Sistemas Informáticos

Curso 2011 – 2012

Proyecto Krowface:

Interfaz de simulación de redes sociales

Álvaro Alonso Moreno
Octavio Miguel González Hernández
Iván Lavera Ulloa

Facultad de Informática
Universidad Complutense de Madrid

Krowface

Interfaz de simulación de redes sociales

Proyecto de Sistemas Informáticos
Facultad de Informática

Universidad Complutense de Madrid

Autores:
Álvaro Alonso Moreno
Octavio Miguel González Hernández
Iván Lavera Ulloa

Profesor director:
Juan Pavón Mestras

Profesor codirector:
Diego Blanco Moreno

Curso 2011 / 2012

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 7 de junio de 2012

Álvaro Alonso Moreno Octavio M. González Hernández Iván Lavera Ulloa

Álvaro Alonso Moreno
Octavio M. González Hdez
Iván Lavera Ulloa

Resumen

Krowface es una herramienta de configuración y simulación de redes sociales online. Permite diseñar, visualizar y analizar la evolución de diferentes redes sociales en tiempo real mediante una interfaz web sencilla y amigable al usuario.

Abstract

Krowface is a simulation tool for online social networks. It's possible to design, visualize and analyze different social network in real time with an intuitive user-friendly web interface.

Palabras claves:

Krowface
Krowdix
Red Social
Simulador
Evolución
Twitter
Facebook
Plugin
Interfaz web
Análisis de Redes Sociales (SNA)

Keywords:

Krowface
Krowdix
Social Network
Simulator
Evolution
Twitter
Facebook
Plugin
Web interface
Social Network Analisys (SNA)

INDICE DE CONTENIDOS

1.- INTRODUCCIÓN

1.1.- SOBRE KROWFACE

Krowface es una evolución del software de simulación y análisis de redes sociales de Krowdix. Sobre este motor se ha incorporado una completa interfaz web, y se ha mejorado el diseño de la arquitectura y el motor de ejecución para permitir simulaciones concurrentes bajo diferentes parametrizaciones.

El principal objetivo del nuevo diseño ha sido crear una interfaz de simulación de redes sociales, que permitiera al usuario de manera fácil y amigable acceder a Krowdix, y explotar todas sus posibilidades. Así como mejorar el antiguo motor de ejecución y simulación, para que también se ejecute desde una interfaz web. Se ha incluido la utilización de Plugins para la ejecución de las acciones y visualizaciones.

Una red social se compone de los usuarios que la forman y de las relaciones entre ellos. Podemos llegar a la conclusión de que estos dos elementos son la estructura que va a sostener una red social, ya que ésta no podría existir sin usuarios así como sin una serie de interconexiones entre ellos que produzca un intercambio de información. A esta información le podemos llamar “contenidos”. En una red social los usuarios están continuamente mostrando a sus contactos una serie de contenidos, ya sean mensajes, fotos, comentarios en contenidos de otros usuarios. Además las relaciones entre usuarios van a hacer formase grupos en la red social. Estos grupos suelen ser más o menos homogéneos en cuando a las aficiones de los usuarios.

Nuestro sistema se divide en dos partes claramente diferenciadas, la parte de configuración, y la parte de ejecución del sistema. La configuración de hace directamente sobre la interfaz web, en el menú de su mismo nombre.
Los principales elementos del modelo de configuración son:

1. Acciones
2. Atributos
3. Tipos de contenido
4. Tipos de grupo
5. Tipos de relaciones

6. Tipos de usuarios de la red social (SNU)
7. Perfiles estáticos
8. Perfiles dinámicos
9. Vistas

Veremos más adelante en qué consisten estos elementos, y como se relacionan entre ellos.

Por otro lado, en la parte de ejecución y simulación del sistema, también ejecutada desde la interfaz web en el menú de Simulación, podemos crear nuevas redes sociales, con unos perfiles determinados, y simular la red social para ver como evoluciona a lo largo del tiempo, según la vista elegida. Todo eso se ejecuta sobre el núcleo java de la aplicación.
Los principales elementos del modelo de ejecución son:

1. El tiempo
2. Contenidos
3. Grupos
4. Relaciones
5. Social Network User (SNU)
6. Atributos

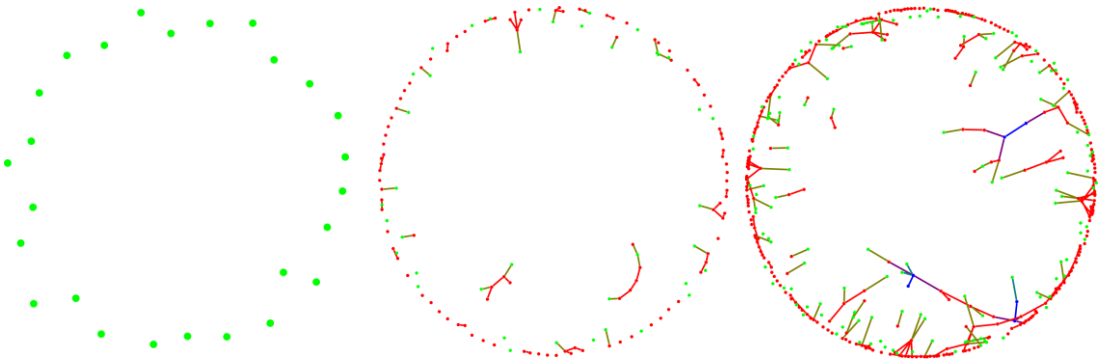
Acabamos de describir los elementos que forman el modelo de nuestra red social. Ahora vamos a hablar de los componentes que harán evolucionar nuestra red.

La red puede evolucionar en muchos sentidos, según como hayamos configurado las acciones, atributos, perfiles… Normalmente la red evolucionará haciendo crecer el número de usuarios, estableciendo nuevas relaciones entre los usuarios, modificando atributos sobre los usuarios, etc.

En nuestras redes sociales todo gira principalmente alrededor del tiempo, los perfiles, las acciones y los atributos.

En el sistema, hay dos tipos de perfiles, por un lado están los perfiles estáticos, que están formados por un nombre, descripción y las acciones asociadas al perfil. Estás acciones están activas en el perfil con un porcentaje, y tienen asociadas unos puntos de acción. En el caso de que los porcentajes de las acciones no lleguen al 100%, se rellenará con una acción por defecto de no hacer nada (NO_ACTION) hasta llegar al total del porcentaje. También existen perfiles dinámicos, que están compuestos por perfiles estáticos, y un atributo, que es el encargado de decidir cuál de los perfiles estáticos se ejecuta en cada momento del tiempo. Según si el atributo es de tipo numérico, lista o String, la distribución de los perfiles se hará de una forma diferente. Teniendo siempre que elegir, un perfil por defecto, para el caso de que el atributo no puede elegir un perfil a ejecutar.

Las acciones, son las que hacen evolucionar el sistema de una forma u otra, pues son las que componen los perfiles, que a su vez componen la red social. Según lo explicado en el párrafo anterior de la ponderación de las acciones en los perfiles, las acciones que tienen mayor peso en el perfil se ejecutarán más veces que las demás. Por ejemplo, si una red social tiene muchos perfiles que tienen la acción de "*crear relación de amistad*", la red evolucionará rápidamente, haciendo que los usuarios tengas cada vez más amigos.



Los atributos, pueden ser de tipo numérico, lista o String.
Están presentes en los perfiles dinámicos, en los tipos de contenidos, tipos de relaciones, tipos de grupos, y tipos de SNU, caracterizando dichos tipos.

Por último, la configuración de los parámetros de simulación: se configura todo desde la interfaz. Cuando creamos una red social nueva, hay que configurar el número de usuarios iniciales, y establecer los perfiles que se van a usar en la simulación. De esta forma podemos modelar una red de usuarios con perfiles posteador y amistoso, y que el 50% de los usuarios tenga un perfil posteador y los demás un perfil amistoso.

Nuestro sistema nos permite hacer la simulación paso a paso (unidad de tiempo), o hacer varios pasos a la vez, y tiene una serie de vistas de los datos generados en esta simulación. Las diferentes vistas van mostrando los cambios realizados en la red para cada instante de tiempo, pudiendo diferenciar fácilmente la evolución de la red en una línea temporal.

1.2.- SOBRE ESTE DOCUMENTO

En este documento vamos a ilustrar tanto el diseño de la aplicación, y la interfaz como el manejo de la misma por parte del usuario final. El propósito es facilitar la comprensión del trabajo realizado a toda aquella persona que esté interesada en conocerlo o ampliarlo.

Hemos querido dejar claros los requisitos del sistema con los que hemos trabajado, de forma que sea fácilmente comprobable si se han cumplido. También se trata de mostrar el diseño y arquitectura del sistema de una manera clara, para dar a conocer los paquetes, las clases y los métodos más importantes. Así como la administración de los datos persistentes. También explicar un poco las librerías más destacadas que utilizamos en el sistema.

También hemos querido plasmar un caso de estudio completo, basado en Twitter, para que se puedan comprobar las capacidades del sistema. Así como un completo manual de usuario, aunque en el *SourceForge,* se podrán encontrar completos "Manuales de desarrollo", así como "Manuales de Plugin". E incluidos en la web, todo el menú de configuración está complementado con "Manuales de uso web", para ayudar al usuario a entender cada uno de los submenús de configuración.

Todo lo mostrado en este documento se puede obtener en la plataforma *SourceForge* en la página http://sourceforge.net/projects/krowdix/, en la carpeta de Krowdix 3.0. De tal forma que quien esté interesado en él, puede bajar el código fuente completo. Si se desea contribuir al desarrollo de Krowface, se puede unir al equipo de desarrolladores, permitiendo el acceso al repositorio SVN, explicaremos más adelante los pasos a seguir para el montaje del entorno del proyecto. Se puede utilizar el sistema, o ampliarlo ya que está disponible bajo licencia GPL.

2.- ESTADO ACTUAL DE LOS SIMULADORES DE REDES SOCIALES

El análisis de redes sociales, Social Network Analisys (SNA) se remonta al menos en la década de 1940 (posiblemente a fines del siglo 19), y el análisis de redes sociales (ARS) como un campo formal ha sido bien establecido en la sociología. El análisis de redes sociales prevé descriptores formales, y por medio de medidas cuantitativas permite realizar pruebas de modelos estadísticos sobre las relaciones y estructuras.

El análisis de redes sociales se originó como un medio para la aplicación de la computación científica para el análisis sociológico cuantitativo, y sigue siendo de gran importancia allí. Se ve un uso intensivo en los estudios de modelos de negocio y estrategias, pero también se aplica intensamente a asuntos militares, y los estudios criminológicos antiterroristas. También ve el servicio a través de una notable variedad de campos, demasiado numerosa para resumir, incluso, incluyendo las ciudades de origen humano / interacción de los animales, como la agricultura y, junto con otros métodos científicos basados en la informática, tales como el modelado basado en agentes y el análisis fractal, el bienestar animal.

Actualmente existen una serie de programas de análisis de redes social tanto de tipo comercial, freeware, y de código libre o abierto. En general el software SNA posee una serie de características comunes.

1. Introducción de datos y manipulación de datos.
2. Técnicas de visualización
3. Rutinas de análisis de redes sociales, divididos en tres tipos de métodos:
    a. Métodos descriptivos para calcular (simple) estadísticas de la red (por ejemplo, la centralidad).
    b. Análisis basados en procedimientos basados en más algoritmos complejos (iterativos) por ejemplo, análisis de clusters.
    c. Modelos estadísticos basados en las distribuciones de probabilidad.

Todos estos programas ofrecen una serie de métricas sobre las redes. Estas métricas se extraen de una serie fórmulas y metodologías que otorga el análisis de redes sociales.

Usando software de análisis de redes se pueden obtener respuesta a preguntas como:

- ¿En qué medida está conectada una entidad a una red?
- ¿Cuál es la importancia real de una entidad en una red?
- ¿Cómo de central es una entidad en una red?
- ¿Cómo fluye la información dentro de una red?

Los programas de análisis de redes sociales, tratan una red como un conjunto de nodos y un conjunto de aristas que relacionan estos nodos. Además usan funciones de la teoría de grafos para extraer la información antes descrita de la red. Además de todas estas métricas usan diferentes vistas gráficas de la red social, representada como un grafo y utilizando distintas maneras de disponer todos estos nodos y sus aristas, como por ejemplo circularmente, jerárquicamente, etc.

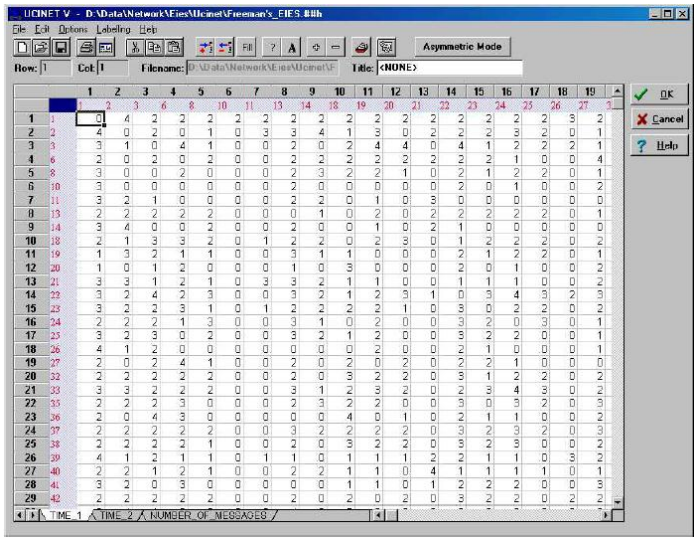
Krowface como software de análisis de redes sociales debe satisfacer todas estas características pero además incluye la potencia del modelado y simulación de las redes sociales. En nuestro sistema podemos modelar la "forma" de una red social mediante la asignación de perfiles a cada entidad y hacerla evolucionar mediante la simulación para posteriormente pasar al análisis de la red creada.

Vamos a ver tres programas actuales de análisis de red sociales, UCInet, Pajek y SocNetV.

2.1.- UCInet

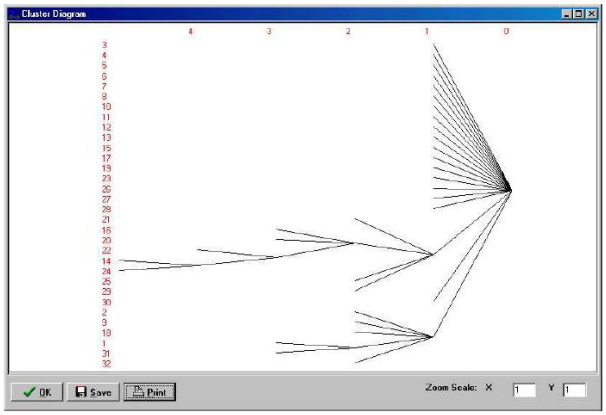
UCINET 6.0 (versión 6.399; Borgatti, Everett y Freeman, 2012) es un amplio programa para el análisis de redes sociales y otros datos de proximidad. Es probablemente el más conocido y más utilizado software para el análisis de sociales de redes de datos y contiene un gran número de rutinas de análisis de red. El programa es un producto comercial, existe una versión de evaluación gratuita, que puede ser ejecutada durante 30 días sin necesidad de registrarse. El manual consta de dos partes: una guía del usuario (gestión de datos y la manipulación) y una guía de referencia (el análisis de redes).

UCINET es un programa de Windows basado en menús, y, como los desarrolladores mismos dicen, "se construye para la velocidad, no para la comodidad" (Borgatti, Everett y Freeman, 1999). La elección de los procedimientos de los menús suele dar lugar a la apertura de un parámetro de formulario donde la entrada para los algoritmos se especifica. Se generan dos tipos de salida se generan: textual de salida, salvo en el diario y les muestra en la pantalla (véase la figura para un ejemplo), y conjuntos de datos que se pueden utilizar como entrada para otros procedimientos. Puede manejar del orden de 30.000 nodos.



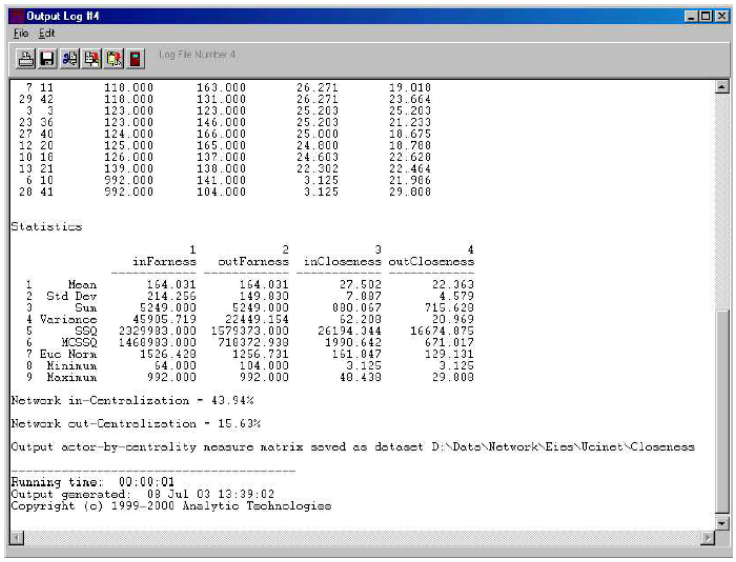
UCINET provee un gran número de herramientas de gestión de datos y la transformación como subconjuntos de la selección, la fusión de los conjuntos de datos, permutando, transposición, o recodificación de datos.
Tiene un lenguaje de álgebra de matrices con todas las características, puede manejar dos modalidades de datos, así como obtener los datos de un modo establece a partir de datos de dos modos. Hay una opción para introducir los datos de atributos y para especificar los valores que faltan. Cabe señalar, sin embargo, que sólo unos pocos procedimientos pueden manejar adecuadamente los valores que faltan. UCINET se distribuye con un gran número de datos de ejemplo conjuntos, incluidos los datos de Freeman EIES.

UCINET contiene herramientas gráficas para elaborar diagramas de dispersión y diagramas de árbol, que se pueden guardar como mapa de bits les (BMP). El programa en sí no contiene los procedimientos gráficos para visualizar las redes, pero tiene un speedbutton para ejecutar el programa NetDraw (Borgatti, 2002), que lee archivos UCINET de forma nativa. NetDraw, es un software desarrollado para la visualización de redes. Además de exportar funciones y Pajek Mago, los datos pueden ser exportados para su visualización en KrackPlot.



El programa contiene una gran cantidad de rutinas de análisis de red para la detección de subgrupos cohesivos (pandillas, clanes, plexos) y regiones (componentes, núcleos), por análisis de centralidad, para el análisis del ego de la red, y para el análisis de los agujeros

estructurales. Como un ejemplo, el resultado de un análisis de centralidad se presenta en la siguiente figura. Para cada nodo, contiene la lejanía de entrada y salida (la suma de las longitudes de las geodésicas de y de todos los demás nodos), y la centralidad de éstos.



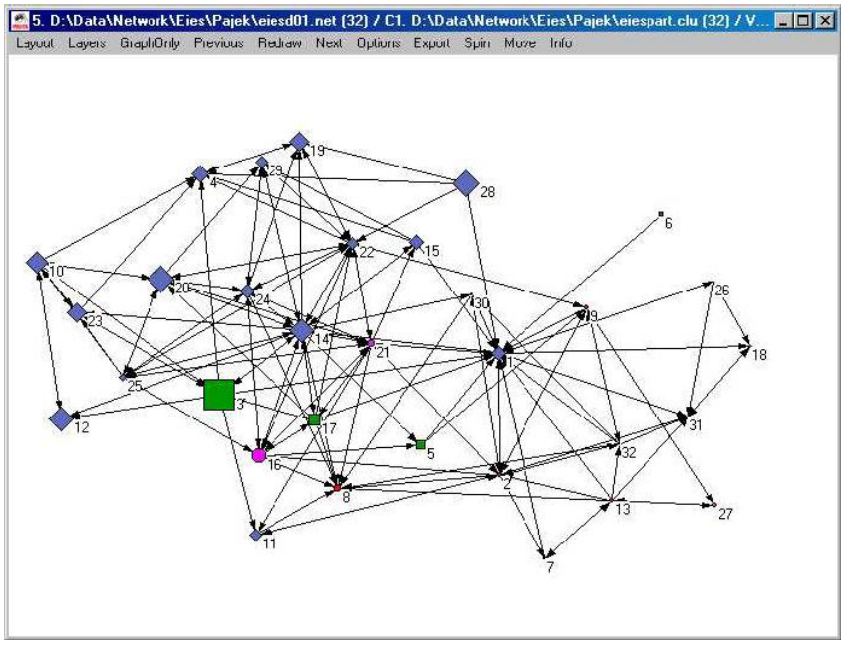
2.2.- Pajek

Pajek es un programa de análisis y visualización de redes, específicamente diseñado para manejar grandes conjuntos de datos. Los principales objetivos del diseño de Pajek son:

1. Facilitar la reducción de una red grande en varias redes más pequeñas que se pueden tratar más con métodos más sofisticados.
2. Proporcionar al usuario herramientas de visualización de gran alcance.
3. Aplicar una selección eficiente de algoritmos de redes.

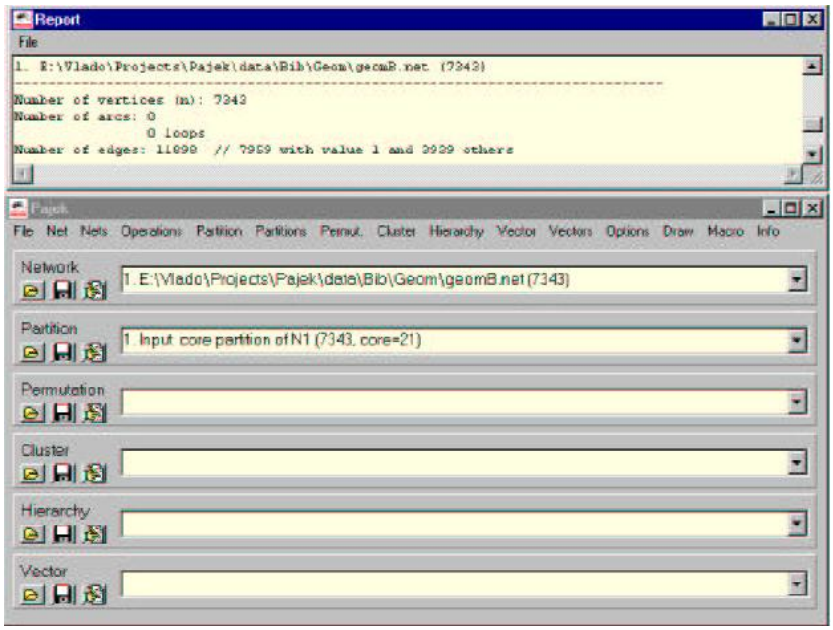
El programa puede descargarse de forma gratuita, y sus desarrolladores están continuamente actualizándolo. No hay ayuda en línea y la documentación disponible no es suficientemente detallada para los usuarios que no son expertos en el análisis de redes.

Pajek puede manejar múltiples redes simultáneamente y redes con eventos en el tiempo. Los eventos en el tiempo resumen el desarrollo o evolución de las redes a través del tiempo (usando unas diferencias horarias como indicadores). En Pajek se pueden analizar redes muy grandes, con más de un millón de nodos, la disposición de memoria en el equipo establece el límite real. Para ahorrar memoria, los nombres y etiquetas de los nodos no se mantienen las redes muy grandes, pero estos se pueden adjuntar más adelante a las pequeñas subredes.



Además de sus propios formatos de entrada, Pajek soporta varios formatos: UCINET DL, genealógicas GED, y algunos formatos molecular: BS (Ball y Stick), Mac (Mac Molecule) y MOL (MDL archivo MOL).
Al ejecutar Pajek obtendrá la ventana principal, Pajek es organizado como una "calculadora" para los datos de la red:

1. Network: objeto principal (vértices y líneas);
2. Partition: a la que pertenece un vértice de racimo;
3. Vector: valores de vértices;
4. Permutation: reordenación de los vértices;
5. Cluster: subconjunto de vértices (por ejemplo, un clúster de la partición);
6. Hierarchy: agrupaciones jerárquicamente ordenadas y vértices.



Pajek contiene opciones de manipulación para todas sus estructuras de datos. Por ejemplo, las redes pueden ser traspuestas, grafos dirigidos cambiados a grafos no dirigidos y viceversa al contrario, las aristas pueden ser añadidas o eliminadas, o la red puede ser reducida por la disminución de clases o extracción de las piezas. El programa también contiene las operaciones básicas de red como recodificación o dicotomización. Por otra parte, las transformaciones amplias para los atributos y las opciones para crear otros datos objetos sobre la base de los atributos (jerarquías, en racimos).

Las propiedades gráficas de Pajek son avanzadas. La ventana de dibujar da al usuario muchas opciones para manipular los gráficos (diseño, tamaño, color, rotación, etc.) Por otra parte, tiene representaciones gráficas de las particiones, los vectores, y combinaciones de particiones y vectores. El dibujo de la red se basa en el principio de que las distancias entre los nodos deben revelar el patrón estructural de la red. Además de diseños sencillos (círculo, al azar), Pajek tiene varios procedimientos automáticos para diseños óptimos. Tiene un algoritmo que supone que los nodos están conectados por medio de muelles, cuya tensión se quiere minimizar.

Última versión: 21-05-2012

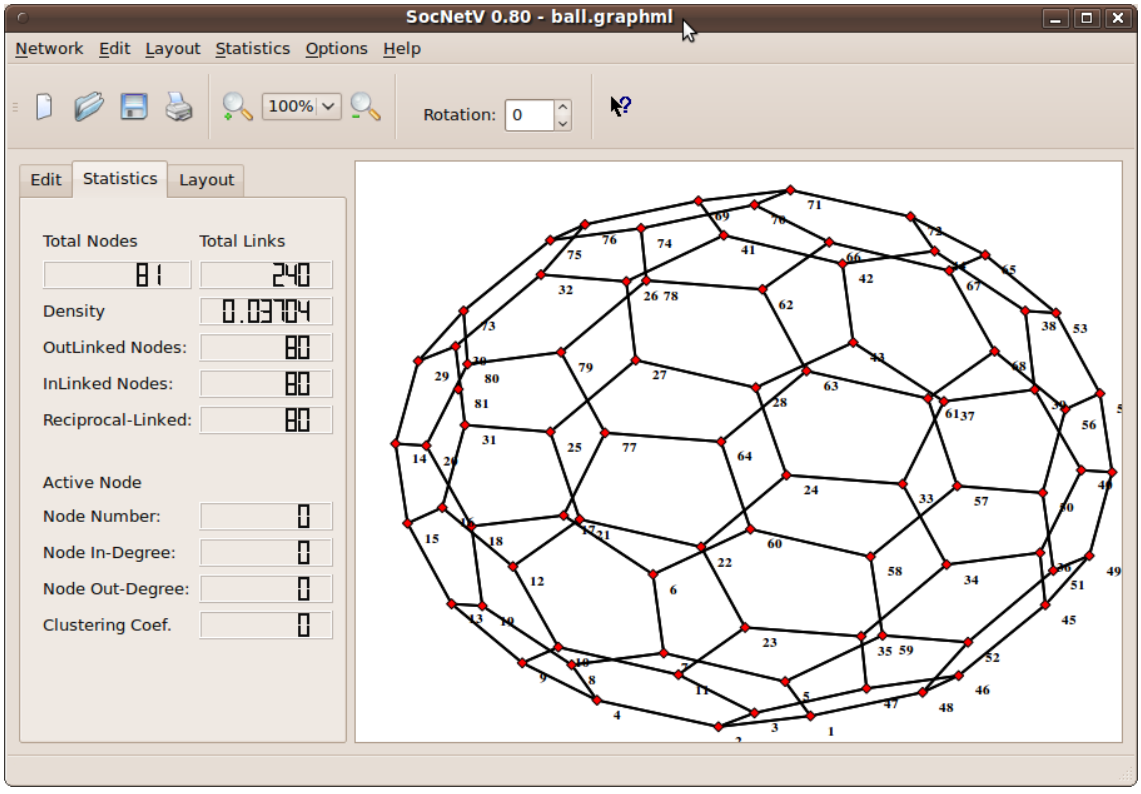
2.3.- SocNetV

Social Networks Visualizer (SocNetV) es una herramienta multi-plataforma, para el análisis y la visualización de las redes sociales. SocNetV le permite construir redes sociales con unos pocos clicks en un lienzo virtual o cargar redes en diferentes formatos (GraphML, GraphViz, Adyacencia, Pajek, UCINET, etc.) y modificarlos para adaptarlos a sus necesidades.

La aplicación puede calcular todas las propiedades de red de base, tales como el diámetro de grafos, y las distancias (la más corta longitud de ruta de acceso), así como las estadísticas estructurales más avanzados, tales como centralidades los nodos y de la red (es decir, cercanía, betweeness, grafos, etc.), coeficiente de agrupamiento , etc.

Dispone de varios algoritmos de diseño para las visualizaciones de sus redes. Además, se pueden crear con unos pocos clicks unas redes aleatorias (Erdos-Renyi, Watts-Strogatz, la red del anillo, etc.).

SocNetV está siendo desarrollado en C++ y Qt, un conjunto de herramientas de código abierto de desarrollo GUI de Nokia. El objetivo principal es la plataforma Linux, pero se puede compilar y ejecutar SocNetV en OS X y Windows, siempre y cuando cuenten con las bibliotecas Qt4 instalado.



Última versión: 13-10-2010

3.- ESPECIFICACIÓN DE REQUISITOS

3.1.- INTRODUCCIÓN

Krowface es una aplicación web innovadora para el análisis del comportamiento y la evolución de una red social. Existen diversas herramientas que tratan parcialmente este aspecto, pero no son integrables entre sí, y la elección de las bondades de una implica el rechazo de ventajas del resto que la elegida no trae. Por ello, cuando se diseña Krowface se intenta pensar en todas las buenas características de las herramientas existentes, además de nuevos aspectos no encontrados en ellas.

El objetivo es conseguir una herramienta flexible para poder reproducir todas las condiciones tanto fijas como variables que proporciona cualquier red social de hoy en día. Además, se pretende dar un aspecto totalmente amigable e intuitivo para facilitar el uso de la aplicación y la comprensión de los datos obtenidos.

En la especificación de requisitos se plasma el comportamiento del sistema, no se hará alusión es este documento a requisitos más bien subjetivos, de distinta interpretación por parte del usuario, como por ejemplo, el uso de un interfaz intuitiva ya que eso puede muy diferente por parte de cada persona.

El propósito es poder definir una serie de metas que nuestro sistema debe cumplir y una vez finalizada la implementación poder comprobar fácilmente si se han llegado a cumplir. Así como facilitar las etapas de la fase de implementación de tal forma que sirva de guía para completar todas las funcionalidades y requisitos de diseño del sistema. Además especificar concretamente todos los aspectos desarrollados en el sistema de tal forma que sea fácilmente comprobable su realización.

3.2.- REQUISITOS

3.2.1.- INTERFAZ WEB DE CONFIGURACIÓN

Requisito	Prioridad	Descripción
REQ001	Alta	El menú de configuración constará de 9 submenús: 1. Acciones 2. Atributos 3. Tipos de contenidos 4. Tipos de grupos 5. Tipos de relaciones 6. Tipos de Usuario de la Red Social (SNU) 7. Perfiles estáticos 8. Perfiles dinámicos 9. Vistas
REQ002	Alta	En este menú se podrá configurar todos los parámetros de una red social, para después poder simular con ellos.
REQ003	Alta	Todos los submenús de configuración deben de ser formularios por pasos, sencillos y lo suficientemente explicativos.
REQ004	Alta	Siempre se mostrará en cada submenú por defecto, un listado de los objetos de ese menú que ya se encuentran en el sistema, pudiendo hacer búsquedas sobre este conjunto.
REQ005	Media	En cada página de los submenús se incorporarán textos explicativos haciendo referencia al uso de esa página.
REQ006	Alta	Cada menú de configuración ha de incorporar una ayuda, en forma de video tutorial, en el que se vea como hay que hacer para ir gestionando cada submenú.
REQ007	Alta	Siempre se mostrará un último paso en forma de Resumen de los datos introducidos antes de guardarlos, y un paso de confirmación de que se han guardado bien los datos.
REQ008	Alta	**Submenú de Acciones:** En este submenú se podrá: 1. Visualizar las acciones que ya están en el sistema. 2. Crear una nueva acción. 3. Editar una acción existente. 4. Eliminar una acción existente. En la creación de una nueva acción se debe introducir la siguiente información: 1. Nombre de la acción. 2. El Plugin asociado a la acción. 3. Una breve descripción (opcional). 4. Puntos de acción por defecto de la acción. 5. Si es acción del sistema o no.

Requisito	Prioridad	Descripción
REQ009	Alta	**Submenú de Atributos:** En este submenú se podrá: 1. Visualizar los atributos que ya están en el sistema. 2. Crear un nuevo atributo. 3. Editar un atributo existente. 4. Eliminar un atributo existente. En la creación de un nuevo atributo se debe introducir la siguiente información: 1. Nombre del atributo. 2. El tipo del atributo (Numérico, Lista o String) I. Si es un atributo Numérico se introducirá un valor cualquiera mayor que cero. II. Si es un atributo de tipo lista, se pondrá introducir todos los atributos que quieras a la lista, y habrá que elegir cuál de ellos, es el atributo por defecto. III. Si es un atributo de tipo String bastará con introducir cualquier cadena de caracteres, con un mínimo de 3 caracteres.
REQ010	Alta	**Submenú de Tipos de contenidos:** En este submenú se podrá: 1. Visualizar los tipos de contenidos que ya están en el sistema. 2. Crear un nuevo tipo de contenido. 3. Editar un tipo de contenido existente. 4. Eliminar un tipo de contenido existente. En la creación de un nuevo tipo de contenido se debe introducir la siguiente información: 1. Nombre del tipo de contenido. 2. Una breve descripción (opcional). 3. Listado de atributos que incorpora el tipo de contenido.
REQ011	Alta	**Submenú de Tipos de grupos:** En este submenú se podrá: 1. Visualizar los tipos de grupos que ya están en el sistema. 2. Crear un nuevo tipo de grupo. 3. Editar un tipo de grupo existente. 4. Eliminar un tipo de grupo existente. En la creación de un nuevo tipo de grupo se debe introducir la siguiente información: 1. Nombre del tipo de grupo. 2. Una breve descripción (opcional). 3. Listado de atributos que incorpora el tipo de grupo.

Requisito	Prioridad	Descripción
REQ012	Alta	**Submenú de Tipos de relaciones:** En este submenú se podrá: 1. Visualizar los tipos de relaciones que ya están en el sistema. 2. Crear un nuevo tipo de relación. 3. Editar un tipo de relación existente. 4. Eliminar un tipo de relación existente. En la creación de un nuevo tipo de relación se debe introducir la siguiente información: 1. Nombre del tipo de relación. 2. Una breve descripción (opcional). 3. Listado de atributos que incorpora el tipo de relación.
REQ013	Alta	**Submenú de Tipos de SNU:** En este submenú se podrá: 1. Visualizar los tipos de SNUs que ya están en el sistema. 2. Crear un nuevo tipo de SNU. 3. Editar un tipo de SNU existente. 4. Eliminar un tipo de SNU existente. En la creación de un nuevo tipo de SNU se debe introducir la siguiente información: 1. Nombre del tipo de SNU. 2. Una breve descripción (opcional). 3. Listado de atributos que incorpora el tipo de SNU.
REQ014	Alta	**Submenú de Perfiles Estáticos:** En este submenú se podrá: 1. Visualizar los tipos de perfiles estáticos que ya están en el sistema. 2. Crear un nuevo perfil estático. 3. Editar un perfil estático existente. 4. Eliminar un perfil estático existente. En la creación de un nuevo perfil estático se debe introducir la siguiente información: 1. Nombre del perfil estático. 2. Una breve descripción (opcional). 3. Listado de acciones que puede ejecutar el perfil.

Requisito	Prioridad	Descripción
REQ015	Alta	**Submenú de Perfiles Dinámicos:** En este submenú se podrá: 1. Visualizar los tipos de perfiles dinámicos que ya están en el sistema. 2. Crear un nuevo perfil dinámico. 3. Editar un perfil dinámico existente. 4. Eliminar un perfil dinámico existente. En la creación de un nuevo perfil dinámico se debe introducir la siguiente información: 1. Nombre del perfil dinámico. 2. Una breve descripción (opcional). 3. Listado de perfiles estáticos que puede ejecutar el perfil dinámico. 4. Un atributo que va a decidir qué perfil se escoge en cada momento.
REQ016	Alta	**Submenú de Vistas:** En este submenú se podrá: 1. Visualizar las vistas que ya están en el sistema. 2. Crear una nueva vista. 3. Editar una vista existente. 4. Eliminar una vista existente. En la creación de una nueva vista se debe introducir la siguiente información: 1. Nombre de la vista. 2. Un Plugin asociado a la vista. 3. Una imagen de la vista seleccionada.

3.2.2.- INTERFAZ WEB DE SIMULACIÓN

Requisito	Prioridad	Descripción
REQ017	Alta	**Submenú de Simulación:** En este submenú se podrá: 1. Crear una nueva simulación 2. Consultar simulación previamente creada 3. Continuar una simulación previamente creada 4. Eliminar una simulación anteriormente creada En la creación de una nueva simulación se debe introducir la siguiente información: 1. Nombre de la simulación 2. Perfiles que intervienen en la simulación 3. Número de puntos de acción 4. Número de usuarios 5. Número de usuarios por perfil 6. Vistas que se quieren generar en la simulación

Requisito	Prioridad	Descripción
REQ018	Alta	**Simulación** En una simulación se podrá: 1. Ejecutar un paso de simulación en las hojas del árbol de ejecuciones de la simulación 2. Ejecutar varios pasos de simulación en las hojas del árbol de ejecuciones de la simulación. 3. Visualizar las visualizaciones seleccionadas en el proceso de creación de la simulación en curso. 4. Crear ramas paralelas de simulación, ejecutando acciones de sistema. 5. Visualizar el número de usuarios. 6. Visualizar la distribución de usuarios por perfiles 7. Visualizar un árbol con los diferentes instantes de simulación.

3.2.3.- MODELO DE RED SOCIAL

Requisito	Prioridad	Descripción
REQ019	Alta	Una red social está compuesta de: 1. Usuarios de la red social 2. Relaciones establecidas entre usuarios 3. Información de perfil asociada a los usuarios 4. Contenidos y grupos creados por los usuarios de la red social.
REQ020	Alta	El estado de la red social se identifica con los usuarios, relaciones, contenidos y grupos creados en cada instante de tiempo.

3.2.4.- MODELO DE USUARIO DE LA RED SOCIAL

Requisito	Prioridad	Descripción
REQ021	Alta	Un SNU (Social Network User) tendrá asociado un perfil.
REQ022	Alta	Un perfil establece cual es el comportamiento del SNU dentro de la red social, qué acciones puede llevar a cabo y con qué probabilidad.
REQ023	Alta	Cada acción tiene un coste de ejecución asociado. Un usuario tiene asociado unos puntos de acción, para cada unidad de tiempo de ejecución de la red social. Con estos puntos ejecutará todas las acciones posibles.
REQ024	Alta	Cada usuario tiene asociado un grupo de tipo TimeLine, al que se refieren los contenidos creados por él.

3.2.5.- RELACIONES ENTRE SNUs

Requisito	Prioridad	Descripción
REQ025	Alta	Entre dos SNUs de la red social puede existir una relación.
REQ026	Alta	Las relaciones son unidireccionales.

3.2.6.- GRUPOS DE LA RED SOCIAL

Requisito	Prioridad	Descripción
REQ027	Alta	Los grupos de la red social se utilizan para relacionar contenidos con algo en común.
REQ028	Alta	Los tipos de grupo son: 1. TimeLine: Todos los contenidos de un SNU. 2. Favorito: Todos los contenidos de usuario, que otro usuario selecciona como favorito. 3. Hashtag: Todos los contenidos con una etiqueta común.

3.2.7.- ACCIONES DE LA RED SOCIAL

Requisito	Prioridad	Descripción
REQ029	Alta	Una acción describe qué puede hacer un usuario dentro de la red social. Ejemplos: hacer un nuevo amigo, enviar un mensaje, crear un contenido, etc.
REQ030	Alta	**Evaluación de ejecución de acción:** **Precondición:** El SNU debe disponer los puntos de acción necesarios para esa acción. **Acción:** Se verifica que el SNU tiene puntos de acción suficientes para ejecutar la acción Se decrementan los puntos de acción del SNU necesarios para ejecutar la acción Se ejecuta la acción. **Poscondición:** El SNU tiene actualizados sus puntos de acción.
REQ031	Alta	**Borrar un contenido:** **Precondición:** El SNU que ejecuta la acción es autor de algún contenido. **Acción:** Recuperar todos los contenidos creados por el usuario. Borrar el contenido con más antigüedad. **Poscondición:** El contenido aparece en el historial de contenidos borrados junto con el tiempo en el que se ha borrado. **Estado de SNM:** Desaparece un contenido y todo lo relacionado con él a partir del instante de borrado y en adelante.

Requisito	Prioridad	Descripción
REQ032	Alta	**Crear grupo de tipo Favoritos:** **Precondición:** El SNU que ejecuta la acción debe tener algún amigo para marcarlo como favorito. **Acción:** El SNU crea un grupo del tipo "Favorites". Se añade una referencia al usuario que se selecciona como favorito. **Poscondición:** Hay un nuevo grupo en la red social. El usuario tiene un amigo más suyo como favorito. **Estado SNU:** Es el autor del nuevo grupo. Otro usuario amigo suyo está asociado a él.
REQ033	Alta	**Crear relación seguidor:** **Precondición:** Tiene que haber usuarios que a los que aún el usuario no lo siga. **Acción:** El SNU se convierte en seguidor de un SNU de la SNM al que no seguía. **Poscondición:** Hay una relación en la red social. **Estado SNU:** Sigue a un usuario que ya estaba en la SNM.
REQ034	Alta	**Crear nuevo usuario:** **Precondición:** Ninguna **Acción:** El SNU crea un SNU nuevo en la SNM pero no se convierte en su seguidor. **Poscondición:** Se ha creado un nuevo SNU en la SNM. **Estado SNU:** Esta acción sólo afecta al SNU que la ejecuta en los puntos de acción.
REQ035	Alta	**Dejar de seguir un grupo:** **Precondición:** El SNU debe haber marcado algún SNU como favorito suyo. **Acción:** El SNU que ejecuta la acción borra como favorito a uno de los usuarios que tiene marcados como favorito. **Poscondición:** El SNU tiene un favorito menos.
REQ036	Alta	**Enviar tweet:** **Precondición:** Ninguna. **Acción:** El SNU crea un nuevo contenido de tipo tweet del cual es el autor. **Poscondición:** Hay un nuevo contenido en la red social.
REQ037	Alta	**Enviar tweet con hashtag:** **Precondición:** Ninguna **Acción:** El SNU crea un nuevo contenido de tipo tweet. Además se crea un nuevo hashtag o se relaciona el tweet con uno existente. Esto se realiza a través de un grupo de tipo Hashtag. **Poscondición:** Hay un nuevo contenido en la red social. Además se relaciona el contenido con un grupo de tipo Hashtag. **Estado SNU:** EL SNU tiene un nuevo contenido.

Requisito	Prioridad	Descripción
REQ038	Alta	**Hacer nuevo amigo:** **Precondición:** Ninguna. **Acción:** El SNU crea un nuevo usuario de la red social. Además el SNU que ejecuta la acción se convierte en seguidor del nuevo usuario. **Poscondición:** Hay un nuevo SNU en la SNM. Éste tendrá un seguidor desde el inicio, el SNU que lo crea. **Estado SNU:** El SNU tiene una nueva relación de seguidor con el nuevo SNU que ha creado.
REQ039	Alta	**"Retweetear":** **Precondición:** EL SNU tiene amigos. **Acción:** El SNU crea un contenido del tipo retweet. Se crea una referencia al tweet original. Éste es un tweet de un amigo del SNU que ejecuta la acción. **Poscondición:** Hay un nuevo contenido de tipo retweet en la SNM. Además se añade la referencia al tweet "retweeteado".
REQ040	Alta	**Seguir grupo:** **Precondición:** El SNU sigue al menos a un SNU que tiene al menos un SNU como favorito. **Acción:** El SNU marca como favorito a un SNU que ya está marcado como favorito por un SNU al que sigue. **Poscondición:** El SNU que ejecuta la acción tiene un SNU más como favorito
REQ041	Alta	**Borrar usuarios amistosos (ACCIÓN DE SISTEMA):** **Precondición:** Ninguna. **Acción:** Se borran todos los usuarios con perfil amistoso en el instante en el que se ejecuta la acción de sistema. Creándose una nueva rama paralela. **Poscondición:** La nueva rama ha eliminado todos los usuarios con perfil amistoso.
REQ042	Alta	

3.2.8.- CONTENIDOS DE LA RED SOCIAL

Requisito	Prioridad	Descripción
REQ043	Alta	Los contenidos son creados por los SNU de la red social.
REQ044	Alta	Todo contenido tiene que tener una referencia obligatoria al grupo TimeLine del creador.
REQ045	Baja	Los contenidos pueden hacer referencia a grupos, contenidos, usuarios…

3.2.9.- TIEMPO DE LA RED SOCIAL

Requisito	Prioridad	Descripción
REQ046	Alta	El tiempo en la red social evoluciona por Instantes. Un instante es una unidad de tiempo "ficticia" que se puede hacer corresponder con unidades de tiempo real.
REQ047	Alta	Durante cada Instante, la red evoluciona de alguna forma porque los URS llevan a cabo acciones. Cada URS tiene asociado un número de unidades de ejecución. El número de unidades de ejecución indican cuantas acciones se pueden hacer por Instante. Un usuario en cada Instante, a medida que ejecuta acciones va consumiendo unidades de ejecución hasta agotarlas. Si con las unidades disponibles no puede ejecutar una acción, el coste de ejecutar la acción en el siguiente Instante se decrementa en tantas unidades como se disponga a la hora de ejecutarla.
REQ048	Alta	El tiempo en la red social puede evolucionar por diferentes ramas paralelas de tiempos, cuando el usuario del programa decida crear una rama y ejecutar alguna acción del sistema.

3.2.10.- REPRESENTACIÓN DE LA RED SOCIAL

Requisito	Prioridad	Descripción
REQ049	Alta	Krowface permitirá visualizar dinámicamente la evolución de la red social.
REQ050	Alta	Existirán distintos tipos de visualizaciones (vistas) de la simulación.
REQ051	Alta	Se podrán ejecutar ramas de tiempo paralelas realizando alguna de las acciones del sistema.

3.2.11.- RENDIMIENTO

Requisito	Prioridad	Descripción
REQ052	Alta	Krowface podrá soportar la ejecución simultánea de 10000 operaciones en total. Una operación es la ejecución de las acciones de un usuario en un instante.

3.2.12.- OTROS

Requisito	Prioridad	Descripción
REQ053	Alta	Se debe guardar en un LOG las acciones que ocurren en la red social.

3.3.- OBSERVACIONES

3.3.1.- REQUISITOS FUNCIONALES

3.3.1.1.- Múltiples redes sociales

Puesto que uno de los objetivos de nuestro programa es el estudio de redes sociales, se hace imprescindible el poder guardar diversas simulaciones para poder establecer comparaciones y bifurcar la evolución de una red social en varias en función de uno o varios parámetros.
Por tanto, se establece que la aplicación debe permitir detener y guardar una simulación, para poder reanudarla en cualquier otro momento desde el mismo instante en que se detuvo, sin perder ningún dato. Las redes sociales deben tener suficiente persistencia como para poder cerrar la sesión y volverla a abrir sin perder las redes sociales guardadas.

3.3.1.2.- Múltiples visualizaciones

Krowface debe permitir mostrar diferentes tipos de visualizaciones (vistas). El usuario de la aplicación podrá acceder a las visualizaciones que haya seleccionado en la creación de la red social. Una vez se generen los resultados de las visualizaciones seleccionadas, se podrá navegar por las diferentes ramas de tiempos, mostrando la visualización generada en cada uno de ellos.

3.3.1.3.- Gestión de perfiles de usuario

Con la finalidad de poder crear usuarios que se comporten de distinta manera, se debe dotar a la aplicación de un sistema de perfiles de usuario configurable. Estos perfiles definirán de alguna manera las tendencias de acción de los usuarios. Se debe incorporar, además, un gestor gráfico para poder crear, borrar y configurar estos perfiles. La distribución de perfiles dentro de la red social evoluciona según la distribución inicial de perfiles al crear la red social. Por ejemplo, si creamos una red con 3 usuarios, cada uno con un perfil, la red social evolucionará manteniendo la distribución del 33% para cada perfil.

3.3.1.4.- Asistente de configuración y creación de redes

La aplicación debe contar con un asistente detallado e intuitivo desde el cual se permita crear y configurar una red desde cero, con la elección del número de usuarios, perfiles de estos.

3.3.2.- REQUISITOS FUNCIONALES

3.3.2.1.- Interfaz gráfica intuitiva

La interfaz gráfica es muy importante, ya que supone el contacto entre el usuario y la aplicación. Se debe intentar que la interfaz web sea lo más autocontenida y explicativa posible, para que la web sirva como tutorial de la misma. Además, se incorporarán video tutoriales de ayuda para el uso de la web.
Es por ello que se fijan una serie de objetivos para la interfaz de Krowface:

1. Debe ser suficientemente intuitiva como para que el usuario pueda manejar todos los aspectos del programa sin tener que investigar demasiado.
2. Debe ser atractiva para que el usuario no se aburra de su uso, y no pierda interés por la aplicación.
3. Debe ser lo suficientemente potente como para poder aprovechar todas las ventajas que ofrece el núcleo de la aplicación, sin perder calidad respecto a los dos factores anteriormente mencionados.

3.3.2.2.- Portabilidad

Nuestro sistema funciona para cualquier sistema operativo, siempre que se acceda desde Google Chrome.

4.- ARQUITECTURA DEL SISTEMA

4.1.- INTRODUCCIÓN

En esta sección se hablará de las distintas decisiones que se han tomado para formar el diseño de Krowface. Algunas de esas decisiones se han tomado basándose en errores vistos en la anterior versión, y otras son simplemente para dar más potencia y utilidad a la aplicación, lo que se traducirá en nuevas posibilidades abiertas tanto para el desarrollo presente como para futuras ampliaciones.

En términos generales, se ha usado un nuevo modelo de datos donde se distingue claramente con vistas de la base de datos bien diferenciadas, la parte de configuración y la parte de ejecución del sistema. En el modelo de datos se distingue entre usuarios, perfiles, perfiles dinámicos, relaciones, acciones, grupos, contenidos... Cada usuario tiene un perfil determinado, que marca la preferencia del usuario para hacer unas acciones u otras. Estas acciones implican crear grupos, contenidos, establecer relaciones etc. Cada acción tiene un coste en puntos.

Por otro lado, un motor de tiempos desarrollado en Java con extensiones via plugins, que permiten añadir nuevas funcionalidades al motor básico de forma fácil y en tiempo de ejecución, sin necesidad de modificar nada en absoluto del código del motor. Además el acceso al motor esta implementado mediante servicios Rest, que permiten intercambiar información en formatos estándar como xml y json.

Para la interfaz de usuario, uno de los grandes cambios de esta versión, usamos una página web completa y amigable, que se divide en dos partes. La configuración, que sirve para introducir los datos (acciones, perfiles, atributos…) que van a formar la red social que queremos simular. Y por otro lado, el menú de la web de simulaciones, en el que se visualiza como va evolucionando la red social en cada paso de ejecución.

4.1.1.- PROPÓSITO DEL SISTEMA

El sistema va a permitir simular el comportamiento de una red social. Es decir, cómo se comportan los usuarios y cómo son sus relaciones en una red cualquiera como pueden ser Facebook, Tuenti o Twitter. Nos centraremos en ésta última para hacer las pruebas de la aplicación. Para ello debe de ser capaz de generar un buen número de usuarios que automáticamente creen relaciones entre ellos, de una forma no aleatoria según sus perfiles y afinidades y se pueda observar la evolución de la red social.

Los perfiles de cada usuario van a marcar el tipo de acciones más importantes que cada usuario va a realizar, de forma que cada usuario puede ejecutar las acciones incluidas en su perfil, pero están acciones están ponderadas al crear el perfil con un porcentaje, por lo tanto se ejecutarán más las que más porcentaje de actuación tengan.

Esta evolución deber ser guardada por el sistema para poder representar mediante imágenes (según los diferentes tipos de vistas) los cambios acontecidos a lo largo del tiempo.

4.1.2.- OBJETIVO DEL DISEÑO

Nuestro diseño debe permitir simular la evolución de la red social Twitter a partir de unos usuarios iniciales. Debe permitir una evolución hasta 10.000 URS sin que su comportamiento se vea afectado. La ejecución se produce de forma secuencial de manera que debe permitir la ejecución de las acciones de 10.000 usuarios en total, es decir, si ejecutamos 4 pasos, en esos 4 pasos no deben ejecutarse más de 10.000 acciones.

Un objetivo del diseño es también conseguir una evolución no aleatoria de la red social, es decir, conseguir una red con un buen número de usuarios entre los que exista una lógica entre sus relaciones y que el sistema no acabe llegando a una red enorme en la que todos los usuarios están relacionados con todos. Para conseguir este objetivo se han realizado los perfiles. El usuario, antes de lanzar una red social nueva, va a poder crear una serie de perfiles. Cada perfil tiene asociado un porcentaje de cada acción, este porcentaje va a marcar el número de veces que ejecutará cada acción. Por ejemplo si creamos un perfil y le asignamos un porcentaje del 50% a la acción crear nuevo amigo, los SNUs que tengan este perfil, se van a dedicar la mitad del tiempo que les corresponde ejecutar acciones a realizar la acción dicha.

Otro objetivo es marcar los pasos de la evolución de la red social. Para esto se ha pensado en los instantes de tiempo. Cada SNU en un instante de tiempo puede realizar una serie de acciones. Al crear la red social, se decide el número de puntos de acción que tienen los usuarios en cada instante. De esta forma, cuando el SNU no puede ejecutar más acciones con sus puntos, el siguiente usuario ejecuta las suyas propias.

También debe ser capaz el sistema de almacenar la evolución de la red social a lo largo de un periodo de tiempo. Para ello vamos a utilizar una base de datos MySQL. Así como representar varias vistas de esta evolución reflejando distintos datos, como la agrupación de los SNUs según sus perfiles, o la evolución de los tweets. Todas estas vistas deben ser fácilmente ampliadas con distintas medidas o visualizaciones de la red social que puedan surgir en un futuro.

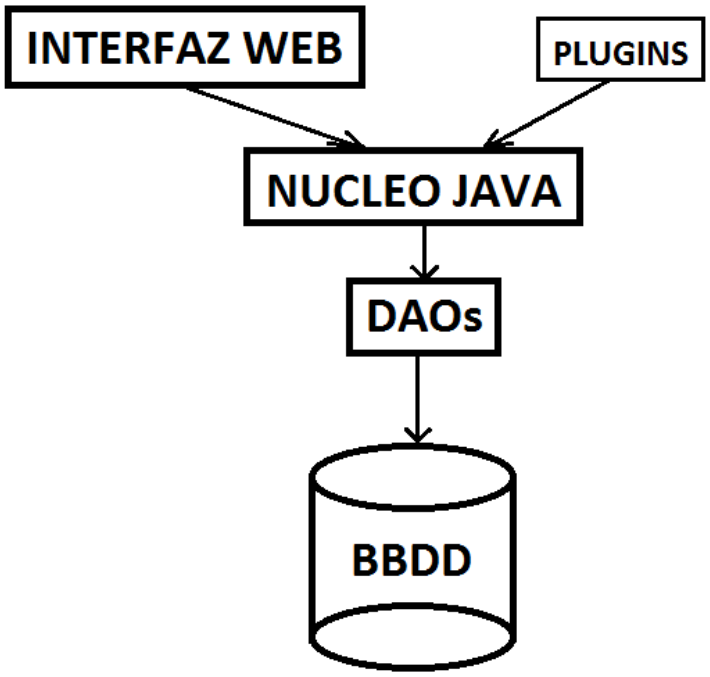
Las acciones deben ser fácilmente creadas también, de manera que los usuarios de la aplicación puedan crearse las que necesiten para crear su simulación.

4.2.- ARQUITECTURA DEL SOFTWARE

4.2.1.- PANORAMA

En lo referente a la arquitectura, se ha intentado separar la gestión de la interfaz, de la gestión de la persistencia y la lógica en 3 capas distintas, de modo que sea sencillo afrontar cambios o ampliaciones en el diseño de cada aspecto.
A continuación se explicará con detalle cada una de estas capas, junto a las decisiones de diseño involucradas.

4.2.2.- DESCOMPOSICIÓN EN SUBSISTEMAS



Podemos diferenciar varios subsistemas o módulos en la aplicación. El módulo más importante, el que podríamos considerar el motor de nuestra aplicación, sería el entorno de ejecución de la red social. Otro módulo usado sería el encargado de llevar a cabo las interacciones con la base de datos MySQL. Y por último la interfaz gráfica que se va a encargar de mostrar las visualizaciones y demás operaciones para crear redes, perfiles y asignarles a estos las acciones, y de mostrar la evolución de las redes sociales.

4.2.2.1.- NÚCLEO DE LA APLICACIÓN

El entorno de ejecución o núcleo de la aplicación se encarga de crear la red social y de que cada URS lleve a cabo sus acciones de forma que vaya evolucionando la red social.

Esta es, quizá, la parte más importante de Krowface, ya que es la que define todas las posibilidades de la aplicación. En esta parte se implementan cuestiones tan importantes como el motor de tiempos, el comportamiento de los usuarios y las acciones, grupos y contenidos con los que los usuarios pueden interactuar.

Acciones

Para representar lo que un usuario puede hacer en la red social, se utiliza el concepto de acción. Todas las acciones tienen una estructura similar, por lo que se han utilizado interfaces para favorecer la generalización y facilitar la creación de nuevas acciones en un futuro. Cada acción implementada es diferente al resto, y tiene efectos distintos. En una red social real, algunas acciones tienen un coste en esfuerzo mayor que otras. Para reflejar esa diferencia, cada acción tiene un coste específico de puntos, mayor cuanto mayor esfuerzo requiera. Estos puntos son por defecto, pero para cada perfil se puede asignar cuántos puntos cuesta la acción.

Perfiles. Diversidad de comportamientos.

Para reflejar la diversidad de usuarios en una red social, se decide implementar perfiles. Cada perfil define las distintas acciones a realizar y su frecuencia. Para ello, se asocia a cada perfil las acciones que se prefieran, y se priorizan en forma de porcentajes. De esta manera, el perfil determinará la proporción entre las acciones que el usuario intentará realizar en la simulación.

Se piensa también en las posibilidades de un perfil dinámico, que se adapte a lo que le va ocurriendo a su usuario. Sin embargo, esto es algo que se decide dejar para futuras versiones, para estudiarlo con detenimiento.

El usuario de la aplicación también puede crear perfiles a los que asociar las acciones que quiera.

Usuarios

Para representar un usuario se usará una entidad llamada SNU. Cada usuario estará asociado a un único perfil. Además, tendrá una lista de acciones, que se rellenará con acciones ordenadas aleatoriamente atendiendo a los porcentajes que marca su perfil. En la simulación, el usuario consumirá y ejecutará acciones de esta lista de la forma en que se explicará más adelante. Los usuarios podrán establecer relaciones de seguidor entre sí, y crear contenidos. También podrán agregar nuevos usuarios a la red.

Motor de tiempos y tabla de usuarios

En cada instante, se forma una lista con todos los usuarios existentes en ese momento, se les da unos puntos de acción para gastar y se manda a cada uno ejecutar acciones de su lista hasta agotar el crédito, o hasta llegar a una acción cuyo coste es mayor al crédito que le queda. Todo esto se realiza de manera secuencial.

Múltiples redes sociales

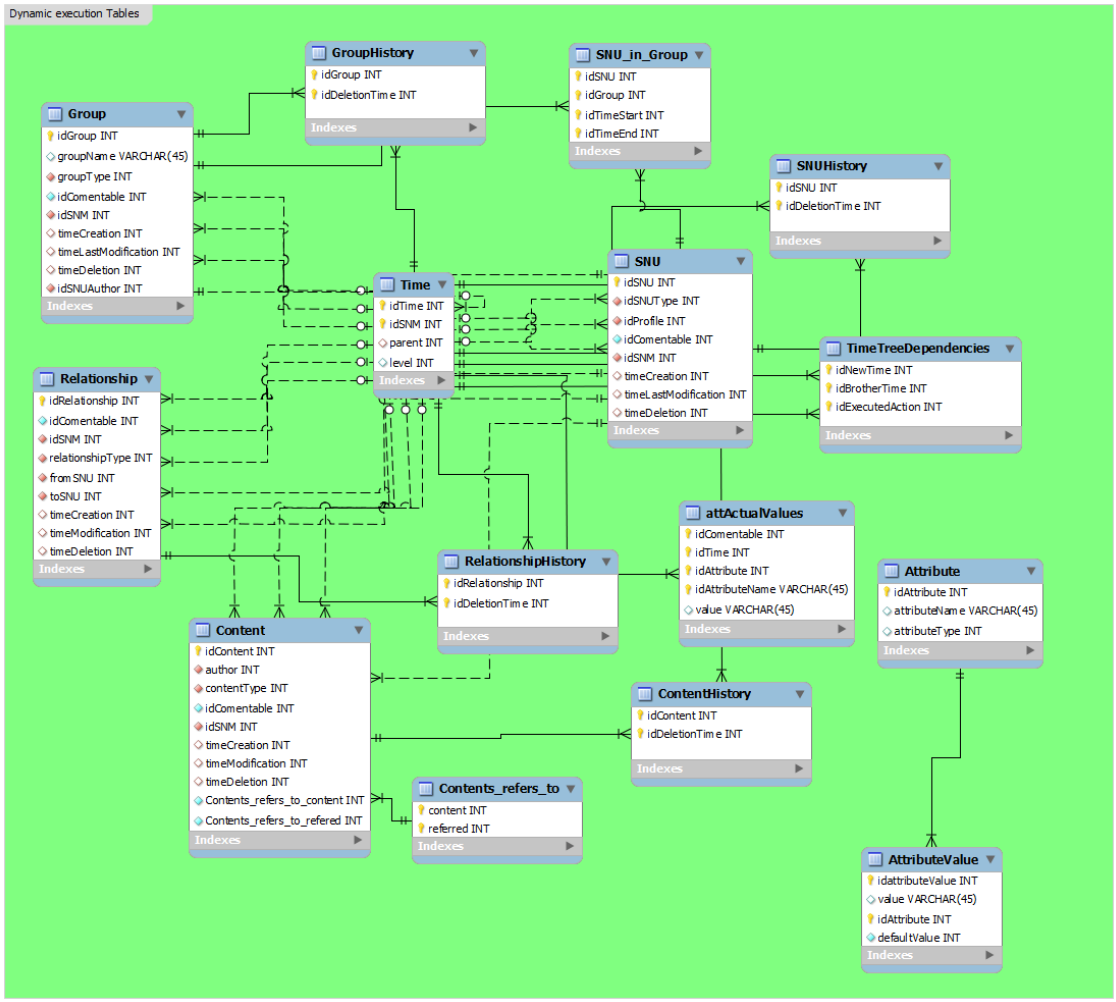
Para poder dar soporte a diversas redes sociales, los perfiles y usuarios en la base de datos tienen guardado el id de la red social a la que pertenecen. De esta manera, podemos recuperar los elementos de la red social necesarios de la base de datos, sólo recuperando los de la red social que estamos simulando.

El módulo de interacción con la base de datos se encarga de crear el modelo de datos de la red social según va recibiendo la información desde el entorno de ejecución. Debido a la penalización que suponen todas las operaciones de entrada salida, este módulo se puede dividir en dos partes, una de recepción de peticiones por parte del entorno de ejecución y otra de comunicación con la base de datos en sí misma. El envío de las peticiones por parte del entorno de ejecución es asíncrono, de forma que éste envía la petición y sigue con su ejecución sin esperar a que esta finalice, de eso se encarga el módulo.

4.2.2.2.- MÓDULO DE LA INTERFAZ WEB

El módulo de la interfaz web, como ya hemos referido antes, se divide en dos grandes submódulos, el de configuración y el de ejecución. El primero sirve para la configuración inicial de todos los parámetros de la red que queremos ejecutar, y el segundo para la visualización y ejecución de la simulación de la red social. Explicaremos los módulos por separado.

EL MODULO DE EJECUCIÓN:



El módulo de ejecución se compone principalmente de:

1. Usuarios
2. Contenidos
3. Grupos
4. Relaciones
5. Atributos
6. Tiempo

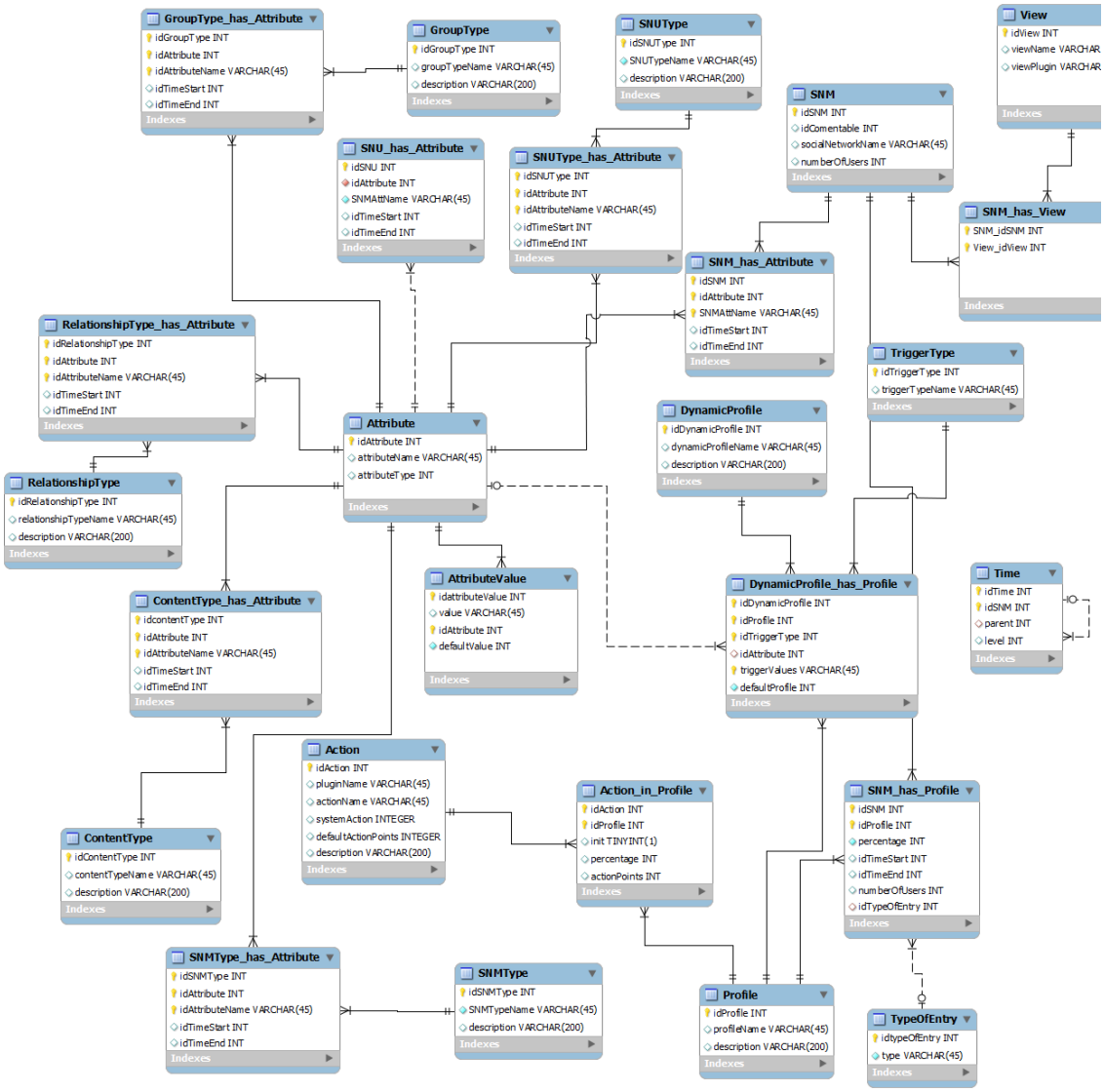
Las cuatro primeras tablas se van rellenando según evoluciona la red social durante su simulación. La tabla de tiempos se utiliza para rellenar el campo de tiempo de creación que tienen estas entidades. Además cada una de ellas tiene una tabla asociada con el sufijo “History”. En esta tabla se guardan las entidades de las que hemos hablado que han sido borradas en algún instante. Estas tablas son muy importantes ya que un elemento de la red social puede ser borrado en distintos instantes. Esto es debido a que tenemos ramas de tiempos paralelas. En esta tabla se guarda el elemento que se ha borrado junto con el identificador el instante en el que ha sido borrado.

Cada uno de estos elementos tiene asociados unos atributos según su tipo. Estos atributos pueden ir variando su valor según la simulación. Para guardar los valores de los atributos utilizamos la tabla attActualValues. En ella se guarda el atributo, el nombre del atributo, el idComentable del elemento al que pertenece el atributo, el tiempo en el que tiene el valor y por supuesto el valor.

La tabla SNU_in_Group se utiliza para gestionar los grupos de favoritos. El autor del grupo es el que tiene a otros usuarios como favoritos. Y los usuarios que aparecen en esta tabla asociados al grupo, son los que ha seleccionado como favoritos.

Por último tenemos la tabla “Contents_refers_to”. Esta tabla relaciona los contenidos creados con algún elemento de la red social. Los contenidos por ejemplo pueden referirse a otros contenidos (retweet), a un grupo con algo en común (hashtag), etc.

EL MODULO DE CONFIGURACIÓN:



El módulo de configuración se compone principalmente de las tablas:

1. Attribute
2. RelationshipType
3. ContentType
4. GroupType
5. SNUType
6. Action
7. Profile
8. DynamicProfile
9. View

Comentaremos las relaciones y tablas más relevantes.

Todas las tablas de tipo (RelationshipType, ContentType, GroupType y SNUType) tienen asociados un atributo de la tabla de atributos que lo caracteriza. Y otros parámetros del uso propio del tipo.

Por otro lado, hemos de destacar la relación entre los perfiles y las acciones, que se hace mediante la tabla Action_in_Profile, dónde se guardan todas las acciones que tiene asociado un perfil, con su porcentaje dentro del perfil y los puntos de acción que cuesta a ese perfil ejecutar la acción.
Por último, cabe destacar la relación entre perfiles estáticos y dinámicos, mediante la tabla DynamicProfile_has_Profile. Como ya hemos dicho un perfil estático tiene asociadas unas acciones con unos porcentajes de uso específicos. Los perfiles dinámicos se componen de perfiles estáticos, teniendo asociado un atributo que decide qué perfil estático se ejecuta en cada momento en el perfil dinámico. También tienen asignado un perfil estático por defecto para el caso de que no pueden seleccionar un perfil con el atributo seleccionado.

4.2.2.3.- MÓDULO INTERACCIÓN CON LA BASE DE DATOS

Mediante esta capa, se pretende independizar completamente la gestión de la base de datos del resto de la aplicación. Esto facilita enormemente las tareas de mantenimiento de la base de datos, al tener delimitadas las clases que interactúan con la misma.

DAOs
Debido a la complejidad del esquema de base de datos, se opta por implementar un DAO para mantener cada tabla. Un DAO será en adelante una clase que implemente todas y cada una de las consultas, actualizaciones y modificaciones de la tabla de la base de datos a la que se asocia. Mediante esta estrategia se consigue ganar flexibilidad y comodidad en caso de que haya que modificar las tablas, debido a que el código afectado está perfectamente localizado y acotado al DAO correspondiente a la tabla modificada.

Punto de entrada común. APIs
La gran cantidad de tablas necesarias en la base de datos, y por consiguiente, de DAOs para administrarlas, hace complejo conocer en cada momento el DAO necesario para gestionar los datos, ya que implica conocer la estructura de tablas para desarrollar algo a nivel de la lógica de negocio. Debido a que esto contradice las intenciones iniciales de mantener la gestión de la base de datos lo más independiente posible, se toma la decisión de implementar una API por elemento de la red social (SNU, motor, contenido, grupo y relación), que contendrá todas las funciones necesarias desde las capas superiores, dejando la responsabilidad de conocer la estructura de tablas y DAOs al programador de estas funciones.
Esta estructura hace posible poder dividir el equipo de desarrollo, de forma que se pueda destinar un subgrupo de personas al desarrollo, mantenimiento y ampliación de esta capa, incluyendo APIs, y otros equipos al desarrollo del resto de la aplicación, sin necesidad de que estos conozcan los detalles de implementación tanto de la estructura de la base de datos, como su gestión y código de mantenimiento. Lo único que necesitan estos equipos es que las funciones de API estén muy bien documentadas, por lo que se ha de poner especial cuidado en este aspecto.

Retardo de conexiones. ConnectionPool

Para hacer una consulta o query a la base de datos, antes hay que abrir una conexión por la que mandarla. Esto es significativamente más lento que la propia consulta, por lo que el proceso entero se ralentiza. Por ello, en aplicaciones con mucha carga de consultas como es Krowface, esta lentitud pasa factura en cuestiones de rendimiento.

Por ello, se decide implementar un pool de conexiones abiertas, llamado ConnectionPool. La misión del pool es abrir una cantidad suficiente de conexiones al arrancar el programa, y servirlas bajo demanda conforme se vayan necesitando, devolviéndose al pool cuando haya terminado de usarse. De esta manera, las conexiones permanecen siempre abiertas y se pueden mandar a través de ellas las consultas sin perder tiempo en operaciones redundantes, como la apertura y cierre de las mismas.

Módulo de visualizaciones de la red social

Por último tenemos el módulo de vistas que se encarga de hacer un modelo de las distintas visualizaciones al que accederá la interfaz gráfica para pintarlas.

Este módulo es el encargado de, una vez activado por parte del usuario, mostrar para cada instante de tiempo una “foto” que indique el estado de la red social en ese momento. Hay varios tipos de visualizaciones, hemos utilizado grafos, gráficas y otros diseños para reflejar las distintas perspectivas con las que podemos mirar el estado de la red social.

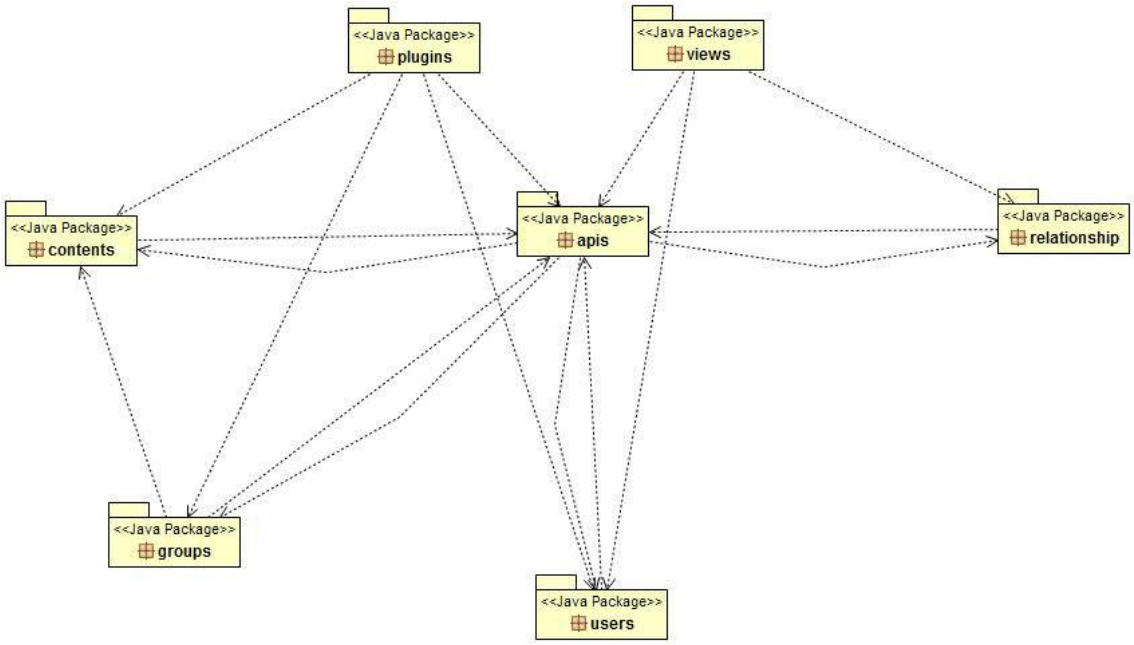
4.2.3.- ADMINISTRACIÓN DE DATOS PERSISTENTES

Para la persistencia de datos vamos a utilizar una base de datos en MySQL. En ella vamos a guardar los SNUs y sus perfiles, así como los contenidos y grupos creados. También las relaciones entre los SNUs. Para guardar la evolución de la red debemos marcar cada elemento con su tiempo de creación y guardar su tiempo de borrado. Al poder hacer ramas paralelas, un usuario puede ser borrado múltiples veces, y lo mismo para relaciones, contenidos o grupos. Por esta razón se introducen unas tablas históricas que recogen los tiempos de borrado de cada elemento. Así, dependiendo de la rama por la que se esté ejecutando, se tienen en cuenta esos elementos o no.

El diagrama de la base de datos utilizada se ha mostrado en el punto anterior.

4.2.4.- DIAGRAMAS DE CLASES (UML)

En este diagrama UML podemos observar las dependencias de los paquetes más importantes de la aplicación:



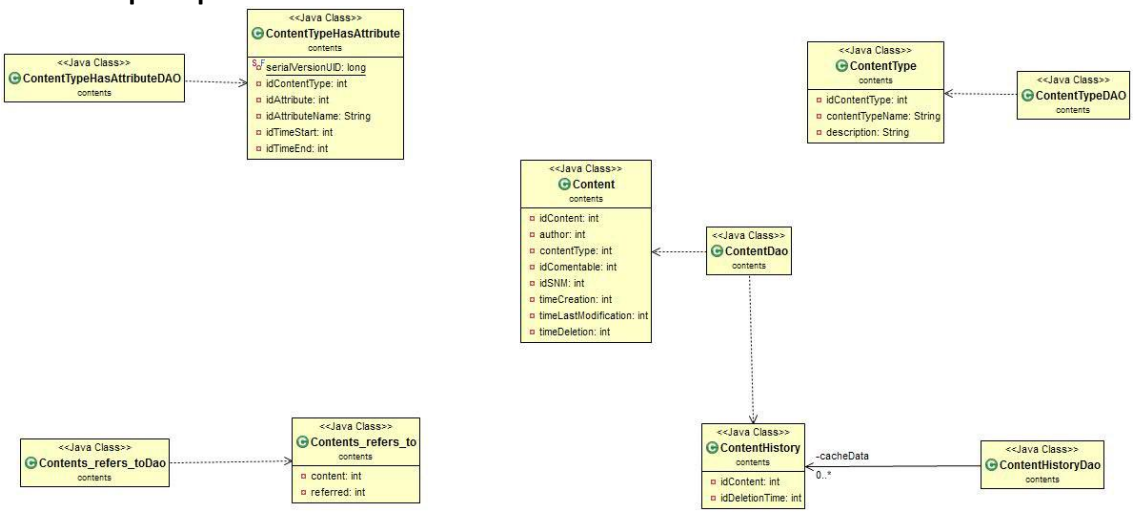
El diagrama muestra cómo las acciones (paquete plugins) y las visualizaciones (paquete views) utilizan los métodos proporcionados por las apis. Además, las 4 entidades principales de la aplicación utilizan y son utilizadas por las apis: usuarios, relaciones, contenidos y grupos.

Paquete Views: Contiene una interfaz genérica de manera análoga a los Plugins que explicaremos en el siguiente apartado.

Paquete Apis: Contiene todas las APIs del sistema (APIS de relaciones, usuarios, grupos, contenidos y motor). En éstas se encuentran los métodos necesarios para hacer más sencillo el uso al programador y al usuario de la aplicación.

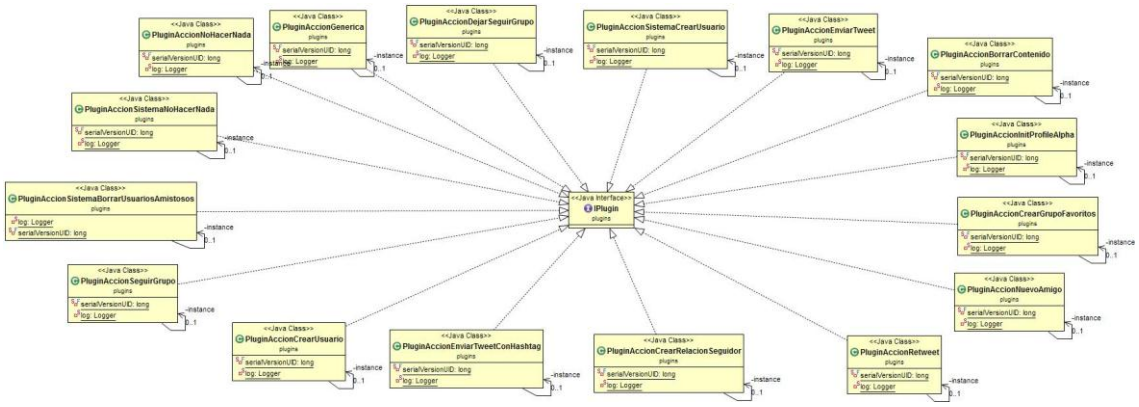
El resto de paquetes se explica a continuación.

Entidades principales



En este caso mostramos la entidad Content pero Group, Relationship y Users son idénticas, exceptuando las clases de Contents_refers_to que son específicas de los contenidos. Se observa que cada clase tiene su DAO correspondiente para el acceso a su tabla en la base de datos. Cada entidad además de su clase, tiene la clase del tipo, la clase de los atributos que tiene el tipo, y la clase de las entidades que se borran durante la ejecución (History). La clase ContentType se responsabiliza de que el tipo asignado a un contenido exista en la base de datos. La clase ContentHistory se utiliza tener acceso a los contenidos que han sido borrados en algún instante de tiempo. La clase ContentTypeHasAttribute asocia unos atributos al contenido. La clase Contents_refers_to se encarga de relacionar el contenido con otros contenidos, usuarios, relaciones, etc.

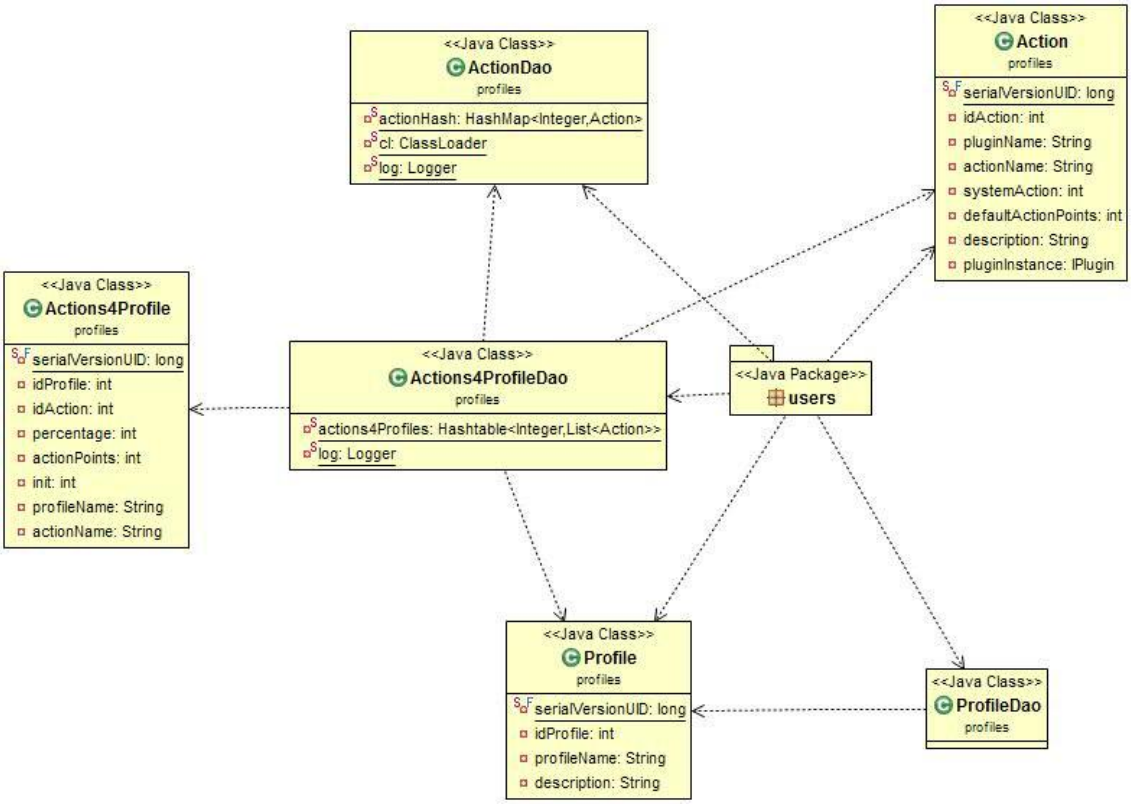
Plugins



Los plugins realizados, y los que creen los usuarios de la aplicación deben implementar la interface IPlugin tal como muestra el diagrama UML.

Los plugins de visualización (Views) se comportan exactamente igual, sólo que tienen una interfaz distinta.

Entidad Users



Esta entidad es especial ya que es la principal de la aplicación. En el paquete users podemos encontrar la estructura que hemos visto anteriormente. La diferencia se encuentra en el número de clases con las que se relaciona. Los usuarios además de tener un tipo, tienen un perfil. A este perfil se asocian unas acciones. Esto es lo que muestra este diagrama. Cada clase con su correspondiente DAO.

5.- LIBRERIAS UTILIZADAS

En este punto, detallaremos las librerías utilizadas en cada uno de los distintos lenguajes en los que esta implementado nuestro proyecto, para un mejor entendimiento de su estructura interna. También describiremos que uso las hemos dado en nuestro proyecto.

5.1 Librerías PHP

A continuación detallamos las librerías utilizadas del lenguaje de programación PHP.

5.1.1.- XML2array

Librería utilizada para convertir ficheros xml recibidos por red en un tipo array de PHP. La utilizamos en el paso de parámetros entre las distintas páginas web.

Extension de PHP Curl

Extensión de PHP utilizada para realizar la llamada a los servicios REST implementados en el motor Java de simulación almacenado en un servidor Tomcat.

5.2 Librerías Javascript

A continuación detallamos las librerías utilizadas del lenguaje de programación JavaScript.

5.2.1.- JQuery

Es un Framework de Javascript, pensado para utilizar Javascript de una forma más cómoda, compatible con todos los navegadores y eficiente. Este Framework admite extensiones. Las utilizadas en nuestro proyecto han sido las descritas a continuación:

Extensión JQuery asmselect
Pluging del Framework JQuery que permite crear cuadros seleccionables desplegables animados.

Extensión JQuery blockUI
Pluging del Framework JQuery que permite crear ventanas de espera animadas entre procesos de carga de una página web mediante Ajax.

Extensión JQuery json
Pluging del Framework JQuery que permite codificar y decodificar estructuras de datos al formato estándar json.

FormToWizard
Librería que utiliza el Framework JQuery que permite la creación de formularios interactivos con pasos sin recargar la página implementada.

5.2.2 Google Char Tools
Librería implementada por Google para visualización de todo tipo de gráficos en páginas web sobre Javascript. Los gráficos concretamente que hemos utilizado en nuestra implementación han sido Pie Chart y Org Chart, para visualizar los datos en los distintos puntos de simulación y en los menús de configuración de la aplicación

5.3 Librerías Java
A continuación detallamos las librerías Java utilizadas en nuestro proyecto.

5.3.1.- Gephi Toolkit
Librería de generación de grafos para el estudio de dominios en investigación. Su diseño está pensado para el estudio de diferentes redes, entre ellas, las redes sociales. Esta librería la utilizamos para generar las imágenes del estado de simulación en cada paso de la simulación

5.3.2.- Log4j
Librería de apoyo a la depuración de programas Java. Esta librería nos ha servido para depurar nuestro sistema y encontrar posibles errores de programación.

5.3.3.- JUnit
Librería diseñada para la creación de casos de prueba en programas Java. Con esta librería hemos realizado los casos de prueba para nuestras clases Java.

5.3.4.- Jersey
Librería que permite la creación de servicios REST. Los servicios REST son URL's http a las que podemos añadir parámetros para obtener resultados concretos. Estos servicios los utilizamos para enlazar la página web, parte visual de nuestro sistema, con la parte background, simulador implementado en Java.

5.3.5.- JSON
Librería diseñada para interpretar código estándar Json en java. Utilizamos esta librería para establecer los diferentes parámetros de configuración de nuestra aplicación.

6.- CASO DE ESTUDIO: TWITTER

Para la realización de la aplicación web, nos basamos en la red social Twitter. A continuación se explica cómo se ha estudiado la red social y cómo hemos modelado el comportamiento y los elementos de ésta.
Lo primero que se observa al estudiar Twitter son los usuarios y los tweets. Así que empecemos por ellos.

Usuarios
Los usuarios son la parte más importante de la red social, sin usuarios la red social no existe. Los usuarios son modelados de forma que tienen un perfil asignado. De esta manera diferenciamos usuarios, y por tanto, diferenciamos las acciones que realizan distintos usuarios. Los usuarios son los que llevan el peso de las acciones en una red social, es por eso por lo que los usuarios de nuestra aplicación son los que ejecutan un número determinado de acciones en cada paso de ejecución.
También son los que generan los tweets y los que realizan copias de tweets de otros usuarios a los que siguen, conocidos como retweets. Los usuarios son los que deciden hacerse seguidor (follower) de otros usuarios de forma que pueden ver los tweets que crean estos.
Además en Twitter, se puede seleccionar un usuario y ver todos los tweets de éste. Es lo que en la aplicación se ha llamado Timeline, que es un grupo único por usuario al que se asocian todos los tweets de cada usuario.

Contenidos
En Twitter los contenidos son los tweets. Para modelarlo, hemos decidido que no solo haya un tipo de contenido, el tweet. El tweet es el contenido más simple de la aplicación. Todo tweet tiene un usuario como autor. Se crea un contenido en la base de datos teniendo una referencia a su autor. Además todos los contenidos pueden referenciar cualquier otro elemento de la red social, usuarios, grupos, relaciones u otros contenidos.
Una de las características más famosas de Twitter son los hashtag. Para implementar los tweets con hashtag, utilizamos un contenido del mismo tipo que un tweet normal, pero la diferencia es que se crea una referencia a un grupo. Cuando un usuario escribe un tweet con hashtag, puede escribir un hashtag nuevo o uno que ya esté creado. Cuando se crea un nuevo hashtag, se crea un grupo al que van a referenciar todos los tweets que contengan el mismo hashtag.

Para implementar el retweet, se ha decidido crear un tipo nuevo de contenido llamado Reteweet. Este contenido es igual que el tweet normal, la diferencia es que hace referencia a otro tweet que ha sido escrito por un usuario de los que el que escribe el retweet es seguidor.

Así conseguimos realizar una simulación del funcionamiento de usuarios y contenidos en twitter. A continuación se explican los demás elementos y su implementación.

Relaciones
La relación en Twitter es también muy famosa, la relación seguidor, conocida como follower mundialmente. Esta relación es unidireccional, ya que un usuario puede ser seguidor de otro que no sea seguidor suyo. Las relaciones se ha implementado de esta manera, así una relación queda determinada por su id único, y guarda el usuario origen y el destino.

Grupos
Hemos implementado tres tipos de grupos distintos. Ya hemos hablado de dos de ellos, empezaremos por ellos.
El primer tipo de grupo que existe es el Timeline, todos los contenidos creados por un usuario hacen referencia a su grupo Timeline. Todo usuario tiene su propio Timeline desde el momento en que escribe su primer tweet o retweet.
El segundo grupo existente, del que ya hemos hablado también, es el grupo utilizado para los hashtags. El grupo de tipo hashtag se utiliza para ser referenciado por todos los contenidos que contienen el hashtag.
El tercer grupo es el de favoritos, un usuario puede seleccionar a otros usuarios como favoritos suyos. En este caso cada usuario tiene un grupo de este tipo y se asocian a él los usuarios que forman parte de los favoritos del autor del grupo.

Acciones
Ya hemos enumerado multitud de acciones implícitamente en los apartados anteriores. En Twitter, se pueden crear contenidos por ejemplo, para eso hemos creado 3 acciones, enviar un tweet, hacer un retweet, y enviar un tweet con hashtag. De esta manera cubrimos todas las posibilidades en lo que a creación de contenidos se refiere. Se ha implementado la acción de crear una nueva relación simulando que un usuario se convierte en seguidor de otro. Además se han creado acciones para marcar a un usuario como favorito, dejar de seguirlo, etc.
Debemos hacer una mención especial a una acción, la de crear un nuevo usuario. Está acción también realizada por un usuario, no existe en Twitter. Un usuario no introduce a otro en ninguna red social, cada usuario se introduce en una red social por sí solo. En nuestro caso hemos decidido que la incorporación de usuarios a la red social la hagan los propios usuarios a través de sus acciones. Esta es la única acción que no tiene una relación directa con una de Twitter. Además hay otra acción que también crea usuarios y además el usuario que crea al otro se convierte en seguidor de este último.

7.- MANUAL DE USUARIO

En este apartado detallaremos los requisitos necesarios para instalar y ejecutar nuestra aplicación. Más tarde, daremos un breve paseo por sus diferentes pantallas y por último, daremos una pequeña explicación de las diferentes visualizaciones que hemos creado en nuestro sistema para representar la red social en cada estado de tiempo.

7.1.- Requisitos del sistema

Debido a que la aplicación que hemos desarrollado es una aplicación de servidor, es decir, es ejecutada en un computador servidor y se accede a ella desde computadores cliente que solicitan la información deseada, los requisitos del computador que actúa como servidor y los requisitos del computador que actúa como cliente, no son los mismos. A continuación detallaremos los requisitos para ambos roles de computadoras.

Requisitos del computador Cliente

- Tener instalado como sistema operativo Windows Xp o superior, MacOs o cualquier distribución Linux.
- Tener instalado en el computador el navegador Google Chrome de Google.
- Tener habilitado el uso de Javascript en el navegador Google Chrome si este no estuviera activado por defecto.

Requisitos del computador Servidor

- Tener instalado como sistema operativo Windows Xp o superior, o cualquier distribución Linux.
- Tener instalado en cualquiera de los dos sistemas operativos anteriores una instancia del servidor de aplicaciones Tomcat 7.0
- Tener instalado en cualquiera de los dos sistemas operativos anteriores una instancia del servidor de páginas web Apache.
- Tener instalado un intérprete de PHP con versión 5.3.1 o superior y que tenga instalado la librería CURL de PHP.
- Tener configurada la instancia del servidor de páginas web Apache para que interprete PHP.
- Tener instalado en cualquiera de los dos sistemas operativos anteriores una instancia del servidor de bases de datos MySQL con versión igual o superior a 5.1.41.

- Tener instalado el jdk de Java versión 1.6 de Oracle
- Cliente svn para descargar el código de nuestro proyecto

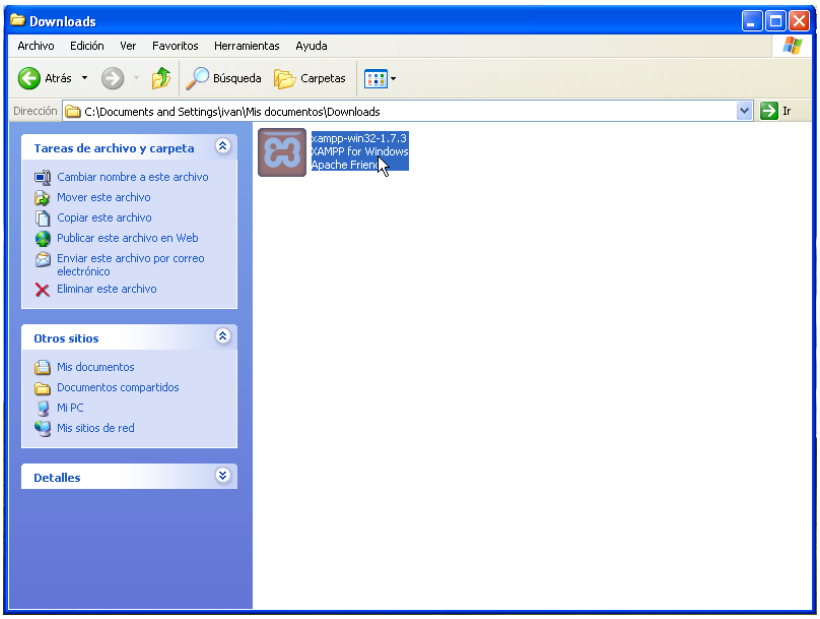
7.2 Instalación

Dada la diversidad de configuraciones posibles en las distribuciones Linux y la no disponibilidad de un equipo con el sistema operativo MacOS, abordamos la explicación de la instalación de nuestra aplicación en una computadora que ejecuta el sistema operativo Windows XP. Cabe destacar que nuestra aplicación solo será necesario instalarse en el computador que actúe como servidor.
Para una mejor comprensión de la instalación de la aplicación, en esta guía de instalación también abordaremos la instalación del software secundario necesario, para que nuestra aplicación se ejecute en perfectas condiciones.

Instalación Servidor Apache y la base de datos MySQL

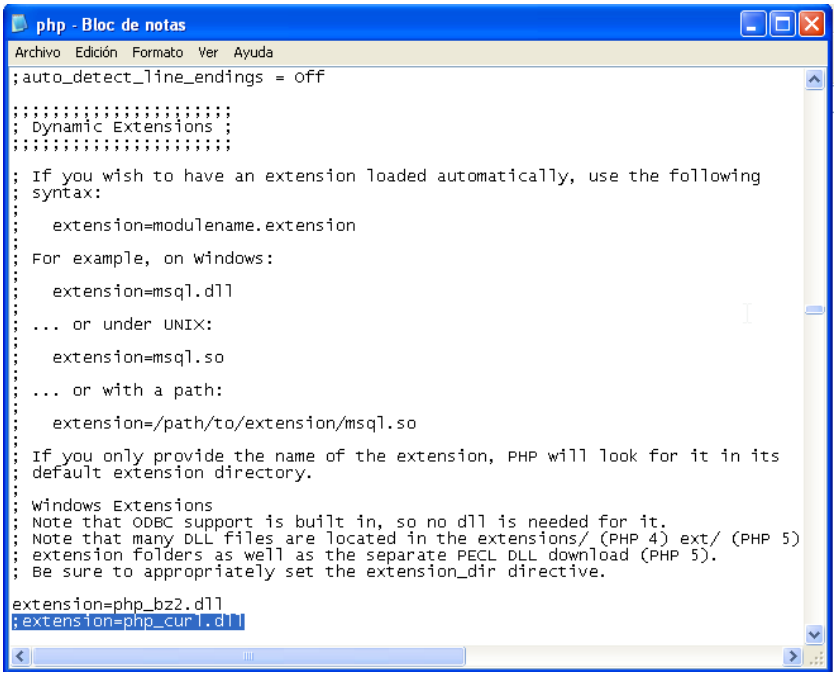
1. Descargamos el paquete de software xampp que contiene un servidor de bases de datos MySQL y un servidor web Apache. La url donde podemos encontrarlo es la siguiente: http://www.apachefriends.org/es/xampp.html

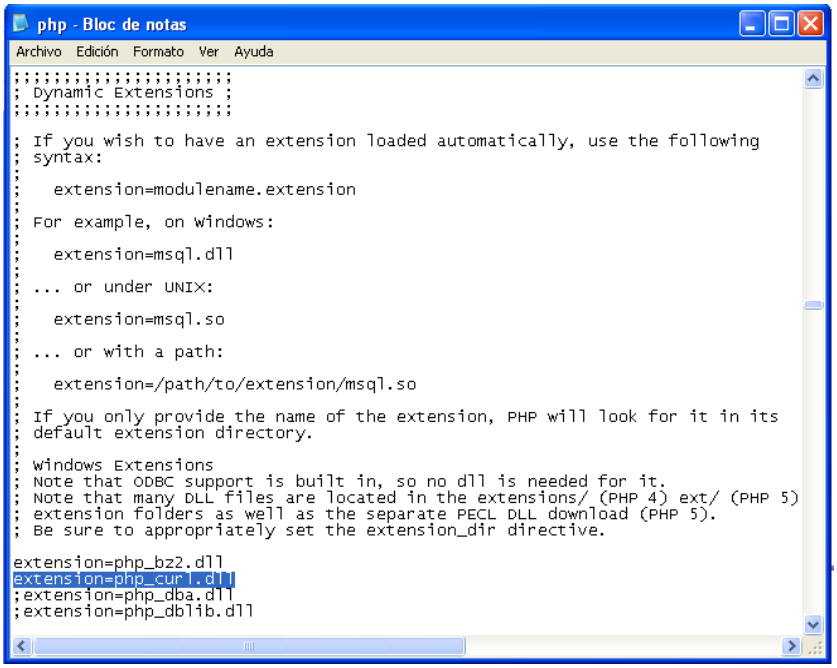
2. Instalamos el paquete el paquete descargado:



3. Seleccionamos como ruta de instalación C:\

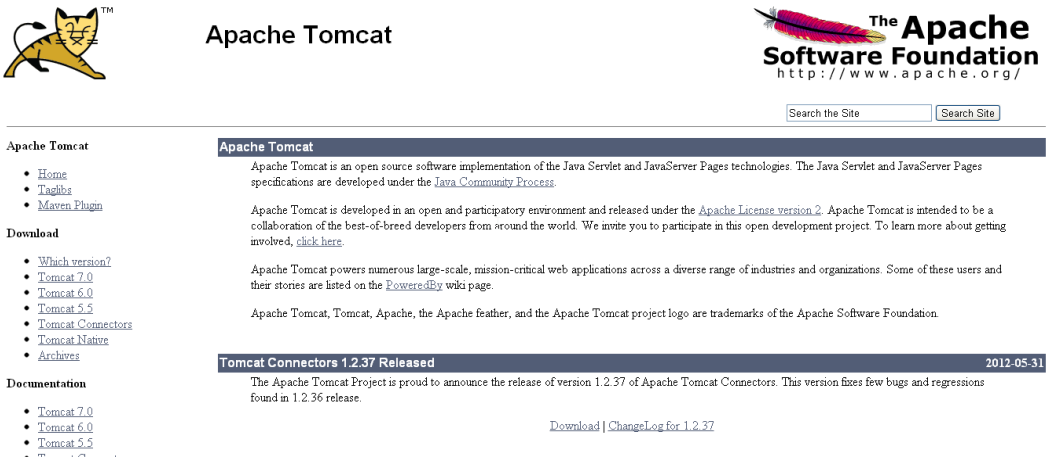
4. El resto de opciones del instalador de xampp las dejamos predefinidas.
5. Editamos el fichero C:\xampp\php\php.ini y cambiamos el contenido indicado en las imágenes


php - Bloc de notas
Archivo Edición Formato Ver Ayuda
;auto_detect_line_endings = Off
;;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;;
; If you wish to have an extension loaded automatically, use the following
; syntax:
;
;   extension=modulename.extension
;
; For example, on Windows:
;
;   extension=msql.dll
;
; ... or under UNIX:
;
;   extension=msql.so
;
; ... or with a path:
;
;   extension=/path/to/extension/msql.so
;
; If you only provide the name of the extension, PHP will look for it in its
; default extension directory.
;
; Windows Extensions
; Note that ODBC support is built in, so no dll is needed for it.
; Note that many DLL files are located in the extensions/ (PHP 4) ext/ (PHP 5)
; extension folders as well as the separate PECL DLL download (PHP 5).
; Be sure to appropriately set the extension_dir directive.
extension=php_bz2.dll
;extension=php_curl.dll

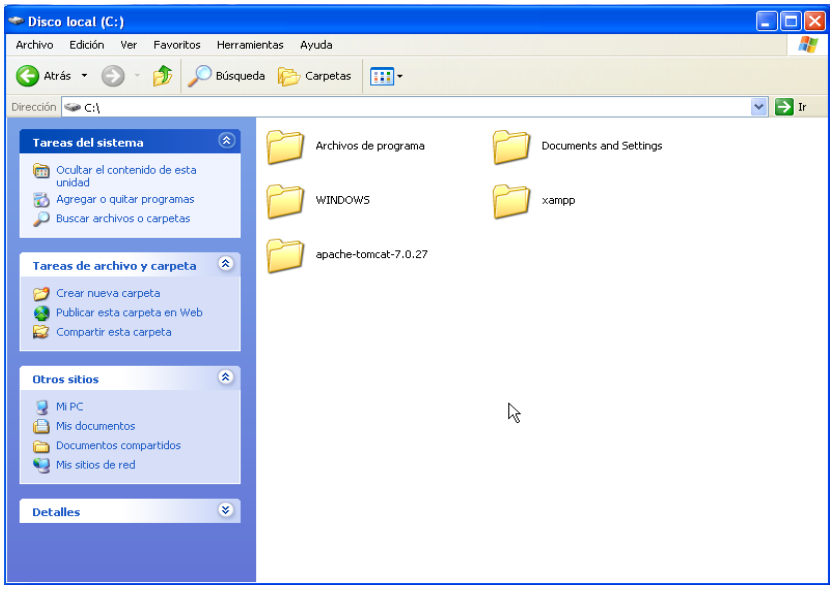

php - Bloc de notas
Archivo Edición Formato Ver Ayuda
;;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;;
; If you wish to have an extension loaded automatically, use the following
; syntax:
;
;   extension=modulename.extension
;
; For example, on Windows:
;
;   extension=msql.dll
;
; ... or under UNIX:
;
;   extension=msql.so
;
; ... or with a path:
;
;   extension=/path/to/extension/msql.so
;
; If you only provide the name of the extension, PHP will look for it in its
; default extension directory.
;
; Windows Extensions
; Note that ODBC support is built in, so no dll is needed for it.
; Note that many DLL files are located in the extensions/ (PHP 4) ext/ (PHP 5)
; extension folders as well as the separate PECL DLL download (PHP 5).
; Be sure to appropriately set the extension_dir directive.
extension=php_bz2.dll
extension=php_curl.dll
;extension=php_dba.dll
;extension=php_dblib.dll

Instalación Apache Tomcat 7

1. Descargamos Apache Tomcat 7 desde la siguiente url: http://tomcat.apache.org/

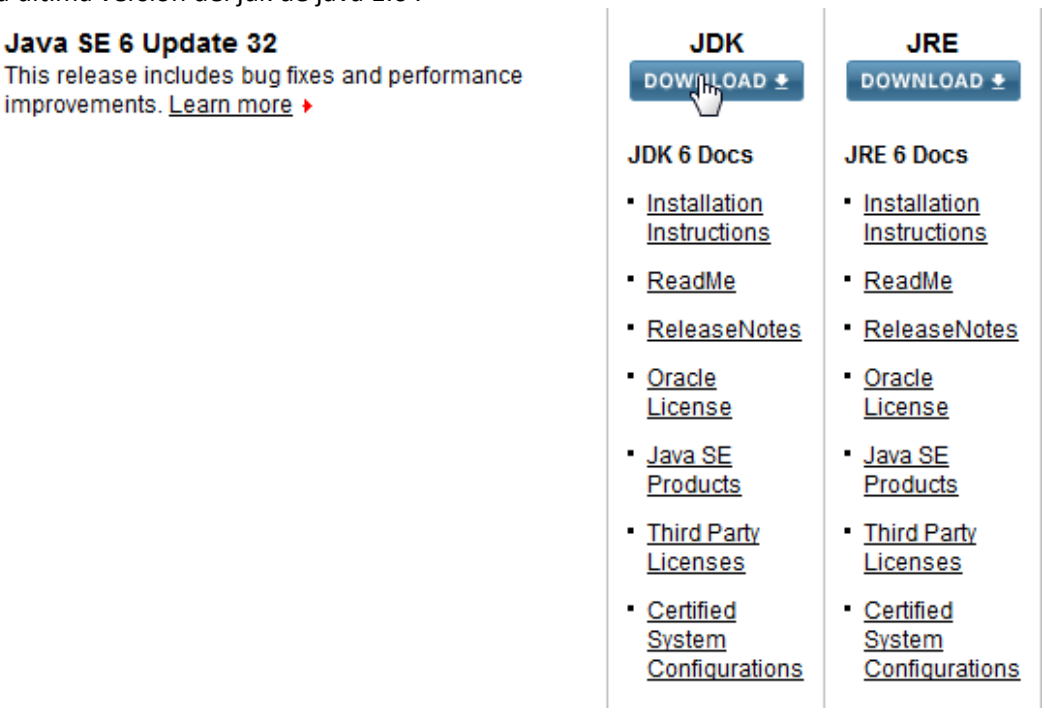


2. Descomprimimos el archivo zip descargado de Tomcat 7 en la ruta C:\ quedando de la siguiente forma el árbol de directorios:

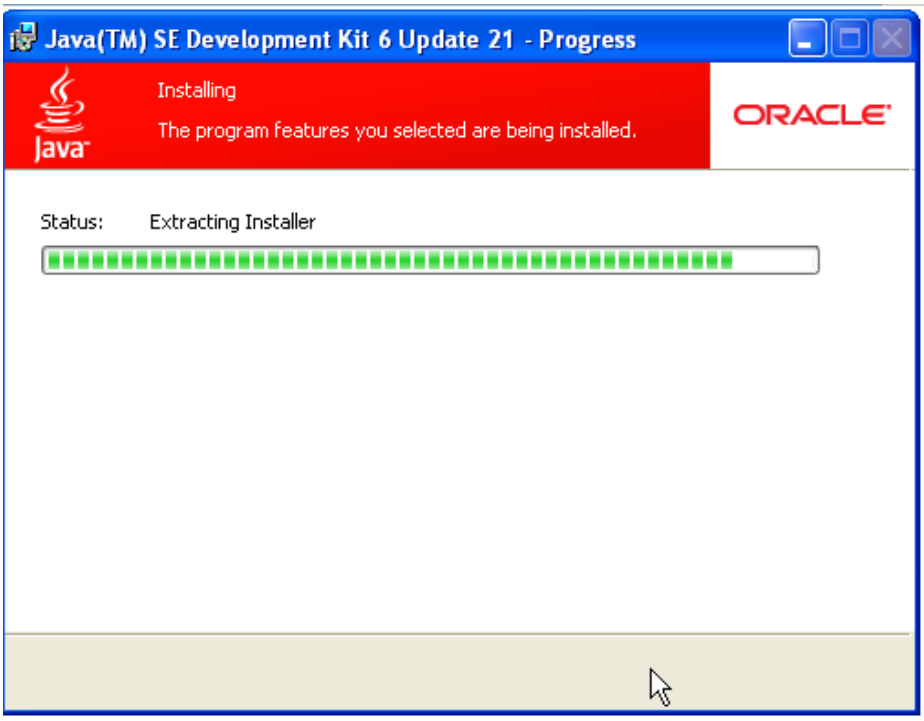


Instalación Jdk de Java

1. Descargamos de la url :
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
   La última versión del jdk de java 1.6 .

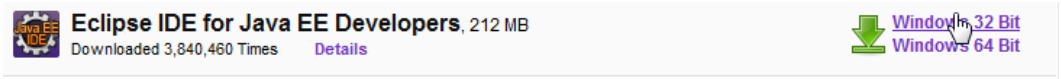


2. Ejecutamos el instalador y seleccionamos todas las opciones por defecto.

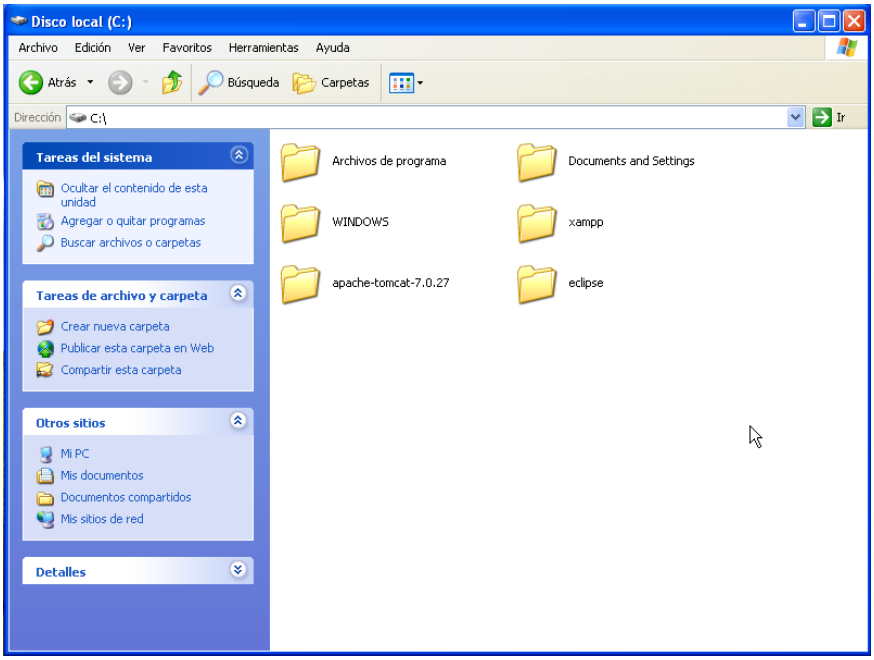


Instalación del entorno de compilación Eclipse

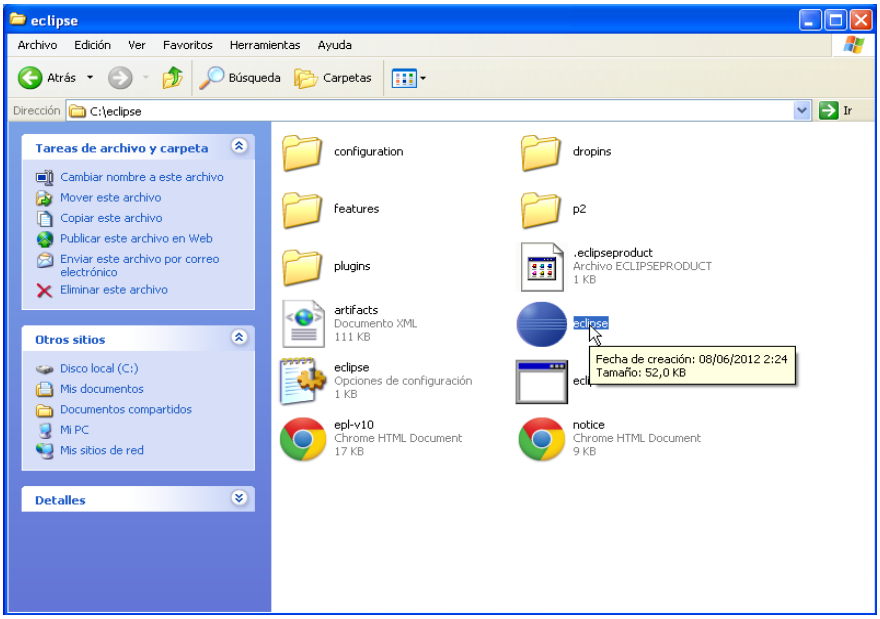
1. Descargamos Eclipse en su versión JEE desde esta URL: http://www.eclipse.org/downloads/



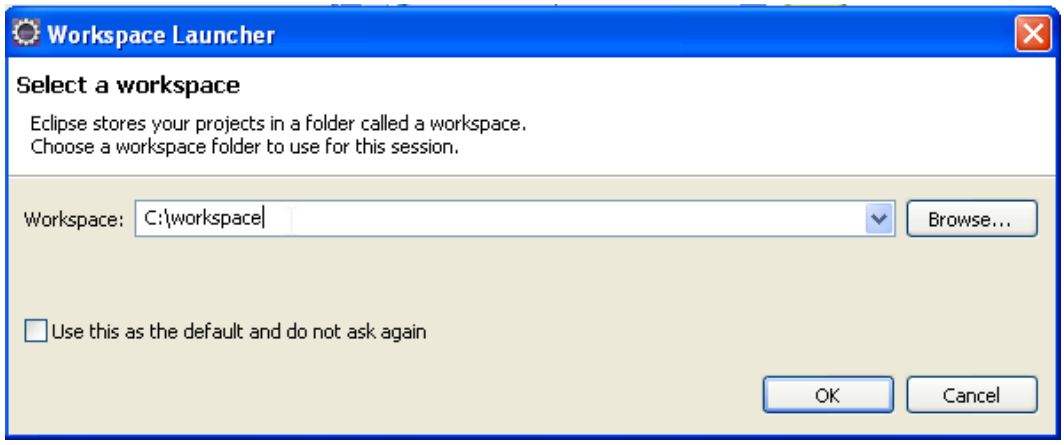
2. Descomprimimos el archivo zip descargado en la ruta C:\ quedando la siguiente estructura de directorios:



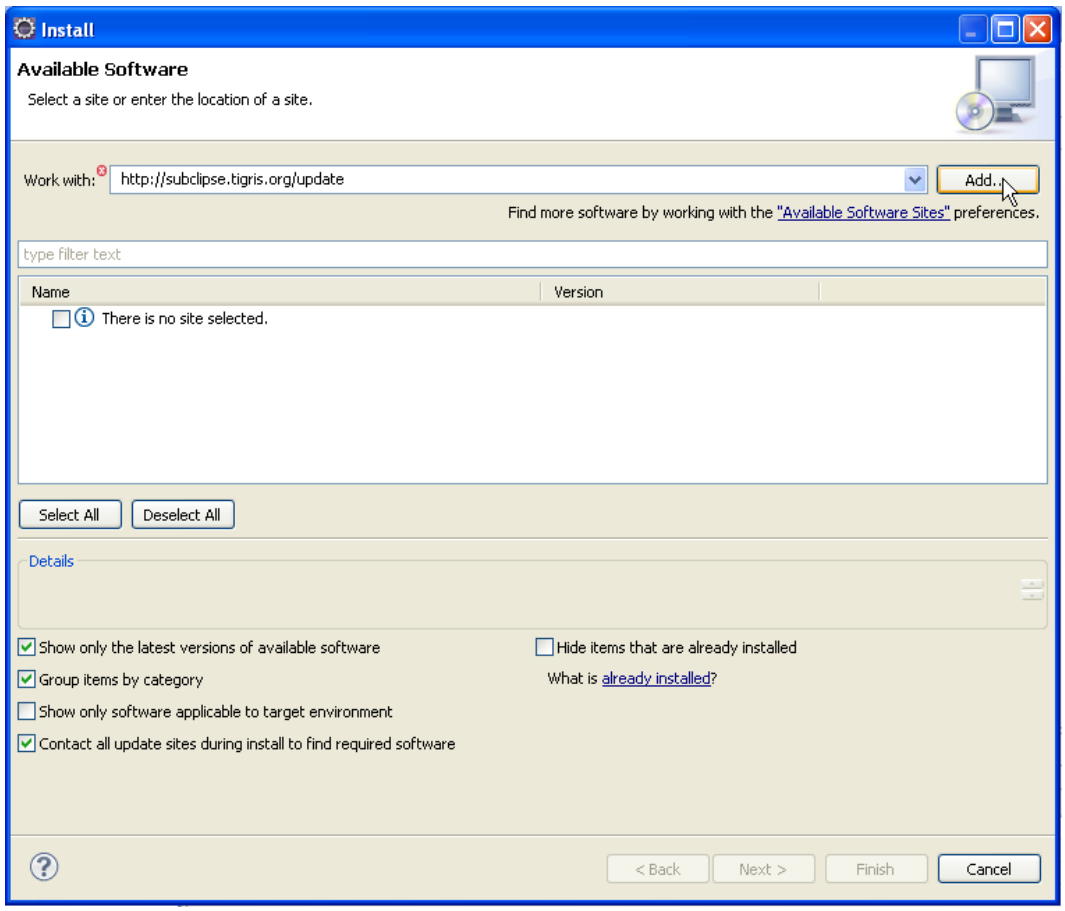
3. Ejecutamos Eclipse haciendo doble click sobre el ejecutable dentro de la carpeta eclipse:



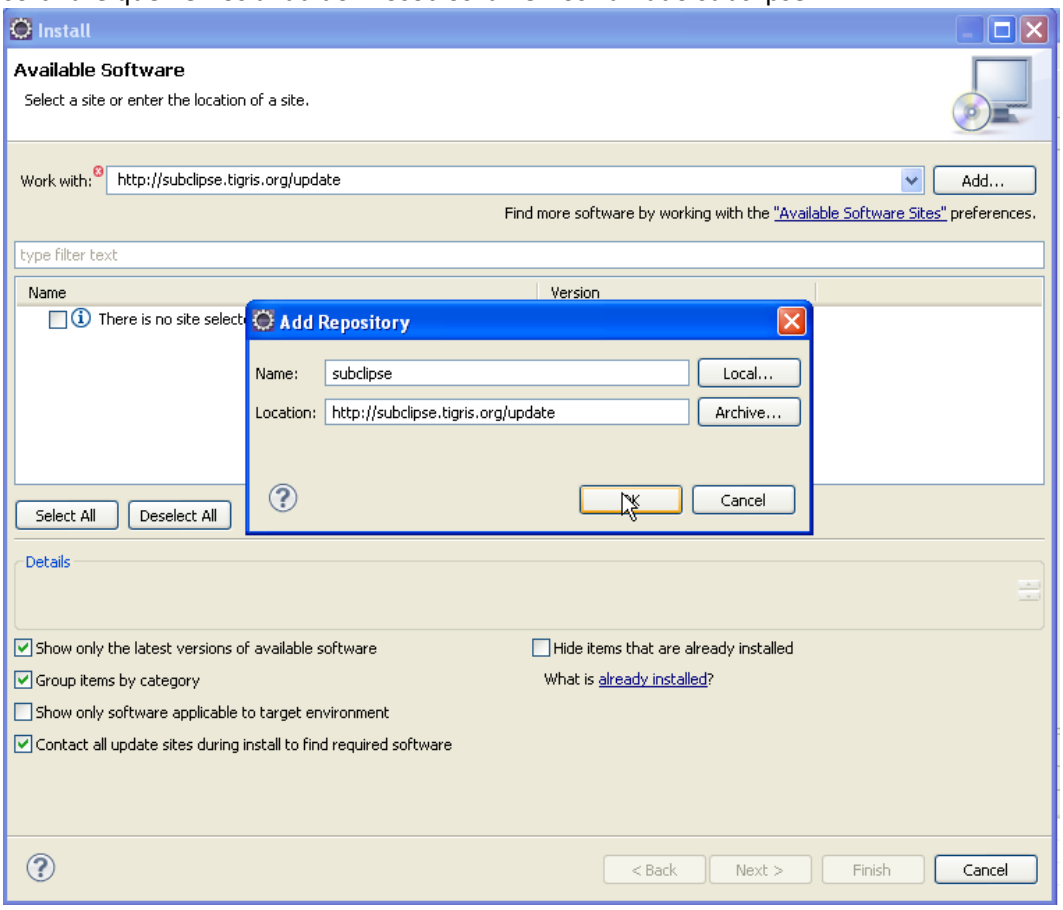
4. Seleccionamos el siguiente workspace:



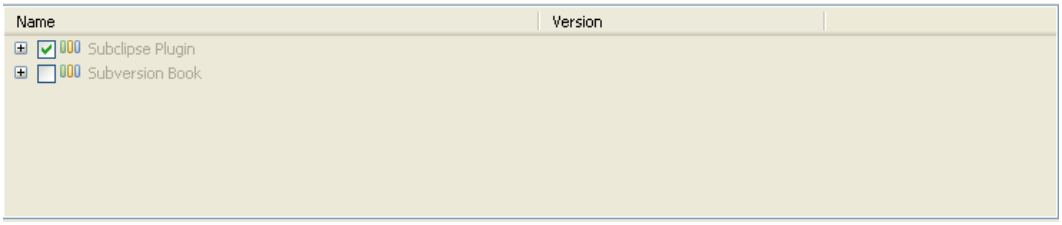
5. Instalamos el plugin subclipse para descargar el código fuente del proyecto de repositorio SVN, pulsando en ***File -> Install new Software*** y añadimos la url de descarga http://subclipse.tigris.org/update en el campo ***Work with***.



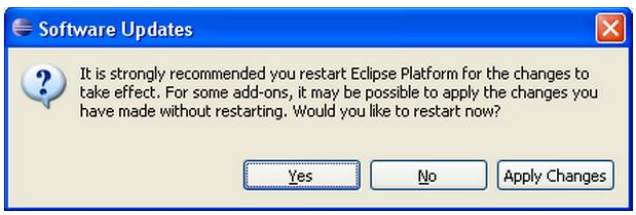
6. Al pulsar sobre el botón ***Add*** deberemos indicar un nombre para la nueva dirección de software que hemos añadido. Nosotros la hemos llamado subclipse.



7. Una vez el centro de software de Eclipse se ha conectado al servidor de aplicaciones de subclipse, nos mostrará un cuadro en pantalla con las aplicaciones disponibles para instalar. Seleccionaremos la aplicación ***Subeclipse Plugin*.**

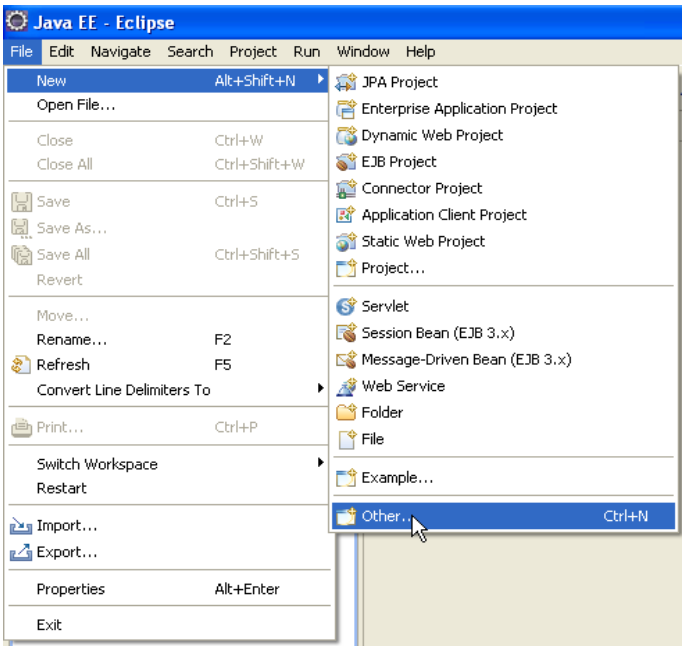


8. Una vez terminado el paso anterior, pulsaremos sobre el botón Finish y se instalará el ***plugin Subeclipse***.
9. Terminado el proceso de instalación, eclipse nos pedirá que lo reiniciemos. Pulsaremos sobre el botón ***yes***, y esperaremos a que se finalice la operación.

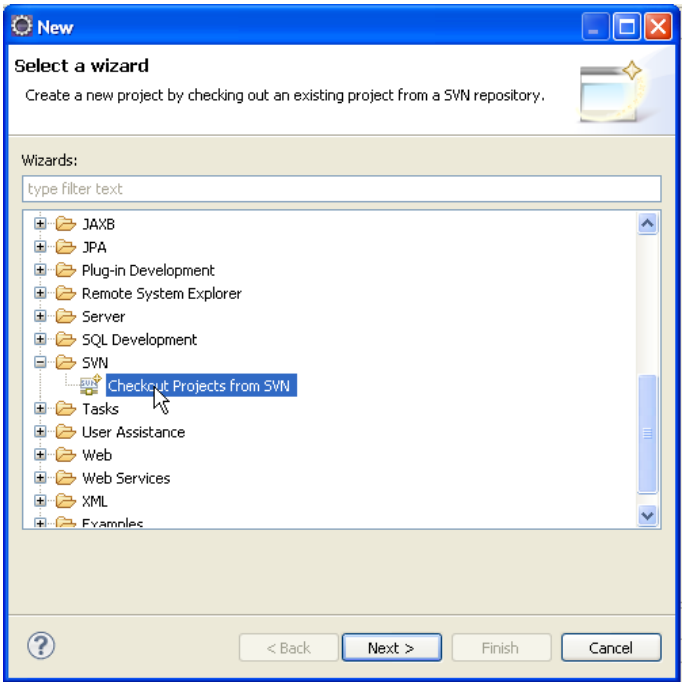


Instalación de la aplicación

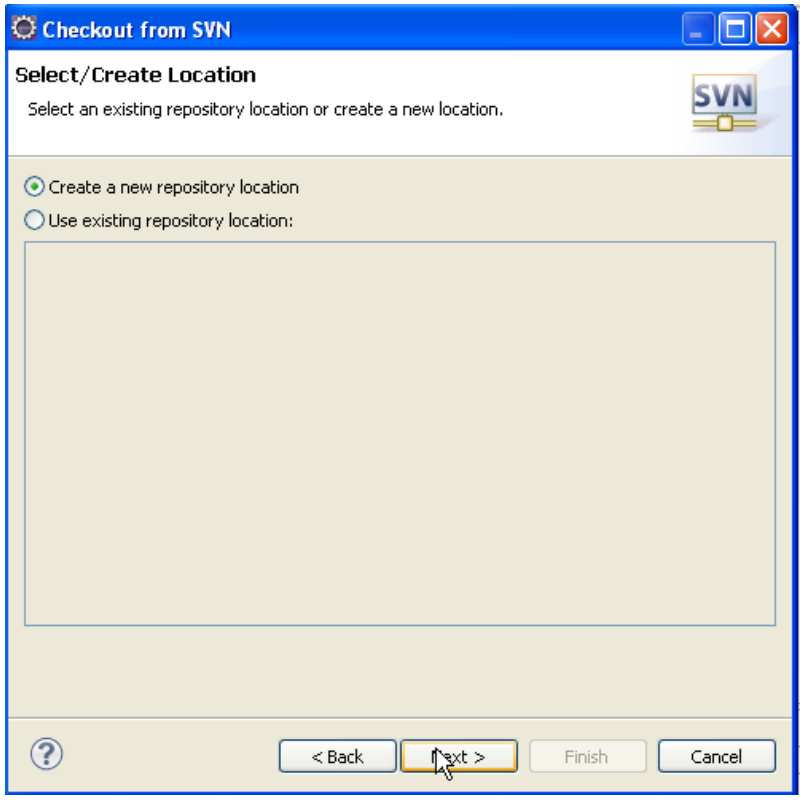
1. Ejecutaremos el entorno eclipse, seleccionando como workspace ***C:\workspace***
2. Crearemos 6 proyectos siguiendo estos pasos:
   Seleccionamos crear un nuevo proyecto como se muestra en la imagen:



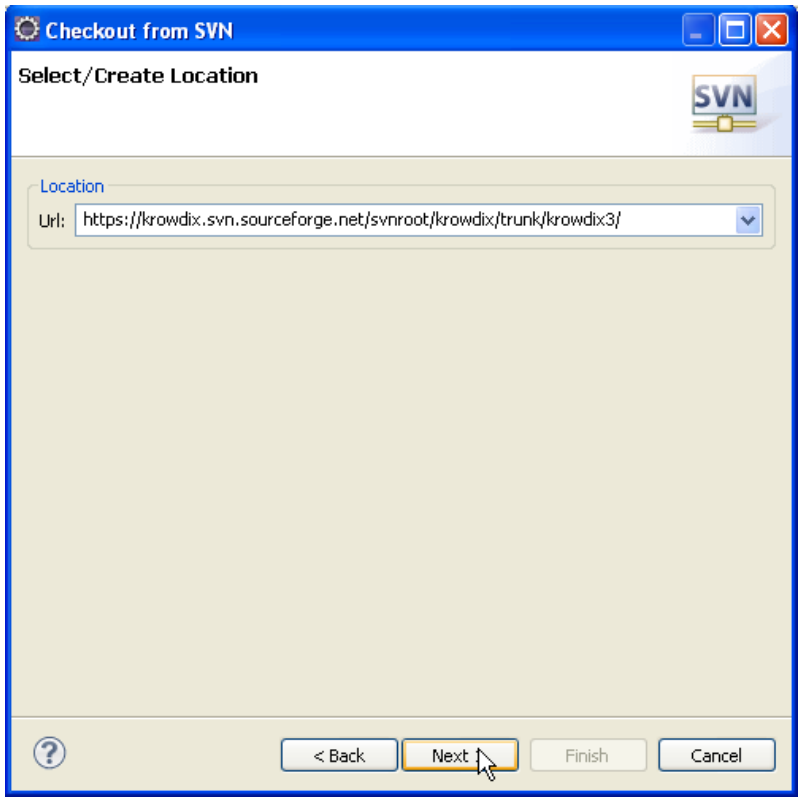
Seleccionamos en el siguiente menú crear un proyecto de SVN como se muestra en la imagen y pulsaremos sobre ***Next****:*



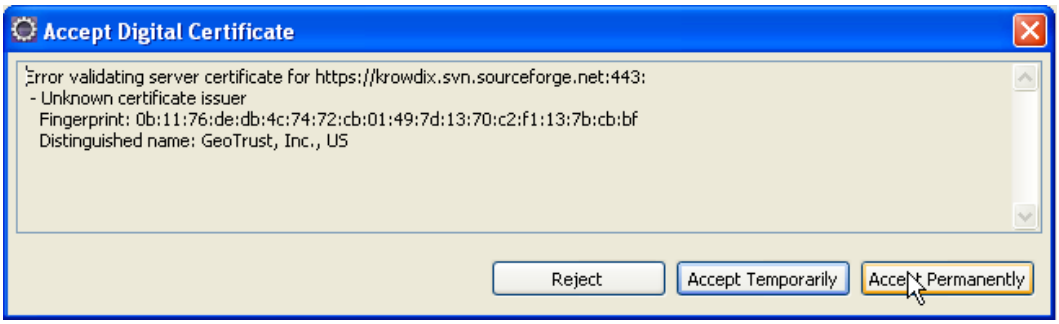
A continuación seleccionamos crear una nueva dirección de repositorio SVN



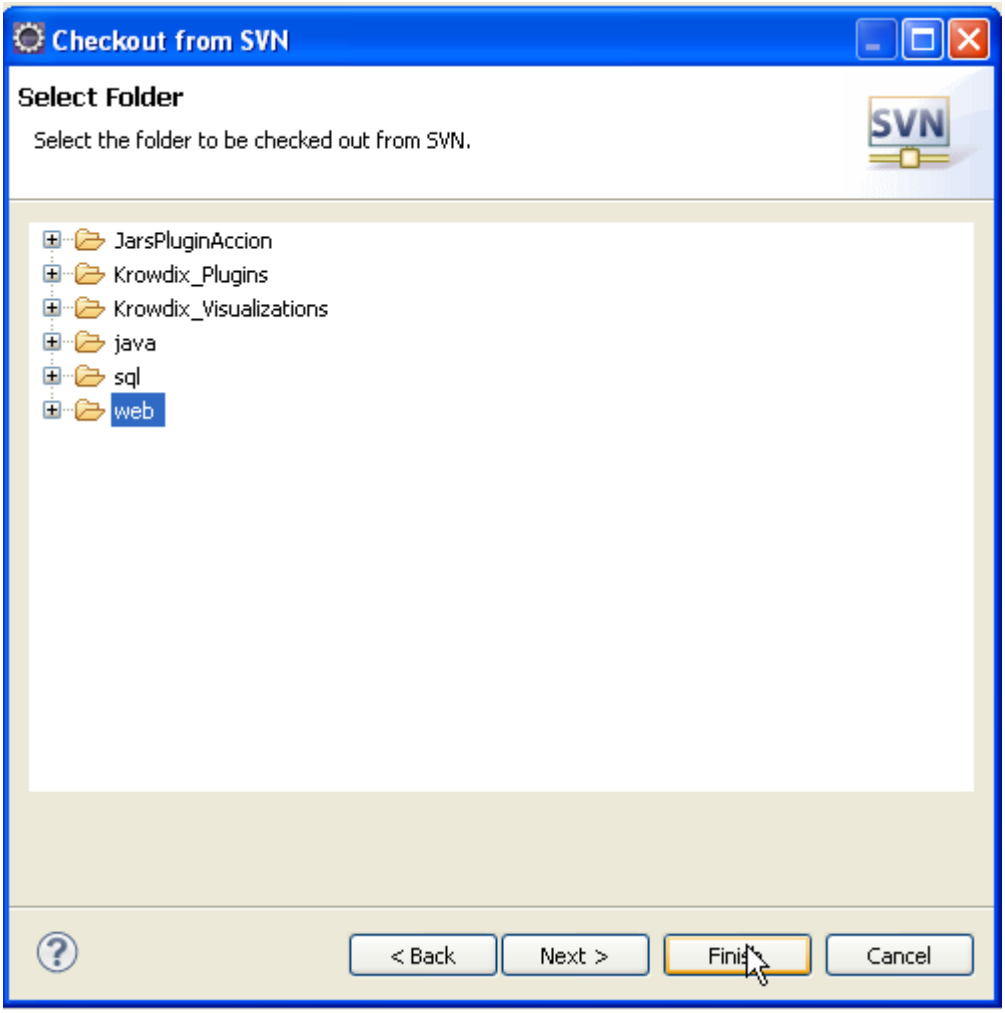
*Insertamos la ruta de la imagen siguiente y pulsamos sobre el botón **Next***



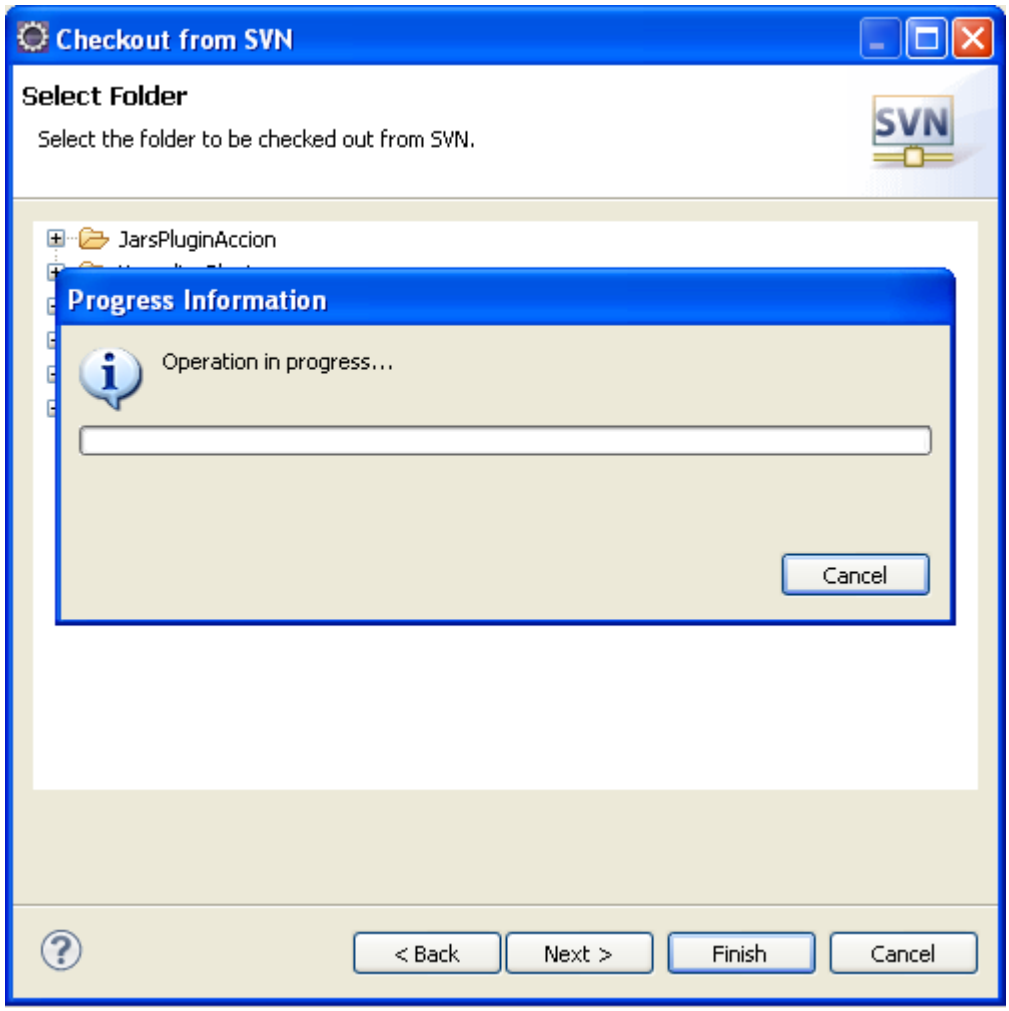
La primera vez que realizamos los pasos anteriores, se nos pedirá aceptar el certificado electrónico del servidor SVN. Pulsaremos sobre ***Accept Permanently*** *para evitar repetir este paso en la creación del resto de proyectos.*



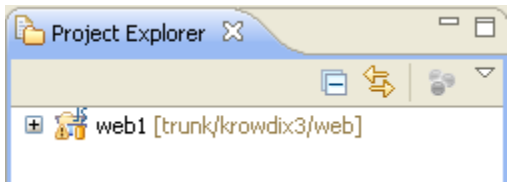
En la siguiente ventana, seleccionaremos la carpeta web y pulsaremos sobre el botón ***Finish.***



Una vez realizados los pasos anteriores, Eclipse descargará los ficheros del servidor SVN.



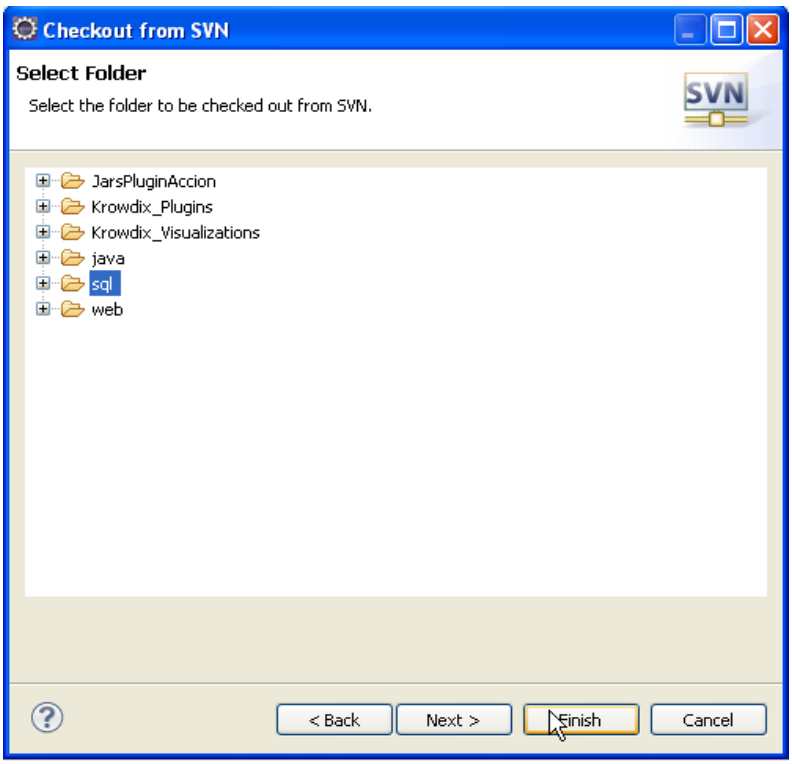
Al terminar de descargar los ficheros del servidor, creará un nuevo proyecto en nuestro workspace llamado web1.



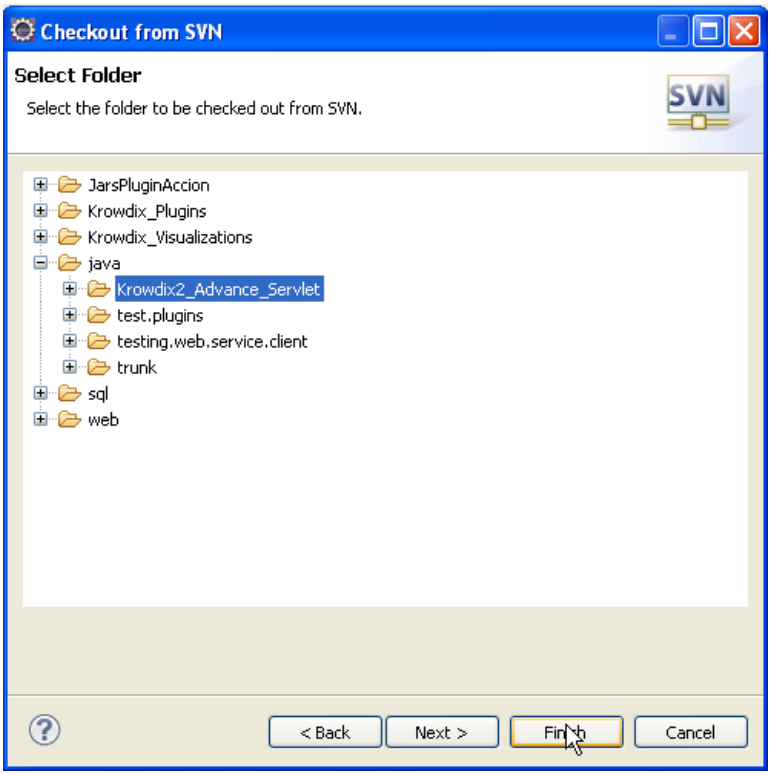
En este punto, hemos terminado de crear el proyecto web1 con la parte web de nuestro proyecto en Eclipse.

Una vez realizados los pasos anteriores, crearemos del mismo modo que el proyecto anterior, 4 nuevos proyectos seleccionando como rutas del servidor SVN las mostradas en las 4 imágenes siguientes.

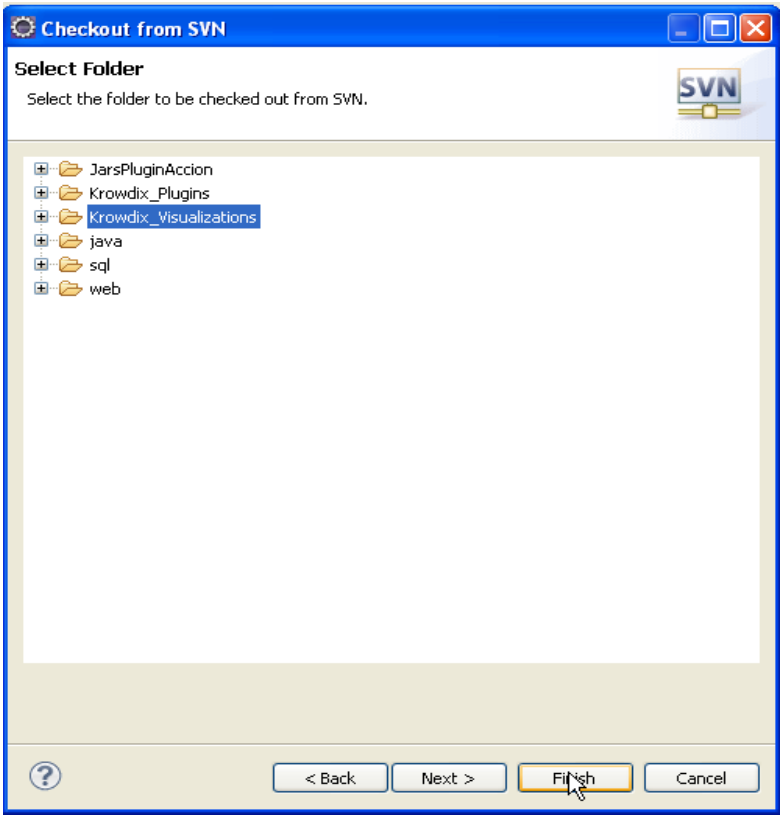
Para proyecto de scripts de base de datos.



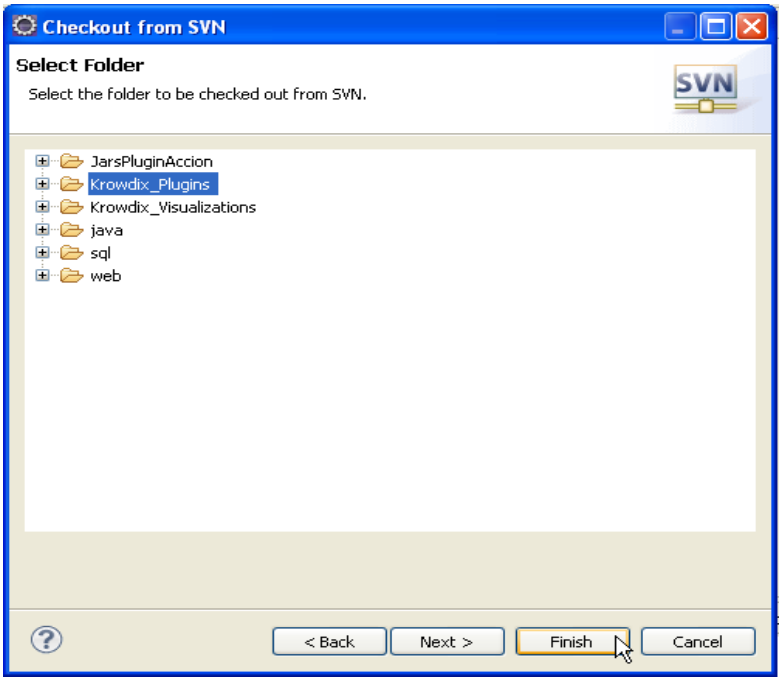
Para proyecto de código java.



Para proyecto de código de generación de las visualizaciones.

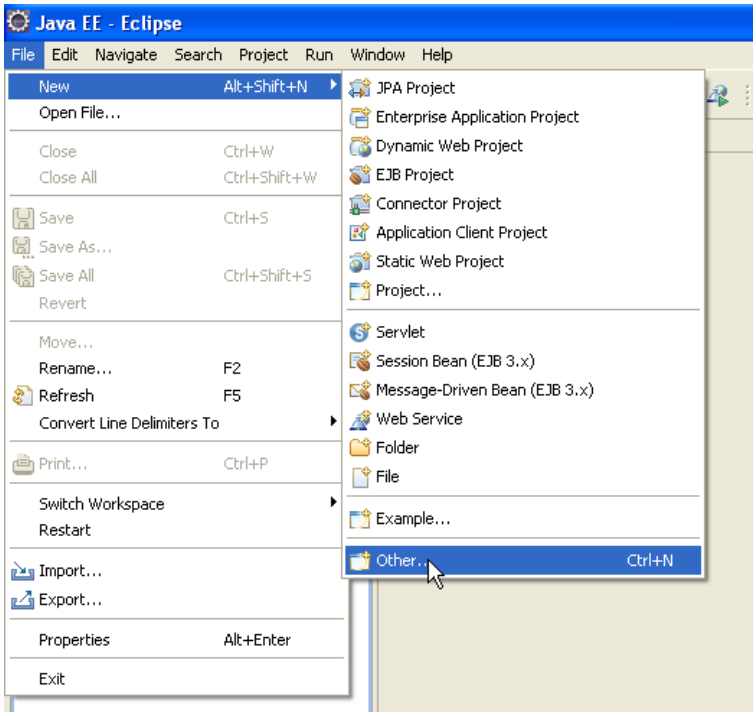


Para proyecto de código de los plugins.

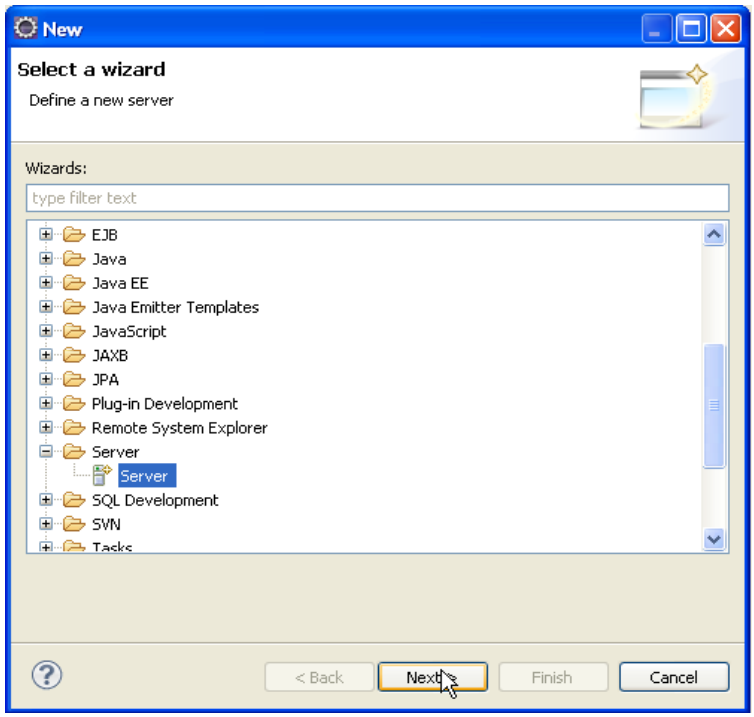


Para la correcta compilación de los proyectos anteriores, crearemos un nuevo proyecto de servidor como explicamos a continuación.

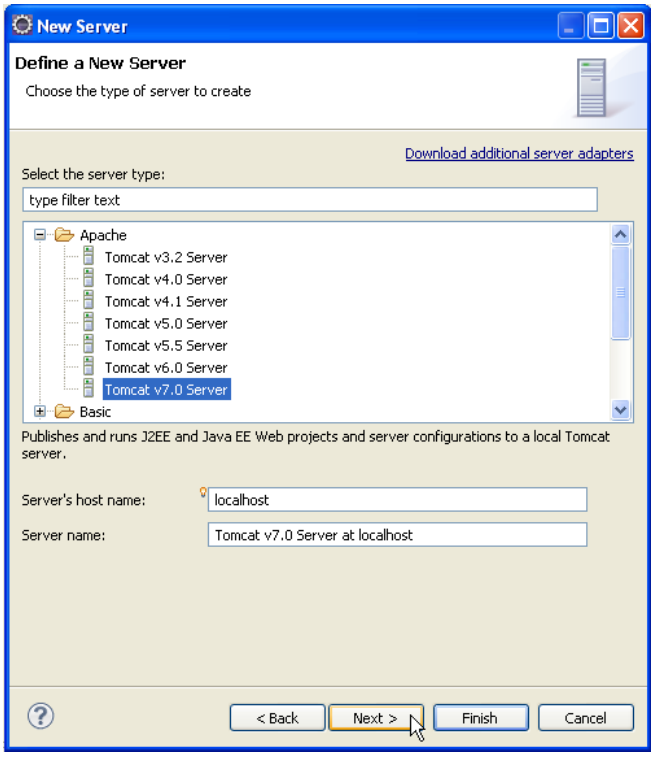
Seleccionamos crear un nuevo proyecto como se muestra en la imagen



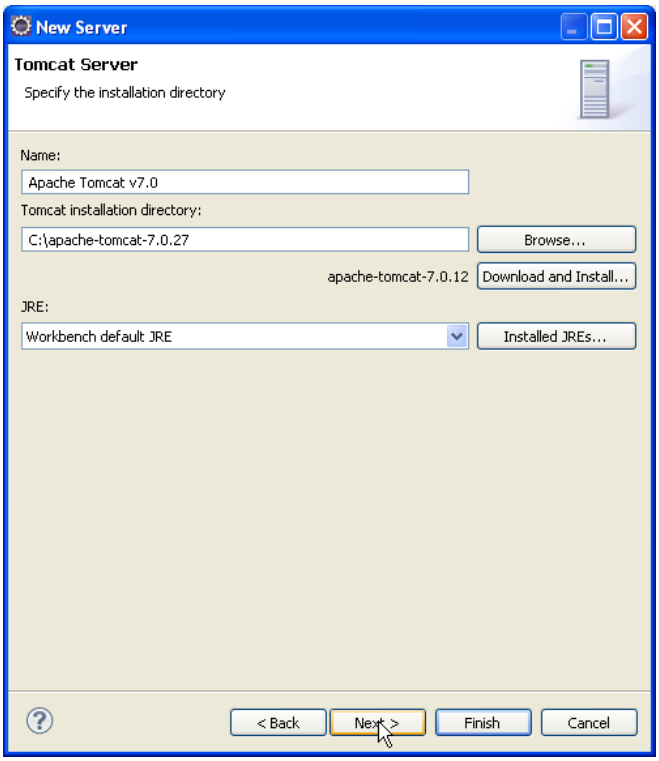
Seleccionamos crear un nuevo proyecto de servidor como se muestra en la imagen y pulsamos sobre el boton ***Next.***



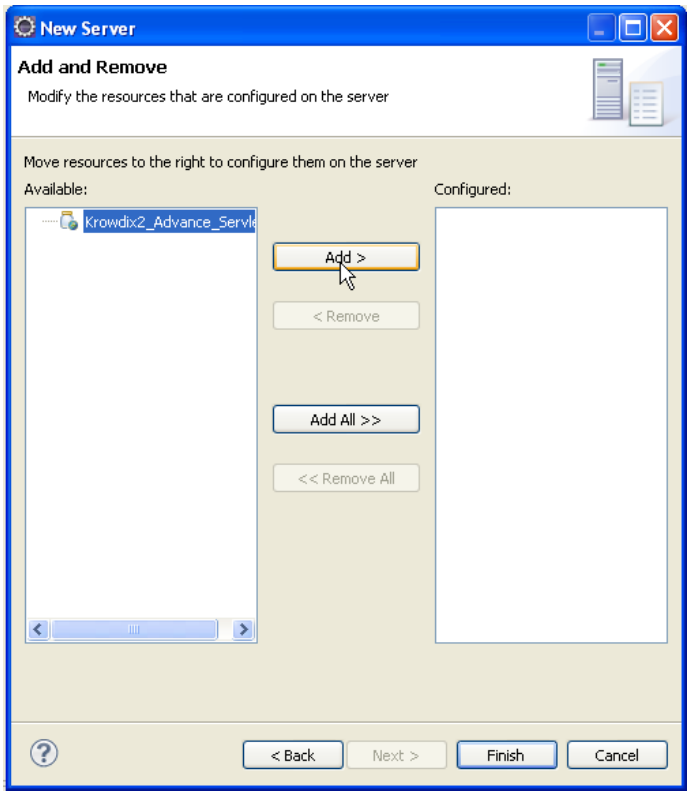
En la siguiente ventana seleccionamos con el ratón la opcion Tomcat v7.0 Server e insertamos los parámetros mostrados en la imagen en todos los campos. A continuación pulsamos sobre el boton ***Next****.*



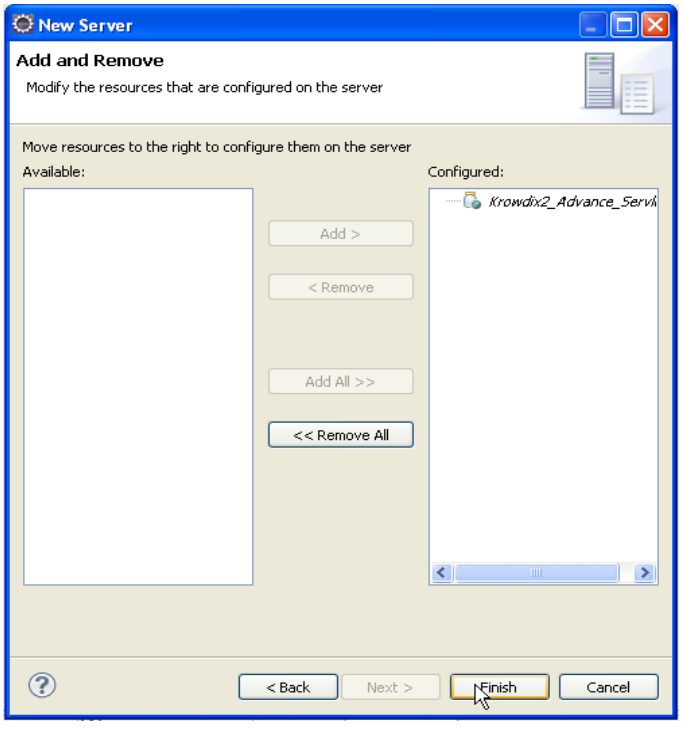
En la siguiente ventana, seleccionaremos los parámetros mostrados y pulsamos sobre **Next.**



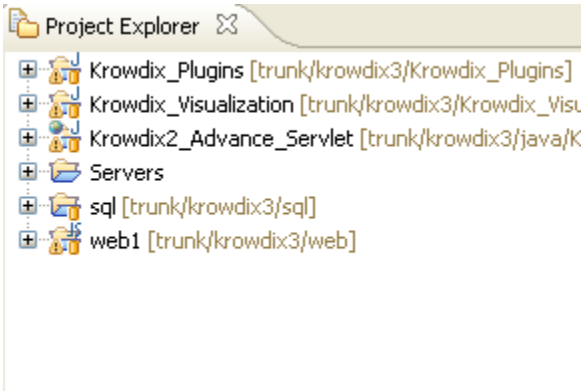
*A continuación en la ventana siguiente, seleccionaremos en el recuadro izquierdo la opción mostrada en la imagen y pulsaremos sobre el boton **Add**.*



*Por último, pulsaremos sobre el botón **Finish**.*



Realizados los pasos anteriores, encontraremos creados en nuestro workspace los proyectos de Eclipse mostrados en la imagen.



7.3 Configuración

Una vez tenemos creados los anteriores proyectos en eclipse vamos a detallar los archivos de configuración que deberemos ajustar para que nuestra aplicación funcione correctamente en nuestro computador servidor.
Dado que la instalación aquí descrita es sobre un entorno Windows con xampp y tomcat, algunos de los pasos a realizar pueden cambiar en otros entornos Linux o Mac. Cuando configuremos un parámetro independiente del entorno, lo indicaremos con una nota, para que cualquier usuario pueda configurar nuestra aplicación en cualquier sistema.

7.3.1 Configuración de la Base de Datos

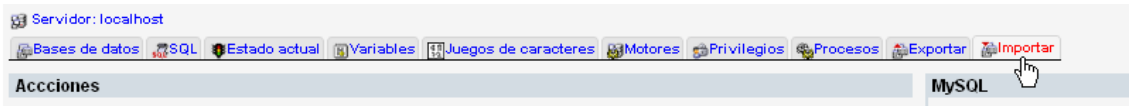
Nuestra aplicación utiliza una base de datos MySQL para su funcionamiento. Los scripts necesarios para su creación, los encontramos en el proyecto de eclipse llamado sql, creado en el apartado anterior.
Para ejecutar los scripts sql seguiremos los siguientes pasos:

1) Abrimos el navegador Google Chrome
2) Insertamos en la barra de direcciones la siguiente url

3) Configuramos el idioma deseado. En nuestro caso hemos seleccionado el idioma Español

4) Seleccionamos la aplicación a la izquierda llamada phpMyAdmin

5) Importamos los scripts de bases de datos seleccionando la opción import y navegando por el explorador de archivos de phpmyadmin hasta c:\workspace\sql\



6) El orden de importación de los scripts de bases de datos será, en primer lugar, crearBaseDatos_v3.sql y en segundo lugar, cargarBaseDatos_V3.sql.

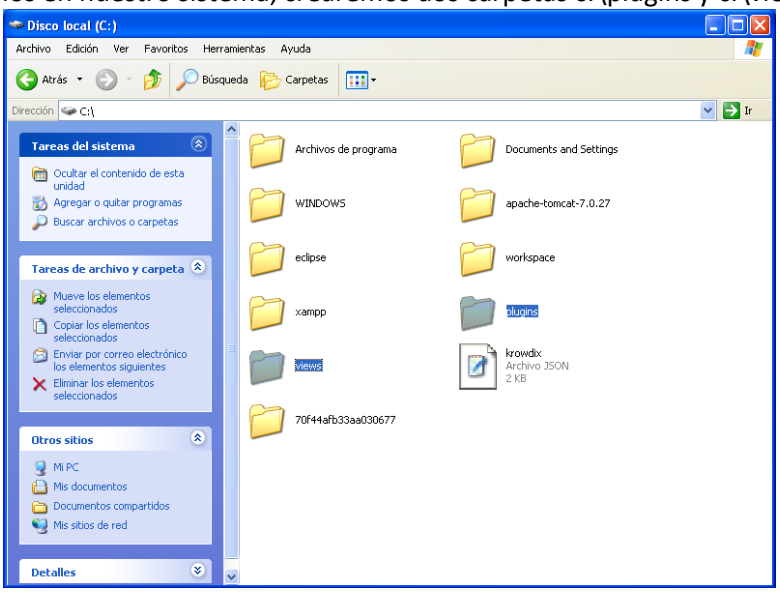
Nota: Podemos usar como alternativa el cliente de bases de datos MySQL Workbench, multiplataforma y para el que facilitamos un archivo de esquema de base de datos llamado MySQLDVE que podemos encontrar en el proyecto sql de eclipse donde se encuentran los scripts de la base de datos MySQL.

7.3.2 Configuración de Views y Plugins de nuestro programa

Dado que nuestro programa pretende ser una herramienta extensible a las necesidades del usuario, algunas de sus funcionalidades son importadas dinámicamente desde ficheros jar ubicados en carpetas independientes al resto del sistema. En este apartado explicaremos como organizar estos ficheros para que nuestro sistema sea capaz de reconocerlos en tiempo de ejecución.

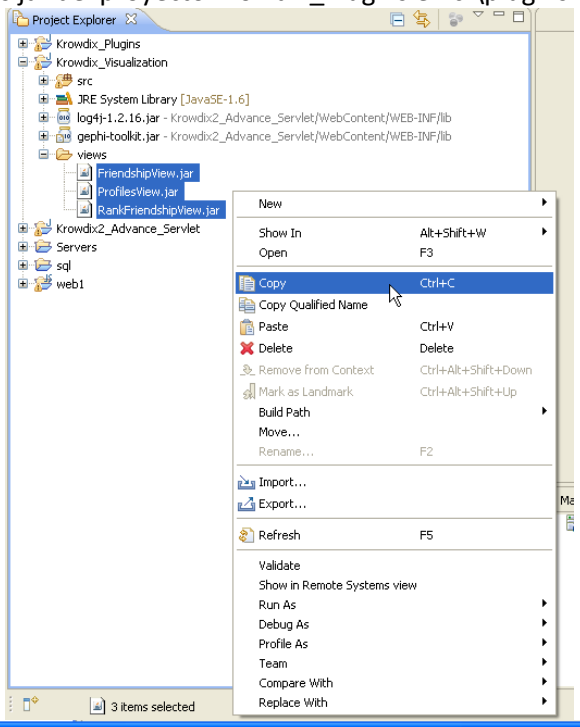
En los proyectos de Eclipse Krowdix_Visualizations y Krowdix_Plugins encontramos las carpetas views y pluginsFinales que contienen los respectivos ficheros jar tanto de visualizaciones como de plugins.

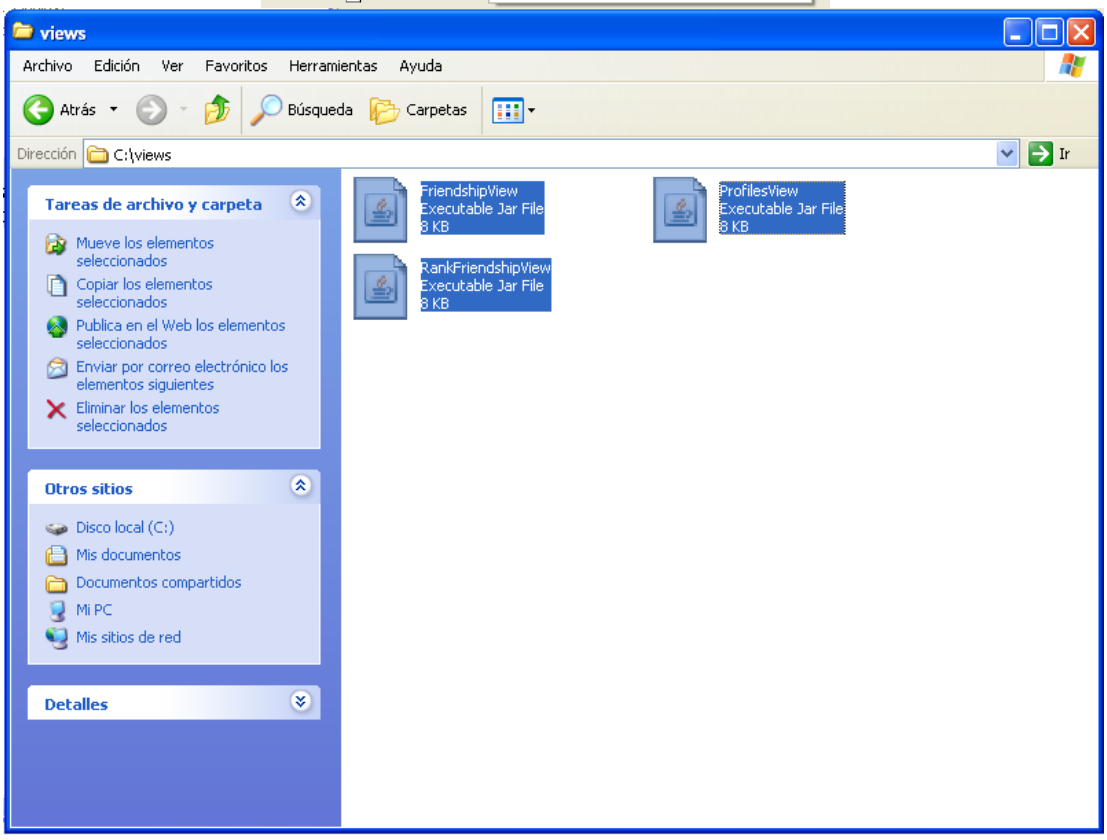
Para integrarlos en nuestro sistema, crearemos dos carpetas c:\plugins y c:\views.

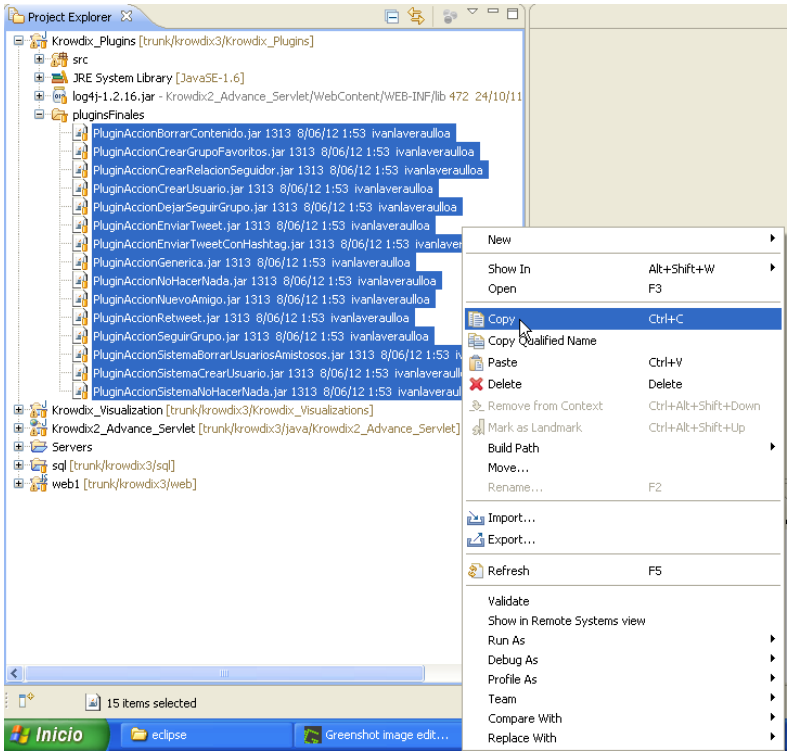


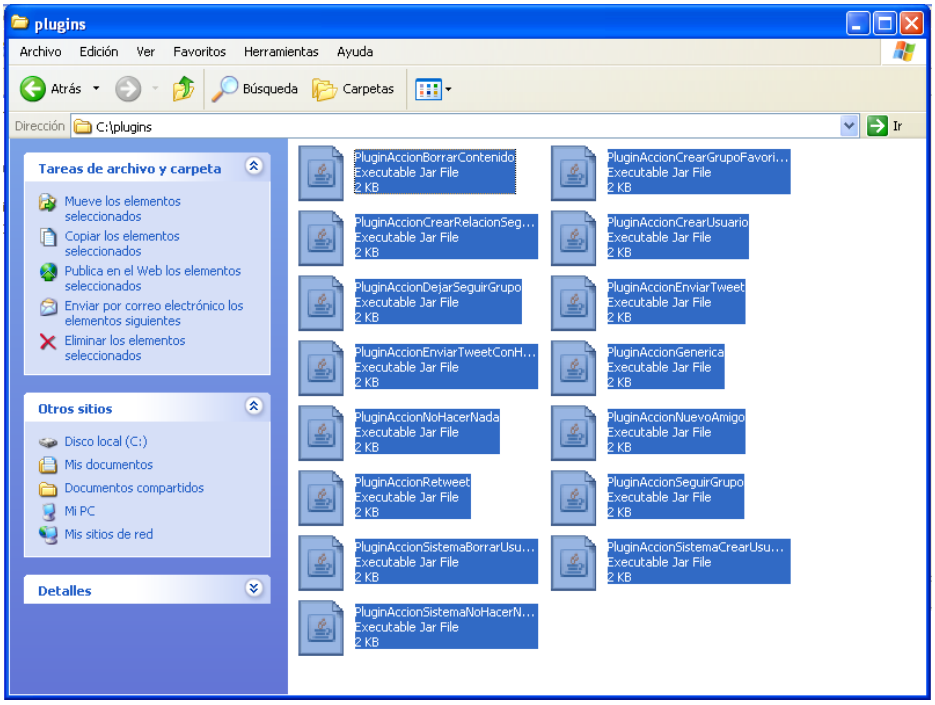
Nota: Podemos ubicar estos ficheros en rutas diferentes independientemente del sistema operativo siempre y cuando, editemos correctamente el fichero krowdix.json como veremos en un apartado más adelante.

A continuación, copiaremos los archivos jar del proyecto Krowdix_Visualizations en c:\views y los archivos jar del proyecto Krowdix_Plugins en c:\plugins



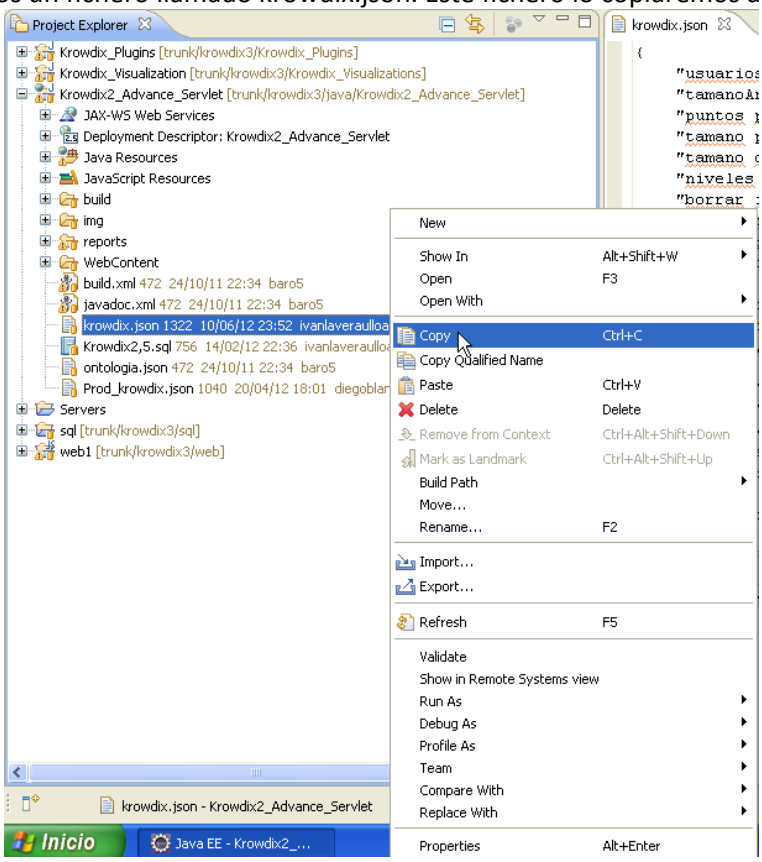


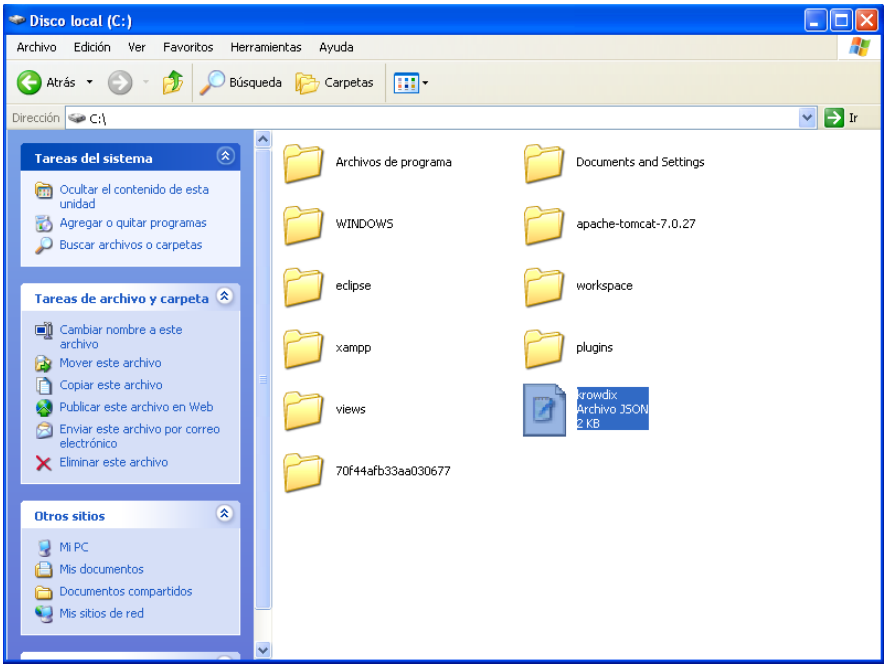

Project Explorer
Krowdix_Plugins [trunk/krowdix3/Krowdix_Plugins]
src
JRE System Library [JavaSE-1.6]
log4j-1.2.16.jar - Krowdix2_Advance_Servlet/WebContent/WEB-INF/lib 472 24/10/11
pluginsFinales
PluginAccionBorrarContenido.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionCrearGrupoFavoritos.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionCrearRelacionSeguidor.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionCrearUsuario.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionDejarSeguirGrupo.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionEnviarTweet.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionEnviarTweetConHashtag.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionGenerica.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionNoHacerNada.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionNuevoAmigo.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionRetweet.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionSeguirGrupo.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionSistemaBorrarUsuariosAmistosos.jar 1313 8/06/12 1:53
PluginAccionSistemaCrearUsuario.jar 1313 8/06/12 1:53 ivanlaveraulloa
PluginAccionSistemaNoHacerNada.jar 1313 8/06/12 1:53 ivanlaveraulloa
Krowdix_Visualization [trunk/krowdix3/Krowdix_Visualizations]
Krowdix2_Advance_Servlet [trunk/krowdix3/java/Krowdix2_Advance_Servlet]
Servers
sql [trunk/krowdix3/sql]
web1 [trunk/krowdix3/web]
New
Show In Alt+Shift+W
Open F3
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Mark as Landmark Ctrl+Alt+Shift+Up
Build Path
Move...
Rename... F2
Import...
Export...
Refresh F5
Validate
Show in Remote Systems view
Run As
Debug As
Profile As
Team
Compare With
Replace With
15 items selected
Inicio
eclipse
Greenshot image edit...


plugins
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás
Búsqueda
Carpetas
Dirección C:\plugins
Ir
Tareas de archivo y carpeta
Mueve los elementos seleccionados
Copiar los elementos seleccionados
Publica en el Web los elementos seleccionados
Enviar por correo electrónico los elementos siguientes
Eliminar los elementos seleccionados
Otros sitios
Disco local (C:)
Mis documentos
Documentos compartidos
Mi PC
Mis sitios de red
Detalles
PluginAccionBorrarContenido Executable Jar File 2 KB
PluginAccionCrearGrupoFavori... Executable Jar File 2 KB
PluginAccionCrearRelacionSeg... Executable Jar File 2 KB
PluginAccionCrearUsuario Executable Jar File 2 KB
PluginAccionDejarSeguirGrupo Executable Jar File 2 KB
PluginAccionEnviarTweet Executable Jar File 2 KB
PluginAccionEnviarTweetConH... Executable Jar File 2 KB
PluginAccionGenerica Executable Jar File 2 KB
PluginAccionNoHacerNada Executable Jar File 2 KB
PluginAccionNuevoAmigo Executable Jar File 2 KB
PluginAccionRetweet Executable Jar File 2 KB
PluginAccionSeguirGrupo Executable Jar File 2 KB
PluginAccionSistemaBorrarUsu... Executable Jar File 2 KB
PluginAccionSistemaCrearUsu... Executable Jar File 2 KB
PluginAccionSistemaNoHacerN... Executable Jar File 2 KB

7.3.3 Edición del fichero krowdix.json

En el proyecto de Eclipse llamado Krowdix2_Advance_Servlet que creamos anteriormente, encontraremos un fichero llamado krowdix.json. Este fichero lo copiaremos a la ruta c:\





Una vez copiado, lo editaremos y modificaremos los campos siguientes como se muestra en la imagen.

```
"URL":"jdbc:mysql://localhost:3306/krowdixDVE",
"USR":"root",
"PWD":"",

"ViewsScreenshots":"C:\\workspace\\web1\\galeriaFotos\\img\\",
"Plugins":"c:/plugins/",
"Views":"c:/views/",
```

Los tres primeros campos URL, USR y PWD hacen referencia a la ruta desde la que tenemos acceso a la base de datos, el usuario y la contraseña de acceso respectivamente.
Los campos Plugins y Views hacen referencia a las rutas donde hemos alojado los .jar copiados en el punto anterior.

El campo ViewScreenshots hace referencia a la ruta donde generaremos las imágenes de visualización de la red social. Siempre en este campo colocaremos la ruta completa para acceder a la carpeta /galeríaFotos/img del lugar donde alojemos el servidor web y la web de nuestro sistema. En un apartado posterior explicaremos como instalar la parte web de nuestro sistema.

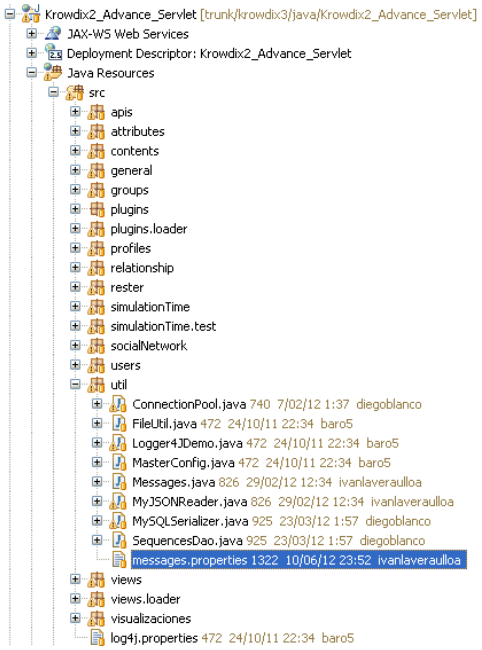
Nota: Los valores aquí introducidos son los valores que tomamos para que siguiendo esta guía se pueda ejecutar el proyecto. Deberemos adaptarlos al entorno de ejecución de la máquina en la que se ejecute nuestra aplicación.

Nota: El fichero krowdix.json lo hemos copiado en la ruta C:\, pero dependerá del entorno de la máquina en el que se ejecute nuestra aplicación. En un apartado posterior explicaremos como configurar la ruta de la ubicación de este archivo.

7.3.4 Configuración motor java de simulación

En este punto vamos a detallar como instalar en el servidor Tomcat, nuestro motor java de simulación de redes sociales.

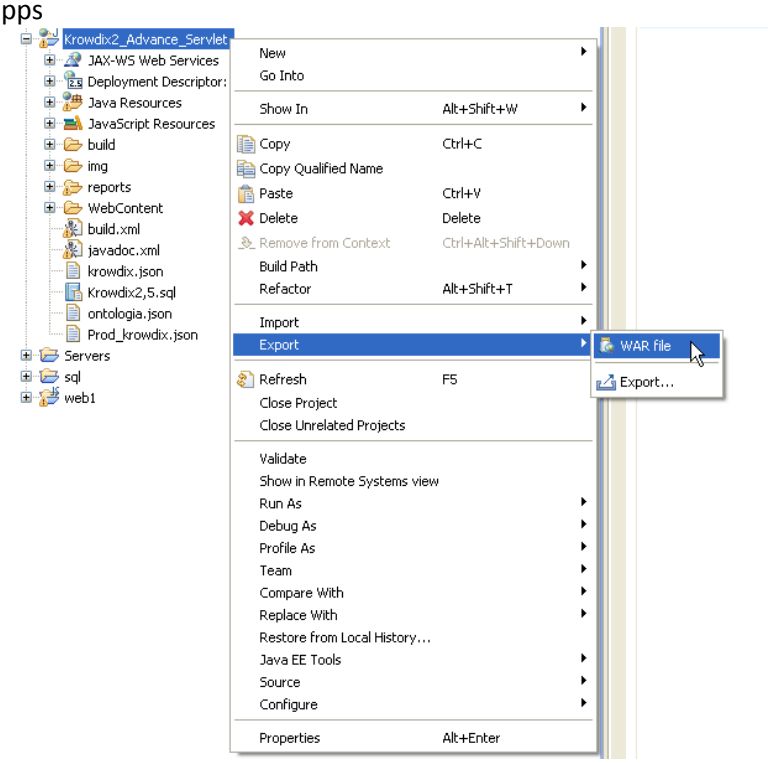
Para ello, editaremos el fichero mostrado en la imagen del proyecto Krowdix2_Advance_Servlet haciendo doble click sobre él desde Eclipse.

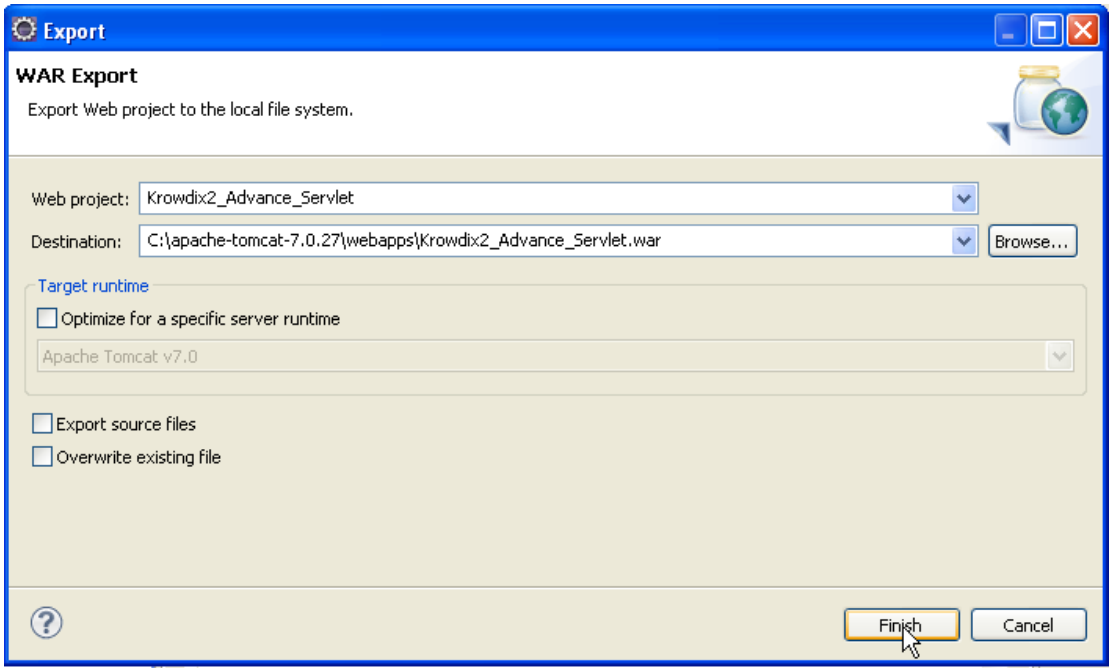


Insertaremos el contenido de la imagen

Nota: Es la ruta en la que se encuentra el fichero krowdix.json comentado en el punto anterior. La ruta dependerá del entorno de ejecución del computador donde ejecutemos nuestra aplicación.

Llegado a este punto, solo queda exportar a un fichero .war el proyecto Krowdix2_Advance_Servlet e instalarlo en nuestro servidor de aplicaciones Tomcat.
Exportamos al fichero .war como se muestra en las imágenes a la ruta C:\apache-tomcat-7.0.27\webapps

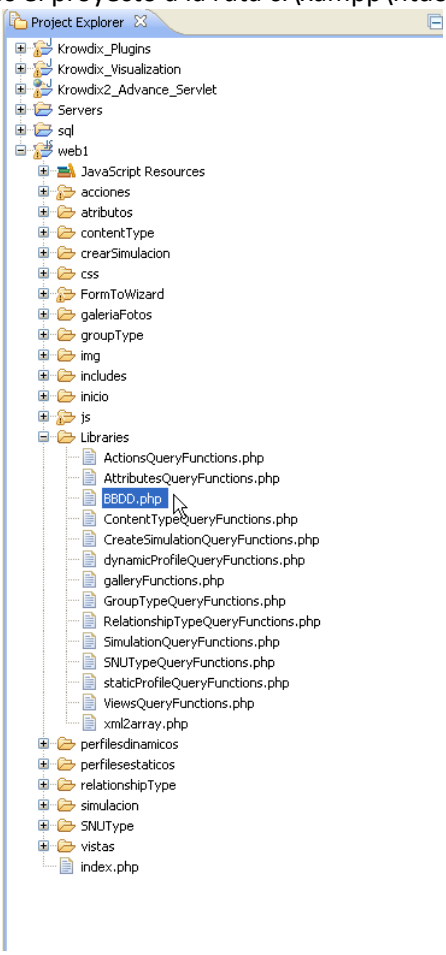




Nota: La ruta C:\apache-tomcat-7.0.27\webapps es dependiente del entorno de instalación con el que estamos elaborando la instalación. Deberemos copiar el fichero .war a la carpeta webapps del directorio de la máquina donde tengamos instalado el servidor de aplicaciones Tomcat.

7.3.5 Configuración de la parte web

En el proyecto anteriormente creado en Eclipse llamado web1, encontramos todos los ficheros que componen la parte web de nuestro proyecto. Para instalar esta parte en nuestra máquina, editaremos primero el fichero BBDD.php mostrado en la imagen y a continuación, exportaremos el proyecto a la ruta c:\xampp\htdocs.



Los valores a introducir en el fichero BBDD.php son los siguientes:
Puerto de escucha del servidor Tomcat(por defecto 8080)

```
function getPortTomcat()
{
    $portTomcat="8080";
    return $portTomcat;
}
```

URL de acceso servidor Tomcat (localhost, ya que lo ejecutamos en nuestra máquina)

```
function getURLServer()
{
    $url="http://localhost";
    return $url;
}
```

ULR de acceso servidor Bases de datos MySQL(localhost, ya que lo ejecutamos en nuestra máquina)

```
function getURLMySQL()
{
  $url="localhost";
  return $url;
}
```

Usuario de la base de datos(root por defecto)

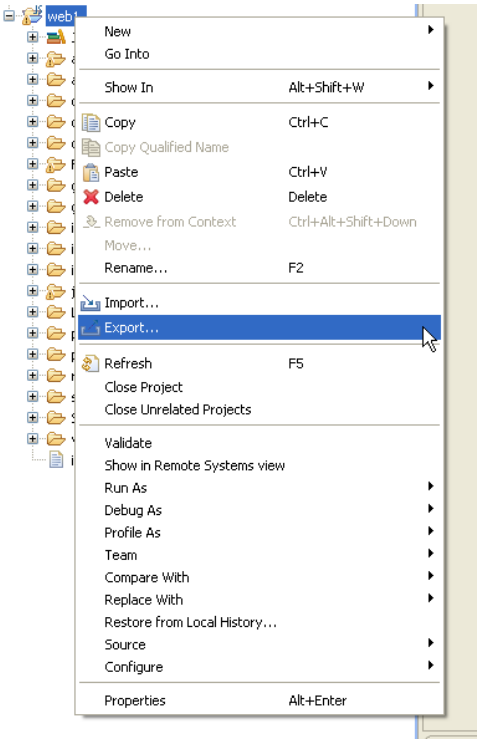
```
function getUserBBDD()
{
 $user="root";
 return $user;
}
```

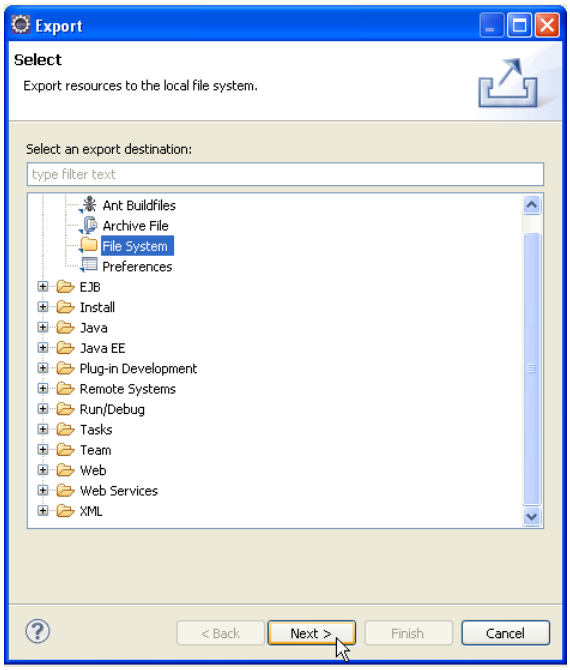
Contraseña de la base de datos(sin contraseña por defecto)

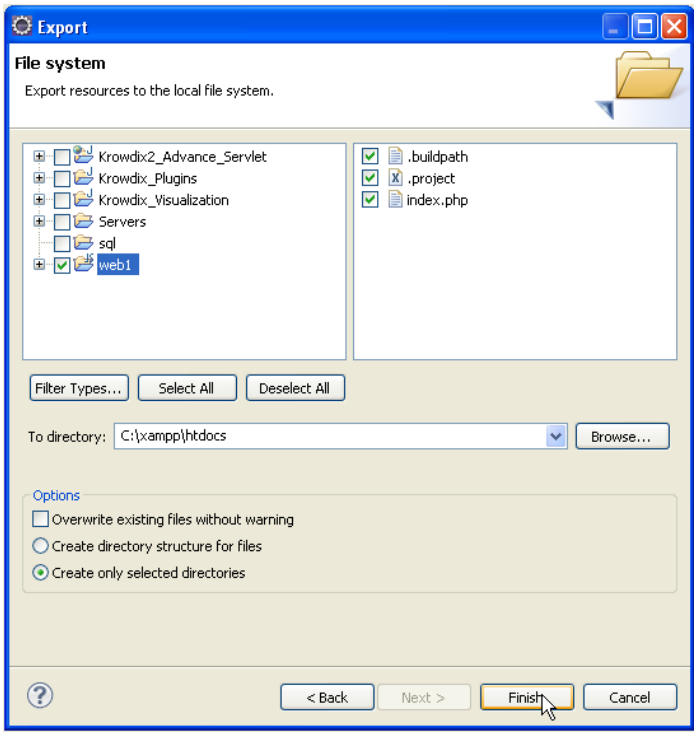
```
function getPassBBDD()
{
 $pass="";
 return $pass;
}
```

Nota: Estos valores anteriores dependerán de la configuración de nuestro sistema. En esta explicación hemos cogido los valores que vienen por defecto en los servidores y la base de datos.

Exportamos el proyecto a la ruta c:\xampp\htdocs siguiendo los pasos siguientes:

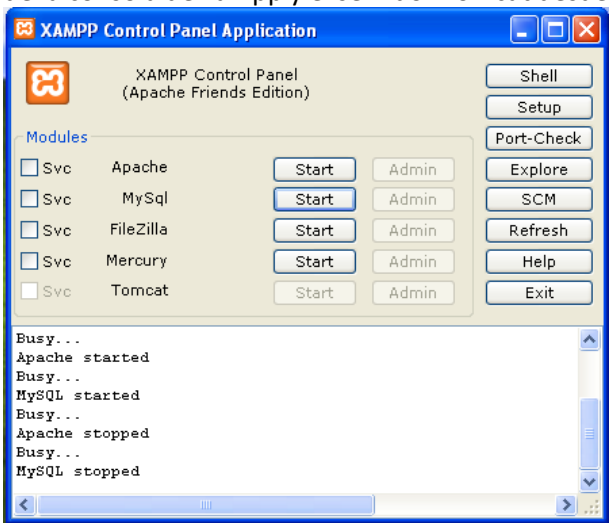


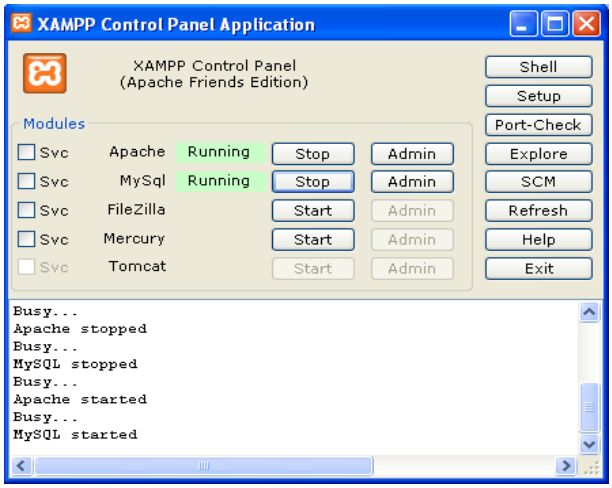


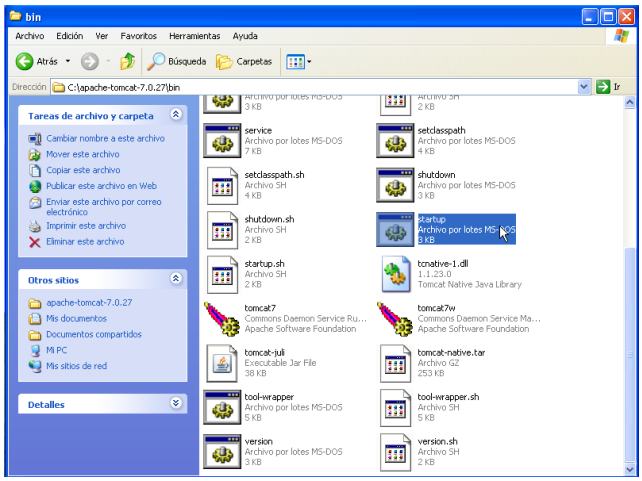


7.3.6 Arranque de la aplicación

Una vez hemos completado todos los apartados anteriores, arrancamos los servidores apache y MySQL desde la consola de xampp y el servidor Tomcat desde su ejecutable.







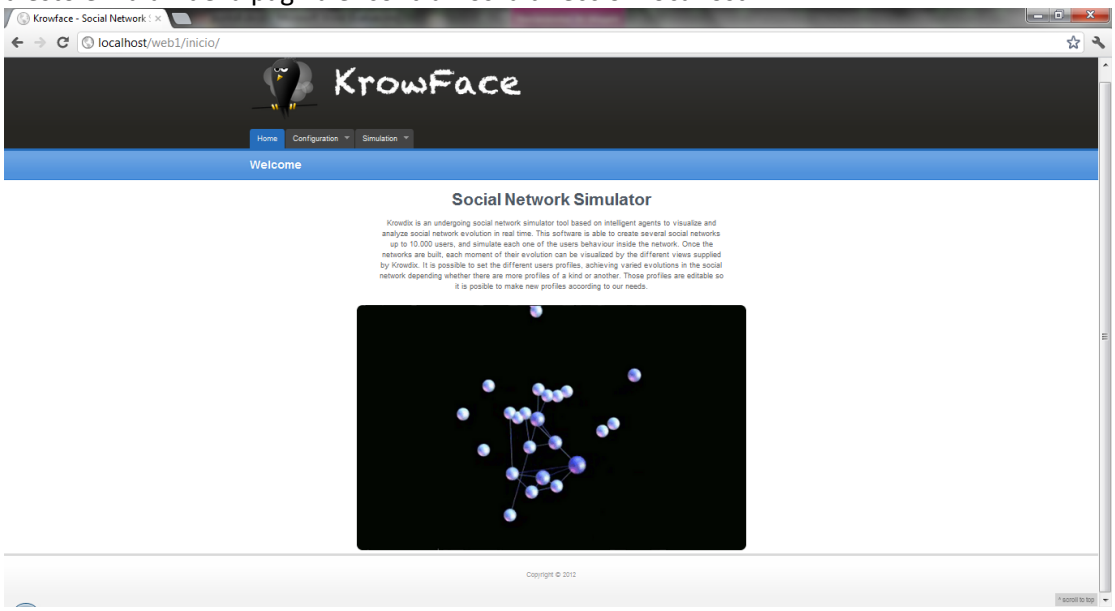
7.4 Configuración Interfaz Web

Para disfrutar de nuestra aplicación ya solo falta abrir el navegador Chrome e insertar la url http://localhost/web1

Nota: La URL anterior dependerá de los parámetros de configuración con los que hayamos configurado la aplicación en los ficheros de configuración descritos anteriormente.

7.4.1 Ventana Principal

Una vez tenemos instalada nuestra aplicación accedemos a su interfaz web mediante el navegador Google Chrome, insertando la url del servidor donde tenemos instalada la aplicación.
Nosotros en este ejemplo, hemos tomado como servidor y cliente el mismo equipo. Debido a esto en la url de la página encontramos la dirección locahost.



Ventana principal
La página principal de nuestra aplicación muestra una pequeña bienvenida y los menús por los que navegaremos para configurar nuestras simulaciones de redes sociales.
La bienvenida a nuestra aplicación se muestra en el centro de la ventana con una pequeña animación y los menús se presentan en la zona superior, debajo de la cabecera de la página.

Disponemos de tres menús

- El menú Home nos permite volver a la página de bienvenida de nuestra aplicación.
- El menú **Configuration** nos permite acceder a los distintos menús de configuración.
- El menú **Simulation** nos permite acceder a la ventana de creación y lanzamiento de simulaciones.

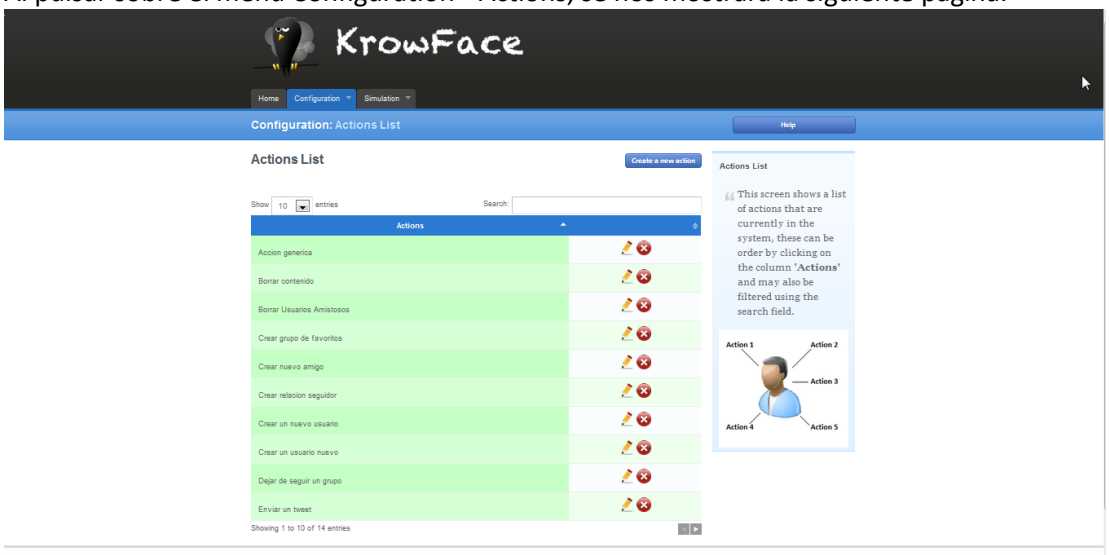
Una vez vamos navegando por los diferentes menus, nos encontraremos un botón arriba a la derecha con el nombre Help, con el cual, al pulsarlo, podremos visualizar un video explicativo de la página en la que nos encontramos y un pequeño ejemplo de uso.

7.4.2 Configuración del sistema

Desde los submenús del menú **Configuration**, podemos crear elementos de simulación que podrán ser asignados en el momento de creación de una nueva simulación.
A continuación mostraremos una pequeña descripción concreta de no de estos menus, concretamente del menú Actions. El resto de submenús están organizados con la misma estructura que el menú Actions, por lo que no será necesario detallar aquí su funcionamiento. Si se tuviese alguna duda de su funcionamiento, podemos acudir al botón Help de la parte superior derecha de la pantalla, en el que encontraremos un video explicativo del menú concreto en el que nos encontramos.

Submenu Actions

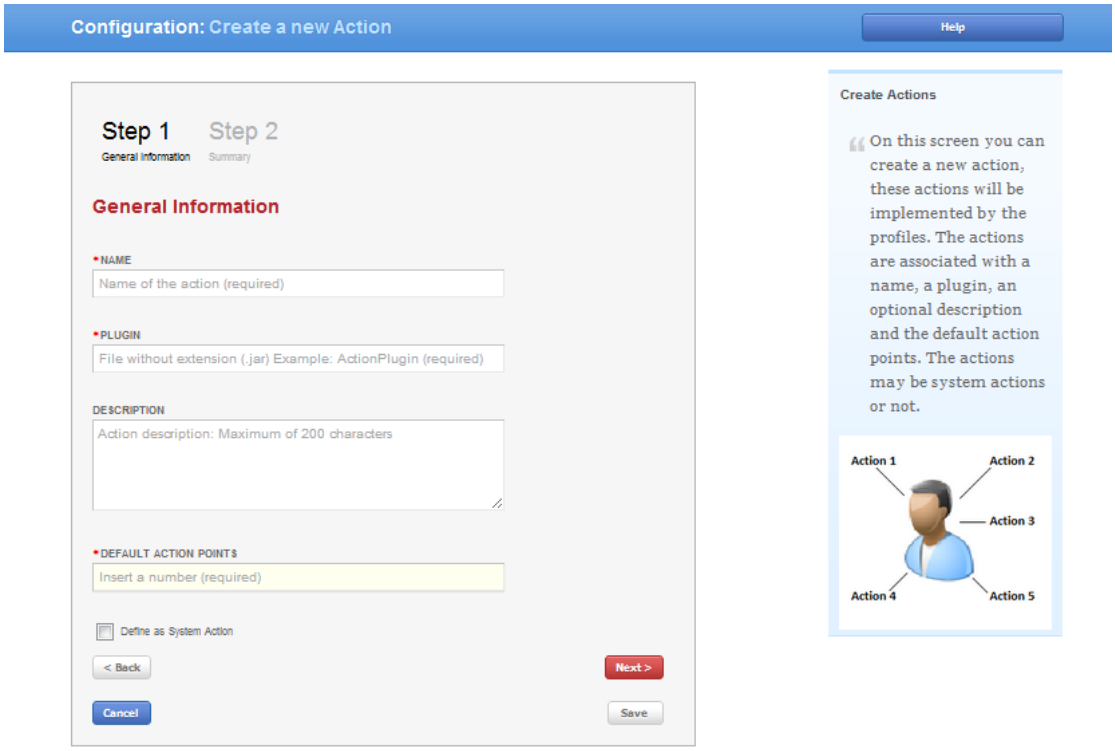
Al pulsar sobre el menú Configuration->Actions, se nos mostrará la siguiente página.



En esta página podemos observar que aparece un listado de todas las acciones creadas en nuestro sistema y una serie de botones que nos permiten realizar diferentes acciones sobre ellas.

En la parte superior del listado observamos un botón con el nombre **Create new action.**

Este botón, nos permite crear una nueva acción en nuestro sistema.
Cuando pulsamos sobre el botón, navegamos a una página que tiene el siguiente aspecto.

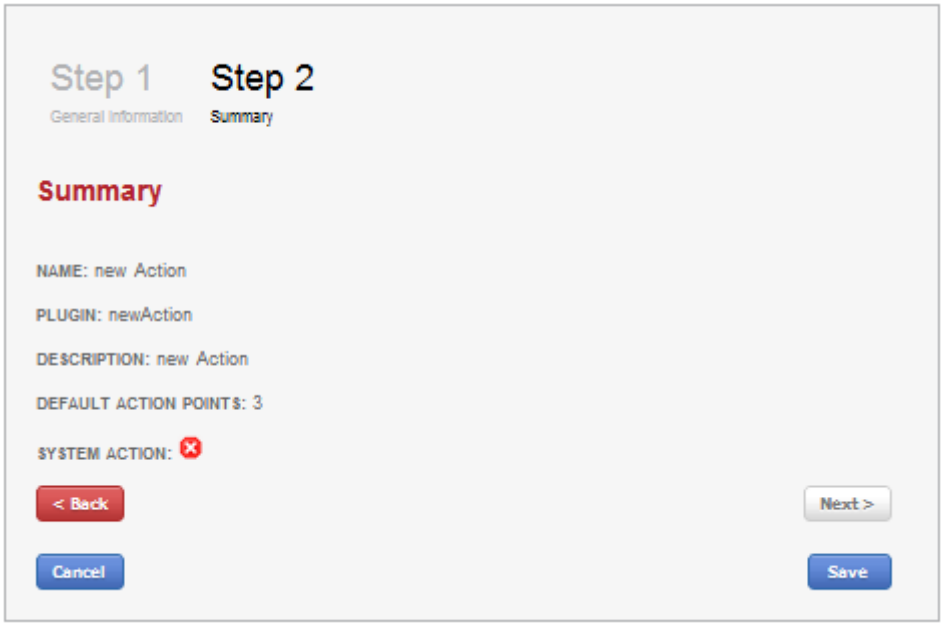


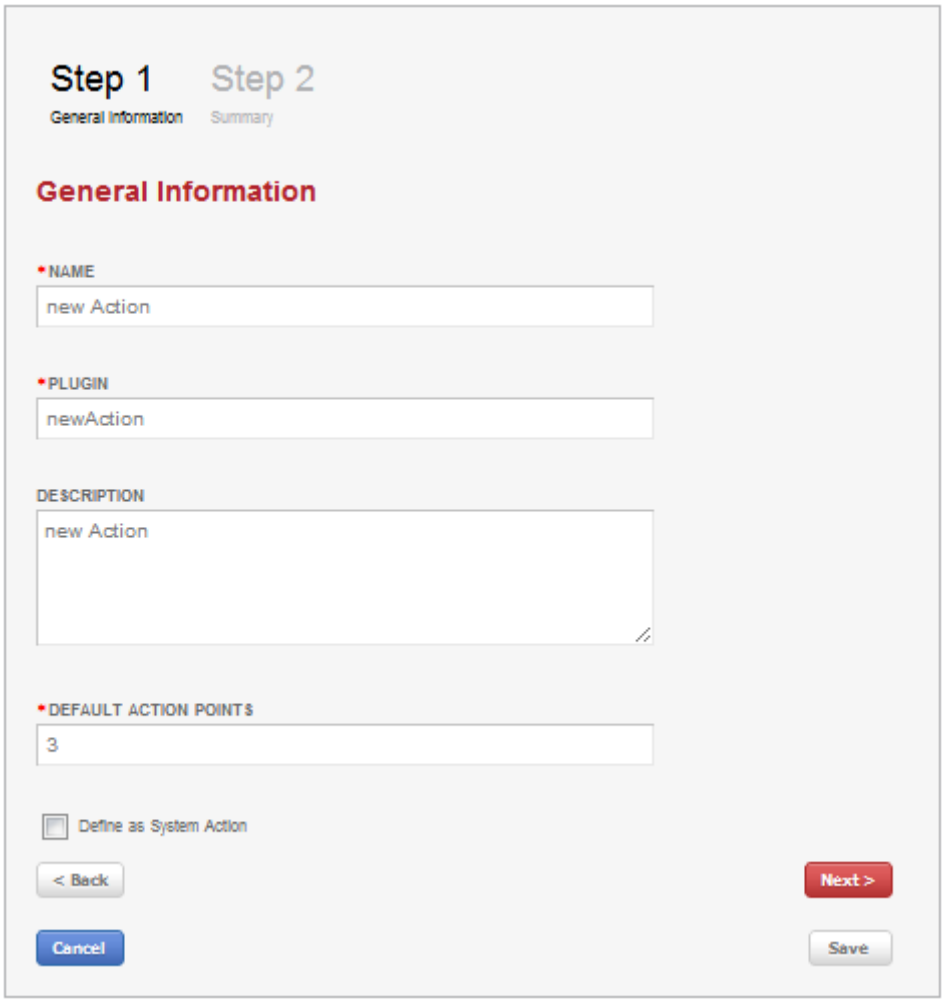
En esta ventana podemos configurar una nueva acción, insertando el nombre de la nueva acción, la ruta al plugin java asociado, una descripción opcional, si es o no acción de sistema y unos puntos de acción de simulación por defecto.

Al pulsar sobre el botón Next, una vez hemos cumplimentado correctamente los diferentes campos del formulario de creación de una nueva acción, se nos mostrará un pequeño informe sobre la operación que se va a realizar sobre nuestro sistema si confirmamos pulsando sobre el botón Save.

Disponemos de un botón cancel, con el que volveremos a la ventana del listado de acciones y que cancelara todo el proceso que hayamos realizado hasta el momento.
El botón back, permite volver a visualizar el formulario de creación de una nueva acción y editar los valores introducidos, si no son de nuestro agrado al haber visualizado el informe previo a la creación.

Una vez pulsemos sobre el botón Save, se insertará la nueva acción en el sistema y podremos hacer uso de ella.





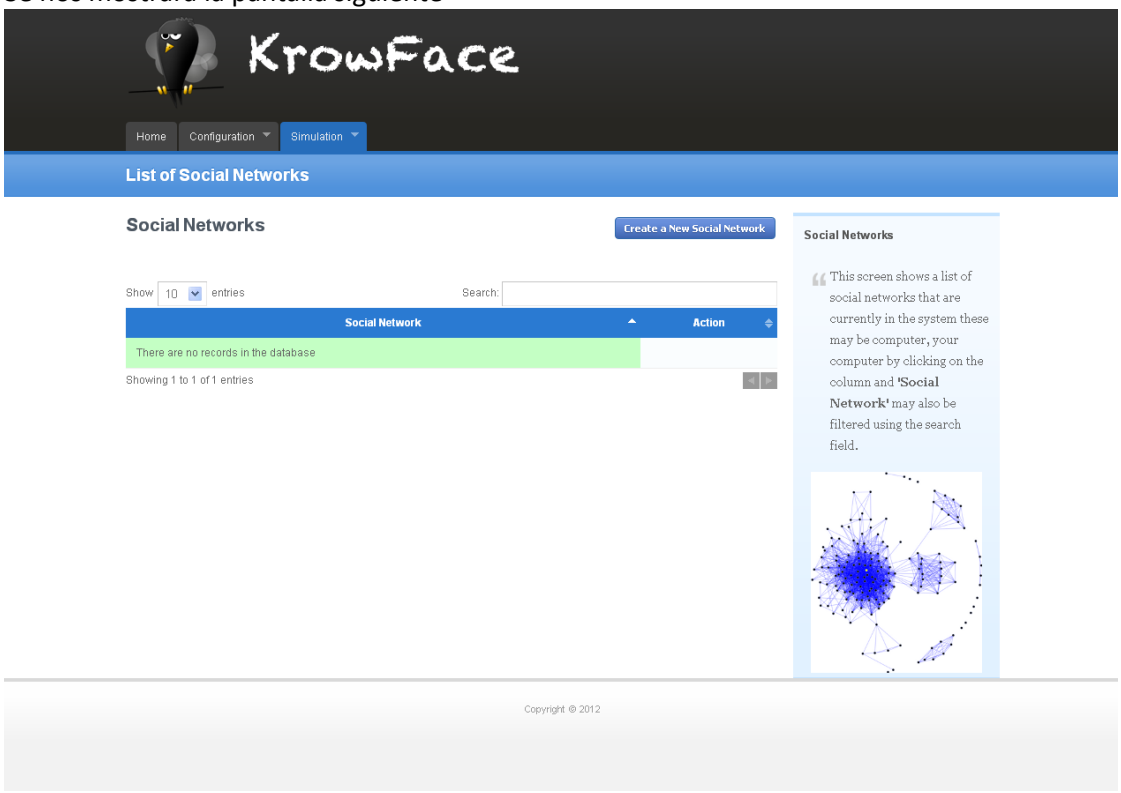
Una vez pulsemos sobre el botón Save, se insertará la nueva acción en el sistema y podremos hacer uso de ella.

7.5 Creación de una nueva simulación

Una vez hemos configurado lo deseado en los submenús del menú configuración, podemos crear una nueva simulación de red social.

Para ello, pincharemos sobre el menú Simulation en la parte superior y luego en el submenú Simulations.

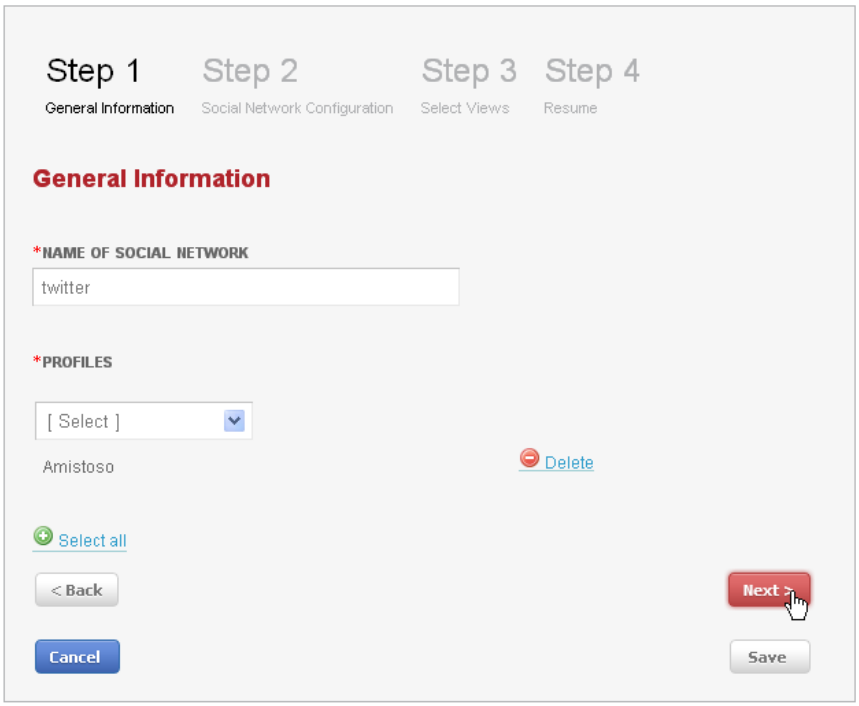
Se nos mostrará la pantalla siguiente



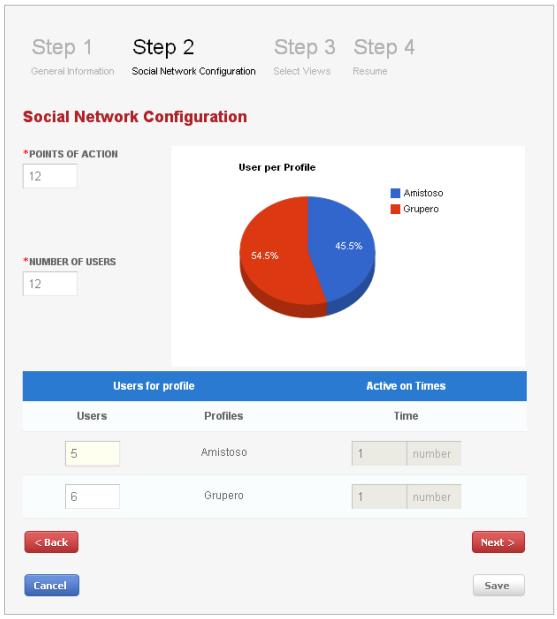
En ella podemos observar una parte central en la que se encuentra la lista de simulaciones de redes sociales anteriormente creadas y un botón en la parte superior que nos permite crear una nueva simulación.

Si pulsamos sobre el botón **Create a New Social Network** vamos a una nueva pantalla, similar a las pantallas de configuración del punto anterior.
La creación de una nueva simulación tiene cuatro pasos:

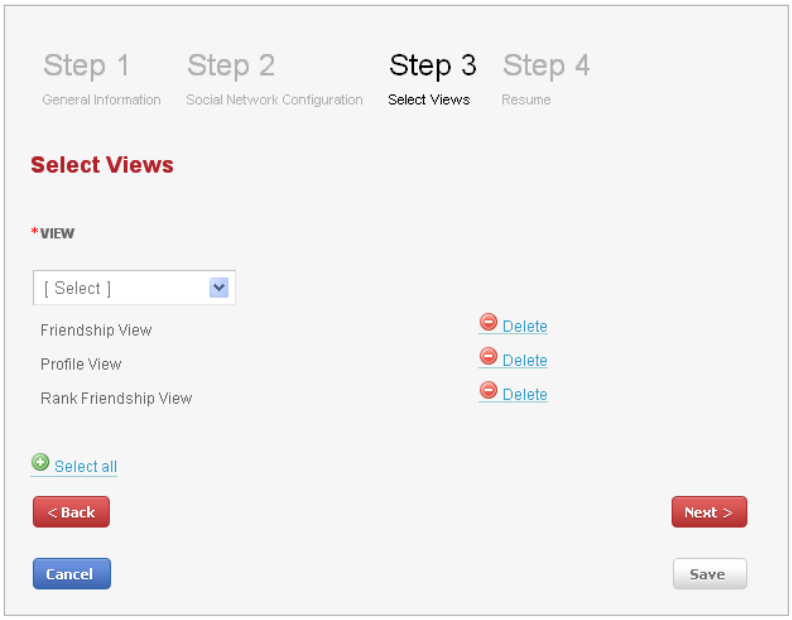
1. Elección del nombre de la red social y perfiles de usuarios que participan en ella



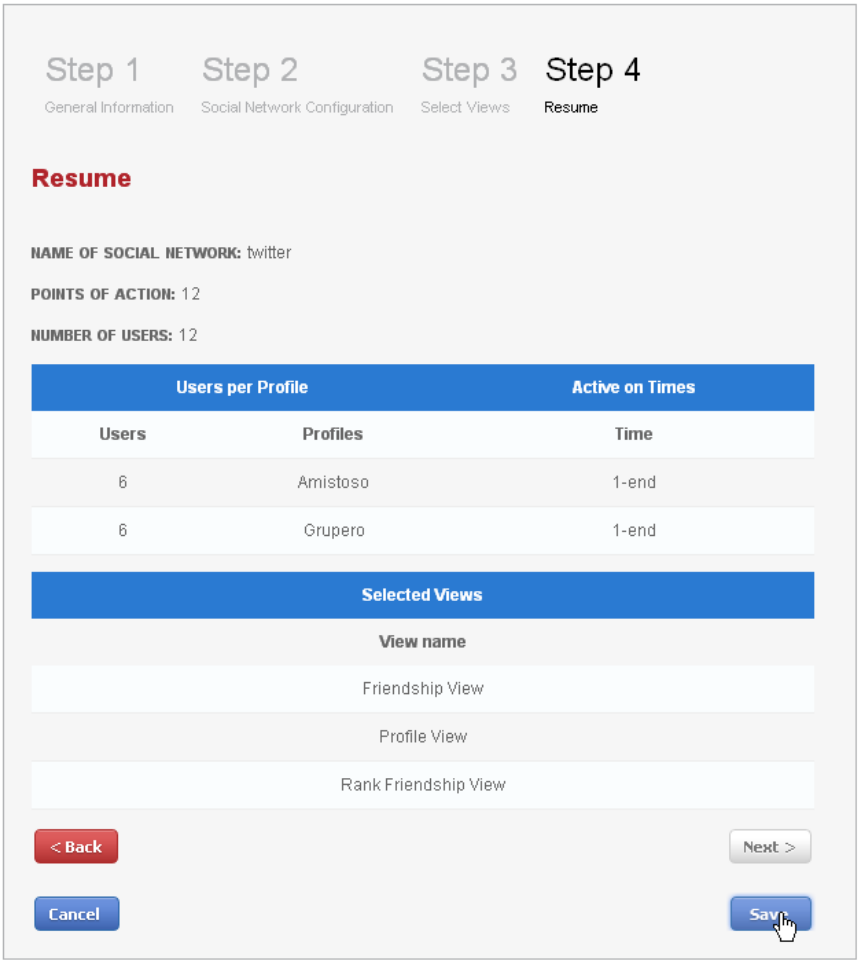
2. Elección de puntos de acción, número de usuarios y reparto entre los perfiles que participan en la red social



3. Selección de los tipos de visualizaciones que se generarán en los pasos de simulación

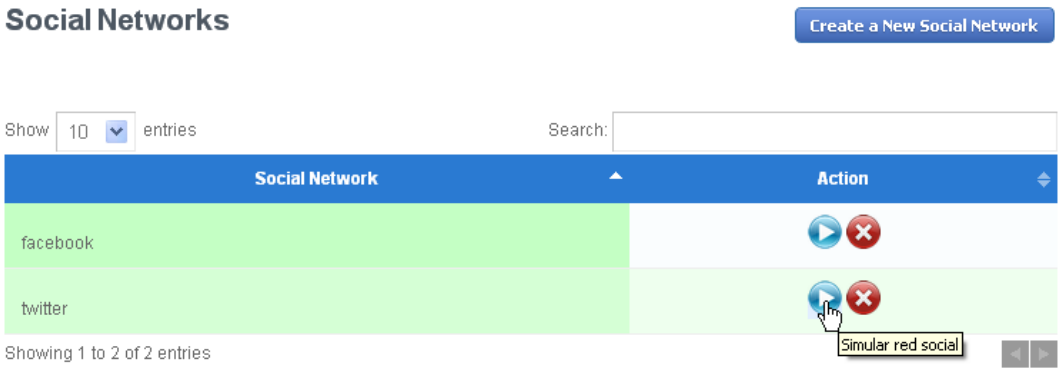


4. Confirmación de creación de la nueva simulación

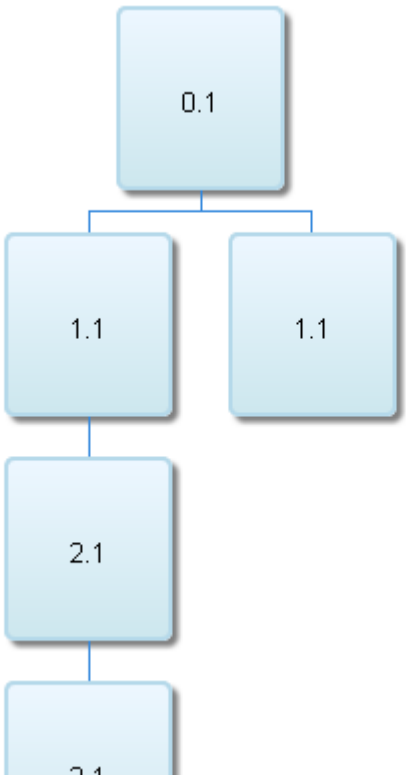


7.6 Simulación

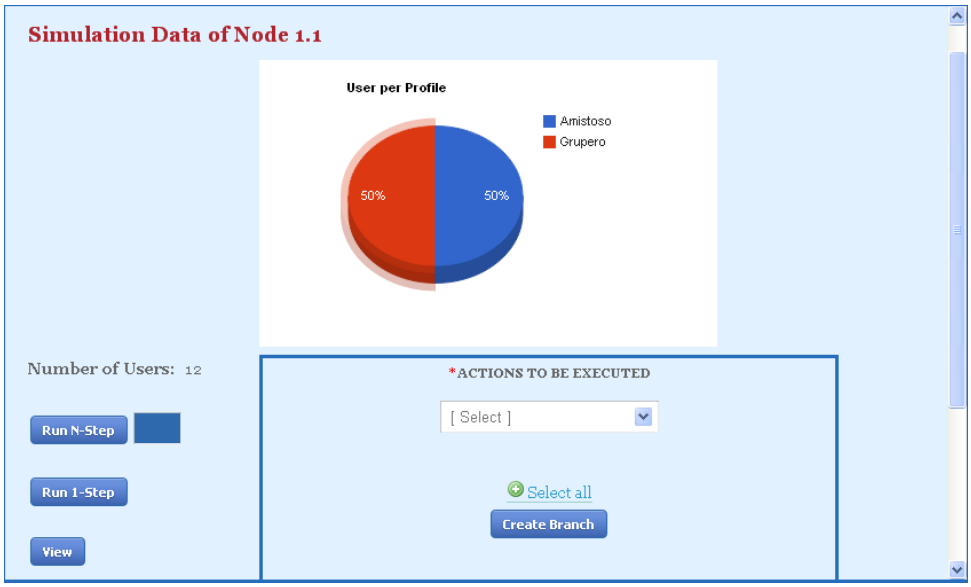
Una vez hemos creado una simulación de una red social siguiendo los pasos del punto anterior, podemos ejecutar la simulación. Para ello pulsaremos con el ratón sobre el menú superior **Simulation->Simulations** y en la lista de simulaciones creadas, pulsaremos sobre el botón  en la simulación que deseemos ejecutar.



Al pulsar sobre el botón play, se nos mostrará una nueva pantalla en la que aparecerá un árbol de tiempos simulados.



Al pulsar sobre el número de los nodos del árbol, se nos mostrará un menú en el que podremos seleccionar diversas opciones de simulación.

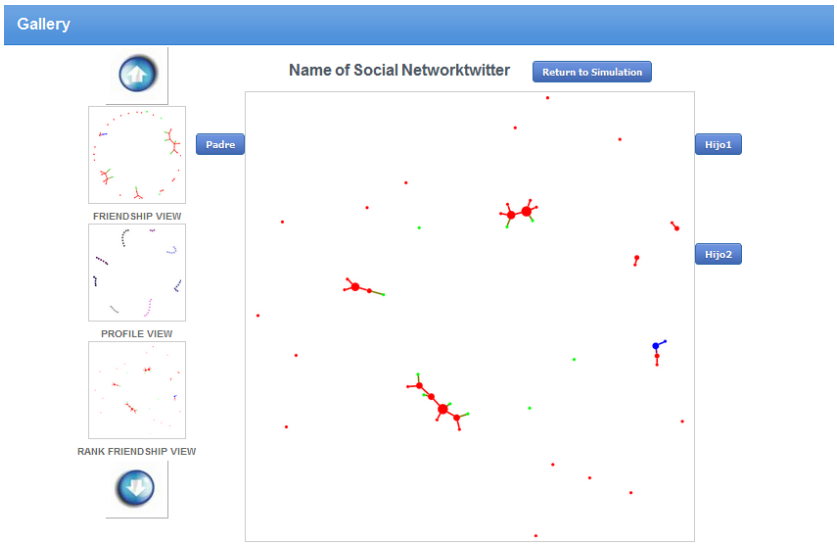


Si pulsamos sobre el botón Run 1-Step, se ejecutará un paso de simulación y se creará un nuevo nodo hijo del nodo que hemos seleccionado previamente.

Si pulsamos sobre Run N-Step, se ejecutarán el número de pasos de simulación que insertemos en el cuadro de inserción a la derecha del botón Run N-Step.

Si seleccionamos en el selector Actions to be Executed acciones a realizar y a continuación pulsamos sobre el botón Créate Branch, se creará un nodo hermano del nodo seleccionado anteriormente, creándose así una nueva rama de simulación.

Si pulsamos sobre el botón View, se nos llevará a una nueva pantalla en la que podremos visualizar el estado de la red social con las visualizaciones seleccionadas en la creación de la simulación.



En esta ventana, en la parte izquierda, podemos seleccionar las diferentes visualizaciones disponibles. En el centro de la ventana nos encontramos una imagen con el estado de la red social en el estado actual. A ambos lados de la imagen central, encontramos botones de navegación que nos permiten desplazarnos por el árbol de simulación.

7.7 Visualizaciones

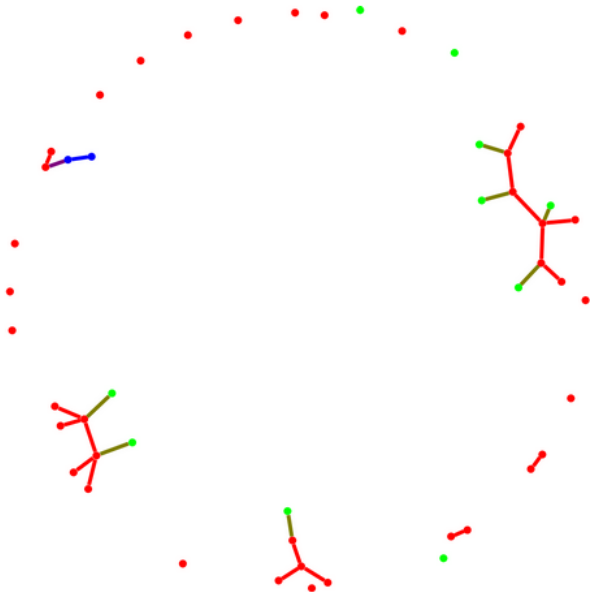
En este apartado vamos a comentar las visualizaciones disponibles en nuestro sistema

7.7.1 Profile View

En esta visualización se muestran los usuarios de cada perfil de la red social agrupados por color.

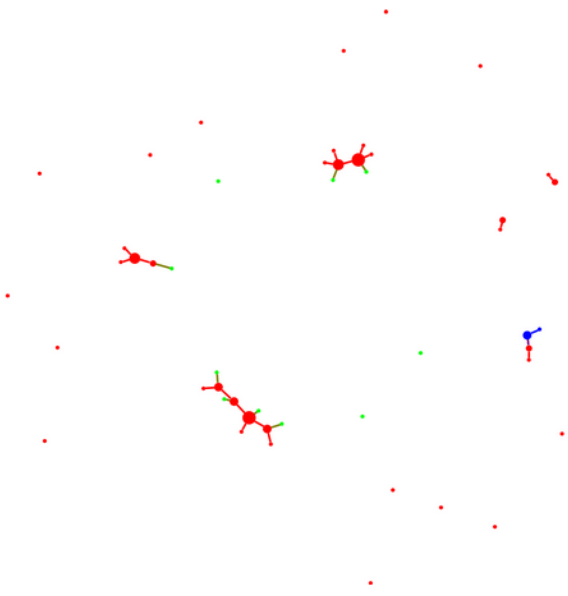
7.7.1 Friendship View

En esta visualización se muestra el grafo de amistades de todos los usuarios de la red social. Los nodos simulan usuarios y las aristas sus relaciones de amistad. Cuando en el momento actual de simulación se ha creado un nuevo usuario, este se representa con color verde. Si el usuario estaba en el paso de simulación previo al actual, se representa de color rojo. Si se crea una nueva relación de amistad, esta se representa de color azul si se ha creado entre dos usuarios que previamente al estado actual, estaban ya en la red social, de color verde si los dos usuarios se crearon en el estado actual y de color marrón si se ha creado una amistad entre un nuevo usuario y que existía en el estado anterior de simulación.



7.7.2 Rank FriendShip View

Es la misma visualización que la Friendship View salvo que en este caso, cuantas más amistades tiene un usuario, mayor es su tamaño de nodo y por consiguiente, podemos diferenciarlo mejor del resto de nodos de usuarios.



8.- AMPLIACIÓN DE FUNCIONALIDAD

8.1.- OBTENER DATOS EN TIEMPO REAL DE UNA RED SOCIAL

Una buena ampliación del proyecto sería que se pudieran obtener los datos directamente de una red social, como Facebook o Twitter, y simular y analizar directamente esos datos. De esta forma podríamos comparar lo que sucede en realidad de la simulación que hacemos nosotros.

8.2.- TENER PREDEFINIDAS REDES SOCIALES PARA PODERLAS SIMULAR

Otra ampliación del proyecto sería que estuvieran predefinidos el conjunto de parámetros que caracterizan una red social como puede ser Facebook, Twitter o Tuenti. Así el usuario de la aplicación podría decidir simular una red social en concreto.

8.3.- AMPLIAR PLUGINS Y VISUALIZACIONES

Otra posible ampliación, aunque no sea una ampliación de funcionalidad, pues nuestro sistema ya permite introducir nuevos Plugin y Views, es la creación de nuevos plugins y visualizaciones a través de las APIs desarrolladas. De esta manera, podremos mejorar las capacidades visuales y de comportamiento de los usuarios mediante la.

9.- GLOSARIO DE TÉRMINOS

Acciones: La forma que tienen los usuarios de interactuar con la red social es mediante la ejecución de acciones a lo largo del tiempo. Estas acciones producen cambios en el estado de la red social, relacionados con los propios usuarios que las ejecutan, o con otros usuarios ajenos. Estas acciones conllevan el establecimiento de nuevas relaciones o la creación de nuevos grupos o contenidos.

Atributos: Los atributos son los que caracterizan al resto de entidades, pudiendo ser de tipo numérico, string o lista.

Contenido: Contenido de la red social. Un contenido es una entidad creada por un SNU. Los contenidos pueden hacer referencia a las demás entidades de la red social.

Grupos: Los grupos son entidades destinadas a relacionar contenidos con usuarios, contenidos con otros contenidos y usuarios con otros usuarios.

Operaciones: Una operación es la ejecución de las acciones de un usuario en un instante.

Perfiles: Clases en las que se dividen los SNU las cuales determinan el comportamiento de cada uno según los porcentajes de realización de acciones que hay determinados en el perfil.

Relaciones: Son entidades destinadas a relacionar unos usuarios con otros.

SNU: Usuario de la red social. Nombramos SNU a cada entidad que se encarga de simular a cada uno de los usuarios de la red social. Representa su perfil así como las acciones y relaciones con otros usuarios. Cada SNU tiene un perfil asociado haciendo que este se comporte de una forma u otra en nuestra red.

Vistas: Representan las diferentes visualizaciones del estado de la red social en cada instante de simulación.

10.- BIBLIOGRAFÍA

- Documentación: Fernando B. López, Alberto J. Ramiro, Ignacio R. Teixeira. Krowdix 2.0: Sistema de simulación y análisis de redes sociales. Proyecto sistemas informáticos (2009-2010). Universidad Complutense de Madrid. Facultad de Informática.

- Relación de webs:
  - http://www.desarrolloweb.com
  - http://www.w3schools.com
  - http://jquery.com/
  - http://gephi.org/