Universidad de las Ciencias Informáticas

Facultad 2

Título: Detección y visualización de comunidades científicas con información extraída del repositorio institucional

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Javier Pereda Portela

Tutor: Vladimir Milián Núñez

Co-tutora: Yarilis Fernández Blanco

Junio de 2016

Declaración de Autoría

Declaro ser el único autor del trabajo titulado **Detección y visualización de comunidades científicas con información extraída del repositorio institucional** y autorizo a la Universidad de las Ciencias Informáticas y a la Facultad 2 para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de junio del 2016.

_____

Firma del Autor

Javier Pereda Portela

_____ _____

Firma del Tutor Firma de la Co-tutora

Vladimir Milián Nuñez Yarilis Fernández Blanco

Datos de contactos:

Javier Pereda Portela

Correo: jpereda@estudiantes.uci.cu

La Habana, Cuba

Ing. Vladimir Milián Núñez

Correo: vmilian@uci.cu

Universidad de las Ciencia Informáticas, La Habana, Cuba

Ing. Yarilis Fernández Blanco

Correo: yfblanco@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

No luchamos por gloria ni honores; luchamos por ideas que consideramos justas.

Fidel Castro.

Dedicatoria

A Fidel en su cumpleaños noventa.

A mis abuelas Caridad y Roselia por estar siempre en los momentos buenos y difíciles de mi vida, por impulsarme a ser un hombre de bien, por enseñarme a no rendirme antes las barreras que se me presentan en el camino y por seguir siempre adelante hasta alcanzar mi sueño.

A mi mamá Eneida porque a pesar de que siempre me pelea, la amo con todo mi corazón y sé que en cada regaño hay algo nuevo que me enseña.

A mi papá Noel que siempre ha sido mi guía, ejemplo en cada momento de mi vida y esto ha influido mucho en que cada día que pasa quiera parecerme más a él.

A mi hermana Tania por ser mi c ó mplice y compañera de tantas travesuras; por su amor incondicional.

En especial a mi abuelo Horacio Pereda que, aunque ya no se encuentra entre nosotros siempre estuve muy orgulloso del abuelo tan maravilloso que la vida me dio, porque me malcriaba, pero tambi é n sab í a cu á ndo regañarme, o llenarme con su alegr í a desmedida, con su sabidur í a y su amor incondicional.

Agradecimientos

A la Revoluci ó n y a Fidel por darme la oportunidad de cumplir mi sueño de convertirme en Ingeniero en Ciencias Inform á ticas.

A mi tutor Vladimir Mili á n N ú ñez por ser un gran amigo, gu í a y maestro. Por su paciencia, dedicaci ó n, fe y confianza en m í .

A mi novia que, aunque lleg ó hace poco tiempo a mi vida me ha impulsado para terminar con é xito esta investigaci ó n.

A mis Abuelas, padres, hermana, t í os y primos que me han incentivado a culminar mis estudios siempre con amor, dedicaci ó n y perseverancia.

RESUMEN

El presente trabajo de diploma se desarrolla en la Universidad de las Ciencias Informáticas (UCI). Tiene como objetivo desarrollar una herramienta para la detección de comunidades científicas y sus principales líderes, a partir de la información extraída del repositorio institucional. En el mismo se realizó un estudio sobre: conceptos asociados al análisis de redes sociales, comunidades científicas virtuales, algoritmos para la detección de comunidades (Fastgreedy, Label Propagation, Walktrap), tecnologías y las soluciones existentes hasta el momento. Además, se analizaron las características de las herramientas seleccionadas para el desarrollo de la propuesta de solución y se pusieron en práctica técnicas de la Ingeniería de Software, basadas en la metodología AUP para una mejor asimilación del negocio. Se definieron las características del sistema y posteriormente se realizó el diseño, implementación y casos de pruebas, corroborando la efectividad de la aplicación. Como resultado final se obtuvo un sistema viable que se puede aplicar a cualquier centro científico-académico, debido a que proporciona la información precisa para la toma de decisiones.

PALABRAS CLAVES: Análisis de Redes Sociales, Detección de Comunidad Científica, Visualización de Grafos, Programación Web.

ÍNDICE

INTRODUCCIÓN

Con el paso del tiempo los hombres han incluido casi genéticamente la necesidad de convertirse en seres sociales, por lo que se unen en grupos específicos que pueden ser por familiaridad, causas o intereses. En el caso específico de las ciencias, los investigadores se agrupan de acuerdo a sus intereses en una o varias áreas específicas de investigación. Dentro de dichos grupos se crean subgrupos o comunidades que tratan temáticas determinadas o específicas a un área de la ciencia en particular, donde los investigadores se relacionan e interactúan entre ellos, formando así las comunidades científicas. Un ejemplo puede ser: el grupo de usuarios de software libre que tiene dentro comunidades de Debian y Ubuntu.

En las comunidades científicas los individuos, además de relacionarse entre ellos, trabajan acorde a sus conocimientos, objetivos y fines comunes. Dentro de estos grupos existen personas que por su accionar sobresalen y lideran la comunidad, liderazgo que puede ser intencionado o no (un líder seleccionado por el grupo o una persona que sobresale por los aportes realizados). El liderazgo en una comunidad se puede determinar al analizar varios factores como son: el número o cantidad de publicaciones realizadas, nivel y/o número de colaboraciones realizadas con otros miembros de la red a la que pertenecen (coautorías) y el número de citas recibidas, por mencionar los principales.

Un papel fundamental en la evolución de estas comunidades lo ha desempeñado el desarrollo de las Tecnologías de la Información y la Comunicación (TIC), gracias a las cuales, el acceso a la información a alcanzado un carácter global y se pone a disposición de los investigadores una mayor cantidad de información científica. Se puede afirmar que estas tecnologías son un componente fundamental para elevar el nivel científico-técnico de la sociedad y las posibilidades de desarrollo de las comunidades científicas. Además, permiten crear, organizar, editar y almacenar la información de forma cada vez más rápido y fácil. Un ejemplo de lo planteado lo representan los repositorios institucionales.

Un repositorio institucional es un archivo donde se depositan, en formato digital, materiales derivados de la producción científica o académica de una institución (universidades, centros de investigación). Su objetivo es facilitar el acceso de la comunidad científica internacional a los resultados de la investigación realizada por sus miembros y aumentar la visibilidad de la producción científica de la institución. Asimismo, contribuir a la preservación de los documentos digitales allí depositados. Los repositorios institucionales suelen incluir tesis doctorales, artículos de carácter científico, ponencias o

comunicaciones a congresos, revistas electrónicas editadas por la institución, materiales docentes, … elaborados por los profesores e investigadores de la universidad o centro de investigación. (1)

A partir de información de los documentos de postgrados que se almacenan en los repositorios de las instituciones, existe un tipo de comunidad científica (2), la cual se puede obtener a partir de la red de coautorías (autores y tutores) o de las citas realizadas a otros documentos del repositorio. A partir de la estructura social de estas comunidades y con la utilización de herramientas informáticas especializadas, es posible realizar análisis que obtengan conocimiento útil e interesante implícito en estos datos. La Universidad de las Ciencias Informática (UCI), como centro de avanzada en el área de las ciencias informáticas, cuenta con un repositorio institucional donde se almacenan documentos de eventos, tesis de pregrado, tesis de postgrado, artículos y revistas[1].

Dentro de los documentos almacenados en los repositorios institucionales, suele dedicársele mayor atención a las tesis de postgrado (maestrías, doctorados y postdoctorados). Las tesis de postgrado son una fuente importante de información ya que en ellas se suele reflejar las tendencias de la disciplina de la ciencia en cuestión, así como las líneas de investigación y proyectos de sus autores, tutores y centros de procedencia y/o ejecución. Valiosa información sobre la cobertura y la profundidad del estudio del campo de investigación se puede obtener del análisis de estos documentos, además de una idea preliminar sobre los líderes científicos de estos centros, ya que estos suelen ser los más citados y los que mayor cantidad de trabajos de postgrado dirigen. Esta información suele obtenerse de las redes de citas y de coautoría[2].

Las redes de coautoría de los documentos de postgrado suelen ser la principal fuente de análisis para tareas de análisis y detección de comunidades científicas. De su análisis se puede obtener importantes resultados como por ejemplo cuales son los principales grupos de investigación y quienes son los líderes de estos grupos. Este análisis permite determinar las líneas de investigación de un grupo y/o institución, así como las personas o grupos a contactar para establecer posibles relaciones de colaboración, o incluso las personas que pueden desarrollar un proyecto de interés para la institución en determinada área de la ciencia.

A pesar de las potencialidades que puede brindar el análisis de las redes de coautoría, no se han encontrado evidencias de este tipo de estudio aplicado a la universidad. Esto trae como consecuencia

[1] http://repositorio_institucional.uci.cu/jspui/

[2] Se consideran redes complejas, donde los nodos de la red son los autores y los enlaces entre los nodos establecen la relación de coautoría en una o varias publicaciones (3).

que los directivos no conozcan con exactitud las condiciones reales de actividad de las líneas de investigación del centro, quienes son las personas que mayor impacto tienes dentro de estas líneas, e incluso puede traer como consecuencia la selección inadecuada de personal o equipos de trabajo para enfrentar una determinada tarea de investigación o desarrollo. Otra consecuencia es que no siempre se conoce quien es la persona con mayor preparación para liderar una comunidad, información que puede obtenerse al realizar este tipo de estudio. Además, este análisis permitiría determinar cuáles son los intereses reales de la comunidad investigativa universitaria y así poder seleccionar o crear los cursos, diplomados, especializaciones, maestrías y doctorados que satisfagan las necesidades de la comunidad científica del centro, lo que repercutiría en una mejor planificación de las actividades de postgrado.

Por lo antes expuesto se plantea como **problema de investigación:** ¿Cómo detectar comunidades y líderes científicos a partir de documentos de postgrado de repositorios institucionales para contribuir a la toma de decisiones en la UCI?

A partir del problema de investigación se define como **objeto de estudio** para esta investigación: análisis de redes sociales en comunidades científicas y como **campo de acción**: detección y visualización de comunidades científicas virtuales en documentos de postgrado.

Para dar solución al problema de investigación mencionado anteriormente se define como **objetivo general**: Desarrollar un sistema informático que permita obtener y visualizar las comunidades científicas virtuales y sus principales líderes en instituciones científico-académica a partir de la información de documentos de postgrado.

Para dar cumplimiento al objetivo general se definen los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Desarrollar un modelo conceptual que funcione como base para la implementación de un sistema informático.
- Definir el diseño que debe cumplir el sistema.
- Implementar las interfaces que muestren los resultados de manera gráfica.
- Realizar pruebas al sistema que comprueben la efectividad del sistema desarrollado.

Para cumplir con lo antes descrito se han trazado las siguientes **tareas de la investigación**:

1. Revisión bibliográfica para generar el marco teórico conceptual en lo referente al desarrollo de sistemas.

2. Estudio de sistemas homólogos para conocer aspectos regulares en el diseño de sistemas que trabajen la detección de comunidades científicas virtuales.
3. Análisis de las herramientas y metodología a utilizar para el desarrollo del sistema.
4. Identificación de los requisitos del sistema.
5. Implementación de la aplicación para la detección de comunidades científicas virtuales.
6. Selección de las técnicas de pruebas a realizar al sistema.
7. Aplicación de las pruebas y recolección de las no conformidades.
8. Resolución de no conformidades y aplicación de las pruebas.

Para dar cumplimiento a los objetivos específicos planteados se hace necesario la utilización de varios métodos científicos de investigación.

Métodos teóricos:

Histórico – Lógico: se utiliza para caracterizar y analizar las redes sociales. Además, para la detección de comunidades científicas teniendo en cuenta sus aspectos más externos a través de su trayectoria, evolución y desarrollo histórico.

Inductivo – Deductivo: Se utiliza para arribar a conclusiones a partir del estudio de documentos relacionados con el análisis de redes sociales y la detección de comunidades científicas, para fundamentar los elementos que conforman esta investigación.

Analítico – Sintético: Se utiliza en el análisis de la información de la presente investigación con el objetivo redactar las conclusiones a las que se arriban.

Métodos empíricos:

Entrevista: se aplica la entrevista no estructurada para identificar los requisitos funcionales del sistema y obtener información sobre la tecnología a utilizar.

Revisión documental: se hace uso de este método para verificar la información relacionada con el análisis de redes sociales, comunidades científicas y repositorios institucionales. Así como el chequeo de las fuentes de información que contienen los elementos referentes al tema enmarcado en el objeto de estudio. Todo esto ayudó en la construcción teórica, la fundamentación del problema y motivó a la elaboración de la solución propuesta.

El trabajo se ha estructurado de la forma siguiente: una introducción, cuatro capítulos, las conclusiones y las recomendaciones. A continuación, un resumen de las diferentes temáticas que se abordan en los capítulos:

Capítulo 1. Fundamentación teórica: Se describen los elementos teóricos que fundamentan el desarrollo de la investigación. Se manejarán conceptos y explicaciones sobre Redes Sociales, Análisis de Redes Sociales y comunidades científicas virtuales. También se dará una descripción de las herramientas, tecnologías y el lenguaje de programación que fueron utilizados para desarrollar la aplicación.

Capítulo 2. Características del Sistema: Se describe detalladamente la propuesta de solución y se muestran los principales artefactos que se generan a partir de haber utilizado AUP.

Capítulo 3. Diseño del sistema: Se muestran los artefactos generados en la etapa de modelado como el modelo de diseño, también se define la arquitectura y los patrones de diseños a utilizar en el desarrollo de la solución.

Capítulo 4. Implementación y Pruebas: En este capítulo se realiza el modelo de implementación y se presentan los resultados de las diferentes pruebas realizadas al sistema implementado, para verificar que cumple con los requisitos funcionales y no funcionales obtenidos en la captura de requisitos.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

Durante los últimos años se han multiplicado los estudios encaminados a analizar la información como factor clave para la toma de decisiones en las instituciones. La información es un recurso que se encuentra al mismo nivel que los recursos financieros, materiales y humanos y constituye el factor de mayor relevancia en la organización del trabajo. Para comprender los elementos anteriormente planteados es necesario definir y analizar ciertos aspectos de relevancia, que a su vez dan soporte teórico-metodológico y conceptual a la presente investigación.

En este capítulo se describen las bases teóricas referentes al análisis de redes sociales. Para ello, se esclarecen algunos conceptos importantes para el entendimiento y el dominio del problema. También se realiza un estudio sobre la aplicación del análisis de redes sociales en la detección de comunidades científicas y las tendencias sobre los temas de investigación.

1.1 Conceptos asociados al dominio del problema

Con el objetivo de que el lector pueda tener una mejor comprensión de los temas que se relacionan en el capítulo, se describen detalladamente a continuación un grupo de conceptos asociados al dominio del problema.

Repositorio Institucional: archivo donde se depositan, en formato digital, materiales derivados de la producción científica o académica de una institución (universidades, centros de investigación). Tiene como principal objetivo facilitar el acceso de la comunidad científica internacional a los resultados de la investigación realizada por sus miembros y aumentar la visibilidad de la producción científica de la institución. Además, contribuye a la preservación de los documentos digitales allí depositados (1). Los Repositorios institucionales suelen incluir tesis de maestría y doctorados, artículos de carácter científico, ponencias o comunicaciones a congresos, revistas electrónicas editadas por la institución, materiales docentes, elaborados por los profesores e investigadores de la universidad o centro de investigación.

Comunidades Científicas: Las comunidades científicas son redes de investigadores que se relacionan entre sí (2). En el caso específico de la presente investigación las comunidades científicas serán las redes de coautoría de documentos de postgrado del repositorio institucional.

Comunidades virtuales: Las comunidades virtuales son el tipo de comunidades creadas, mantenidas y que accede a través de Internet. Estas pueden ser vistas como una red social, dado que la red de equipos

que se conectan a sus miembros puede ser representada como un conjunto de enlaces con sentido social y se representan a través de un grafo (5).

Comunidades científicas virtuales: Entre las comunidades virtuales se pueden encontrar las comunidades científicas virtuales que son aquellas comunidades en las que los miembros se relacionan mediante temas de investigación en común de cualquier tema (5).

1.2 Soluciones informáticas existentes

Para la conformación del marco teórico de la investigación se realizó un estudio sobre las principales investigaciones realizadas respecto a la detección de comunidades científicas virtuales. Aunque se analizaron un grupo trabajos interesantes, solamente se relacionan en este documento tres, debido a que son los que mayor afinidad y relevancia presentan para la investigación.

"Análisis y Visualización de Comunidades Científicas con Información extraída de la web":

El doctor Rafael M. Gasca y los ingenieros Fernando de la Rosa y Sergio Pozo en su investigación, desarrollan un software que representa gráficamente las estructuras de los trabajos de la Jornada de Ingeniería de Software y Bases de Datos. El sistema informático trabaja con redes de coautoría y co-palabras, con lo que analizan las áreas temáticas. El software fue desarrollado en java y la representación de los resultados se muestra a través de un grafo donde el tamaño de los nodos varía en dependencia de su importancia dentro de la red. Aunque esta aplicación cumple con la mayoría de los requerimientos de la presente investigación, no se tiene acceso a la misma, ya que no se encuentra disponible públicamente ni la aplicación ni el código fuente de la misma. Por otro lado, está orientada a un problema específico y su fuente de datos es la base de datos bibliográfica DBLP (*Digital Bibliograph & Library Proyect*), y no un repositorio institucional (6).

"Efficient Crawling of Community Structures in Online Social Networks":

En su trabajo, el master Bas Van Kester, desarrolla una aplicación orientada al ámbito del comercio electrónico, en el cual se utilizan los algoritmos Label Propagation, Fastgreedy, Spinglass y Edge Betweenness para detectar comunidades en grafos generados a partir de los datos obtenidos de los perfiles de usuarios de redes sociales en línea. La herramienta, aunque se encuentra enfocada a las redes

sociales de ocio y su objetivo final es la creación de perfiles de compra para la realización de campañas de *marketing,* alejándose del objetivo principal de la presente investigación, es un punto de partida para el estudio y selección los algoritmos de agrupamiento a utilizar (7).

"Affinity Analysis between Researchers using Text Mining and Differential Analysis of Graphs"

En su trabajo Trigo y Brazdil, proponen un procedimiento para detectar afinidades entre diferentes investigadores a partir de una red de coautoría, a partir de datos bibliográficos obtenidos desde DBLP. En su trabajo analiza y utiliza varios algoritmos para detectar los investigadores de mayor relevancia en un subgrafo, y a partir de ellos identificar afinidad entre los investigadores teniendo en cuenta los que se relacionan mediante coautoría y poder identificar aquellos que pueden relacionarse. Este trabajo sirve como punto de partida para el estudio del algoritmo seleccionado para la detección de los líderes de las comunidades científicas (4).

1.3 Redes Sociales

Las redes sociales son el nombre con que se denomina a los grupos de individuos que guardan alguna relación social entre sí (8). Con el surgimiento de las TIC surgen nuevas redes sociales de carácter global entre las que se encuentran Facebook, Twitter e Instagram. Sin embargo, existen otras redes que tienen un carácter más científico y que la relación existente entre sus miembros puede ser un tema de investigación.

Las redes sociales se definen como un conjunto finito de actores (individuos, grupos, organizaciones, comunidades y sociedades) vinculados unos a otros a través de una relación o un conjunto de relaciones sociales.

1.3.1 Propiedades de las Redes Sociales

Las redes sociales se representan mediante grafos y se utilizan técnicas de la teoría de grafos para estudiar las estructuras sociales. Sin embargo, existen diferencias en la forma en cómo se aplican los distintos conceptos de la teoría de grafos sobre el Análisis de Redes Sociales. Este último adopta muchas de las propiedades de las redes sociales, en esta sección se estudian las diferentes propiedades que presentan una red social en base a su forma, distribución y similaridad entre los conjuntos de nodos y relaciones que existen en la red (9).

Distancias en las redes

La distancia es una medición de la teoría de grafos que permite definir propiedades más complejas sobre la posición de los individuos y sobre la estructura de la red. Una trayectoria es el número de enlaces que existen entre dos nodos en el grafo. La distancia geodésica es el número mínimo de enlaces que llevan de un nodo a otro (trayectoria mínima). Esta medida es muy utilizada en el Análisis de Redes Sociales, ya que permite obtener el camino más eficiente entre dos actores. Sin embargo, para el caso de las redes sociales el camino más corto no siempre es el buscado y dependerá del tipo de red social para determinar el tipo de distancia a utilizar (9).

El diámetro de un grafo está definido como el máximo geodésico de todos los vértices dentro de un grafo. El mismo representa el tamaño del grafo y permite saber que tan grande es, se dice que las redes de mundo pequeño poseen un diámetro pequeño. En la figura 1.1 se muestra esta propiedad en la red social del Club de karate de Zachary, utilizada en varios trabajos de investigación de redes sociales (10, 11) principalmente para trabajos de detección de comunidades.

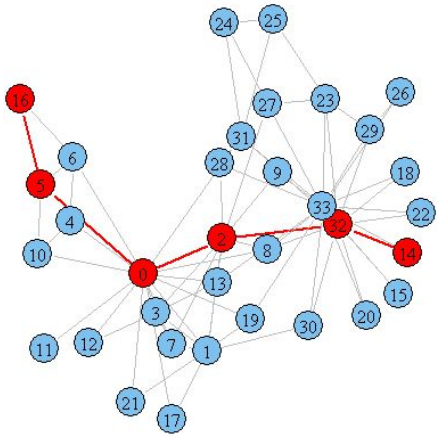


Figura 1.1: Ejemplo del cálculo del diámetro para la red del club de karate de Zachary.

Tipos de interacción

La forma en que interactúan los nodos dentro de un grafo permite establecer un nivel de conexión entre ellos y sus relaciones. A este fenómeno se le llama conectividad y permite reducir las trayectorias entre dos vértices. En un grafo con conectividad alta se puede ir de un nodo a otro con trayectorias más cortas (9).

La reciprocidad en las redes sociales es un fenómeno presentado en grafos dirigidos, también llamado simetría de vínculos. Este fenómeno se presenta cuando en un grafo existe un enlace que va de A hacia B y uno que va de B hacia A. La simetría de vínculos en un grafo reduce el diámetro de la red e incrementa la conectividad de la red (9).

Coeficiente de Agrupamiento

El coeficiente de agrupamiento es una métrica de similaridad que mide qué tanto está agrupado (o interconectado) un vértice. Para el presente trabajo se utilizó el coeficiente de agrupamiento local y global para estudiar la similaridad entre nodos de la red. El coeficiente de agrupamiento local mide el nivel de agrupamiento o interconexión de un vértice con sus vecinos; donde el vecindario de un nodo i está formado por el número de nodos que están adyacentes a él, es decir, aquellos nodos con los que se mantiene una relación (9). El coeficiente de agrupamiento global indica el nivel de agrupamiento de los nodos con respecto a la red total. En (12) Watts y Strogatz proponen el cálculo del coeficiente de agrupamiento para redes de mundo pequeño.

Cliques

Un clique en la teoría de grafos es un conjunto de vértices en el cual para cada vértice existe una arista que los conecta. Un clique es un subgrafo en el cual cada vértice está conectado a cada otro vértice del grafo, es decir, el subgrafo puede ser considerado como un grafo completo. En la figura 1.2 se muestra un grafo completo, si es un subgrafo de una red se considera que es un clique de tamaño 6 (9).

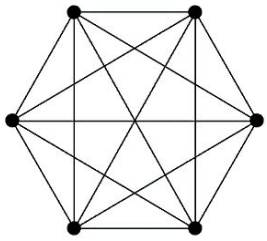


Figura 1.2: Ejemplo de grafo completo o clique de tamaño 6.

1.3.2 Clasificación de Redes Sociales

De modo general en la actualidad no existe una clasificación que permita especificar los tipos de redes sociales, estas se agrupan en dependencia del área de investigación. Una red social puede ser clasificada según el enfoque del estudio.

Las **redes basadas en su tamaño** dependen del diámetro de la red, es decir, la distancia mayor entre dos actores en la red. No existe una medida exacta para poder determinar cuándo una red social es grande o pequeña, por lo que dicho valor dependerá de lo que se quiera representar con la información.

- **Redes a pequeña escala:** Pueden ser analizadas por la mayoría de los sistemas para visualización de redes y generalmente utilizan al conjunto de datos completo para su análisis. Un ejemplo de este tipo de redes sociales es la red formada a partir de la colaboración de investigadores, donde los investigadores son los nodos de la red y los enlaces están formados por los artículos en los que uno o más investigadores participan.

- **Redes a gran escala:** No pueden ser analizadas fácilmente y en muchas ocasiones suelen realizarse las mediciones en base a una porción representativa de la red. Aún no existe sistema alguno que permita visualizar los tipos de redes de manera completa. Un ejemplo es la formada por los sistemas de correo electrónico, donde las cuentas de usuario son los actores en la red y un enlace es representado por un usuario en la lista de contactos, donde resultaría difícil tratar de representar una red de todas las cuentas de correo que interactúan en la Web.

Las **redes basadas en la evolución** dependen de los cambios que sufre la red a través del tiempo, pueden ser de cualquier tamaño y tener distintas formas.

- **Redes Estáticas:** No sufren ningún tipo de alteración cuando son sujetas a estudio, por lo que mantienen la misma estructura desde el momento en que son analizadas hasta el final de su estudio.

- **Redes Dinámicas:** Sufren cambios en su estructura debido a la incorporación o eliminación de nuevos actores y las relaciones entre ellos. Existen muchas redes de este tipo, pero la más importante es la propia Web, ya que se encuentra en constante cambio y su tamaño aumenta cada

vez más al pasar el tiempo. Diferentes fenómenos se presentan en este tipo de redes y muchos trabajos han sido propuestos como son la predicción de vínculos (13) y los modelos de crecimiento (14).

Las **redes basadas en su origen** dependen de su fuente de datos de origen. Muchas de estas redes pueden representar comunidades virtuales y/o del mundo real (15).

- **Redes fuera de línea (Off-line):** Son aquellas en las que las relaciones sociales son establecidas sin la intervención de un medio electrónico. Es decir, la administración y conocimiento de las relaciones recae exclusivamente en el conocimiento del individuo sin ayuda de un sistema software que le permita llevar la gestión de contactos. Un ejemplo de este tipo de redes fue la red generada para el caso del presunto suicidio del científico británico David Kelly (16), dicha red fue generada a partir de documentos del gobierno sobre el caso.

- **Redes en línea (On-line):** Son redes que dependen altamente de medios electrónicos y se mantienen ligadas a los cambios en la tecnología de los sistemas. Ejemplo de estas redes son Facebook, Twitter, Instagram, entre otras.

Las **redes basadas en su topología** dependen de la complejidad de la red.

- **Redes Simples:** Son estructuras sencillas y pueden ser fácilmente analizadas con conceptos básicos de la teoría de grafos.

- **Redes Complejas:** Su estudio está basado en el análisis empírico de las redes de mundo real, se dice que son complejas porque presentan propiedades no triviales como el cálculo del coeficiente de agrupamiento y centralidad de la red. Ejemplo de este tipo de redes son las redes aleatorias (17), las de mundo pequeño (18, 12) y las libres de escala (19).

En la presente investigación la red social que se analizará se puede clasificar como: de pequeña escala, dinámica, en línea (on-line) y compleja.

1.3.3 Estructuras de las Redes Sociales

La estructura de una red social está basada en individuos u organizaciones que están conectados por una o más relaciones tales como: amistad, contactos profesionales, parentesco, entre otros. Las redes sociales en términos de la teoría de grafos representan una estructura basada en grafos complejos. También pueden ser representadas como grafos dirigidos, no dirigidos, bipartidos y completos.

El componente gigante de las redes sociales

En el año 2000 se publica un artículo (20) sobre las características de la Web, entre sus principales descubrimientos encontraron que un número pequeño de nodos presentaba alto agrupamiento, a este fenómeno le llamaron Componente Fuertemente Conectado (CFC) o Componente Gigante (CG) y también demostraron que existía la presencia de más de uno. Esto permitió diferenciar ampliamente a las redes aleatorias de las redes de mundo real, ya que las aleatorias presentan una distribución homogénea entre sus nodos, es decir, no hay presencia de CFC y los nodos presentan una distribución de grado similar.

Un CFC es definido como un conjunto de nodos tal que para cualquier par de nodos u y v en el conjunto hay un camino entre u a v. La figura 1.3 muestra la conectividad de la Web, donde el CFC es el núcleo de la red y el conjunto de nodos en IN son los nodos que entran en el CFC y el OUT es el conjunto de nodos que salen del CFC. Los tentáculos representan a los nodos que conectan a un CFC con otros tipos de formaciones, ya sea otro CFC o islas de nodos.

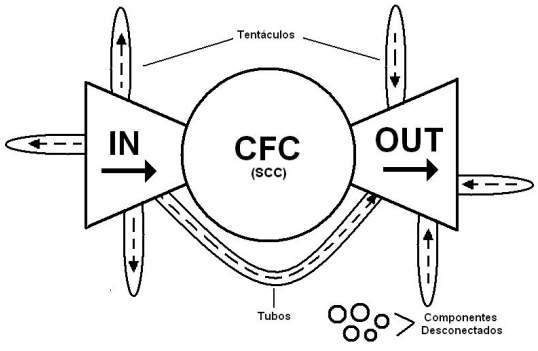


Figura 1.3: Conectividad de la Web, donde el CFC es el núcleo de la red.

Redes Complejas

En trabajos recientes (21, 22) se ha demostrado que las redes de mundo real son complejas, en sentido a las diferentes características topológicas que son derivadas de la teoría de las redes aleatorias. Un grafo complejo contiene cuantiosos subgrafos diferentes. Muchos sistemas en la naturaleza están construidos por un alto número de conexiones dinámicas (por ejemplo: redes neuronales y el Internet) y existen diferentes modelos de redes complejas como las redes aleatorias, mundo pequeño y libre de escala que permiten estudiar el comportamiento de una gran cantidad de redes del mundo real.

Redes aleatorias

Este tipo de redes son generadas a partir de agregar nodos de manera aleatoria a un conjunto de datos fijos. Este tipo de redes presentan caminos muy cortos entre nodos y un coeficiente de agrupamiento bajo, además siguen una distribución Binomial o de Poisson como en la figura 1.4. Existen diferentes modelos que han sido propuestos para este tipo de redes como el modelo Erdös-Rényi (ER) (23) y el de Gilbert (24), siendo estos los primeros aplicados a las redes sociales.

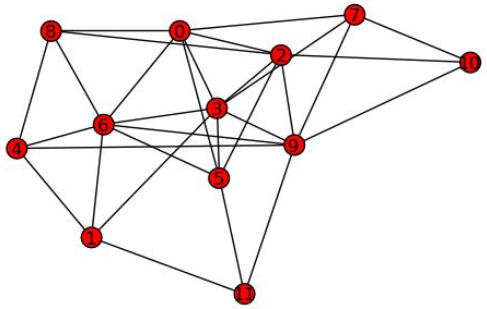


Figura 1.4: Ejemplo de una red aleatoria con 12 vértices y 29 aristas.

Redes de mundo pequeño

Las redes de mundo pequeño son estructuras que tienen un diámetro pequeño y exhibe un alto agrupamiento. El problema del mundo pequeño fue planteado en 1967 por Stanley Milgram (15), donde textualmente se hace estas dos preguntas:

1. *"Comenzando con dos personas cualquiera en el mundo, ¿Cuál es la probabilidad que ellos se conozcan?"*

2. *"Dados dos personas cualesquiera en el mundo, persona X y persona Z, ¿Cuantos enlaces intermedios son necesarios antes que X y Z se conecten"*

Estas interrogantes motivaron a Milgram a generar una serie de experimentos, que consistió básicamente en monitorear a un grupo de personas que interactuaban enviando cartas a desconocidos. Los resultados mostraron que la red de la sociedad humana presenta una estructura de mundo pequeño y que los individuos están a "seis grados de separación". Watts (25) describe la teoría de los seis grados y plantea que "el mundo es un pañuelo", es decir, cualquiera en el mundo permanece conectado a otra persona a través de otras personas y que el número de personas intermedias es menor a cinco. Watts y Strogatz proponen un modelo basado en el fenómeno de las redes mundo pequeño (12).

1.4 Análisis de Redes Sociales

El análisis de redes sociales es un conjunto de métodos y técnicas que han alcanzado gran importancia en la sociedad actual. Está destinada a estudiar cómo se relacionan entre sí las personas u otras entidades como instituciones y naciones, en las siguientes líneas se hace mención a varias definiciones.

"El análisis de redes sociales, también denominado análisis estructural, se ha desarrollado como herramienta de medición y análisis de las estructuras sociales que emergen de las relaciones entre actores sociales diversos (individuos, organizaciones, naciones, etc.)" (26).

"En la actualidad los análisis de redes sociales se han posicionado con estudios que nos permiten conocer la relación entre grupos y fuentes de investigación. Al entender a la colaboración científica como un proceso natural dentro del proceso de la comunicación científica." (27) *"mediante la aplicación del análisis de redes sociales, a través de las redes de co-autoría, que nos permite conocer con exactitud los colegios invisibles y su influencia dentro de una disciplina"* (28).

"Los análisis de redes sociales nos ofrecen toda una metodología para analizar cómo se organizan los individuos e instituciones, es decir estos análisis reflejan la interacción entre individuos (autores, instituciones) con las estructuras sociales que se establecen entre ellos" (29).

A partir de las definiciones anteriores, se asume la ofrecida por Gretzel que define el análisis de las redes sociales como una herramienta, siendo esta premisa el objetivo de la investigación. Su implementación y

utilización es indispensable debido a los beneficios que ofrece y tributa al éxito de la organización. La puesta en práctica del análisis de redes sociales permitirá encontrar las relaciones existentes entre las fuentes de información que cuenten con alguna similitud (publicación de tesis, artículos científicos e informes); ya sea por el autor o el contenido descrito en la misma. Logrando así una interpretación más acertada de las relaciones contenidas en las bases de datos de las redes de documentos de postgrado.

La utilización del análisis de redes sociales ha traído consigo grandes beneficios para las empresas y organizaciones que la han aplicado. El hecho de su implementación ha permitido extraer información útil demostrando que puede ser usada efectivamente para la toma de decisiones.

1.4.1 Beneficios del análisis de redes sociales

El análisis de las redes sociales, aplicado a la gestión de la información y el conocimiento en las organizaciones facilita:

- Actualizar la información: área o actividad que permite poner al día, renovar o proporcionar el comportamiento de la información contenida en las empresas.
- Optimizar la comunicación entre áreas de trabajo (acción y efecto de mejorar, perfeccionar, para que la comunicación entre la organización se realice de la forma más rápida posible).
- Visualizar las redes en temas de investigación (acción de mostrar la estructura entre los diferentes temas de investigación, siendo de ayuda interna para el estudio, organización, y solución de problemas).
- Prestar servicio al usuario en tiempo real (conjunto de actividades encaminadas a responder las necesidades que presenta el usuario) (30).

1.4.2 Detección de Comunidades Virtuales

En una red social, es común la aparición de grupos densamente conectados de vértices, sólo con las conexiones más dispersas entre los diferentes grupos. La capacidad para detectar estos grupos -llamados comunidades- podría ser de relevancia en el estudio de una red, revelando nuevos vínculos y grupos de nodos (31).

Las estructuras de comunidad en las redes durante mucho tiempo han sido estudiadas por múltiples disciplinas y se han relacionado estrechamente con la teoría de grafos, para representar gráficamente las

ideas de la informática y el agrupamiento jerárquico de la sociología (32). En la literatura se han propuesto diferentes algoritmos para resolver el problema descrito. Ejemplos de ellos son: los algoritmos de división que son capaces de detectar los vínculos entre las comunidades y eliminarlos posteriormente desde la red, los algoritmos de aglomeración que fusionan nodos o comunidades similares de una manera recursiva y los métodos de optimización que se basan en la maximización de una función objetivo (33).

Modularidad

Con el fin de detectar la calidad de la estructura de la comunidad, es decir, la calidad de las particiones obtenidas por el algoritmo de detección de la comunidad, es usada a menudo una métrica llamada Modularidad determina por Newman (34):

$$Q=\frac{1}{2m}\sum_{c}\sum_{i,j\in c}(A_{ij}-\frac{k_i k_j}{2m})$$

Donde A es la matriz ponderada de adyacencia entre el nodo i y j. $k_i=\sum_j A_{ij}$ es la suma de los pesos de todos los vértices que viene dentro o fuera del nodo i. C_i es la comunidad a la que se le asigna i y $m=\frac{1}{2}\sum_{ij} A_{ij}$.

La modularidad es un valor escalar en el intervalo [-1, 1] y mide la densidad de vértices interiores de las comunidades en comparación con vértices inter-comunidades. Esto significa que la modularidad puede ser entendida como el número de aparición de aristas dentro de los grupos menos el número esperado en una red idéntica con las aristas colocados al azar (31). Los valores de modularidad pueden ser positivos o negativos, sin embargo, de acuerdo con Newman, los valores positivos de la modularidad indican que la red tiene una posible existencia de comunidad. Por lo tanto, el autor sugiere buscar divisiones con alta modularidad, ya que indica una red con muchas posibilidades de tener bien definidas y claras particiones que forman comunidades.

1.4.3 Algoritmos para la detección de comunidades virtuales

Se han desarrollado diversos algoritmos para la detección de comunidades en grafos, cada uno con sus especificidades en cuanto a complejidad computacional, medidas para la validación de las comunidades, entre otros indicadores.

Edge Betweenness

Es un algoritmo particional que se basa en el recuento de los caminos más cortos entre los vértices. Por lo tanto, se define que las aristas con un bajo valor de intermediación pertenecen a la misma comunidad y las aristas con un alto valor de intermediación diferentes comunidades (35).

El algoritmo que se propone para la identificación de las comunidades se indica a continuación:

1. Calcular la intermediación para todas las aristas de la red.
2. Retirar la arista de la más alta intermediación.
3. Recalcular la intermediación para todas las no afectadas.
4. Repetir desde el paso 2 hasta que las aristas se mantengan.

Este algoritmo no se utilizará en esta investigación porque para grafos grandes (con 1000 nodos o más) no funciona.

Fastgreedy

El algoritmo Fastgreedy fue propuesto por Clauset en el 2004 y es una versión mejorada del algoritmo propuesto por Newman, que utiliza una estrategia voraz para la detección de una comunidad. El método puede ser representado como un árbol, en el que las hojas son los vértices de los nodos de redes originales y los correspondientes acoplamientos internos y se denomina dendrograma. Utiliza grafos ponderados y busca determinar los cambios en la modularidad que se derivan a partir de la fusión iterativa de comunidades. Las comunidades que el algoritmo detecta son excluyentes (36).

El algoritmo Fastgreedy se basa en:

1) Cada nodo de la red es considerado como una sola comunidad.
2) Calcular el cambio en la modularidad del grafo (= Q_{ij}) al crear una nueva comunidad con las comunidades i y j.
3) Combinar las dos comunidades que generan el mayor cambio en la modularidad (= Q_{ij}).
4) Repetir los pasos 2 y 3 hasta que el cambio de la modularidad sea negativo (= Q_{ij} X 0).

La aplicación del algoritmo de detección de comunidades produce como resultados:

1) Una partición de la red o grafo:
    - Agrupaciones de nodos, elementos del conjunto V de nodos (o vértices), que conforman las comunidades.
    - Enlaces inter-comunidades, elementos del conjunto E de enlaces que conectan las comunidades.
2) Un valor de modularidad.

Label Propagation

Dentro de las redes complejas, las redes reales tienden a tener estructura de comunidades. Label Propagation es un algoritmo (37) para la búsqueda de comunidades. En comparación con otros algoritmos (36) Label Propagation tiene ventaja en su tiempo de ejecución, la cantidad de información a priori necesaria acerca de la estructura de redes (no requiere que ningún parámetro sea conocido de antemano). Tiene como desventajas que no producen ninguna solución única, sino varias soluciones.

En condición inicial los nodos tienen una etiqueta que indica la comunidad a la que pertenecen. La pertenencia a una comunidad cambia, a partir de las etiquetas que los nodos vecinos poseen. Este cambio está sujeto a la cantidad máxima de las etiquetas dentro de un grado de los nodos. Cada nodo se inicializa con una etiqueta única a continuación, las etiquetas se difunden a través de la red. En consecuencia, los grupos fuertemente relacionados alcanzan una etiqueta común rápidamente. Cuando muchos de estos grupos densos (consenso) se crean en toda la red, que continúan expandiéndose hacia el exterior hasta que sea posible hacerlo (37).

El proceso en 5 pasos: (37)

1) Inicializar las etiquetas en todos los nodos de la red. Para un nodo determinado $x, C_x(0)=x$.
2) Establecer a t + 1.
3) Organizar los nodos de la red en un orden aleatorio y establecerlo en “ .

4) Para cada x ∈ “ elegido en ese orden determinado, permitir que $C_x(t)=f(C_{x_{i1}}(t),\ldots,C_{x_{im}}(t),C_{x_{i(m+1)}}(t-1),\ldots,C_{x_{ik}}(t-1))$. f aquí devuelve la etiqueta que ocurre con mayor frecuencia entre los vecinos.
5) Si cada nodo tiene una etiqueta con el número máximo de los vecinos y luego se detiene el algoritmo. Si no, se establece t + t ” 1 y se repite desde el paso 3.

Walktrap

Con el fin de agrupar los nodos en comunidades, Walktrap en (38) introduce una distancia *r* entre nodos que trata de capturar la estructura de las comunidades en el grafo. Esta distancia debe ser mayor si los dos nodos pertenecen a comunidades distintas, en caso contrario; cuando los nodos pertenezcan a la misma comunidad, la distancia debe ser pequeña. Este valor es calculado a partir de la información entregada por {paseos aleatorios} en el grafo.

Se consideran paseos aleatorios en el grafo *G* de una longitud dada *t*. Se usará la información proporcionada por las probabilidades P_{ij^t} relacionadas con ir de i a j en t pasos. La longitud t del paseo aleatorio debe ser lo suficientemente largo para reunir información suficiente acerca de la topología del grafo. Para comparar dos vértices i y j utilizando estos datos, debe tenerse en cuenta lo siguiente:

- Si dos nodos i y j están en la misma comunidad, la probabilidad P_{ij^t} seguramente será alta. Sin embargo, que la probabilidad $P_{ij\,t}$ sea alta, no necesariamente implica que i y j estén en la misma comunidad.
- La probabilidad P_{ij^t} está influenciada por el grado k j debido al hecho que paseo aleatorio tiene una alta probabilidad de ir a los nodos de grado alto.
- Dos nodos de una misma comunidad tienden a {ver} al resto de los nodos de la misma manera. Es decir, si los nodos i y j pertenecen a la misma comunidad, probablemente se tenga que P_{ik^t} + P_{jk^t} .

Teniendo en cuenta lo anterior, se da a continuación una definición de distancia entre nodos:

Definición 1.1: Sean i y j dos nodos en un grafo G, entonces se define una distancia r_{ij} , para dichos nodos como:

$$r_{ij} = \left(\sum_{l=1}^{n} \frac{\left(P_{il}^{t} - P_{jl}^{t} \right)^2}{k_l} \right)^{1/2}.$$

Definición 1.2: Sean C_1 y C_2 dos comunidades. Se define una distancia entre estas dos comunidades como: $r_{C_1 C_2} = \left(\sum_{l=1}^{n} \frac{\left(P_{C_1 l}^{t} - P_{C_2 l}^{t} \right)^2}{k_l} \right)^{1/2}$

Una vez hechas estas definiciones, se puede describir el proceso del algoritmo de Walktrap de la siguiente forma:

1. Se empieza desde la partición P_1 + ''vQ v∈ VQdel grafo en que las n comunidades han sido reducidas a un sólo nodo. En primer lugar, se calcula las distancias entre todos los nodos adyacentes. A continuación, esta partición evoluciona mediante la repetición de las operaciones (2-4):
2. Se eligen dos comunidades C_1 y C_2 en P_k de acuerdo a un criterio basado en la distancia entre comunidades,
3. Fusionar estas dos comunidades en una nueva comunidad $C_3 = C_1 \cup C_2$, y crear la nueva partición $P_k + 1 = \left(P_k \{C_1, C_2\} \right) \cup \{C_3\}$, y
4. Actualizar las distancias entre comunidades (adyacentes).

Después de n %1 iteraciones, el algoritmo termina y se obtiene $P_n = \{V\}$. Cada paso define una partición P_k del grafo en comunidades, las cuales dan una estructura jerárquica (dendograma).

Luego de analizados los algoritmos se decidió seleccionar para este trabajo Fastgreedy, Walktrap y Label Propagation debido a que se ajustan a las necesidades del sistema.

1.4.4 Miembros claves en las comunidades virtuales

Los miembros claves juegan un rol decisivo en la construcción y desarrollo de cualquier comunidad, ya que ellos no solo producen muchas de las contribuciones, sino también alientan a otros miembros a participar, compartir conocimiento, buscar nuevos miembros y pueden moderar los diferentes tipos de actividades dentro de la comunidad.

Técnicas de descubrimiento

En la bibliografía del Análisis de Redes Sociales hay técnicas para descubrir que miembros de una comunidad son los más influyentes. Se basan en la idea de medir la interacción de un individuo con los miembros de la comunidad, porque los miembros claves están en el centro de la comunidad. Es común aplicar algoritmos básicos, como HITS y PageRank para listar miembros de acuerdo a su importancia dada por sus conexiones con otros para obtener miembros clave basado en sus respectivos temas de interés.

HITS

Para una búsqueda realizada en la web por un humano, un sistema de clasificación basado en el texto de las páginas web hace referencia a un enorme número de páginas, que puede que ni siquiera sea relevante para el tema, lo que causa un gran problema de clasificación sin siquiera garantizar información útil para la persona. Para darle solución a dicho problema Kleinberg propuso un algoritmo para clasificar las páginas pertinentes a un tema de búsqueda utilizando la estructura de enlaces de cada uno de ellos denominado HITS (Hyperlink-induced topic search). En (39) Kleinberg describe su algoritmo como una herramienta para extraer información de manera efectiva en un entorno web, basado en el descubrimiento de fuentes de información autoritarias sobre cualquier tema de búsqueda.

Kleinberg plantea un sistema de clasificación diferente para páginas web, como resultado de una consulta a una web basada a una clasificación que determinó, por ejemplo, con páginas autorizadas. Una página es una autoridad si contiene información relevante y valiosa para un tema específico, es decir, si el creador de la página p incluye un enlace a la página q, que ha "conferido" autoridad a que (excluyendo los enlaces con fines de navegación, tales como "volver al menú principal"). Mientras que una página puede ser un contenedor si anuncia una página autorizada, es decir, una página que contiene enlaces útiles hacia la autorizada, lo que ayuda al motor de búsqueda que apunte en la dirección correcta.

HITS clasifica las páginas que son bienes de páginas autorizadas y que son buenos contenedores. Esta se realiza mediante la asignación de un peso de autoridad y un peso de contenedor para cada página, dependiendo del número de veces que una página se señaló.

Con esta clasificación Kleinberg construye subgrafos dirigidos focalizados a la web $G=(V,E)$, donde los nodos se corresponden con las páginas y una arista $(p,q)\in E$ es un enlace de la página p a q. Este gráfico toma en consideración a los pesos de los arcos como se ha indicado anteriormente. El algoritmo comienza con un subgrafo "raíz" que contiene páginas con unas altas apariciones de las palabras de búsqueda, y luego otro subgrafo se construye con todas las aristas que salen y entran del subgrafo "raíz", esto se llama subgrafo "semilla" y probablemente tienen una gran autoridad sobre las páginas con el tema. HITS actualiza dinámicamente los pesos de la "semilla" basada en que un buen centro aumenta el peso de la autoridad de las páginas que apunte, y una buena autoridad aumenta el peso de las páginas que apuntan a ella.

Este algoritmo toma la teoría de grafos al modelado de páginas web y sus conexiones,
ha sido útil para Análisis de Redes Sociales en el descubrimiento de los miembros clave. Como se indica en (39), HITS puede ser utilizado en las redes sociales, al igual que con la noción de autoridad se puede medir la utilización, impacto y la influencia de los miembros de una comunidad.

PageRank

PageRank es el algoritmo que utiliza el motor de búsqueda de Google en la Web. Este algoritmo sigue las mismas ideas básicas del algoritmo HITS utilizando la estructura de enlaces del gráfico de malla para hacer un sistema de clasificación de cada página web en relación con un determinado tema de búsqueda. La diferencia es que HITS utiliza el subgrafo "semilla" para cada consulta que cambia con una búsqueda diferente, por lo tanto, la clasificación de los Nodos de la colección (páginas web) también cambian.

A diferencia de HITS, PageRank utiliza la estructura de enlaces web como un grafo dirigido.
El algoritmo modela la navegación web teniendo en cuenta que un usuario puede navegar a través de una página siguiendo sus enlaces, pero de repente cambiar a otra página con una pequeña probabilidad pero positiva. Para recuperar este comportamiento, Brin y Page añaden una constante p a la matriz PageRank, M, como sigue:

$$M=A\,(1-p\,)-B_{p}$$

donde p es la probabilidad constante mencionada llamada factor de amortiguamiento. A es la adyacencia, matriz ponderada del grafo y para un número n de las páginas web. Por lo tanto, cada página tiene probabilidades de ser elegidos (39).

Entonces, el algoritmo sigue a la tarea de calcular el vector PageRank, que es el único vector propio correspondiente al valor propio 1 de la Matriz PageRank hasta que converge, lo que resulta en un ranking de páginas web. Como se puede imaginar, este sistema de clasificación se ha utilizado en el análisis de redes sociales para ordenar los miembros de las comunidades por la forma en que el algoritmo trata a la red web como un grafo, de manera similar a una red comunitaria.

1.5 Tecnologías y herramientas utilizadas para el desarrollo de la aplicación

Como resultado del proceso investigativo de las principales tecnologías y herramientas para el desarrollo del sistema informático, se realiza la siguiente selección de las mismas, teniendo como principales criterios las altas prestaciones que brindan y por encontrarse dentro de las herramientas y tecnologías no privativas. A continuación, se presenta la selección realizada:

1.5.1 Metodología de desarrollo

Las metodologías de desarrollo marcaron un hito en la historia de la producción de software, ya que con las mismas se logra organizar el proceso de desarrollo y a su vez posibilita el trabajo en equipo y la asunción de proyectos de mayor envergadura y complejidad.

Se define la metodología como: *"El conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal."* (40)

Para producir un software de manera eficiente, que cumpla con los requisitos del cliente, es necesaria una metodología que sirva como guía para todos los participantes en el proceso de desarrollo. A partir de esta necesidad se aplicó el método de Boehm y Turner (41), el cual a través del análisis de sus cinco criterios que son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta, permitió determinar que la metodología más adecuada para el desarrollo del sistema debería ser una metodología ágil.

En la siguiente imagen se representa cómo se comportan los 5 criterios que propone este método en el entorno de desarrollo del sistema, así como también cual es el terreno más adecuado en el cual se debe encontrar la metodología de desarrollo de software a utilizar.

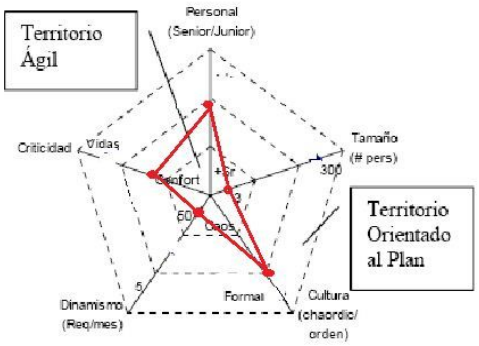


Figura 1.4: Representación de la estrella de BOEHM y Turner para el desarrollo del sistema

Para el desarrollo de esta tesis se ha seleccionado como metodología de desarrollo Agile Unified Process (AUP). Ya que proporciona un ambiente de desarrollo de software iterativo e incremental. A menudo, es calificado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto de software, pero dado que es un marco de procesos, puede ser adaptado ya que es muy semejante a RUP, AUP propone los mismos roles, artefactos de RUP, pero en una versión simplificada; sólo se utilizan los artefactos que son imprescindibles y realmente necesarios para la realización del producto. Además, AUP adopta muchas de las técnicas ágiles de “ P y otros procesos ágiles, pero aún mantienen la formalidad de RUP.

De igual forma AUP se preocupa especialmente de la gestión de riesgos, es decir, propone para aquellos elementos con alto riesgo, prioridad en el proceso de desarrollo y que sean abordados en etapas tempranas del mismo.

(AUP) es una versión simplificada del Proceso Unificado de Desarrollo (RUP). Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software, usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP puede verse más simple que las versiones más clásicas del proceso unificado, mientras que, desde el punto de vista ágil, puede entenderse como una versión más pesada que las metodologías ágiles. (42).

La utilización de AUP ofrece numerosas ventajas al equipo de desarrollo, entre ellas se pueden mencionar (42):

- **El personal sabe lo que están haciendo**. Las personas no van a leer documentación de procesos detalladamente, pero quieren alguna guía de alto nivel o capacitaciones de vez en cuando. Este producto proporciona vínculos a muchos de los detalles, si está interesado, pero no los obliga ni los impone.
- **Simplicidad**. Todo se describe brevemente utilizando pocas páginas y no miles de ellas.
- **Agilidad**. Permite responder de manera rápida y apropiada a los cambios.
- **Enfocar las actividades de alto nivel**. Se centra en las actividades que realmente cuentan, no en cada cosa posible que podría pasarle al proyecto.
- **Independencia de Herramientas**. Se puede usar el conjunto de herramientas que se desee con AUP.

La siguiente figura representa a AUP en sus dos dimensiones:

- El eje horizontal representa el tiempo en las cuatro fases en que se descompone el proceso.
- El eje vertical representa la serie de flujos de trabajo que lo construyen gradualmente.

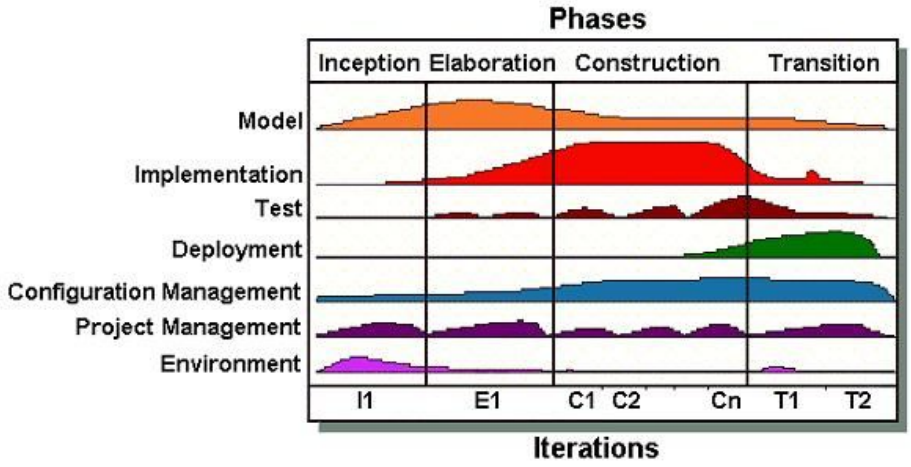


Figura 1.5: Proceso Unificado Ágil

Como se aprecia en la Figura 1.5, AUP está compuesto por cuatro Fases que son: (42)

- **Inicio**: El objetivo es identificar el alcance inicial del proyecto, una arquitectura base del sistema, obtener una primera propuesta de presupuesto del proyecto, así como una aceptación de los involucrados.
- **Elaboración**: El objetivo es establecer un plan de proyecto y una arquitectura correcta del sistema.
- **Construcción**: El objetivo es construir un software funcional sobre una base regular e incremental, las cuales cumplan con las prioridades más importantes para los involucrados o usuarios del proyecto.
- **Transición**: El objetivo es validar y desplegar el sistema en su ambiente de producción.

Como se aprecia en la Figura 1.5, AUP está compuesto por siete Disciplinas o Flujos de Trabajos que son (42):

- **Modelado:** El objetivo de esta disciplina es entender el negocio de la organización e identificar una solución viable para abordar el dominio del problema.
- **Implementación:** Este método se encarga de transformar sus modelos en código ejecutable y realizar pruebas en un nivel básico.
- **Prueba:** Consiste en ejecutar una evaluación de los objetivos para asegurar la calidad. Esto incluye encontrar defectos, validar que el sistema función como fue diseñado y verificar que los requerimientos están completos.
- **Despliegue:** Planifica la entrega del sistema y ejecutar el plan para que el sistema esté a disposición de los usuarios finales.
- **Gestión de configuración:** Se dedica administrar el acceso a los entregables o productos del proyecto. Esto incluye no sólo el rastreo de versiones del producto en el tiempo, sino que también incluye controlar y administra los cambios que ocurran.
- **Gestión de proyectos:** Su misión es dirigir las actividades que se llevan a cabo en el proyecto. Esto incluye administración del riesgo, la dirección de personas (asignar tareas, seguimiento de los procesos, etc.), y coordinar con los sistemas y personas fuera del alcance del proyecto para que este termine a tiempo y dentro del presupuesto.
- **Entorno:** Su tarea es apoyar el resto de los esfuerzos por garantizar que, el proceso adecuado, la orientación (normas y directrices) y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario.

1.5.2 Lenguaje de Modelado

Como lenguaje de modelado se seleccionó el Lenguaje Unificado de Modelado (UML), el cual se representa de manera visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas (43).

La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Da soporte de apoyo a la mayoría de los procesos de desarrollo orientados a objetos. UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo (43).

1.5.3 Herramienta Case:

Visual Paradigm V 9.0

La Ingeniería de Software Asistida por Computadora (CASE por sus siglas en inglés), constituye la aplicación de métodos y técnicas utilizadas para ayudar a las actividades del proceso del software, como el análisis de requerimientos, el modelo de sistemas, la depuración y las pruebas. Las herramientas CASE representan una forma que permite modelar los procesos de negocios de las empresas y desarrollar los sistemas de información.

Visual Paradigm es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Utiliza UML como lenguaje de modelado. Es una herramienta que genera la documentación del proyecto automáticamente en varios formatos como Web o PDF, soporta la realización de ingeniería tanto directa como inversa. Presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas que soportan la Ingeniería de Requerimientos. Dentro sus características están su robustez, usabilidad y portabilidad (44).

1.5.4 Herramienta de Desarrollo (IDE)

PyCharm V 4.5

PyCharm es un Entorno de Desarrollo Integrado (acrónimo de Integrated Development Enviroment, IDE por sus siglas en inglés) que tiene un sistema de licencia dual desarrollado por JetBrains en el lenguaje Java para el desarrollo de aplicaciones en Python, PyPy, Iron Python, Jython, Django, Flask, Pyramid, Web2Py, entre otros. Una herramienta pensada para escribir, compilar, depurar y ejecutar programas, donde existe, además, un número importante de módulos para extenderla.

PyCharm presenta un editor de códigos inteligente, que entiende los detalles específicos de Python y ofrece extraordinarios mejoradores de la productividad: formateo automático de código, finalización de código, refactorizaciones, importación automática, navegación de código con un solo clic, y mucho más. Respaldadas por avanzadas rutinas de análisis de código, estas características hacen de PyCharm una poderosa herramienta tanto en las manos de los desarrolladores Python profesionales como en las de quienes recién comienzan a usar la tecnología. Tiene además la posibilidad de auto chequear el código escrito para que cumpla con los estándares y la guía de estilos de la comunidad Python, comúnmente conocido como PEP8 (63).

1.5.5 Lenguaje de Programación

Python 2.7

Para el desarrollo de esta aplicación se seleccionó a Python como lenguaje de programación ya que este tiene un grupo de ventajas.

- **Repositorio de librerías:** Existen repositorios donde se almacenan librerías que se utilizan para el desarrollo de las aplicaciones.

- **Multiplataforma:** Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

- **Tipado Dinámico:** No es necesario definir el tipo de datos para las variables que se declaran en la aplicación, se puede simplemente haberla creado, asignado un entero y posteriormente asignarle un valor booleano sin que exista un error.

- **Funciones y librerías:** Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos.

- **La curva de aprendizaje es más baja:** El lenguaje Python es sencillos de aprender.

- **Gran comunidad de colaboradores y documentos:** Existe en internet una gran cantidad de documentación y de foro.

- **Gratuito:** Una ventaja fundamental de Python es la gratuidad de su intérprete, se puede descargar desde la página web: http://www.Python.org.

1.5.6 Framework de desarrollo web Django 1.8.12

Un framework (marco de trabajo) es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Django es un framework web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código. Se centra en automatizar todo lo posible y se adhiere al principio DRY (DonÁt Repeat Yourself, o no te repitas a ti mismo). Mantiene de forma rigurosa un diseño limpio en su propio código. Fomenta el bajo acoplamiento: filosofía de programación que dice que las distintas partes de la aplicación deben ser intercambiables.

1.5.7 HTML5

HTML es un lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto,

imágenes, entre otros. Es un estándar a cargo de la World Wide Web Consortium (W3C), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación (45).

HTML5 es la quinta versión importante del estándar e introduce algunos elementos que constituyen ventajas de su empleo y hacen que se ajuste a las necesidades actuales de la web. Por otro lado, algunos aspectos de diseño también son incluidos en el lenguaje, así como también algunos detalles de navegación. Dentro de sus ventajas sobresalen la compatibilidad y facilidad que plantea su aprendizaje debido al reducido número de etiquetas en las que se apoya. Con el uso de HTML5, se puede reducir la dependencia de los plugins que se tienen que tener instalados para poder ver una determinada página o sitio web (45).

1.5.8. CSS3

CSS (Cascading Style Sheets, u Hojas de Estilo en Cascada) es la tecnología desarrollada por el W3C con el fin de separar la estructura de la presentación. Es un lenguaje constituido a base de selectores, propiedades y valores que tiene como objetivo dar aspecto visual al código HTML (46). La novedad más importante que aporta CSS3, como tercera versión importante de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complicaban el código de la web (47).

1.5.9 Bootstrap 3.0

Bootstrap es un framework de código abierto cuyo objetivo es facilitar el desarrollo de aplicaciones o páginas web teniendo una colección de plantillas CSS, HTML y plugins JavaScript.

Los diseños creados con Bootstrap son simples, limpios e intuitivos. Con sólo agregar algunas clases y el markup correcto se pueden lograr casi sin esfuerzo grupos de botones, barras de navegación, dropdowns, formularios, etc., sin tener que escribir una línea de código CSS. Las aplicaciones que utilizan Bootstrap adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, esto se denomina diseño adaptativo. Esto significa que el diseño gráfico de la página se ajusta dinámicamente,

tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles). Diseñado por expertos para algunos de los problemas más difíciles de diseño web moderno (48).

En esta investigación es utilizado principalmente para el maquetado de los componentes, aunque también se utilizan las demás funcionalidades que el mismo proporciona, pensando en la compatibilidad con los distintos navegadores y dispositivos en que pueda ser utilizada la solución.

1.5.10. JavaScript V 1.7

JavaScript es uno de los lenguajes de script u orientado a documento, más usados en Internet. Creado por la empresa Netscape9, para añadir interactividad a las páginas web. Es compacto y basado en objetos, diseñado para el desarrollo de aplicaciones cliente-servidor a través de Internet (49).
Es un lenguaje interpretado e independiente de la plataforma que permite dar respuestas a eventos iniciados por el usuario, como la entrada de datos en un formulario o la elección de algún enlace del documento HTML. La principal ventaja de JavaScript es que este proceso sucede sin ningún tipo de transmisión de datos a través de Internet, de tal forma que cuando un usuario escribe algo en un formulario, no es necesario que sea verificado, devuelto y enviado al servidor (49).

1.5.11. Gestor de Base de Datos PostgreSQL 9.4

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar la definición de los datos, el mantenimiento de la integridad de éstos dentro de la base de datos, el control de su seguridad y privacidad; así como la manipulación de los mismos (50).

PostgreSÑL es un SGBD relacional, robusto y confiable. Soporta gran variedad de sistemas operativos como Linux y UNI“ en la mayoría de sus variantes y Windows 32/64bit. Se caracteriza por ser un sistema de alto rendimiento y gran flexibilidad, lo que permite a los desarrolladores generar nuevas aplicaciones o mantener las existentes. Se desempeña mejor en ambientes con altas cargas de usuario y consultas complejas, donde la integridad de los datos es muy importante (51).

Teniendo en cuenta los aspectos mencionados anteriormente se selecciona dicha herramienta, en su versión 9.0, para gestionar la base de datos del sistema.

1.5.12. PgAdmin III

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SÑL simples hasta desarrollar bases de datos complejas. Su interfaz gráfica soporta todas las características de PostgreSÑL y facilita enormemente la administración del mismo.

La aplicación también incluye un editor SÑL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados y mucho más. Fue desarrollado por una comunidad de expertos en PostgreSÑL alrededor de todo el mundo y está disponible en más de una docena de idiomas. Es multiplataforma y puede funcionar bajo GNU / Linux, FreeBSD, Mac, Windows y Solaris (52).

1.5.13 Apache 2.4

Es un servidor web, altamente configurable de diseño modular, gratuito y multiplataforma. Es un software robusto y estable que proporciona la confianza de todos aquellos que lo utilizan. Es uno de los servidores web más populares del mundo y su condición de software multiplataforma le permite funcionar en sistemas operativos privativos y de código abierto.

1.6 Conclusiones parciales

- El análisis de las soluciones existentes sirvió de base para la conceptualización y posterior desarrollo de un sistema para la detección y visualización de comunidades científicas.

- El análisis de los principales conceptos asociados al dominio del problema permitió tener un mejor entendimiento y comprensión de las temáticas que conforman la investigación, así como de los factores que influyen en la situación problemática actual.

- El uso de herramientas y tecnologías, como las seleccionadas para el desarrollo de la solución, permitirá lograr mayores resultados y garantizará la obtención de un producto de calidad que satisfaga las necesidades del cliente.

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se detallan las características que debe tener el sistema a desarrollar, a partir de la descripción general de la propuesta de solución y los procesos que serán objeto de automatización, se crea el modelo de dominio e identifican los requisitos funcionales y no funcionales del sistema.

2.1 Propuesta de solución

La solución propuesta en la presente investigación es una aplicación WEB, que pone a disposición de las personas e instituciones una herramienta que detectará las comunidades científicas y sus líderes. El sistema debe ser capaz de reconocer ficheros con las extensiones GML, PICKLE, PAJEK, GraphML o NET. A los ficheros se le aplicará el algoritmo de clusterización (Fastgreedy, Label Propagation o Walktrap) que seleccione el usuario. Posteriormente se utilizará el algoritmo Pagerank para detectar los líderes de comunidades. Finalmente se mostrarán los resultados en canvas dinámicas, que representan las comunidades detectadas y sus respectivos líderes, a través de estructura de grafo. La aplicación también debe exportar los resultados en un documento PDF o imágenes en formato png.

2.2 Modelo de dominio

Un modelo de dominio es: {*Una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés*} (53).

En el modelo de dominio están representados los principales conceptos y las relaciones entre ellos, permitiendo la comprensión de los conceptos fundamentales y el contexto en el que se desarrolla el sistema.

Un gestor de información actualiza la base de datos con formato de grafos. El investigador tiene la misión de realizar el proceso de analizar los elementos tomados de la base de datos de forma manual. A partir de los resultados obtenidos en el análisis se generan los reportes que contienen las comunidades científicas y sus respectivos líderes; apoyando así a la toma de decisiones.

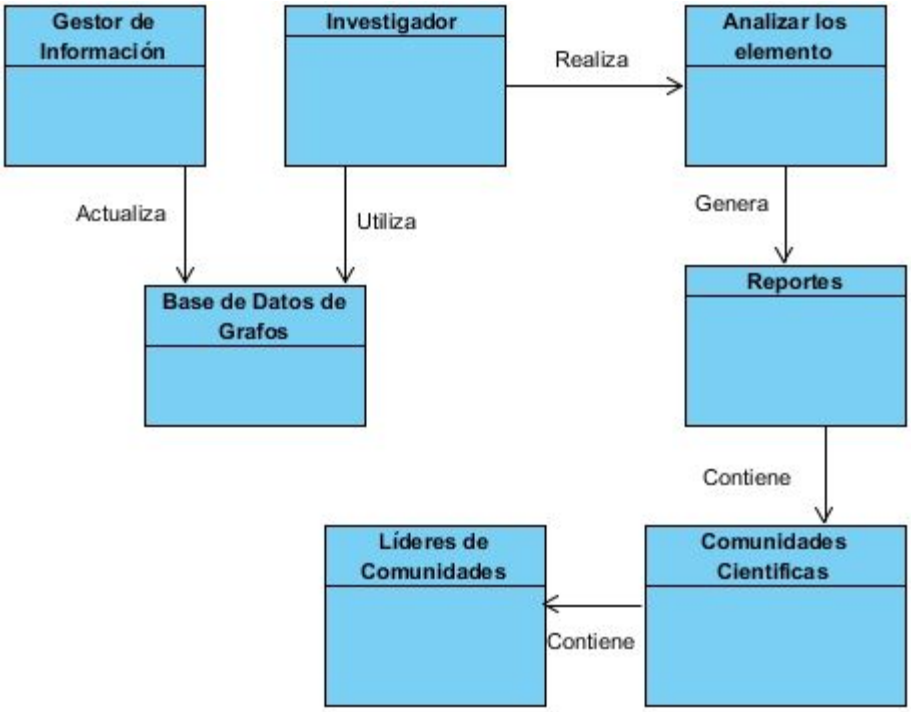


Figura 2.1 Modelo de Dominio.

2.2.1 Descripción del modelo conceptual

Gestor de Información: Persona o herramienta encargada de actualizar la base de datos de grafos para ser utilizadas por el investigador.

Base de Datos de Grafos: Es la base de datos que contiene la información asociada a los archivos insertados por el gestor de información, que poseen un formato y extensión específica.

Investigador: Persona encargada de utilizar los elementos la base de datos de grafos y realizar el análisis de forma manual.

Análisis de elementos: Es el empleo de los algoritmos por el investigador para obtener el resultado que arroje la investigación.

Reporte: Es un documento escrito en Word y exportado a PDF donde se salvan los resultados devueltos a partir del análisis de los elementos.

Comunidades Científicas: Son subredes del grafo general que representan a los autores asociados por temas de investigación y son generadas a partir del análisis de los elementos.

Líderes de Comunidades: Es la persona con mayor importancia en cada una de las Comunidades Científicas.

2.3 Relación de los requerimientos

2.3.1 Listado de los requerimientos funcionales

Los requisitos funcionales constituyen las capacidades o condiciones que el sistema debe cumplir. Estos deben ser comprensibles por los clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos en forma medible y verificable (54).

Para la obtención de los requisitos funcionales se utilizó el método de entrevista (descrita en la introducción) determinándose los siguientes requisitos:

RF1: Cargar fichero a la base de datos de grafos

1.1. Cargar fichero a la base de datos de grafos con extensión GML.

1.2. Cargar fichero a la base de datos de grafos con extensión PICKLE.

1.3. Cargar fichero a la base de datos de grafos con extensión PAJEK.

1.4. Cargar fichero a la base de datos de grafos con extensión GraphML.

1.5. Cargar fichero a la base de datos de grafos con extensión NET

RF2: Listar ficheros

RF3: Analizar la información del fichero.

RF4: Exportar resultados del análisis de redes sociales

4.1: Exportar resultados del análisis de redes sociales en archivos de tipo PDF.

4.2: Exportar los resultados del análisis de redes sociales en archivos de tipo PNG.

2.3.2 Definición de los requerimientos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe tener. Estas propiedades o cualidades son las características que hacen al sistema atractivo, usable y confiable (54).

Los requisitos no funcionales que el sistema debe cumplir son:

Apariencia o interfaz externa.

- **RNF 1**. Los colores utilizados en el diseño de las interfaces deben ser: negro, blanco y azul.
- **RNF 2**. Todos los textos y mensajes en pantalla aparecerán en idioma español.

Usabilidad.

- **RNF 3**. El sistema contara con botones que describan de forma clara las funciones.

Software.

Servidor

- **RNF 4.** El servidor requiere un Python 2.7 o superior, las librerías Pattern 2.6 y Igraph 0.7, servidor apache 2.4 y Django 1.8

Estaciones de trabajo o PC Cliente

- **RNF 5.** En las estaciones de trabajo se requiere tener instalado un navegador web.

Hardware.

Servidor

- **RNF 6.** El servidor requiere un disco duro que contenga 10 GB.
- **RNF 7.** El servidor requiere 2 GB de memoria RAM.
- **RNF 8.** El servidor requiere un procesador Dual Core a 1.6GHz.

Estaciones de trabajo o PC Cliente

- **RNF 9.** En las estaciones de trabajo se requiere 256 MB de memoria RAM.
- **RNF 10.** En las estaciones de trabajo se requiere un procesador a 1GHz.

2.4 Modelo de Casos de Uso del Sistema

El modelo de casos de uso se define como el conjunto de todos los casos de uso; es un modelo de la funcionalidad y entorno del sistema (53).

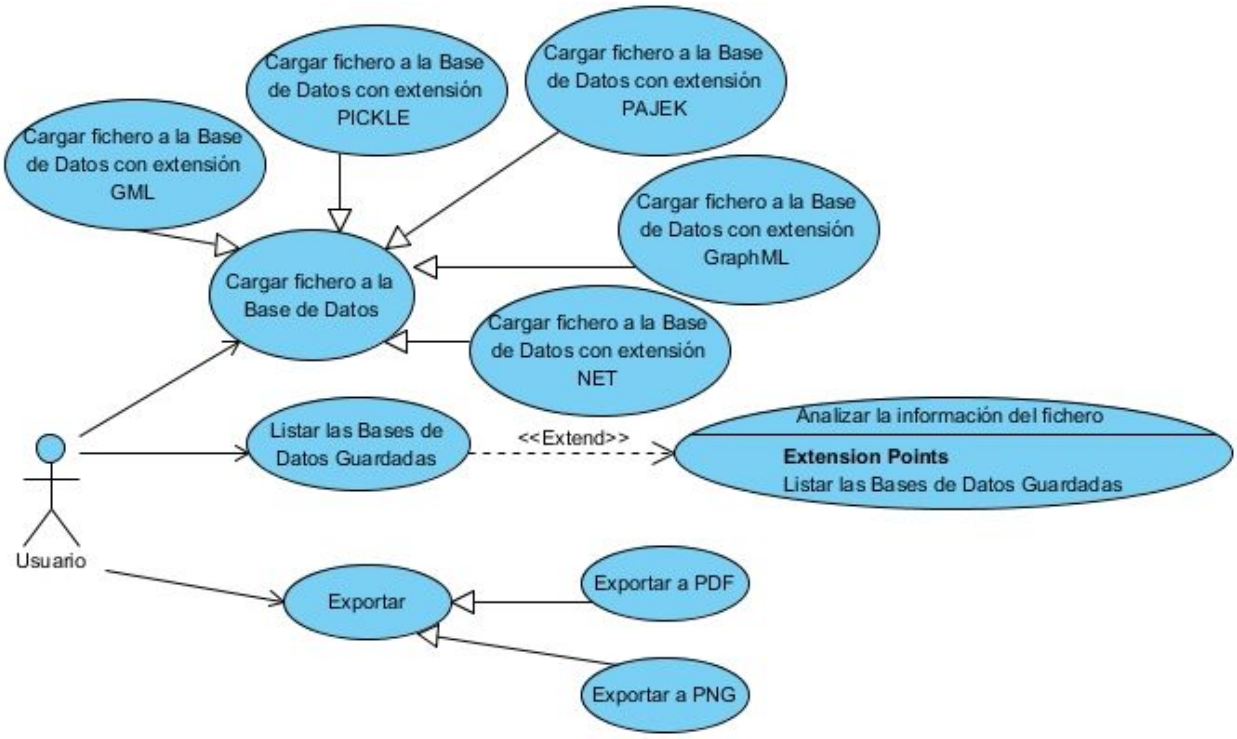


Figura 2.2 Diagrama de Casos de Uso

2.5 Especificación de los Casos de Uso del Sistema

Mediante la especificación de los casos de uso se puede describir paso a paso la secuencia de eventos que los actores utilizan para completar un proceso a través del sistema. A continuación, se presenta la descripción del caso de uso Cargar fichero a la Base de Datas Bibliográfica que es el encargado de iniciar el flujo principal del sistema.

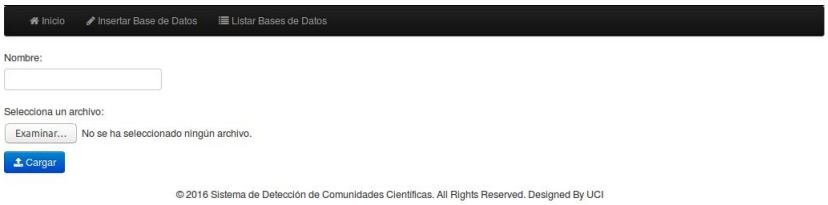
2.5.1 Descripción del CU Cargar fichero a la Base de Datos

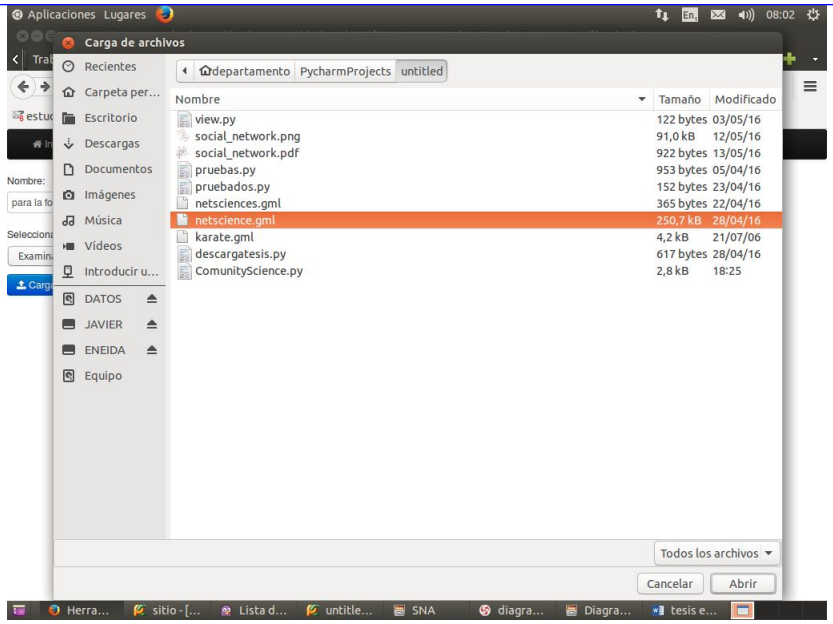
Objetivo	Cargar un fichero a la Base de Datos.
Actores	Usuario
Resumen	El CU comienza cuando el usuario desea cargar una base de

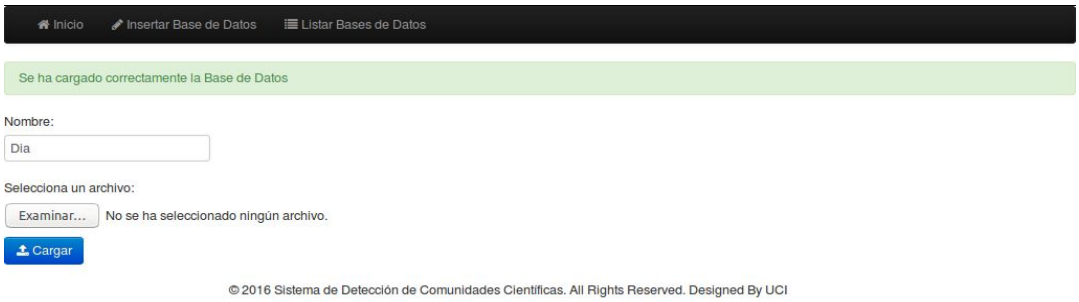
	datos con una estructura de grafos en el sistema. En dependencia de la extensión que posea la base de datos, la aplicación lee la extensión y verifica si es una extensión válida. Cuando el sistema reconoce la extensión de la base de datos guarda la información en la base de datos y termina el CU.
Complejidad	Baja
Precondiciones	El fichero cargado a la aplicación es una base de datos que debe tener extensión GML, PICKLE, PAJEK, GraphML o NET.
Postcondiciones	

Flujo de eventos	
Flujo Básico	
Actor	Sistema
	1. El sistema muestra una interfaz de bienvenida con un Menú que contiene las siguientes opciones: Inicio (página de Bienvenida) Insertar Base de Datos (Permite cargar en el sistema una base de datos con extensión GML, PICKLE, PAJEK, GraphML o NET). Listar Bases de Datos
2. El usuario selecciona la opción {Insertar Base de Datos}.	2. El sistema muestra la interfaz para escribir nombre y escoger el archivo que se desea cargar. Y permite: ➢ Cargar el fichero
3. El usuario escribe el nombre, busca el archivo que desea subir al servidor y oprime el botón {Cargar }	3. El sistema verifica que el archivo seleccionado esté en alguno de las siguientes extensiones: GML, PICKLE, PAJEK, GraphML o NET.
	4. El sistema identifica la extensión del fichero.
	5. El sistema re-direcciona a la página Insertar Base de Datos muestra el siguiente mensaje de

	información: {La base de datos ha sido cargada satisfactoriamente.}	
Relaciones	CU Incluidos	No
	CU Extendidos	No
Prototipo elemental de interfaz gráfica de usuario		




Aplicaciones Lugares
08:02
Carga de archivos
Recientes
Carpeta per...
Escritorio
Descargas
Documentos
Imágenes
Música
Vídeos
Introducir u...
DATOS
JAVIER
ENEIDA
Equipo
departamento
PycharmProjects
untitled
Nombre
Tamaño
Modificado
view.py 122 bytes 03/05/16
social_network.png 91,0 kB 12/05/16
social_network.pdf 922 bytes 13/05/16
pruebas.py 953 bytes 05/04/16
pruebados.py 152 bytes 23/04/16
netsciences.gml 365 bytes 22/04/16
netscience.gml 250,7 kB 28/04/16
karate.gml 4,2 kB 21/07/06
descargatesis.py 617 bytes 28/04/16
ComunityScience.py 2,8 kB 18:25
Todos los archivos
Cancelar
Abrir


Inicio
Insertar Base de Datos
Listar Bases de Datos
Se ha cargado correctamente la Base de Datos
Nombre:
Dia
Selecciona un archivo:
Examinar...
No se ha seleccionado ningún archivo.
Cargar
© 2016 Sistema de Detección de Comunidades Cientificas. All Rights Reserved. Designed By UCI

Para acceder al resto de las descripciones de los casos de uso de sistema (ver Anexo 1).

2.6 Conclusiones

- La construcción del modelo de dominio permitió que se pudiera tener un mayor entendimiento del negocio, lo cual sirvió como punto de partida para identificar las funcionalidades que el sistema debe tener para satisfacer las necesidades del cliente.
- El diagrama de casos de uso permitió tener una vista global del sistema a desarrollar.
- La solución propuesta consiste en una aplicación web que automatiza el proceso detección y visualización de comunidades científicas en la UCI, posibilitando conocer los principales líderes en las distintas áreas de investigación de la universidad.

Capítulo 3 DISEÑO DEL SISTEMA

En el presente capítulo se tratan los temas relacionados con el diseño de la propuesta de solución, se describe la arquitectura base y se muestran algunos de los artefactos generados en la etapa del modelado que guiarán la implementación del sistema.

El objetivo del diseño es lograr una representación arquitectónica de un sistema, que a su vez pueda servir como marco de trabajo para llevar a cabo las actividades del diseño más detalladas (55).

3.1 Arquitectura del Sistema

"La arquitectura del software alude a la estructura global del software y a las formas en que la estructura proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura es la estructura jerárquica de los componentes del programa (módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. Sin embargo, en un sentido más amplio, los componentes se pueden generalizar para representar los elementos principales del sistema y sus interacciones" (55).

3.2 Diseño del sistema

3.2.1 Patrones Arquitectónicos

Un patrón arquitectónico define la estructura básica de una aplicación, provee un conjunto de subsistemas predefinidos, incluyendo reglas, lineamientos para conectarlos y pautas para su organización. Constituye una plantilla de construcción.

Los patrones arquitectónicos expresan una organización estructural para un sistema, permiten estructurar los componentes y subsistemas del mismo. La abstracción más alta en cuanto al uso de patrones, se obtiene a través de patrones arquitectónicos.

Modelo-Vista-Controlador

El patrón Modelo-Vista-Controlador (MVC) según su nivel de abstracción se puede clasificar como un patrón de arquitectura. Este tipo de patrón especifica una serie de subsistemas y sus responsabilidades respectivas e incluye las reglas y criterios para organizar las relaciones existentes entre ellos (56).

EL patrón MVC está acorde con la concepción inicial del sistema, este ayudará a obtener una estructura bien definida y se logrará que el proyecto esté organizado.

Este patrón está formado por tres niveles (57):

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Como ya se había planteado, para el desarrollo de la aplicación se empleará el framework Django, el cual a pesar de que hace uso intensivo del patrón arquitectónico MVC, realiza algunas modificaciones a la forma de llamar sus elementos y respectivas funciones y establece su propia filosofía para interpretarlo (58). Django toma al modelo tal y como es, pero la forma de ver la vista y el controlador varía de cierta manera. Los desarrolladores de Django identifican en su arquitectura los mismos beneficios que en el MVC, pero orientados a lograr un mayor énfasis hacia el desarrollo ágil, y después de tener un conjunto de herramientas informáticas que forman un framework coherente y dan sentido a su flujo de trabajo, culminaron sus esfuerzos con algo a lo que han denominado patrón Modelo-Plantilla-Vista o Model-Template-View (MTV), por sus siglas en inglés.

Una vista en MTV, comparte el nombre y algunos elementos con la definición original del MVC de vista, sin embargo, tiene más en común con el controlador, puesto que se encarga de realizar las tareas de la lógica, combinando algunas responsabilidades de las vistas tradicionales con la integridad de las tareas del controlador. Una vista acepta las entradas del usuario (incluyendo una simple petición de información), las manipula en concordancia con la lógica de la aplicación y muestra un retorno accesible para que los usuarios puedan ver la información representada por modelos. Las vistas son definidas normalmente como funciones estándar de Python que son llamadas cuando un usuario requiere una url específica. En términos de la web, incluso la más pequeña petición de información es considerada una acción, por eso las vistas son manipuladas de forma tal que permitan modificaciones de los datos a enviar en torno a la petición. Las vistas tienen acceso a los modelos, consultando y modificando la información necesaria para cumplir la tarea requerida por el usuario (58).

Por otra parte, una plantilla, es la encargada de presentar los datos al usuario, la tarea de cómo se presentan los mismos es tarea de la plantilla. Las plantillas en Django tienen la importancia suficiente para considerarlas una parte esencial del framework y una capa independiente que interactúa con el resto de las capas del patrón. Django provee a las plantillas de un lenguaje simple por encima del HTML para cumplir su propósito de manejar los detalles de cómo es mostrada la información a los usuarios, pudiendo incluso aplicar lógica en la presentación desde el servidor (58).

3.2.2. Patrones de diseño

Un patrón de diseño puede ser caracterizado como una regla de tres partes, la cual expresa una relación entre cierto contexto, un problema y una solución. Para el diseño de software, el contexto permite a quien usa el patrón entender el entorno en el cual reside el problema y qué solución pudiera ser apropiada dentro de ese entorno. Un conjunto de requisitos, incluyendo limitaciones y restricciones, actúan como un sistema de fuerzas que influencia en cómo el problema puede ser interpretado dentro de su contexto y cómo la solución puede ser aplicada efectivamente (59).

Los patrones de diseño hacen que sea más fácil reutilizar buenos diseños y arquitecturas. Los patrones de diseño ayudan a elegir las alternativas de diseño que hacen que un sistema sea reutilizable, y evitan aquellas que dificultan dicha reutilización. Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. En definitiva, los patrones de diseño ayudan a un diseñador a lograr rápidamente un buen diseño (60).

Patrones GRASP

Los patrones General Responsibility Assignment Software Patterns (en español Patrones Generales de Software para la Asignación de Responsabilidades) (GRASP) son parejas de problema solución con un nombre, que codifican buenos principios relacionados frecuentemente con la asignación de responsabilidades. Estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, garantizando la máxima capacidad de reutilización del código (61).

A continuación, se reflejan los patrones GRASP empleados en el diseño del sistema:

Patrón Experto: Se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

Patrón Creador: Las clases que tienen la responsabilidad de crear objetos contienen toda la información necesaria para construir los mismos. Se pone de manifiesto al utilizar las class-based-views [3]de Django.

Patrón Alta Cohesión: Se aplica en la mayoría las clases del diseño, ya que en cada una solo se implementan las funcionalidades que le corresponden. El patrón alta cohesión se puede evidenciar en las plantillas encargadas de mostrar los datos al usuario y en las clases manejadoras de los formularios.

Patrón Bajo Acoplamiento: Cada clase se comunica con un número mínimo de clases posible. El patrón bajo acoplamiento se pone de manifiesto en la relación presente entre las clases del sistema, al establecer un número mínimo de dependencias entre estas, un ejemplo de ellas puede ser la de los formularios, vistas, modelos y plantillas donde para cambiar el contenido presente en la plantilla basta con editar el contenido del formulario sin necesidad de cambiar el contenido del resto de las otras clases.

Patrón Controlador: Está representado por una clase a la cual se le asigna la responsabilidad de las operaciones del sistema, ofrece también una guía para tomar decisiones apropiadas. Este patrón se refleja en la clase views que es la encargada de controlar las acciones que realiza el usuario con la interfaz de la aplicación y dar respuesta a las peticiones realizadas.

Patrones GoF

Los patrones Gang of Four (en español Banda de los cuatro) (GoF) surgen como una forma indispensable de enfrentar los problemas propios de la codificación. Fueron dados a conocer en el libro {Design Patterns–Elements of Reusable Software} de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides. El empleo de los mismos garantiza una solución efectiva y a prueba de fallos para cada problema en cuestión, que no puede ser solucionado por las vías normales (60).

---

[3] Las class-based-views son vistas que permiten realizar de forma sencilla, las funcionalidades básicas en casi todos los sistemas (Crear, Actualizar, Eliminar, Listar)

A continuación, se reflejan los patrones GoF empleados en el diseño del sistema:

Patrón Front Controller (Controlador Frontal): Django posee una implementación de Controlador Frontal que despacha las peticiones hacia métodos o clases, que en la práctica son páginas controladoras. Antes del despacho, la petición es procesada por varios filtros (middlewares).

Patrón Madiator (Mediador): Encargado de manejar la interacción entre los diferentes subsistemas. Realiza el mapeo objeto–relacional (ORM) a cargo del motor de Django. Simplifica y facilita los resultados de las consultas realizadas a la base de datos, así como brinda varias funcionalidades adicionales, que consultas SÑL tradicionales no proveen.

3.3 Diseño del Sistema

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, tanto funcionales como no funcionales y las restricciones que se le imponen. Es usado como entrada esencial en las actividades relacionadas a la implementación y representa a los casos de uso en el dominio de la solución.

3.3.1 Modelo del diseño

"El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso y se utiliza como abstracción del modelo de implementación y el código fuente} (55).

Está conformado por otros modelos. La metodología AUP utilizada en el desarrollo de la solución, plantea que no es necesario crear todos los modelos que conforman al modelo de diseño, solamente los que sean apropiados al contexto donde se aplica. Los modelos creados para la implementación del sistema se relacionan a continuación.

Diagrama de Paquetes

Los diagramas de paquetes muestran la descomposición jerárquica lógica de un sistema, así como las dependencias entre las partes lógicas que lo conforman. A continuación, se muestra el diagrama de paquetes del sistema, en el cual se puede apreciar que la estructura organizativa de los paquetes responde a la arquitectura definida para el desarrollo de la herramienta.

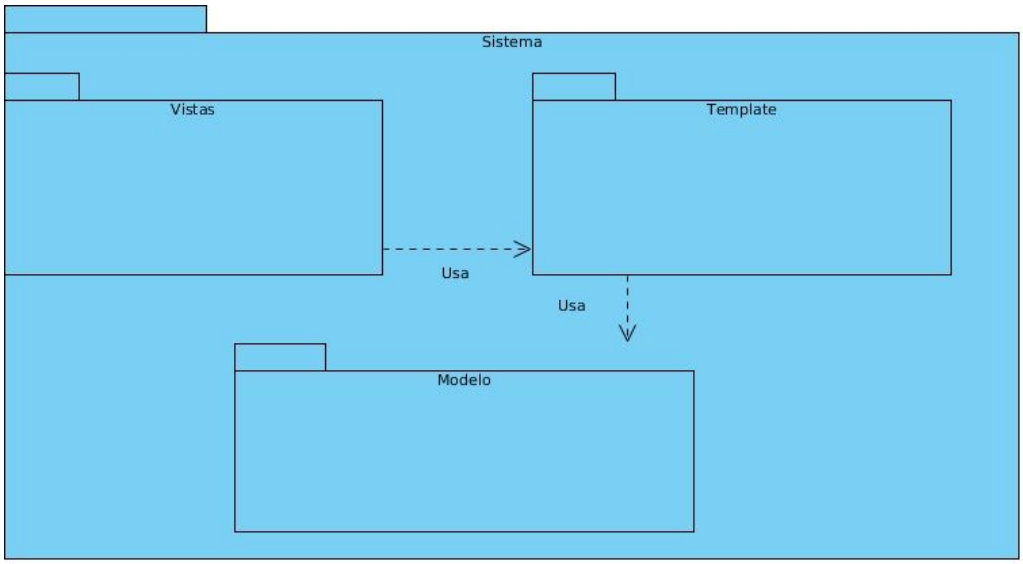


Figura 3.1 Diagrama de Paquetes

Diagrama de Clases

Los diagramas de clases son diagramas estáticos que exponen las diferentes clases que componen un sistema y cómo se relacionan entre ellas. Sirven para mostrar tanto lo que el sistema puede hacer (análisis), como de qué manera puede ser construido (diseño). Son la base de los diagramas de componentes y despliegue. Permiten no sólo visualizar, especificar y documentar modelos estructurales, sino también construir sistemas ejecutables, aplicando ingeniería directa e inversa (55).

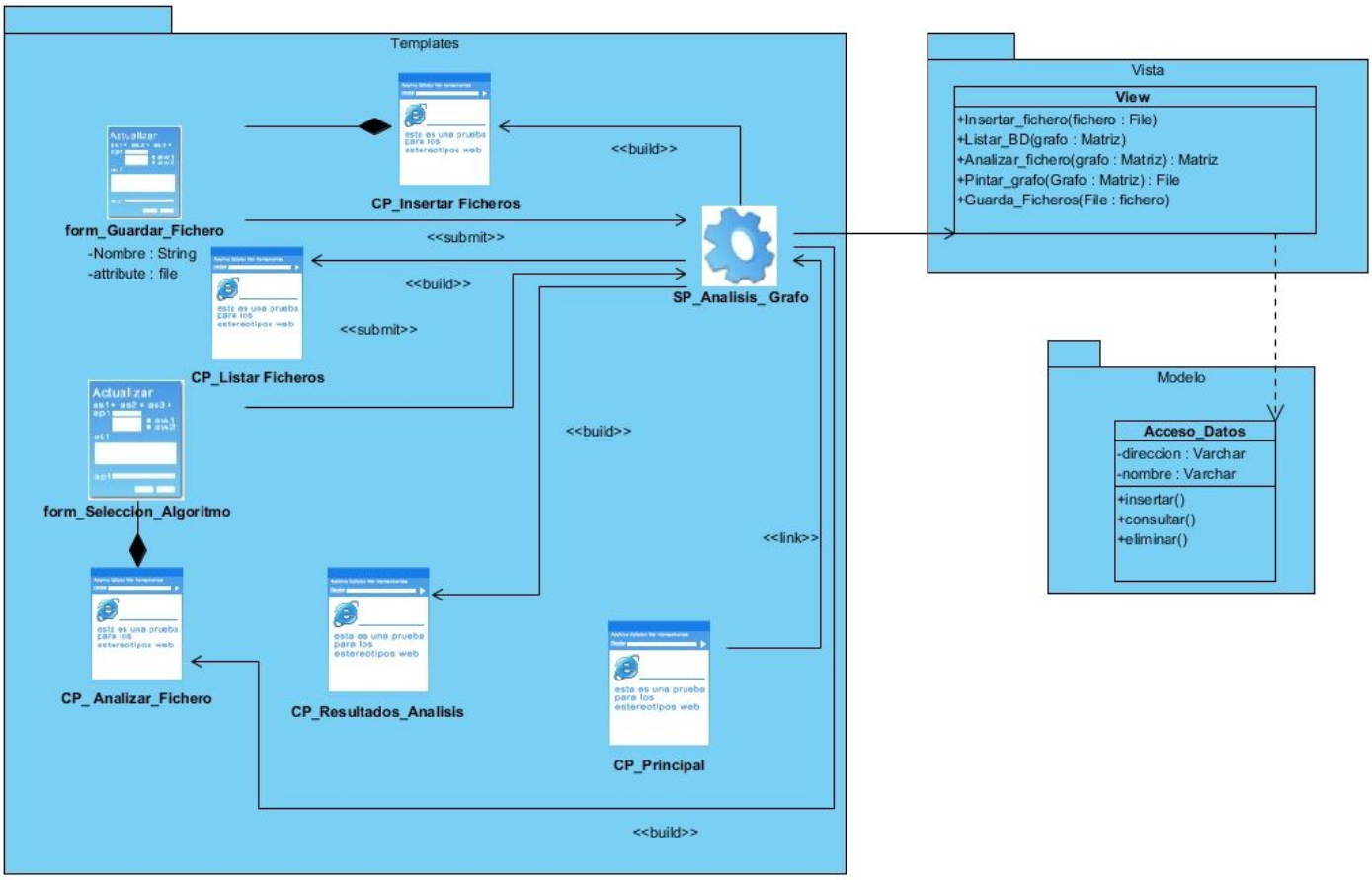


Figura 3.2 Diagrama de Clases del Diseño

3.4 Modelo Físico de Datos

Describe las representaciones físicas de los datos persistentes utilizados en la herramienta y que serán almacenados en base de datos.

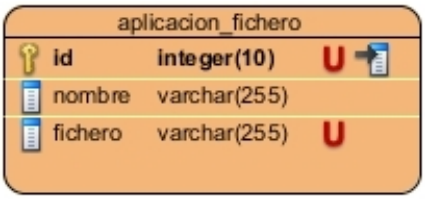


Figura 3.1. Diagrama Entidad-Relación usado para representar el modelo de datos.

3.4.1 Descripción de la tabla del Modelo Físico

aplicación_fichero		
Descripción	Esta tabla contendrá los datos que referencia a los ficheros con los grafos	
Atributo	Tipo	Descripción
Id	integer	Identificador del fichero
Nombre	varchar	Nombre del fichero
Fichero	varchar	Dirección relativa del fichero

3.5 Conclusiones

- Con la definición de una arquitectura flexible, adaptada a las funcionalidades del sistema, se logra realizar un diseño robusto y conforme a las buenas prácticas.
- El uso de patrones de diseño permite obtener un sistema lo más organizado y robusto posible.
- La realización de diagramas, como el diagrama de diseño, permite tener una visión clara de lo que se va a realizar.

➢

Capítulo 4 IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo a partir de los resultados obtenidos del modelado, se realiza el modelo de implementación. Además, se aplican diversas pruebas para comprobar la calidad y funcionalidad del sistema informático propuesto.

4.1 Modelo de Implementación

El modelo de implementación describe como los elementos del modelo de diseño, se implementan en términos de componentes como ficheros de código fuente y ejecutables. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes entre ellos (54).

4.1.1 Diagrama de componentes

{Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño}. (54)

Los diagramas de componentes permiten describir los elementos físicos que integran el sistema y las relaciones que existen entre ellos. Los componentes representan todos los tipos de elementos de software que entran en la producción de aplicaciones informáticas. Pueden ser simples archivos, paquetes, y/o bibliotecas cargadas dinámicamente. A continuación, se exponen el diagrama de componentes asociado al sistema implementado.

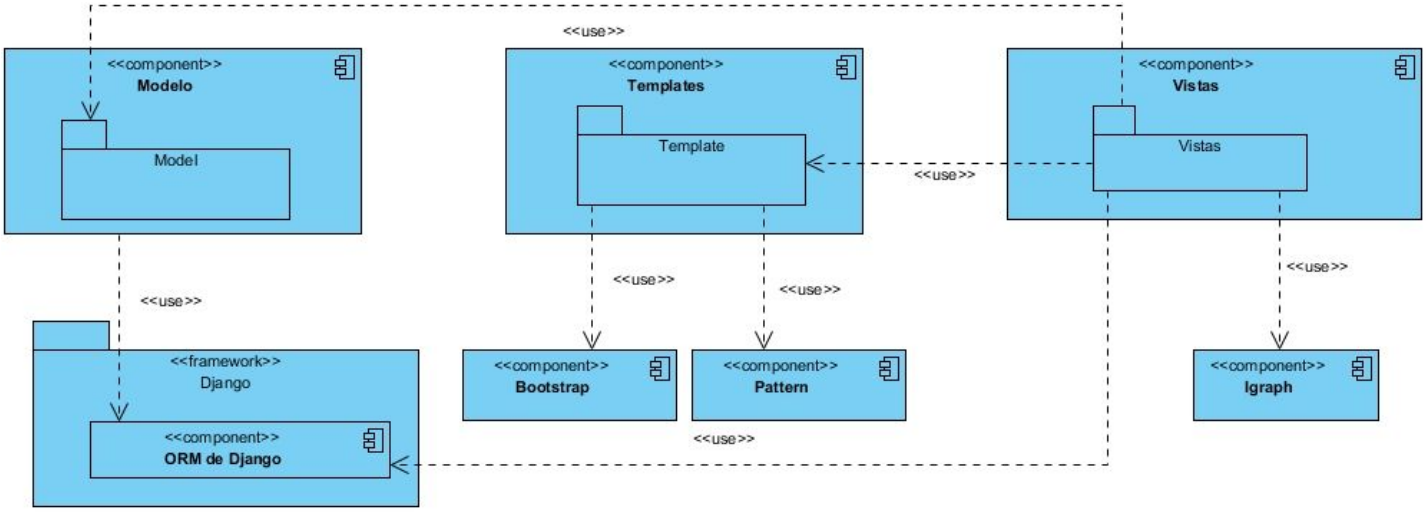


Figura 4.1 Diagrama de Componentes del sistema

Descripción de los Componentes

- Templates: Este componente contiene las interfaces encargadas de la interacción con el usuario.
- Vistas: El componente contiene las interfaces y clases controladoras del sistema, en las cuales se almacena todo el comportamiento lógico del sistema.
- Modelo: El componente contiene las interfaces y una descripción de la tabla de la base de datos, como una clase de Python.
- ORM de Django: Este componente se encarga de transformar la información del modelo a consultas SÑL que reconozca el servidor de bases datos y viceversa.
- Bootstrap: Este componente es una biblioteca CSS la cual ayuda a construir aplicaciones atractivas.
- Pattern: Es una biblioteca de Python que se utiliza para el trabajo con grafos en la web.
- Igraph: Es una biblioteca de Python que se utiliza para el trabajo con grafos.

4.1.2 Diagrama de despliegue

Un diagrama de despliegue modela la topología del hardware sobre el que se ejecuta un sistema. Muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos.

Un nodo es un elemento físico que existe en tiempo de ejecución. Representa un recurso computacional que, por lo general, tiene memoria y capacidad de almacenamiento. Además, representa el despliegue físico de un componente. (54)

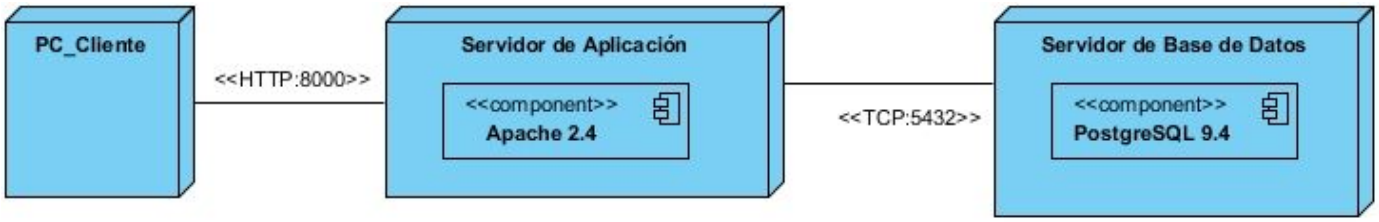


Figura 4.2 Diagrama de Despliegue

4.2 Estándares de codificación

Los estándares de codificación comprenden todos los aspectos de generación de código en un proyecto y permiten entender de manera rápida, fácil y sencilla el código empleado en el desarrollo de un software. Garantizan un mantenimiento óptimo del código por parte de los programadores, independientemente de si son los creadores del mismo. Reflejan un estilo armonioso como si un único programador hubiera escrito todo el código de una sola vez. La confección de estos estándares debe ser definida al comienzo de la implementación para garantizar que todos los programadores trabajen de manera coordinada.

Para lograr este objetivo se utilizó la Guía de estilo para el código Python (PEP 8). Esta guía posee una gran cantidad de convenciones para escribir código legible, dentro de las cuales se destacan [52]:

- Usar cuatro espacios por indentación.
- Nunca mezclar tabulaciones y espacios.
- Limitar todas las líneas a un máximo de caracteres (120 en este proyecto).
- Separar funciones de alto nivel y definiciones de clase con dos líneas en blanco, mientras que las
- definiciones de métodos dentro de una clase son separadas por una línea en blanco.
- Codificación UTF–8 en todos los módulos.
- Las importaciones deben estar en líneas separadas.
- Evitar usar espacios en blanco innecesarios.

- Utilizar el estilo CamelCase para nombrar clases, y el lower_case_with_underscores para funciones y métodos.

4.3 Modelos de Prueba

Las pruebas de software son actividades con las cuales el sistema o parte de él es ejecutado bajo condiciones específicas. Los resultados obtenidos permiten realizar una evaluación del mismo con el objetivo de valorar su calidad. Las pruebas constituyen un elemento fundamental para garantizar la calidad del software (62).

4.3.1 Técnicas de Evaluación Dinámica

La aplicación de estas técnicas se conoce como pruebas de software. Estas pruebas de software se pueden agrupar en:

- **Caja blanca o estructural:** Se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código.

- **Caja negra o funcional:** Son las pruebas que se llevan a cabo sobre la interfaz de usuario, se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Examina aspectos del modelo principalmente del sistema sin tener mucho en cuenta la estructura interna del software.

4.3.2 Pruebas de Caja Blanca

Pruebas de Unidad

Con el objetivo de comprobar el funcionamiento del sistema se le aplicaron pruebas de unidad, enfocadas a los elementos más pequeños del sistema que brindan una funcionalidad, para comprobar que actúen correctamente. Esta prueba se realizó a los componentes del sistema comprobando que desempeñaran las funciones esperadas, además de las pruebas hechas por el equipo de desarrollo durante el proceso de implementación de las diferentes funcionalidades.

Para la aplicación de las pruebas unitarias se utilizó la funcionalidad Test que trae integrada el IDE de desarrollo utilizado en la implementación del sistema.

En la siguiente figura se muestra un ejemplo del código de las pruebas realizadas a la Vista que es la encargada de modelar toda la lógica del negocio:

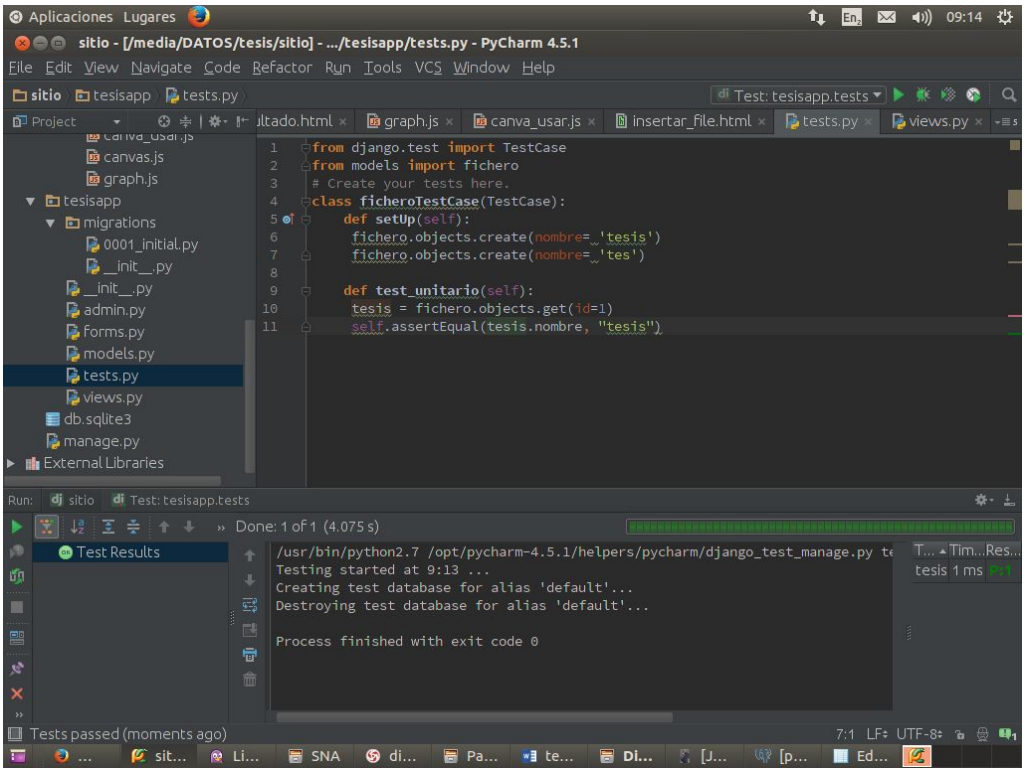


Figura 4.3 Fragmento de código de las pruebas unitarias del modelo

Resultados de las Pruebas de Caja Blanca

Al sistema se le realizaron dos iteraciones de pruebas unitarias. En la primera iteración se pudo validar el 75& de las funcionalidades, se encontraron 5 no conformidades que fueron resueltas. Una vez resuelta las no conformidades encontradas se aplicó una segunda iteración de pruebas en la cual no se detectaron no conformidades. La siguiente gráfica muestra la relación de estos datos:

Figura 4.4 Resultados del Test Unitario

4.3.3 Pruebas de Caja Negra

Descripción de los Casos de Pruebas

Las descripciones de casos de prueba que se utilizarán para las pruebas de la aplicación están diseñadas con el objetivo de verificar los requerimientos del usuario. A continuación, se muestra la descripción del caso de prueba cargar base de datos.

Escenario	Descripción	Nombre	Abrir fichero	Respuesta del sistema	Flujo central
EC 1.1 Cargar una base de datos	EC 1.1 Se carga un fichero correctamente	V	V	Se carga la página listar bases de datos y en la misma el sistema muestra el mensaje:	1. Seleccionar la opción {Insertar

correcta.				{Se ha cargado correctamente la Base de Datos.}.	Base de Datos} en el Menú Principal. 2. Introducir el nombre y la base de datos y buscar el fichero. 3. Oprimir el botón "Cargar".
EC 1.2 Cargar una base de datos incorrecta.	EC 1.2 Se carga un fichero incorrectamente	I	V	Se carga la página listar bases de datos y en la misma el sistema muestra el mensaje: {Se ha cargado incorrectamente la Base de Datos.}.	
		Se introducen 76 o más caracteres	Tiene que estar en uno de estos formatos: GML, GraphML, PAJEK, NET, PICKLE		
		V	I		
		Se introducen hasta 75 caracteres	No se encuentra en ninguno de estos formatos: GML, GraphML, PAJEK, NET, PICKLE		
EC 1.3 Se carga un fichero de modo incompleto	EC 1.3 Dejan campos en blanco	NA	V	Se carga la página listar bases de datos y en la misma el sistema muestra el mensaje: {No se pueden dejar los campos incompletos porque	
		No aplica el escenario incompleto porque no acepta	Tiene que estar en uno de estos formatos: GML, GraphML,		

		valores nulos, o es obligatorio	PAJEK, NET, PICKLE	tienen carácter obligatorio}.	
		V	NA	Se carga la página listar bases de datos y en la misma el sistema muestra el mensaje: {No se pueden dejar los campos vacíos porque tienen carácter obligatorio}.	
		Se introducen hasta 75 caracteres	No aplica el escenario incompleto porque no acepta valores nulos, o es obligatorio		

Para acceder al resto de las pruebas (ver Anexo 2).

Resultados de las Pruebas de Caja Negra

Al sistema se le realizaron dos iteraciones de pruebas de caja negra. En la primera iteración se pudo validar un 80& de las funcionalidades que se desea que el sistema realice, se determinaron cuatro no conformidades que fueron resueltas. En la segunda iteración el 100& de las funcionalidades arrojaron resultados satisfactorios.

Figura 4.5 Resultados de las pruebas de caja negra

4.3.4 Entorno de ejecución de las pruebas

Las pruebas de software realizadas se llevaron a cabo en un entorno de pruebas con las siguientes características:

- Procesador: Intel(R) Core(TM) i3 CPU 2.27 GHz.
- Memoria RAM: 4.00 GB.
- Sistema Operativo: Ubuntu 15.10.

Resultados de las Pruebas

De forma general al sistema se le realizaron dos iteraciones de pruebas. La primera iteración devolvió que en el sistema funcionaba correctamente en un 77.5& de las funcionalidades. Entre las no conformidades que se encontraron se pueden mencionar:

- Validación omitida de los datos nulos en algunas funcionalidades.

- Validación omitida del contenido del fichero cargado.
- Faltas de ortografía en algunos mensajes del sistema (mayúsculas incorrectas).
- Colores inapropiados en el diseño de las interfaces.

Las no conformidades encontradas en la primera iteración fueron resueltas satisfactoriamente. La segunda iteración de las pruebas arrojó que el sistema funciona correctamente en un 100& de sus funcionalidades.

4.4 Conclusiones

- El estándar de codificación definido siguiendo la guía de estilo para el código Python (PEP 8), permite tener un mejor entendimiento y mantenimiento de la solución, posibilitando que otros desarrolladores comprendan el código para una futura utilización o modificación.
- La realización del diagrama de despliegue permite tener una concepción de lo que se necesitará para desplegar la solución.
- Con la realización de pruebas al sistema queda comprobado que la solución propuesta se ejecuta correctamente y sin errores.

CONCLUSIONES

Al concluir la presente investigación se obtienen resultados que dan cumplimiento satisfactorio al objetivo general planteado inicialmente.

- El análisis de las soluciones existentes evidenció la necesidad del desarrollo de una nueva solución que permita la detección y visualización de comunidades científicas.
- Con la realización de un sistema informático que permita obtener las comunidades científicas virtuales y sus principales líderes en instituciones científico-académica se logra proveer a la Universidad de las Ciencias Informáticas de una importante herramienta para el apoyo en la toma de decisiones.
- La herramienta, metodología y lenguaje de modelado identificado a partir de la investigación de las herramientas y tecnologías actuales para el desarrollo de software garantizaron la realización de los artefactos necesarios para el posterior diseño e implementación de la solución.
- Las pruebas realizadas al sistema demuestran su correcto funcionamiento.

RECOMENDACIONES

Una vez cumplido el objetivo de la presente investigación se recomienda:

- Aplicar el sistema en la Universidad para la detección de los grupos de investigación y líderes ocultos que no son del dominio de la dirección del centro.
- Desplegar el sistema en otros centros científico-académicos.
- Utilizar el sistema para la detección de comunidades y líderes de otros centros de investigación y/o desarrollo para el estudio de posibles alianzas y colaboración.
- Agregarle el soporte de nuevas extensiones de ficheros.
- Agregarle nuevos algoritmos para la detección de comunidades.
- Agregar la posibilidad de realizar análisis sobre redes de co-palabras y redes de citas.

REFERENCIAS BIBLIOGRAFICAS

(1) Biblioteca de la Universidad de León 2014 [En línea] http://biblioteca.unileon.es/ayuda-formacion/repositorio-institucional

(2) GALINDO C' CERES, Jesús. Construcción de una comunidad virtual. Signo y pensamiento, 2013, vol. 19, no 36, p. 93-102.

(3) ORTIZ MU#OZ, Ernesto; HIDALGO DELGADO, Yusniel. Detección de comunidades a partir de redes de coautoría en grafos RDF. Revista Cubana de Información en Ciencias de la Salud, 2016, vol. 27, no 1, p. 90-99.

(4) TRIGO, Luís; BRAZDIL, Pavel. Affinity Analysis between Researchers using Text Mining and Differential Analysis of Graphs. ECML/PKDD 2014 PhD session Proceedings, Nancy, France, 2014, p. 169-176.

(5) Sebastían A Ríos and Felipe Aguilera. Web intelligence on the social web. In Advanced Techniques in Web Intelligence-I, p 225–249. Springer, 2010.

(6) F. de la Rosa T., S. Pozo y R. M. Gasca. Análisis y Visualización de Comunidades Científicas con Información extraída de la web, 2014.

(7) VAN KESTER, S. Efficient crawling of community structures in online social networks. 2011. Tesis Doctoral. TU Delft, Delft University of Technology.

(8) LOZARES COLINA, Carlos. La teoría de redes sociales. Papers: revista de sociologia, 1996, no 48, p. 103-126.

(9) OLIVARES, Cristian Paolo Mejia. Análisis de Redes Sociales a Gran Escala. 2010

(10) M. Girvan and M. E. J. Newman. Community structure in social and biological networks. Proceedings of the National Academy of Sciences of the United States of America, 99(12):7821–7826, June 2002.

(11) M. E. J. Newman. Fast algorithm for detecting community structure in networks. Sep 2003.

(12) Duncan J. Watts and Steven H. Strogatz. Collective dynamics of Ásmall-worldÁnetworks. Nature, 393(6684):440–442, June 1998.

(13) David L. Nowell and Jon Kleinberg. The link prediction problem for social networks. In CIKM Á03: Proceedings of the twelfth international conference on Information and knowledge management, pages 556–559, New York, NY, USA, 2003. ACM.
(14) A. Capocci, V. D. P. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, and G. Caldarelli. Preferential attachment in the growth of social networks: The internet encyclopedia wikipedia. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 74(3), 2006.
(15) H. Zanghi, C. Ambroise, and V. Miele. Online and offline social networks: Use of social networking sites by emerging adults. Applied Developmental Psychology, 29:420–433, August 2008.
(16) Seth Richards. A social network analysis into the david kelly tragedy. Connections, 26:25–32, 2005.
(17) Mark E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. Proceedings of the National Academy of Sciences of the United States of America, 99(Suppl 1):2566–2572, February 2002.
(18) S. Milgram. The small world problem. Psychology Today, 2(1):60–67, 1967.
(19) Cohen 2002. R. Cohen, D. Avraham, and S. Havlin. Structural properties of scale-free networks, 2002.
(20) Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web: experiments and models. In Proceedings of the Ninth International World-Wide Web Conference (WWW9, Amsterdam, May 15-19, 2000-Best Paper). Foretec Seminars, Inc. (of CD-ROM), Reston, VA, 2000.
(21) S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang. Complex networks: Structure and dynamics. Physics Reports, 424(4-5):175–308, February 2006.
(22) R. Ferrer I Cancho and R. V. Solé. The small world of human language. Proc R Soc Lond B Biol Sci, 268(1482):2261–2265, November 2001.
(23) Mark E. J. Newman, Albert L. Barabási, and Duncan J. Watts, editors. The Structure and Dynamics of Networks. Princeton University Press, 2006.
(24) E. N. Gilbert. Random graphs. The Annals of Statistics, 30(30):1141–1144, 1959.
(25) Duncan J. Watts. Six degrees: The science of a connected age. WW Norton Í Company, 2003.
(26) Gretzel, Ulrike. 2001. Social Network Analysis: Introduction and Resources. [En línea] Noviembre de 2001. [Citado el: 20 de febrero de 2013.] http://lrs.ed.uiuc.edu/tse-portal/analysis/socia network-analysis/.
(27) González-Alcaide, G., y otros. 2008. 11, s.l. Redes de coautoría y colaboración institucional: Revista de Neurología, 2008, Vol. 46.

(28) Molina., I. s.l.: Apuntes de Ciencia y Tecnología ˝La ciencia de las redes˝, 2004, Vol. 11.

(29) Sanz M. Análisis de Redes Sociales: o como representar las estructuras sociales subyacentes. 2003.

(30) Enciso., Guillermo E. 2010. [En línea] 29 de septiembre de 2010. [Citado el: 5 de marzo de 2016.] http://www.slideshare.net/existaya/5-ventajas-de-usar-redes sociales-en-su-empresa.

(31) Mark EJ Newman. Modularity and community structure in networks. Proceedings of the National Academy of Sciences, 103(23):8577–8582, 2006.

(32) Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. Physical review E, 69(2):026113, 2004.

(33) Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10): P10008, 2008.

(34) NEWMAN, Mark EJ; GIRVAN, Michelle. Finding and evaluating community structure in networks. Physical review E, 2004, vol. 69, no 2, p. 026113.

(35) M. Girvan and M. E. J. Newman. Community estructure in social and biological networks. Proc. Natl. Acad. Sci. USA, 99(12):7821–7826, 2002.

(36) A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Physical Review E,79(066111), 2004.

(37) U.N. Raghavan – R. Albert – S. Kumara "Near linear time algorithm to detect community structures in large-scale networks", 2007

(38) P. Pons and M. Lapaty. Computing communities in large networks using random walks. http://arxiv.org/abs/physics/0512106, 2005.

(39) Jon M Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 46(5):604–632, 1999.

(40) Real Academia Española - 22a edición. [En línea] http://lema.rae.es/drae/.

(41) Boeras Velázquez, M {Aplicando el Método de Boehm y Turner}. [En línea] http://publicaciones.uci.cu/index.php/SC/article/view/891.

(42) AUP The Agile Unified Process. [En línea] http://www.cc.una.ac.cr/AUP/html/philosophies.html.

(43) Jacobson, Ivar, Rumbaugh, James and Booch, Grady. El lenguaje Unificado de Modelado. Manual de Referencia. Madrid: Addison Wesley, 2007.

(44) Visual Paradigm [En línea] https://www.visual-paradigm.com/features/

(45) HICKSON, I. y HYATT, D. 2009. HTML5. 2009. Consultado el 5 de marzo de 2016. Dirección: http://dev.w3.org/html5/spec/Overview.htmlı html-vs-xhtml

(46) BARCIA, D. 2003. ¿Ñué es CSS? 2003. Consultado el 5 de marzo de 2016. Dirección: http://www.maestrosdelweb.com/introcss/.

(47) ' LVAREZ, M. A. 2008. Introducción a CSS3. 2008. Consultado el 5 de marzo de 2016. Dirección: http://www.desarrolloweb.com/articulos/introduccion-css3.html .

(48) LERNER, Reuven M. At the forge: Twitter bootstrap. 2012. vol. 2012, no 218.

(49) GONZ' LEZ, J. y GAUDIOSO, E. 2001. Aprender y formar en Internet. 2001. Cap. 6. JavaScript.

(50) ' LVAREZ, S. 2007. Sistemas gestores de bases de datos. Introducción a este concepto y características especiales. 2007. Consultado el 6 de marzo de 2016. Dirección: http://www.desarrolloweb.com/articulos/sistemasgestores-bases-datos.html.

(51) PostgreSÑL. 1996-2016. Consultado el 6 de marzo de 2016. Dirección: http://www.postgresql.org/about/.

(52) PgAdmin. 2007. Dirección: http://www.purosoftware.com/programacion-bases-de-datos/11-pg-admin-3.html.

(53) Larman, Craig. UML y Patrones. Segunda Edición.

(54) ACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. El proceso unificado de desarrollo de software. Reading: Addison Wesley, 2000.

(55) Pressman, Roger S. Ingeniería del Software Un enfoque práctico. Ñuinta Edición.

(56) LLEONART, E.; GARC˜A, A. y ROVIRA, M. 2010. Patrones. 2010.

(57) VELOSO, P. 2008. Uso de patrones de arquitectura. 2008.

(58) ALCHIN, M. 2012. Pro Django. Second Edition. www.it-ebooks.info: Apress, 2012.

(59) PRESSMAN, R. S. 2013. Software Engineer. A practitioner¨s Approach. 7ma Edición. 2013.

(60) GAMMA, E.; HELM, R.; JOHNSON, R. y VLISSIDES, J. 2002. Patrones de Diseño. Elementos de software orientado a objetos reutilizable. Madrid. España.: Addison Wesley, 2002.

(61) VISCONTI M. y Astudillo, H. 2011. Fundamentos de Ingeniería de Software. Universidad Técnica Federico Santa María. Departamento de Informática. 2011.

(62) UDA Universidad de Alcala. [En línea] http://www2.uah.es/jcaceres/uploaded/teoria/LenguajesProgramacion/tema1.pdf.

(63) VAN ROSSUM, G.; WARSAW, B. y COGHLAN, N. 2001. Pep 8, Style guide for python code. 2001.

ANEXOS

Anexo 1. Descripciones de CU del sistema

Listar Bases de Datos

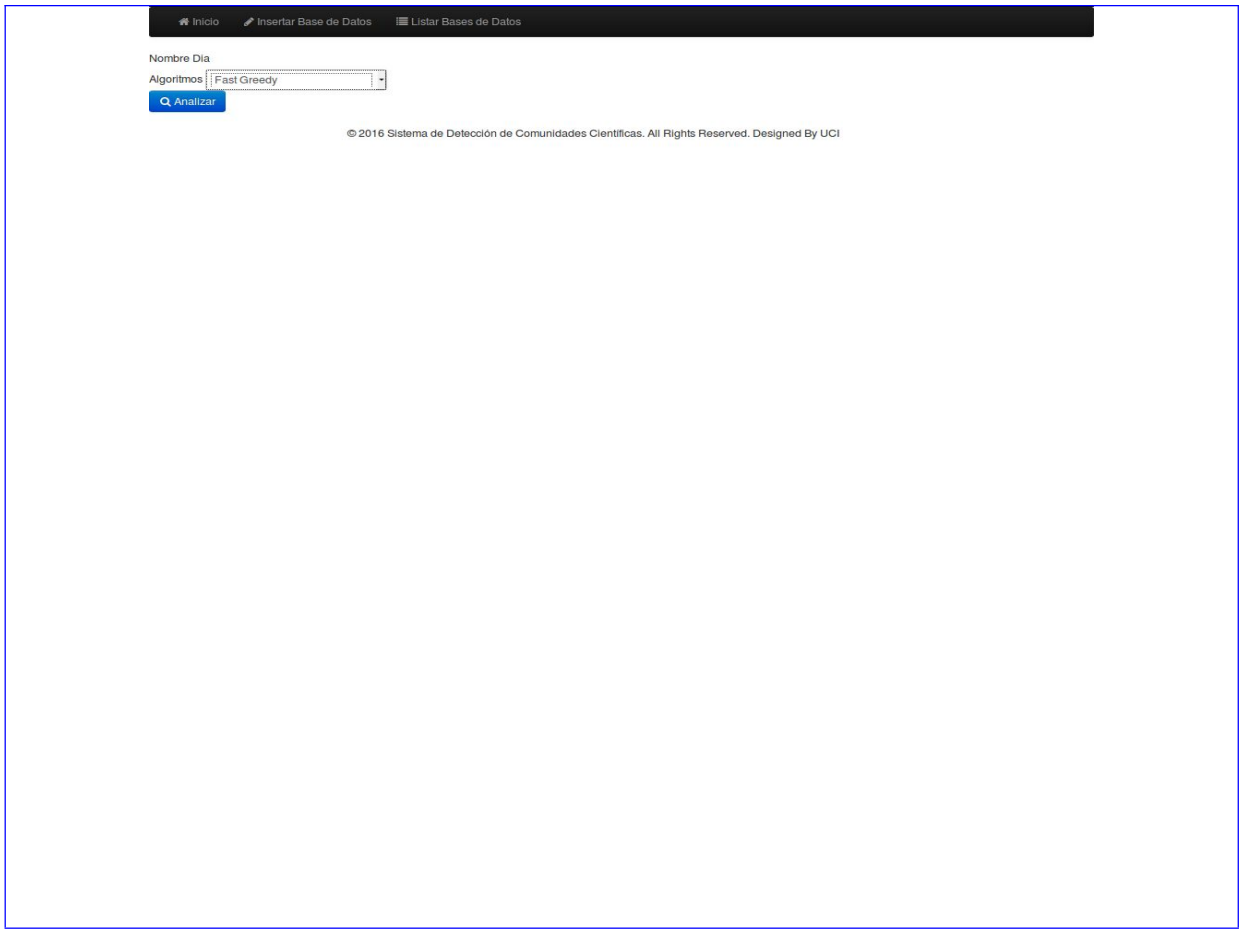
<table>
<tr><td>Objetivo</td><td colspan="3">Realizar un listado con todas las bases de Datos</td></tr>
<tr><td>Actores</td><td colspan="3">Usuario</td></tr>
<tr><td>Resumen</td><td colspan="3">El CU comienza cuando el usuario desea ver todas las bases de datos cargada. Este Presiona el botón listar bases de datos y observa las bases de datos cargadas. Cuando se hayan mostrados todas las bases de datos termina el CU.</td></tr>
<tr><td>Complejidad</td><td colspan="3">Alta</td></tr>
<tr><td>Precondiciones</td><td colspan="3">-</td></tr>
<tr><td colspan="4">Flujo de eventos</td></tr>
<tr><td colspan="4">Flujo Básico</td></tr>
<tr><td colspan="2">Actor</td><td colspan="2">Sistema</td></tr>
<tr><td colspan="2"></td><td colspan="2">1. El sistema muestra una interfaz de bienvenida con un Menú que contiene las siguientes opciones:
➢ Inicio (página de Bienvenida)
➢ Insertar Base de Datos (Permite cargar en el sistema una base de datos bibliográfica con extensión GML, PICKLE, PAJEK, GraphML o NET).
➢ Listar Bases de Datos</td></tr>
<tr><td colspan="2">2. El usuario oprime el botón Listar Bases de Datos.</td><td colspan="2">2. Sistema muestra una interfaz donde se observa una lista con los nombre, las direcciones de las bases de Datos y un botón con el icono de una lupa para analizar la base de datos</td></tr>
<tr><td rowspan="2">Relaciones</td><td>CU Incluidos</td><td colspan="2">No</td></tr>
<tr><td>CU Extendidos</td><td colspan="2">No</td></tr>
</table>

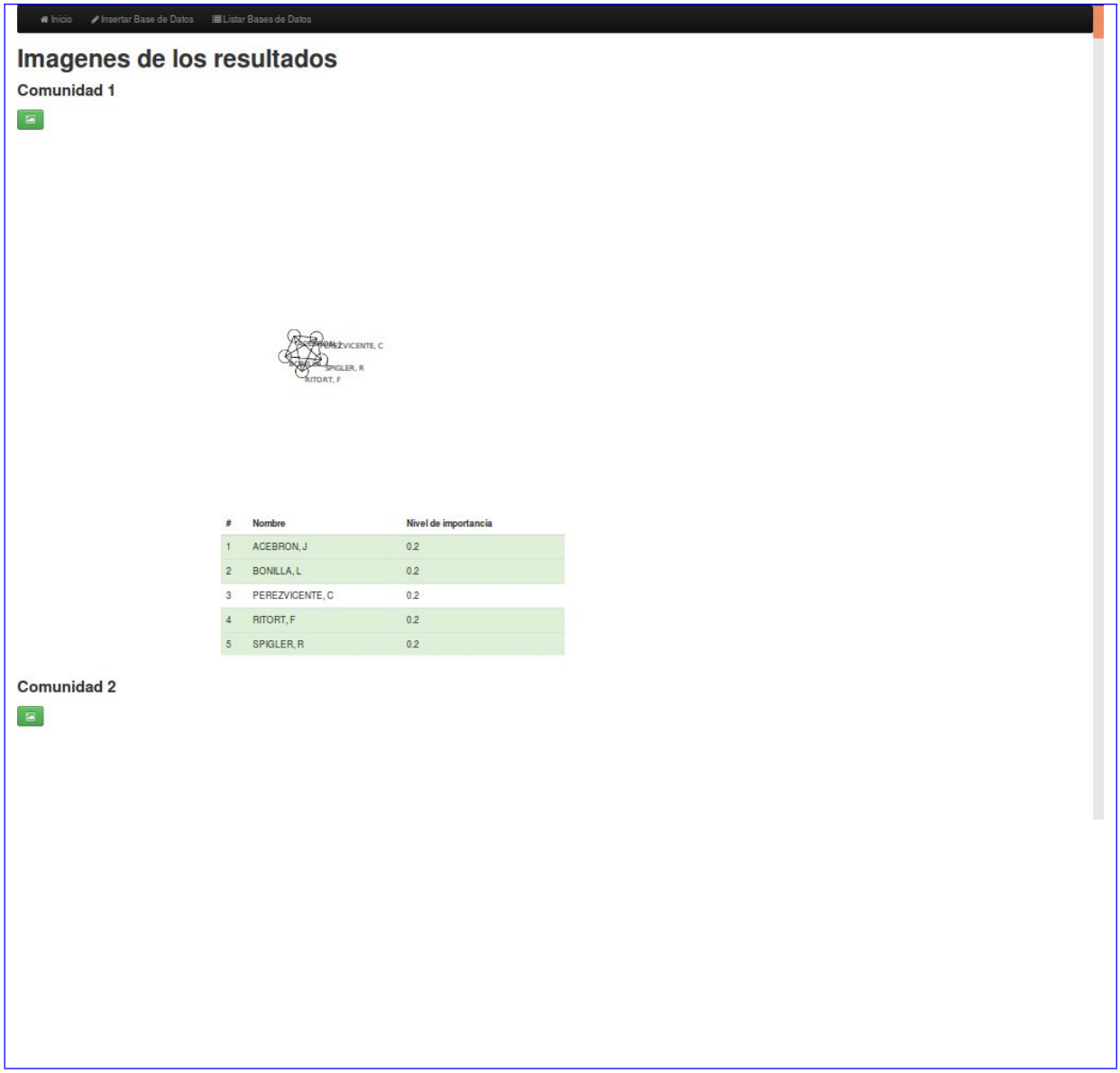
Prototipo elemental de interfaz gráfica de usuario

Analizar Grafo

Objetivo	Realizar análisis del Grafo.
Actores	Usuario

Resumen	El CU comienza cuando el usuario desea comenzar el análisis de una base de datos cargada. Este selecciona la base de datos lee los datos, realiza el análisis y muestra los grafos de las distintas comunidades. Cuando se haya completado el análisis base de datos termina el CU.	
Complejidad	Alta	
Precondiciones	El sistema debe tener al menos una base de datos de grafos en el servidor.	
Flujo de eventos		
Flujo Básico		
Actor	Sistema	
	1. El sistema muestra el listado de las Bases de Datos.	
2. El usuario oprime el botón el botón con la lupa de la base de datos que desea analizar	2. Sistema muestra una interfaz donde se observa El nombre de la base de datos seleccionada un campo desplegable con tres algoritmos (Fastgreedy, Label Propagation y Walktrap) Un botón cargar	
3. El Usuario selecciona el algoritmo y oprime el botón Analizar	3. El sistema carga la base de datos, realiza el análisis de para la determinar comunidades y detectar los líderes de las mismas	
	4. El sistema dibuja los grafos de las diferentes comunidades	
Relaciones	CU Incluidos	Listar Base de Datos.
	CU Extendidos	No
Prototipo elemental de interfaz gráfica de usuario		


Inicio
Insertar Base de Datos
Listar Bases de Datos
Nombre Dia
Algoritmos
Fast Greedy
Analizar
© 2016 Sistema de Detección de Comunidades Científicas. All Rights Reserved. Designed By UCI



Exportar a PNG

Objetivo	Exportar los grafos a PNG.
Actores	Usuario
Resumen	El CU comienza cuando el usuario desea guardar como imagen

	uno de los grafos resultantes. Este selecciona el botón exportar PNG, convertir u grafo en una imagen. Cuando se haya exportado la imagen termina el CU.	
Complejidad	Baja	
Precondiciones	El sistema debe mostrar los resultados del análisis	
Flujo de eventos		
Flujo Básico		
Actor	Sistema	
	1. El sistema muestra los grafos resultantes del análisis donde cada uno trae un botón para exportar los grafos	
2. El usuario oprime el botón el botón con la una figura con un PNG	2. La canva es transformada en imagen y salvarla en la PC cliente	
Relaciones	CU Incluidos	Listad de grafos resueltos.
	CU Extendidos	No
Prototipo elemental de interfaz gráfica de usuario		

Exportar a PDF

Objetivo	Exportar los grafos a PDF.
Actores	Usuario
Resumen	El CU comienza cuando el usuario desea guardar como documento PDF todos los resultados del análisis. Este selecciona el botón exportar PDF, convierte todos los grafos en imágenes y los guardas en un documento PDF. Cuando se haya

	exportado la el documento termina el CU.		
Complejidad	Baja		
Precondiciones	El sistema debe mostrar los resultados del análisis		
Flujo de eventos			
Flujo Básico			
Actor	Sistema		
	1. El sistema muestra los grafos resultantes del análisis donde cada uno trae un botón para exportar los grafos		
2. El usuario oprime el botón con la una figura con un PDF	2. Todas las canvas que representan grafos se convierten en imágenes y lo guardan en el documentos		
Relaciones	CU Incluidos	Listado de grafos resueltos.	
	CU Extendidos	No	
Prototipo elemental de interfaz gráfica de usuario			

Anexo 2: Caso de Prueba del sistema

Analizar base de Datos

Escenario	Descripción	Fichero	Respuesta del sistema	Flujo central

EC 1.1 Analizar una base de datos correcta.	*EC 1.1 Analizar una base de datos correctamente.*	*V*	*Se carga la página respuesta muestra los grafos de las comunidades científica.*	*1. Seleccionar el algoritmo de detección de comunidades.* *2. Oprimir el botón "Analizar".*