.4.1. *Eclipse Platform.* 2000, 2008.

Licencia de este documento

Autorización

Autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

En Madrid, a 26 de junio de 2009

Daniel Alonso Fernández Mario Arnaldos Navarro César Montenegro Portillo