UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO

LICENCIATURA EN SISTEMAS DE INFORMACIÓN

TRABAJO FINAL DE GRADUACIÓN

Aplicación Práctica del Enfoque de Gestión de Conocimiento: Sistema web basado en aplicaciones sociales como soporte al trabajo colaborativo en áreas académicas de nivel superior

Autores:

Germán Ezequiel LESCANO

Ricardo Daniel PACHECO TOLEDO

Profesora Guía**:**

MABEL DEL VALLE SOSA

Septiembre de 2011

UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

TRABAJO FINAL DE GRADUACIÓN DE LA LICENCIATURA EN SISTEMAS DE INFORMACIÓN

APLICACIÓN PRÁCTICA DEL ENFOQUE DE GESTIÓN DE CONOCIMIENTO: SISTEMA WEB BASADO EN APLICACIONES SOCIALES COMO SOPORTE AL TRABAJO COLABORATIVO EN ÁREAS ACADÉMICAS DE NIVEL SUPERIOR

Autores:

....

Germán Ezequiel Lescano Ricardo Daniel Pacheco Toledo

Profesora Guía:

.....

Ing. Mabel del Valle Sosa

————————————————————————————————————*

Aprobado el día ..... del mes de ..... del año 20.....

por el Tribunal integrado por

....

....

Santiago del Estero – Argentina

A la memoria de mí querida madre

Para mi familia y muy particularmente para mi abuela Angela quienes supieron comprender todo el tiempo que dedique en mi proceso de formación descuidándome muchas veces de ellos.

Germán Ezequiel Lescano

Para toda mi familia, especialmente para mi mamá Carmen que me dio la posibilidad de formarme profesionalmente y para mi novia Alejandra que me acompañó y alentó durante este proceso de formación.

Ricardo Daniel Pacheco Toledo

Agradecimientos

Agradecemos muy particularmente a nuestra profesora guía Ing. Mabel del Valle Sosa por su desinteresada colaboración y aporte de conocimientos, manifestado en comentarios, correcciones y críticas que enriquecieron nuestro Trabajo Final de Graduación.

Asimismo, extendemos nuestro agradecimiento a las autoridades y profesores de la Facultad de Ciencias Exactas y Tecnologias que nos apoyaron en los diferentes eventos sobre tecnologías informáticas que organizamos junto a otros compañeros.

Finalmente agradecemos a nuestros compañeros de trabajo que nos supieron alentar en todo momento durante nuestro proceso de desarrollo del trabajo y especialmente a nuestros compañeros de carrera Ricardo Miranda y Pablo Santana.

G.L. y R.P.
Santiago del Estero, Argentina
Agosto de 2011

TABLA DE CONTENIDO

RESUMEN

Las últimas tecnologías de la información y la comunicación, sumado a las innovaciones Web han hecho de esta última una "Plataforma". Herramientas tales como las de *sindicación*, las *wikis*, los *blogs* y los *podcasts* gradualmente se están volviendo más populares dentro del campo de la Educación Superior. Sin embargo, sus grandes potencialidades no han sido completamente exploradas en este ámbito.

Actualmente, nos encontramos frente a una sociedad que se encuentra en un proceso de transición cultural, identificada como sociedad del conocimiento y caracterizada por el desarrollo vertiginoso de conocimiento científico y tecnológico. Sumado a ello, la tendencia hacia el trabajo colaborativo y el compartir conocimientos que se observa en los ámbitos educativos de nivel superior, dan cuenta de la necesidad de disponer de mejores infraestructuras que faciliten el desenvolvimiento de sus funciones primordiales, principalmente la función académica, que permitan dar respuesta a las nuevas demandas sociales y permanecer en entornos altamente cambiantes y competitivos.

En base a lo expuesto anteriormente, en este trabajo se realiza el diseño y construcción a nivel de prototipo de un sistema web basado en aplicaciones sociales, para el ámbito de la Educación Superior, que facilite el trabajo colaborativo y el desarrollo de las prácticas académicas, y a la vez, promueva la generación colaborativa, la transmisión y el uso de conocimientos científicos y tecnológicos; contribuyendo a la explicitación de los conocimientos disponibles y a generar valor agregado a partir de los mismos, según el modelo propuesto por Nonaka-Takeuchi.

Palabras claves

Gestión del Conocimiento, Aplicaciones sociales, Trabajo colaborativo, Educación Superior

INTRODUCCIÓN

En nuestra sociedad, el conocimiento es el componente más importante de cualquier actividad, y constituye la fuerza que orienta el cambio y la innovación [32]. Las instituciones educativas, en particular las universidades, tienen el rol clave de proveer educación fidedigna y de alta calidad, y contribuir con el desarrollo del conocimiento; sin embargo, en la actualidad no son las únicas fuentes de información y conocimiento, debido a la expansión de las nuevas formas de comunicación, siendo hoy la más notable, Internet[28].

En este contexto, las instituciones de Educación Superior están enfrentando fuertes presiones para ajustar sus métodos para crear, compartir y preservar el conocimiento, debido a los cambios tecnológicos de los últimos años, en los cuales, las tecnologías de la información y la comunicación han abierto oportunidades sin precedentes de interacción social y construcción de conocimiento de a pares [28].

Por las cuestiones mencionadas anteriormente, el presente trabajo se centra en la aplicación y uso de las herramientas propias de la Web, como instrumentos que brindan la posibilidad de hacer frente a los desafíos antes mencionados. Concretamente, se plantea el diseño y construcción a nivel de prototipo, de un sistema web basado en aplicaciones sociales, que facilite el trabajo colaborativo y de soporte al desarrollo de las prácticas académicas de nivel superior. El sistema se construye bajo el enfoque de gestión del conocimiento y se lo verifica a través del modelo de producción de conocimiento planteado por Nonaka-Takeuchi [31].

El trabajo se organiza en cinco capítulos y cuatro anexos. En el capítulo I se describe el problema, se establecen los objetivos del trabajo, se fundamenta la importancia del mismo y se analizan antecedentes de trabajos relacionados con la propuesta de solución efectuada. En el capítulo II se desarrollan los marcos referenciales, a saber: teórico conceptual, metodológico y empírico, sobre los que se fundamenta este trabajo. En el capítulo III se presenta el sistema web propuesto que integra aplicaciones sociales, se describe el modelo tecnológico y cada una de las capas que lo integran. En el capítulo IV se presenta la evaluación del sistema y se explica de qué modo da soporte a los procesos y actividades de la gestión del conocimiento en un ámbito académico universitario. Finalmente, en el capítulo V se presentan las conclusiones del trabajo realizado.

Con respecto a los anexos, en el Anexo I se detallan los diseños de clases de los prototipos desarrollados; en el Anexo II se describen las interfaces de las principales pantallas de los prototipos que fueron programados en su totalidad; en el Anexo III se muestran los resultados de las pruebas que se consideraron significativas para comprobar el funcionamiento de la aplicación; en el Anexo IV se explica en detalle cómo fue configurada y programada la red neuronal de soporte para búsqueda y, en el Anexo Vddddd se presenta un glosario con los principales términos que pueden ser de ayuda para comprender de una mejor manera el trabajo.

CAPÍTULO I
PROBLEMAS, OBJETIVOS Y ALCANCE DEL TRABAJO

1.1. PLANTEAMIENTO DEL PROBLEMA

Los grandes cambios que se vienen dando en nuestra sociedad debido a los avances vinculados a las tecnologías de la información y la comunicación, plantean nuevos desafíos a las instituciones educativas, principalmente a las de nivel superior, de ajustar sus métodos para crear, compartir y preservar el conocimiento [28].

Asimismo, estas instituciones deben estar preparadas para recibir a las nuevas generaciones que crecieron en estos entornos sociales, a las cuales Neil Howe y William Strauss bautizaron como "Millenials", quienes poseen capacidades de multitarea sin precedentes, aliados a las expectativas de interacciones rápidas con los canales de información y un deseo intrínseco por estar conectados [18].

A su vez, el uso generalizado de Internet, y de las aplicaciones sociales en particular, en menos de una década han cambiado profundamente la naturaleza de la comunicación al permitir una mayor participación de personas en la generación de contenidos online.

Todas estas transformaciones provocan que la sociedad demande a las instituciones educativas la formación de ciudadanos con nuevas habilidades (capacidad para administrar la sobrecarga de información, construir redes de interacción con otras personas, crear comunidades, comunicarse de manera innovadora en el ámbito de su profesión, y poseer espíritu crítico y creativo) para desenvolverse adecuadamente en los nuevos contextos [23].

Sin embargo, hasta ahora las instituciones educativas no han reaccionado a estos cambios ni han estimado las nuevas oportunidades que pueden surgir [23]. Particularmente en Argentina, y más precisamente en nuestra región, Noroeste

Argentino, tampoco se vislumbra una estrategia que busque sacar provecho de los avances en las Tecnologías de la Información y la Comunicación (TICs).

En particular, en el ámbito de la Universidad Nacional de Santiago del Estero (UNSE), se observan algunos problemas básicos, tales como, dificultades en el acceso a fuentes de información comunes, desconocimiento de los recursos existentes (tanto materiales como de recursos humanos), la repetición de trabajo o duplicación de esfuerzos y en consecuencia, la obstaculización al desarrollo de procesos de creatividad y la optimización del desarrollo de las actividades académicas (de docencia e investigación). Algunos de estos problemas mencionados ya fueron detectados en [29].

Con el propósito de agilizar el trabajo académico, en el sentido de mejorar el acceso a la información y conocimiento generado, usar, distribuir y compartir los recursos creados, en este trabajo se propone un sistema web basado en el enfoque de gestión del conocimiento, que permita enfrentar los problemas detectados en nuestra universidad y contribuya con la creación de una comunidad de práctica académica colaborativa.

1.2. ANTECEDENTES

En el ámbito educativo la mayoría de las investigaciones están orientadas al fenómeno e-learning, hecho que motivó el estudio de Plataformas especializadas para la Gestión del Aprendizaje (LMS, en inglés, *Learning Management System*) y Plataformas especializadas para la Gestión de Contenidos (CMS, en inglés, *Content Management System*) [17]. Estos estudios se centran en analizar cómo mejorar la forma de gestionar el ciclo de vida de los contenidos digitales en las organizaciones, es decir, la creación, la distribución, la preservación, y la reutilización de estos recursos a través de las redes. Sin embargo, como plataforma para la generación y transmisión de conocimientos académicos consideramos que puede ser mejorada al incorporar otras herramientas Web 2.0 (por ejemplo bookmarking y networking) o usar contenidos generados por los miembros en aplicaciones sociales ya existentes (por ejemplo, Twitter, Wordpress, Blogger, entre otros).

Otro antecedente es el proyecto llevado a cabo en la Universidad Peruana de Ciencias Aplicadas [17]. En dicho proyecto se plantea un modelo que busca gestionar y

valorar los contenidos académicos producidos por sus docentes e investigadores. Para lograrlo se plantea el desarrollo de un sistema web, conocido como Delfos, a través del cual lo que se persigue fundamentalmente es generar un espacio para publicar y compartir contenidos digitales de apoyo para el proceso de enseñanza-aprendizaje [17]. Considerando este caso particular, observamos que puede ser mejorado si se aplican los principios vinculados a las aplicaciones sociales, como ser, disponer de medios a través de los cuales interactuar con otras personas y opinar con respecto a un tema en particular, contar con facilidades para poder compartir y disemeninar contenidos, tener herramientas a través de las cuales se puedan colaborar en la construcción de materiales colaborativos, disponer de posibilidades para valorar recursos, entre otros.

Por último, otros proyectos interesantes están vinculados con la apertura de cursos universitarios completos online lo cual se constituyen en enormes fuentes de transmisión de conocimiento. Básicamente se pueden mencionar tres grandes proyectos: MIT Opencourseware [35], Webcast.Berkeley [34] y el OpenLearn Project [33]. En ellos se busca contribuir con la diseminación y la producción colaborativa de conocimiento académico y científico alrededor del mundo. Estas iniciativas son un claro ejemplo de los resultados que se pueden obtener a través de la potencia de la colaboración.

1.3. OBJETIVOS

En este trabajo se han definido como objetivos generales los siguientes:

- Proporcionar un Sistema Web al ámbito de la Educación Superior que permita hacer frente a los nuevos desafíos que le plantea la sociedad del conocimiento.
- Promover nuevos enfoques de transmisión de conocimientos en la comunidad educativa de la UNSE.

Los objetivos específicos definidos son:

- Diseñar un sistema web integrando aplicaciones sociales como soporte a las actividades académicas de investigación y aprendizaje en el ámbito universitario.

- Construir una herramienta de tipo *networking* académico para la comunicación de los miembros de la institución y facilitar el descubrimiento de los mismos.
- Desarrollar un portal académico personalizable para el acceso a fuentes de información - conocimiento.

1.4. SOLUCIÓN PROPUESTA

La solución se basa en la integración de aplicaciones sociales: *networking*, *wikis*, *foros*, *blogs* (Wordpress y Blogger) y aplicaciones de *microblogging* (Twitter). A su vez se ofrecen algunos canales para la interacción entre los usuarios y la circulación de los conocimientos tácitos. De esta manera, a través de estas múltiples herramientas de cooperación se pretende potenciar la inteligencia colectiva, la cual es una forma de inteligencia que surge de la colaboración y el aporte de muchos individuos, para encontrar una ventaja individual y colectiva mayor que si cada participante permanece sólo.

El sistema web propuesto con enfoque de gestión del conocimiento responde al modelo de conversión de conocimiento planteado por Nonaka y Takeuchi [31]. Por otra parte, el sistema como un todo proporciona "facilitadores" para las distintas etapas de un proceso de gestión del conocimiento genérico, por ejemplo, los contenidos consumidos desde *Twtter*, *Delicious*, *Wordpress*, *Blogger* y la información proporcionada por las herramientas de sindicación sirven en la etapa de descubrimiento; o por ejemplo, los mecanismos para colocar "marcas" que se ofrecen en el directorio de usuarios sirven como herramientas de clasificación.

El alcance del trabajo abarca el desarrollo de los siguientes puntos:

- Análisis y diseño del sistema propuesto. En este punto se analizan los requisitos del sistema y se efectúa el diseño completo del mismo, proporcionando productos entregables tales como por ejemplo el modelo de clases, el modelo de entidad – relación y el modelo de flujo de datos.
- Desarrollo del prototipo de portal. Constituye la interface principal del sistema web. La función esencial es servir de *puerta de acceso* al usuario a un conjunto de recursos y de servicios, como por ejemplo un buscador, extractos de

contenido relavante extraídos desde la wiki, el foro, el directorio de usuarios, los sitios de cátedra y el directorio de audio y video.

Para cumplir con el objetivo de construir un portal *personalizable*, en el sentido de que el usuario configure qué visualizar, como estrategia se aplica el concepto de "widget". Un widget es un objeto con interfaz de usuario *propia* que el usuario puede seleccionar y ubicarlo en el portal. En este caso el prototipo posibilita al usuario insertar *pestañas* sobre las cuales se pueden organizar los widgets que se deseen visualizar. Se desarrollaron algunos widgets para demostrar el funcionamiento del portal.

- Desarrollo del prototipo de directorio de usuarios. Es una de las prestaciones más importantes que ofrece esta aplicación ya que los usuarios pueden disponer de una sección común a través de la cual comparten información sobre experiencias laborales, educativas, intereses y especialidades. Esta aplicación incluye además una casilla de correo y un chat para que el usuario pueda enviar mensajes instantáneos a otros que estén "conectados" en el mismo. Dentro del sistema global, este directorio constituye una aplicación relevante por el hecho de que sirve de fuente de información sobre los usuarios (recopila información sobre sus intereses, especialidades, antecedentes laborales, antecedentes educativos, antecedentes de investigación, publicaciones que le son de interés, acceso a sus contenidos producidos en sus blogs, acceso a sus publicaciones en Twitter, sus cuentas en las demás aplicaciones del sistema como el portal, la wiki, el foro, el directorio de audio y video, entre otros). Todos estos datos sirven como materia prima para efectuar recomendaciones de contenido al usuario.
- Programación de las funcionalidades de la capa de filtrado e inteligencia colectiva. Los algoritmos que se implementaron son: 1) los que permiten descubrir patrones sobre los datos (clustering); 2) los que permiten visualizar de una manera apropiada los datos resultantes del análisis de clustering que se efectúe (escalado multidimensional); y, 3) los que sirven para apoyar los procesos de búsqueda de contenido generados en el sistema. Este último utiliza una red neuronal donde el proceso de entrenamiento se basa en la interacción que los usuarios hacen del motor de búsqueda y del feedback que los mismos proporcionan.

- Integración de aplicaciones existentes: wiki; foro; RSS y servicios web de Blogger, Wordpress, Twitter y Delicious.

1.5. JUSTIFICACIÓN

Los grandes cambios que se vinieron produciendo en la última década como producto de los avances del área de las Tecnologías de la Información y la Comunicación (TICs) plantean nuevos retos a la sociedad y en particular a las instituciones educativas [12, 17, 18, 28].

Por tal motivo, a través del presente trabajo se proporciona un enfoque que permita a las instituciones de Educación Superior, y en particular a la UNSE, hacerse eco de estas manifestaciones y aprovechar los avances de las TICs de manera de usarlas como herramientas que permitan darle soporte en su misión social y contribuir a solucionar los problemas detectados.

Con el sistema desarrollado se espera conseguir el logro de los siguientes beneficios:

- Incrementar la accesibilidad y disponibilidad de los recursos y conocimientos generados a partir de las actividades académicas;
- Potenciar los procesos de colaboración, permitiendo a los profesores y alumnos interactuar para desarrollar contenidos educativos;
- Disponer de un ambiente de aprendizaje atractivo (recursos multimedia), al soportar modos personalizados de recuperar, administrar y transformar la información.
- Disponer de nuevos formatos para la diseminación, adquisición y administración del conocimiento.

.

CAPÍTULO II
MARCOS REFERENCIALES

2.1. MARCO TEÓRICO Y CONCEPTUAL

2.1.1. DATO, INFORMACIÓN, CONOCIMIENTO Y SABIDURÍA

Un aspecto controversial relacionado con el objeto de estudio es definir qué se entiende por conocimiento. Los conceptos datos, información y conocimiento muchas veces son confundidos y se los tiende a usar como sinónimos. Por tal motivo, es de fundamental importancia poder establecer las distinciones entre los mismos.

Los **datos** son un conjunto discreto de hechos objetivos acerca de eventos. Carecen de sentido, porque describen parcialmente lo que sucede y no proporcionan juicio ni interpretación, ni permiten la toma de decisiones. Por sí mismos no dicen nada significativo [32].

La **información** es considerada como datos dotados de relevancia y propósito. Persigue cambiar la forma en que el receptor percibe algo, influir sobre su juicio y su comportamiento. El receptor es el que da categoría de información a un determinado hecho [32].

A diferencia de los datos, la información tiene sentido. No sólo tiene el potencial de modelar al receptor, sino que en sí misma tiene *forma*, está organizada con algún propósito. La relevancia entre datos e información es la siguiente: los datos se convierten en información cuando se les añade sentido mediante varios métodos (Figura II.1):

- Contextualización. Se sabe para qué propósito fueron recolectados.
- Categorización. Se conocen las unidades de análisis o los componentes claves de los datos.
- Cálculo. Los datos se analizan matemática y estadísticamente.
- Corrección. Se eliminan los datos erróneos.
- Condensación. Se resumen los datos disponibles.

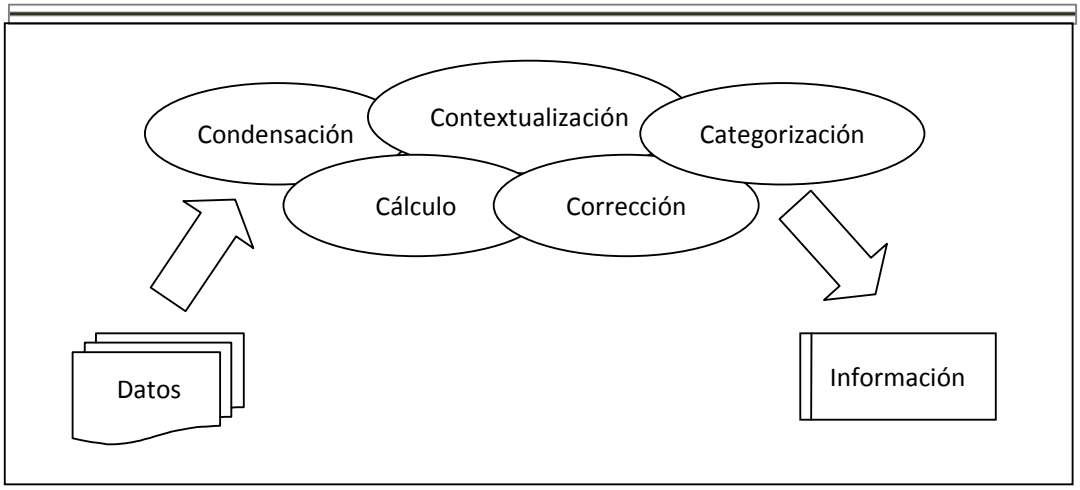


Figura II.1. Métodos que permiten transformar los datos en información. Adaptado de [44]

Con respecto al **conocimiento**, Thomas Daveport lo define como "una mezcla fluida de experiencias, valores, información contextual y apreciaciones expertas que proporcionan un marco para su evaluación e incorporación de nuevas experiencias e información. Se origina y aplica en la mente de los conocedores. En las organizaciones a menudo está embebido no sólo en los documentos y base de datos, sino también en las rutinas organizacionales, en los procesos, prácticas y normas", citado en [31].

Nonaka y Takeuchi definen el conocimiento desde una perspectiva organizacional como la unidad básica de análisis para explicar el comportamiento de la organización [31].

En [1] consideran que el conocimiento es una creencia personal justificada que incrementa la capacidad individual para tomar acciones efectivas. Las acciones pueden ser descritas en términos de habilidades y competencias físicas, actividades intelectuales y cognitivas o ambas.

Para [30] el conocimiento es simplemente **información accionable**. Accionable se refiere simplemente a la noción de relevante. Por información relevante se hace referencia a información que este disponible en el momento adecuado, en el lugar justo, en el contexto adecuado y en el modo justo de manera que cualquiera (no sólo el productor), pueda recuperarla para ayudarse en la toma de decisiones.

La relación clave entre información y conocimiento se expresa en la idea comúnmente aceptada de que el conocimiento en el contexto de las organizaciones no es más que información accionable [30]. Este autor afirma que si se puede usar una

determinada información para hacer aquello que se intenta hacer, entonces esta podría decirse que, se convierte en conocimiento. El conocimiento es información almacenada o capturada junto con su contexto.

El conocimiento no es algo claro, tajante o simple. Todo lo contrario, es poco claro, difuso, parcialmente estructurado y parcialmente desestructurado. Es intuitivo, difícil de comunicar y difícil de expresar en palabras e ilustraciones, y una buena porción de este no es almacenado en base de datos, pero sí en la mente de las personas quienes trabajan en la organización [1]. Está presente en las conexiones, en las conversaciones entre personas, en las intuiciones basadas en la experiencia, y en la habilidad de las personas para comparar situaciones, problemas y soluciones. Sólo una porción minúscula de este conocimiento tácito se logra formalizar en base de datos, libros, manuales, documentos y presentaciones; el resto descansa en la mente de las personas.

En este trabajo se emplea el concepto de conocimiento proporcionado por Amrit Tiwana [30] debido a que tiene una implicación práctica del concepto y se condice con esta propuesta de aplicar un enfoque práctico de gestión del conocimiento que pueda ser implementado al ámbito de la Educación Superior.

Por último, la sabiduría es vinculada con los principios, la introspección, la moral, los arquetipos, tratando de dar respuestas al porqué de las cosas.

Una característica distintiva entre dato, información y conocimiento es el método de transferencia. A medida que uno se acerca a la producción de sabiduría, la información se vuelve no programable, luego el uso de tecnologías de información se reduce o se elimina [16]. En la figura II.2 se ilustra esta observación.

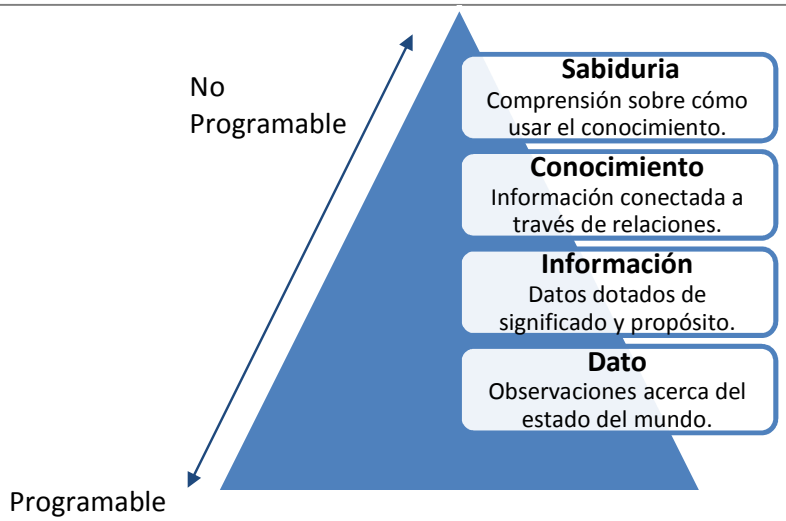


Figura II.2. Evolución del conocimiento. Adaptado de [44]

2.1.2. TIPOS DE CONOCIMIENTO: TÁCITO Y EXPLÍCITO

El conocimiento tiene su origen en la mente de los individuos, como síntesis de diversos componentes: creencias, experiencias, inteligencia, intuiciones, juicios, valores, etc. Este conocimiento puede ser transmitido mediante el lenguaje y la observación. Además, podemos servirnos de diversos medios para transcribir determinados componentes del conocimiento mediante su codificación formal: base de datos, documentos, correos electrónicos, esquemas, webs, etc.

Esta situación del conocimiento en las mentes de las personas y en medios físicos ha dado lugar a la clasificación ampliamente aceptada que contempla dos categorías [1]:

- *Conocimiento tácito*: es el conocimiento personal, almacenado en las cabezas de los individuos, difícil de formalizar, registrar y articular y que se desarrolla mediante un proceso de prueba y error que va conformando el conocimiento del individuo sobre las más diversas materias. Es producto de una actividad social e individual [16]. Está presente en las habilidades personales, en las creencias, en los valores, en los ideales, en la creatividad, en los pensamientos, en los procesos de innovación, entre otros.
- *Conocimiento explícito*: es el conocimiento almacenado en medios físicos, puede ser fácilmente compartido entre las personas, es estructurado y se refleja en cuestiones tales como políticas, procedimientos, patentes, marcas, investigaciones y habilidades registradas.

Una manera de documentar el conocimiento tácito es usando historias, metáforas y analogías [16]. Las historias ayudan a diseminar las experiencias y lecciones aprendidas, las metáforas combinan ideas en imágenes más expresivas, y las analogías ayudan a formar nuevas ideas al compararlas con otras familiares.

2.1.3. EL CONOCIMIENTO EN LAS ORGANIZACIONES

Las organizaciones actuales se enfrentan a un problema similar a la sobrecarga de datos de un par de décadas atrás pero ahora con respecto a la información debido a los volúmenes que de la misma fluyen en nuestros días. Tener mucha información es, en algunos casos, mejor que tener poca, por lo cual se hace necesario descubrir cómo enfocarse en la información que es relevante y aplicable. Por otro lado, al tener una gran cantidad de información resulta difícil darle sentido, por ende esta situación tampoco es beneficiosa. De ahí la necesidad de manejar un concepto de conocimiento como lo plantea Amrit Tiwana.

El conocimiento en las organizaciones es soportado por procesos y estructuras tanto formales como informales para su adquisición, compartimiento y utilización. Los trabajadores del conocimiento o empleados frecuentemente comunican y asimilan valores, normas, procedimientos y datos empezando desde las etapas tempranas de la socialización (cuando la persona llega a la organización y lentamente se va haciendo más dispuesto a compartir), y continuando con el proceso a través de las discusiones e intercambios tanto formales como informales.

Mucho del conocimiento que fluye en una organización, ya sea una empresa pequeña o una de gran envergadura, se crea durante el acto de la colaboración y la acción. Por esta razón, soportar los esfuerzos colaborativos es una parte crítica de cualquier iniciativa de Gestión del Conocimiento.

La actividad social que se genera a través de los procesos de colaboración es de tal importancia que tienen la capacidad de incidir en las maneras de ver los hechos. Por ejemplo, como se plantea en [30] el mismo dato que podría representar información útil para una persona, podría significar nada más que un elemento insignificante para otros. Sin embargo, **muy frecuentemente, cuando los datos se comparten y se distribuyen entre las personas dentro de una organización, estos empiezan a volverse crecientemente útiles a medida que las personas los perciben como tal**.

2.1.4. ACTIVIDADES DEL PROCESO DE GESTIÓN DE CONOCIMIENTO

La creación de conocimiento es una actividad social. Por esta razón, como se afirma en [31], se puede aceptar el convenio de que las actividades del proceso de gestión del conocimiento en una empresa requieren la existencia de al menos dos personas. En tal sentido, estas personas requerirán de ciertos mecanismos que les brinden el soporte necesario para poder colaborar en la tarea de trabajar con el conocimiento.

El proceso de gestión del conocimiento está vinculado con la creación, captura, integración y utilización del conocimiento dentro del contexto de una organización [16].

Como se señala en [31], un modelo genérico de proceso de gestión del conocimiento contempla las actividades de: descubrimiento, captura, clasificación y almacenamiento, distribución y diseminación, compartimiento y colaboración, y utilización de los conocimientos. En la figura II.3 se esquematiza el modelo genérico propuesto por [31].

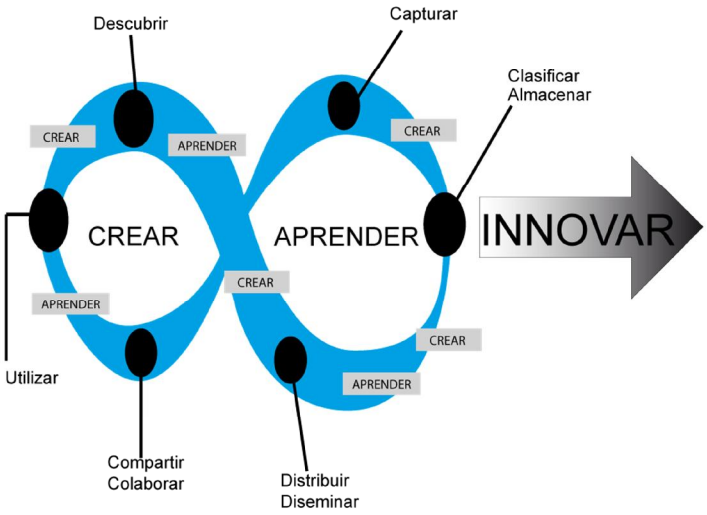


Figura II.3. Actividades del proceso gestión del conocimiento. Extraído de [31]

El objetivo último de la gestión del conocimiento es lograr la innovación, lo cual es fundamental para las organizaciones actuales por el hecho de que deben competir en ambientes crecientemente competitivos y globalizados.

En este modelo genérico, las actividades de creación y aprendizaje son considerados como super-actividades que impregnan a todas las demás. La interacción

entre las actividades es múltiple y cruzada, pudiéndose recorrer en cualquier dirección. El conocimiento puede ser creado al integrar el conocimiento interno existente y las experiencias o al analizar la información disponible. Algunos autores argumentan que en esta etapa, la tecnología es importante porque puede facilitar la creación de nuevo conocimiento a través de la síntesis de datos e información que ha sido capturada desde diferentes fuentes [31].

A través de la actividad de descubrimiento la organización lo que busca es conseguir fuentes de conocimiento, las cuales variarán de una organización a otra. Sus principales fuentes están relacionadas con sus clientes, sus proveedores, sus competidores, entidades reguladoras, y en fin todo el ambiente inmediato de la organización, conocido como ambiente de tarea.

La actividad de captura consiste en localizar y obtener los conocimientos que puedan ser de interés para la acción organizacional. Involucra traer el conocimiento desde afuera hacia la organización.

A través de la clasificación y almacenamiento se busca ordenar la información recolectada en taxonomías, codificarla y almacenarla, para facilitar su posterior recuperación.

La distribución y diseminación hace referencia al proceso de socialización del conocimiento, lo cual es necesario para contribuir con la utilización del mismo.

Con la actividad de compartir y colaborar lo que se busca es incentivar a la creación y fortalecimiento del conocimiento organizacional. Esta es una actividad importante por el hecho de que busca romper la tendencia natural de las personas a monopolizar los conocimientos como un mecanismo de defensa de su puesto de trabajo. La colaboración involucra la transferencia del conocimiento de una persona (o varias) a otra (otras). Por tal motivo, es importante que las organizaciones implementen diferentes métodos que soporten el compartir los diferentes tipos de conocimientos.

Por último, con la aplicación o utilización del conocimiento, la práctica de la gestión del conocimiento llega a su fin cuando el conocimiento almacenado y compartido es utilizado para el beneficio de su organización. La gestión del conocimiento no tiene ningún valor si el conocimiento creado y almacenado no es

utilizado. Mayor conocimiento es creado a medida que el conocimiento es aplicado y utilizado.

2.1.5. Modelo de Conversión De Conocimiento Propuesto Por Nonaka-Takeuchi

La interacción del conocimiento tácito y explícito da lugar a procesos de creación de conocimiento. La figura II.4 muestra el modelo de proceso de conversión de conocimiento propuesto por Nonaka y Takeuchi [31], por el cual el conocimiento es transformado entre sus formas tácitas y explícitas. Ninguno de estos procesos ocurre en forma aislada, sino que naturalmente ocurren en un ambiente de trabajo [16].

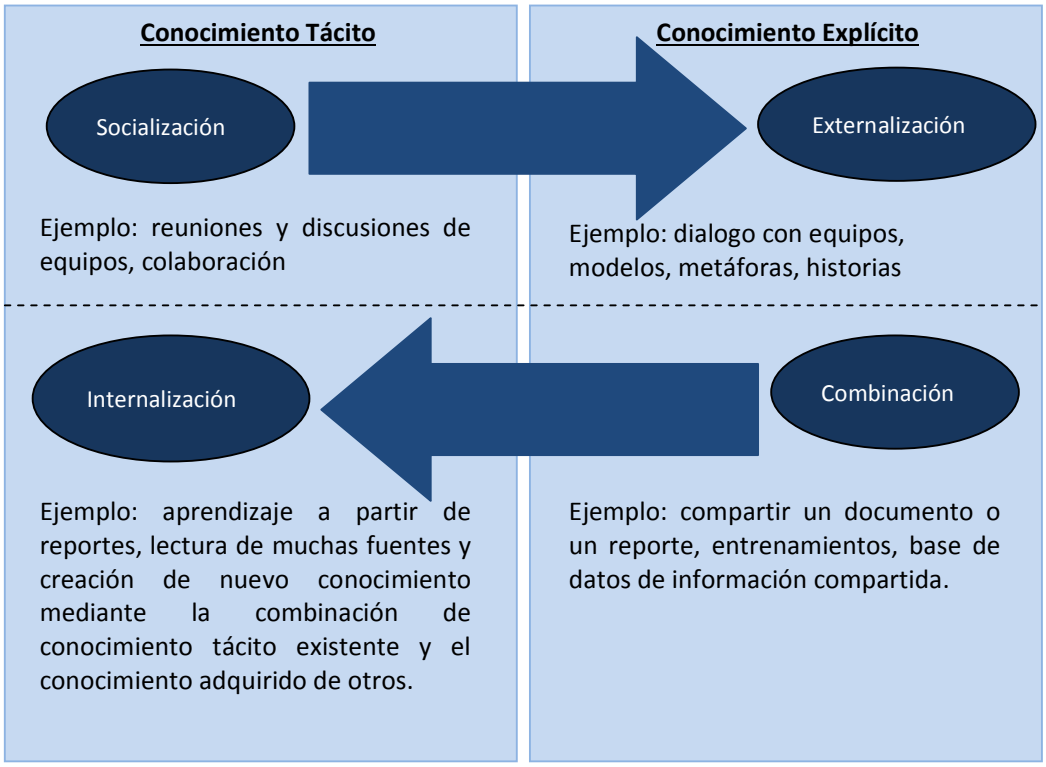


Figura II.4. Proceso de conversión del conocimiento

La conversión de "tácito a tácito" se produce en la **socialización** donde los individuos discuten y colaboran. El conocimiento "tácito converge en explícito" cuando se busca formalizar y codificar el conocimiento personal, este proceso es conocido como **externalización**. La conversión del conocimiento "explícito a tácito" produce la **internalización** a través del cual una persona crea su propio conocimiento tácito a partir del aprendizaje y síntesis del conocimiento compartido por otros. Por último, la conversión de "explícito a explícito", conocido como **combinación**, se produce al

integrar las piezas de conocimiento explícito, ocurre por ejemplo a través de los mensajes de correo electrónico, reportes, base de datos y otros tipos de información compartidos.

2.1.6. PAUTAS PARA FACILITAR LA TRANSFERENCIA DE CONOCIMIENTOS TÁCITOS

Tres aspectos pueden considerarse para facilitar la transferencia de los conocimientos tácitos:

a. Los procesos de colaboración,
b. El hecho de que toda transferencia de conocimiento involucra información como estado intermedio y,
c. Las comunidades de práctica.

a). El proceso de crear, compartir y aplicar el conocimiento inherentemente involucra la colaboración [30]. Este proceso necesita ser soportado por tecnologías informales [30]. En este sentido Tiwana sostiene que la demanda de formalidad hecha por las tecnologías frecuentemente interrumpe las relaciones informales más productivas entre los trabajadores del conocimiento. Por ejemplo, en los motores de búsqueda si un término de búsqueda no se corresponde exactamente con las palabras claves las páginas web nunca serán mostradas en los resultados.

Para soportar estos procesos de colaboración se requiere que las tecnologías de la comunicación promuevan la generación de "comunicaciones ricas" (conversaciones que puedan permitir a las personas conversar casi de la misma manera como si estuviesen cara a cara). Por ejemplo, haciendo uso de los chats, aplicaciones de audio en vivo, el teléfono, y otros mecanismos de comunicación informal. En la figura II.5 se representa estos dos tipos de transferencias: formal e informal que deben ser soportados en un sistema de gestión de conocimiento. La transferencia informal facilita el intercambio de conocimientos tácitos, los cuales una vez que hayan sido asimilados o internalizados pueden ser transmitidos a través de medios formales como ser un artículo, registrado en una base de datos, expresado a través de un video, etcétera. Esta transferencia formal, como los ejemplos lo plantean, promueven el intercambio de conocimientos explícitos.

b). El paso esencial para la transferencia de conocimiento tácito entre personas es la **conversión del conocimiento tácito en información** y luego volver esta a conocimiento tácito. La figura II.6 esquematiza esta observación.

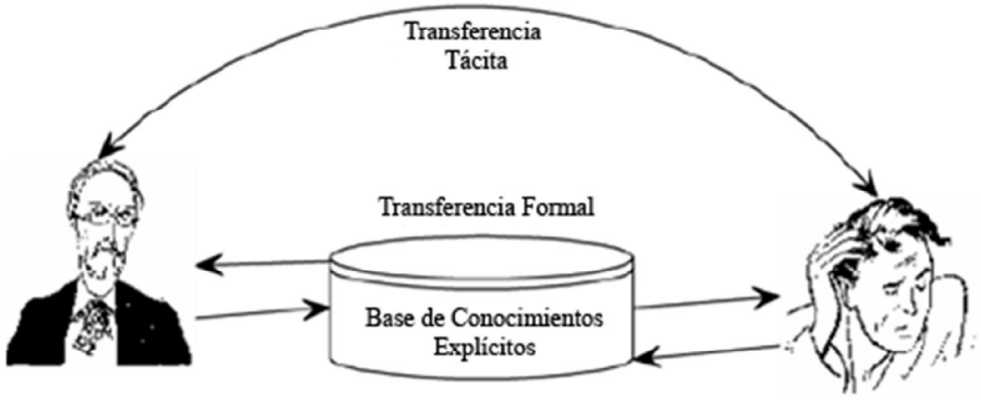


Figura II.5. La transferencia del conocimiento se puede dar a través de canales formales e informales. Adaptado de [30]

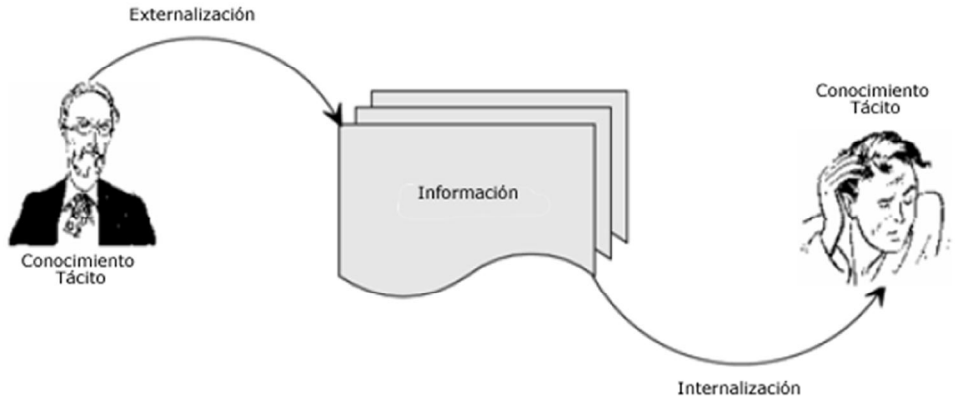


Figura II.6. La transferencia de conocimiento involucra información como un estado intermedio. Adaptado de [30]

Cuando esta transferencia ocurre a través de procesos formales tales como captura de conocimiento en base de datos o a través de mecanismos informales tales como conversaciones, la información como estado intermedio está casi siempre involucrada. Esto implica que **el conocimiento puede ser transferido a través de algo tan complejo como la intranet o base de datos de discusión, o a través de algo tan simple como un teléfono, fax o una conversación cara a cara**. La tecnología no es un precursor para el intercambio de conocimiento pero es un habilitador para situaciones en las cuales no se puede llevar a cabo la transferencia de conocimiento cara a cara.

c). Las **comunidades de práctica** son cruciales para asegurar que el conocimiento tácito sea capturado y compartido [30]. Una comunidad de práctica es un grupo de personas quienes trabajan juntos en la misma área de interés dentro de una organización

[5]. En otros términos puede decirse que a través de la práctica, una comunidad de práctica desarrolla una comprensión compartida de qué hacer, cómo hacerlo y cómo esta se relaciona con otras comunidades y sus prácticas [5].

El foco de una comunidad de práctica es mejorar el proceso de compartir el conocimiento. Una comunidad de práctica puede ser realizada de dos maneras: la primera es a través de reuniones cara a cara en las cuales las interacciones entre los individuos dentro de la comunidad puede ser sobre la base del "uno a uno" o del "uno a muchos". El segundo modo es a través de ambientes potenciados por tecnologías. La combinación de ambos enfoque genera mejores resultados [5].

2.1.7. CARACTERÍSTICASDE UNA ARQUITECTURA DE GESTIÓN DE CONOCIMIENTO

Amrit Tiwana, a modo de analogía, reconoce que el teléfono dispone del mejor conjunto de características que se podrían desear para un sistema que soporta flujos de conocimientos de manera efectiva, puesto que cuando se habla por este medio se puede transmitir contexto, significados, actitudes y tono, junto con información y datos. Además de estas características, estos dispositivos están profundamente embebidos en las comunidades de una organización.

De esta analogía con el teléfono se derivan las características básicas que cualquier sistema que sea de soporte al conocimiento debe poseer:

- Aceptación. El sistema debería ser bien aceptado en la comunidad en la cual se lo usará, no sólo en la comunidad que lo crea.
- Herramientas que faciliten comunicaciones ricas.
- Un modo a través del cual se permita circular el contexto, significados, opiniones, tonos y prejuicios.
- Poder de captación. El usuario no debería sentirse como si estuviera usando algo que no usaría si se le da la opción de elegir.
- Transparencia para el usuario. Representa el hecho de que el sistema como medio pasa desapercibido por el usuario, es algo ubicuo y no debería interferir en su actividad normal.
- Soporta la jerga informal usada en las organizaciones.

La característica inminentemente social de la gestión del conocimiento queda reflejada como una enseñanza de los procesos de investigación que llevó adelante la comunidad de inteligencia artificial (IA). Esta comunidad llevó años intentando probar cómo encapsular conocimiento en un repositorio. Aunque los esfuerzos de la comunidad de IA en esta dirección fueron infructuosos, no fueron en vano: estos esfuerzos han permitido hacer dar cuenta que **la inteligencia humana y el conocimiento no pueden ser completamente codificados**. Con esto en mente, un sistema de gestión de conocimiento no debería buscar eliminar la necesidad de la interacción humana directa. Hay mucho contexto (tales como el tono de las conversaciones o las expresiones faciales) que no pueden ser bien representadas en algún tipo de base de conocimiento o repositorio.

Por otro lado, se necesita un sistema de manera tal que los miembros de la organización puedan obtener información que ellos necesitan desde cualquier lugar y a cualquier hora. Por esta razón, es muy importante adherirse a los estándares industriales tales como HTML, XML, TCP/IP, y ODBC a fin de poder implementar un sistema de gestión de conocimiento rápidamente, con facilidades de extenderlo y personalizarlo en el futuro [30].

Por último, [30] sostiene que se requiere un modelo de Gestión del Conocimiento integrativo e interactivo. *Integrativo* en el sentido de que sea capaz de acumular en repositorios de conocimientos distribuidos los contenidos capturados explícitamente; e *interactivo* en el sentido que es necesario para soportar lo integrativo. Esta segunda característica debe habilitar la interacción entre las personas y proveer un canal básico para compartir el conocimiento tácito; permitiendo que los productores y consumidores de contenido frecuentemente intercambien sus roles.

2.1.8. WEB 2.0

Web 2.0 es un término ampliamente usado en la actualidad, sin embargo, como ocurre con el concepto de conocimiento, todavía no existe una amplia aceptación acerca de su significado. El término surgió a fines del año 2001 con la explosión de la burbuja "punto COM" y fue acuñado por los organizadores de las conferencias de la editorial O'Reilly.

La web 2.0 es un concepto que intenta explicar la naturaleza de las nuevas aplicaciones y servicios que se están ofreciendo en Internet. A continuación se exponen definiciones extraídas de [11] proporcionadas por diferentes autores:

Dorion Carroll, vicepresidente de ingeniería de *Technorati*, considera la Web 2.0 como un medio que permite conectar a las personas y les proporciona la oportunidad de disponer de entornos abiertos para compartir ideas. Las personas pueden publicar contenido rápidamente y de manera fácil con la ventaja de poder ser descubierto y leído fácilmente, al tiempo que permite a otras personas reaccionar ante estos contenidos. En estas aplicaciones existe más la noción de Web potenciadas por personas, a través de empleo de gestos sociales como realizar enlazados (linking), leer, votar, o enviar y compartir contenidos vía mail.

Raju Vegesna, portavoz de *Zoho*, una compañía que produce productos que facilitan la colaboración y el trabajo de oficina, manifiesta que se habla de Web 2.0 cuando los usuarios tienen la posibilidad de comunicarse con los productores de contenido. Es también la combinación de sabiduría de la gente y la Web lecto-escribible.

Richard MacManus, fundador de *Read/Write Web*, sostiene que definir lo que significa el término Web 2.0 es difícil. Considera que podría ser la combinación de sitios sociales y software, y redes sociales tales como *MySpace* y *Facebook*. Pero también contenido generado por el usuario, como por ejemplo, *YouTube*. Es todo esto más tecnologías como RSS y el uso de Web Services y APIs para relacionar sitios y ponerlos juntos.

Shaun Walker, presidente y jefe arquitecto de *DotNetNuke*, analiza que la Web 2.0 puede ser particionada en diferentes cuadrantes en términos de funcionalidad y experiencia de usuario. Un aspecto es que el contenido es generado por el usuario, donde el público está ahora interesado en contribuir con contenido a Internet como un bien público, lo cual es diferente si se compara con el pasado donde el consumidor sólo consumía el contenido. Ellos ahora tienen la capacidad de poder interactuar, expresar sus opiniones, y volverse parte de un gran repositorio de conocimiento en la esfera de la web.

Para Walker, las redes sociales es otra parte importante de la Web 2.0 que permite enlazar el contenido generado por el usuario y la asociación de diferentes usuarios. Otro

cuadrante considera que son los buscadores y la sindicación a contenido, puesto que son herramientas importantes ante la emergencia de gran cantidad de contenido. Por último, observa que una cuestión importante en las aplicaciones Web 2.0 es proporcionar interfaces de usuario ricas, a través de la adopción de tecnologías como AJAX, Flash y Microsoft Silverlight.

Por último, según O´Reilly, la Web 2.0 es un concepto difícil de definir, más bien propone visualizarla como un conjunto de principios y prácticas [22]:

1. Utilizar la web como plataforma: se refiere al paso desde el concepto o idea del sistema operativo instalado en el ordenador a la plataforma de estándares abiertos de la web, que permiten acceder a la aplicación desde cualquier sistema operativo.

2. La base de datos es el centro del desarrollo: toda aplicación web utiliza una base de datos especializada, sin embargo la ventaja competitiva se obtiene cuando esa base de datos es difícil de replicar, ya sea por su contenido básico y específico o por el valor agregado que se le otorgue.

3. Ofrecer el software o aplicación como un servicio en desarrollo: actualmente el software se ofrece como un servicio a través de la web, y no como un producto cerrado para un sistema operativo determinado, es decir, con capacidad de actualización constante, para mejorar la aplicación y el servicio. De allí que los servicios de la Web 2.0 estén en versión beta constante.

4. Utilizar modelos de programación abiertos: en contraposición a las aplicaciones de escritorio con sistemas propietarios, la Web 2.0 utiliza un conjunto de sistemas abiertos para lograr sus aplicaciones.

5. Ofrecer experiencias de usuario ricas y simples: referido a la posibilidad de proveer interfaces de usuarios más ricas que las tradicionales, con características similares a las aplicaciones de escritorio que puedan ser accedidas desde cualquier terminal conectada.

6. Generar aplicaciones para más de un dispositivo: actualmente, las aplicaciones son planteadas como servicios integrales para distintos tipos de dispositivos, PC, móvil o dispositivos portátiles.

2.1.9. APLICACIONES SOCIALES

Algunas de las aplicaciones sociales más conocidas según [10], son los blogs, las wikis, los canales de marcadores sociales, los sitios de uso compartido, las redes sociales y las tecnologías de sindicación y notificación.

Blogs. El término blog originalmente proviene de la frase "web-log", el cual se refiere a una simple página web que contiene párrafos de opinión, información, entradas diarias personales, o links ordenados de manera cronológica a partir de la entrada más reciente, al estilo de un diario online [7]. El proceso de "bloguear" se caracteriza por "postear" (escribir y publicar artículos) y comentar los contenidos de los blogs generando un intercambio de visiones entre el/los autores del blog y los lectores [3]. Las entradas de los blogs pueden incluir videos dependiendo del software de "blogueo" o del servicio que se use.

Relacionado a este concepto está el término de *blogósfera* con el cual colectivamente se engloba todos los blogs como una comunidad o red social.

Otros aspectos que hacen tan populares a los blogs como medios de comunicación es el conjunto de tecnologías que emplean para permitir construir una comunidad entre usuarios: enlaces permanentes, enlaces inversos (trackback) y RSS. Los enlaces permanentes son links persistentes a determinados posteos de un blog. El trackback se trata de un enlace inverso que permite conocer qué enlaces apuntan hacia un determinado post; de ese modo, avisa a otro weblog que se está citando uno de sus posts. El concepto referente a RSS se trata posteriormente en esta misma sección.

Wikis. Es un sistema que permite a una o más personas construir un cuerpo de conocimientos a través de un conjunto de páginas web interconectadas, usando un proceso de creación y edición de páginas.

La palabra "wiki" se originó a partir de un término hawaiano el cual significa "rápido" o "super rápido". El concepto fue usado por primera vez en 1994 por Ward Cunningham para desarrollar de manera rápida y colaborativa contenido para la web. El sitio wiki más popular es *Wikipedia*.

La característica distintiva de una wiki es que cualquiera puede editarla actuando de esta manera como una herramienta colaborativa que facilita de un modo más efectivo el desarrollo de trabajos grupales.

La naturaleza "abierta" hace a las wikis más flexibles que los tradicionales sitios web basados en editores. Cuando una nueva información se hace disponible puede ser agregada inmediatamente a la wiki sin necesitar la aprobación de un editor. Similarmente, cuando un error es detectado por un lector éste puede corregirlo sin necesidad de contactarse con el administrador del sitio o el autor del documento. Además, los documentos de una wiki pueden ser escritos usando una sintaxis que relativamente no es complicada, y características tales como el control de versión hacen que las wikis sean bien recibidas en ambientes de trabajo colaborativo.

Sin embargo, esta naturaleza abierta de las wikis presenta una serie de problemas. Debido a que no existe un control de calidad por parte de "editores", errores pueden ser insertados accidentalmente o incluso de manera deliberada. Los lectores, por otro lado, podrían confundirse con la información provista en una wiki de confianza. Otro problema es que información incorrecta, contenido difamatorio y publicidades pueden ser insertados, y otros contenidos pueden ser borrados o sobreescritos. Sin embargo, generalmente la información incorrecta se repara en breve tiempo al revertir una página a su versión previa.

Canales de marcadores sociales. Proveen a los usuarios de facilidades para registrar ("bookmark") páginas web y marcar aquellos registros con palabras significantes ("tag") que permitan describir la página que está siendo registrada. Este proceso es conocido como "tagging" [20]. El principio básico del "tagging" es que los usuarios finales tienen la potencia de crear sus propios índices en vez de que esta actividad sea sólo realizada por expertos; los tags asignados están inmediatamente disponibles en la web [36].

A través del tiempo el conjunto de palabras claves crea construcciones de registros con "tags" comunes, y los usuarios pueden buscar ítems registrados por "tags" similares, debido que aquellas páginas que hayan sido consideradas que valen la pena ser resguardadas y clasificadas con uno o más *"tags"*, tienen mayor *"visibilidad"* y probabilidad de ser encontradas. Esto hace que las herramientas de bookmarking puedan algunas veces ser más efectivas que los motores de búsqueda para encontrar recursos en Internet.

Folksonomía es un término relacionado al concepto de *"tagging"* en el sentido de que hace referencia a una colección de *"tags"* creados por un individuo para uso

personal. El tagging social fue visto como un modo útil de conseguir una clasificación de contenido fiable a partir de la colaboración de una gran cantidad de personas [36].

Sitios de uso compartido de archivos multimedia. Estos servicios almacenan archivos multimedia compartidos por el usuario, y permiten buscarlos y visualizarlos. Además de servir como un medio en donde se puede apreciar la actividad creativa, estos servicios pueden constituirse en valiosos recursos educativos.

Como ejemplos de estos servicios se pueden mencionar a *YouTube* [46] (para compartir archivos de video), *iTunes* [40] (para compartir podcasts y vidcasts), *Flickr* [39] (para compartir fotos), *Slideshare* [45] (para compartir presentaciones), *DevianArt* [37] (para compartir trabajos artísticos) y *Scribd* [43] (para compartir documentos).

Un tipo de sitio particular para compartir archivos multimedia son los de *podcasting*, modo por el cual el usuario puede estar actualizado con los contenidos de audio recientes subidos a un determinado sitio web. Estas grabaciones de audio son conocidas como *"podcasts"*, usualmente en la forma de charlas, entrevistas, conferencias, las cuales pueden ser escuchadas ya sea a través de una computadora de escritorio o por medio de un reproductor MP3 [3]. Originalmente los podcasts fueron llamados "audio blogs" y fueron el esfuerzo inicial de agregar audio en los primeros blogs. Los oyentes de los podcasts normalmente se adhieren a sus canales RSS para recibir nueva información sobre nuevos podcasts que están disponibles. Los vidcasts o vodcasts son versiones de video de los podcasts.

Redes sociales o "social networking". Las redes sociales son estructuras que describen las relaciones sociales entre individuos. Cada nodo en la red es una persona, y las aristas entre los nodos son las conexiones entre individuos, donde el peso de las aristas puede ser usado para denotar el grado de la "amistad" [10].

Las redes sociales son utilizadas principalmente para mantener relaciones informales entre individuos. En estas aplicaciones se ofrecen funcionalidades básicas para chatear con miembros de la red, compartir información, etcétera. Los usuarios que se unen a una red social tienen que llenar un formulario con información sobre su perfil tales como su nombre, la fecha de nacimiento y una foto. Estos datos se hacen disponibles a otros miembros de la red que pueden "ver" sus contactos y también los amigos de sus contactos (amigos de segundo grado). Esta característica facilita claramente crear nuevas conexiones en la red.

Entre las redes sociales más conocidas se pueden citar a Facebook [38] y MySpace [42] (redes para socializar), LinkedIn [41] (redes de contactos profesionales), Second Life [44] (simulación de mundos virtuales), entre otros. Estas aplicaciones permiten a los usuarios describirse a ellos mismos y sus intereses, y generalmente implementan las nociones de amigos, "ranking", y comunidades.

Sindicación y notificación. En un mundo de contenidos compartidos y en el cual constantemente se producen nuevos contenidos, es útil ser capaces de mantenerse actualizados de una manera fácil de los nuevos contenidos o de sus actualizaciones, particularmente si uno está interesado en múltiples fuentes de información sobre múltiples sitios web. Cada uno de los cambios que se detectan en las fuentes se denomina *"feed"*. Comunmente se usa un lector de feed para centralizar todos los cambios recientes en las fuentes de interés, y un usuario puede fácilmente usar un lector para ver los nuevos contenidos o sus cambios. Bajo esta escena, existen dos protocolos que se encargan de estas funcionalidades: RSS (Really Simple Syndication) y Atom. Estos lectores regularmente extraen de los sitios designados los feeds, los muestran en forma de resumen, y permiten a los usuarios ver tales entradas de manera completa.

De los protocolos mencionados en el párrafo anterior, RSS es el más conocido por el hecho de que provocó un avance significativo en la arquitectura fundamental de la web [22]. RSS es un formato XML que permite a los usuarios conocer acerca del contenido de sitios webs, blogs, o podcasts que proveen RSS, sin tener que necesariamente visitar el sitio [3].

2.2. MARCO METODOLÓGICO

2.2.1. GUÍA PARA LA CONSTRUCCIÓN DE SISTEMAS DE GESTIÓN DE CONOCIMIENTO DE AMRIT TIWANA

La guía propuesta por Amrit Tiwana para la construcción de sistemas de gestión del conocimiento se organiza en diez etapas y se basa en los fundamentos teóricos propuestos por Davenport [30]. Las diez etapas se agrupan en cuatro fases: *evaluación de la infraestructura*; *análisis, diseño y desarrollo del sistema de gestión del conocimiento*; *despliegue* y *evaluación* (Tabla II-1).

En esta propuesta se llevan a cabo las tareas involucradas en las etapas 3 y 7 por ser las que indican las actividades necesarias para diseñar e implementar un sistema de gestión de conocimiento, objetivo fundamental que perseguimos con el trabajo.

Tabla II-1. Etapas de la guía metodológica de Amrit Tiwana

ETAPAS		FASES
1	Analizar la infraestructura existente.	**FASE I** **Evaluación de la infraestructura**
2	Alinear la gestión del conocimiento con la estrategia de negocio.	
3	Diseñar la infraestructura de gestión del conocimiento.	**FASE II** **Análisis, diseño y desarrollo del sistema de gestión del conocimiento**
4	Auditar/inventariar los activos y sistemas de gestión del conocimiento.	
5	Diseñar el equipo de gestión del conocimiento.	
6	Crear anteproyecto de gestión del conocimiento.	
7	Desarrollar el sistema de gestión del conocimiento.	
8	Desplegar el sistema, con la metodología orientada a resultados.	**FASE III** **Despliegue**
9	Gestionar el cambio, la cultura y las estructuras de incentivos.	
10	Evaluar rendimiento, medir retorno de la inversión y refinar el sistema.	**FASE IV** **Evaluación**

La etapa 3 involucra las siguientes actividades:

1. Comprender los componentes de la infraestructura de conocimiento.
2. Identificar fuentes de conocimiento internas y externas que deben ser integradas.
3. Seleccionar componentes de la tecnología de la información (TI) para buscar, crear, ensamblar y aplicar conocimiento.
4. Identificar elementos de la capa de interfaz: clientes, servidores, puertas de enlace y plataformas.
5. Seleccionar la plataforma colaborativa.
6. Identificar y comprender los componentes de la capa de inteligencia colaborativa.
7. Equilibrar mecanismos basados en *push* y *pull* para la entrega de conocimiento.
8. Crear mecanismos de perfiles para la entrega de contenido.

9. Alinear los componentes de la TI en el modelo de gestión de conocimiento propuesto por Nonaka y Takeuchi para validar las elecciones.

En la etapa 7 se llevan a cabo diferentes tareas orientativas que abarcan el desarrollo de:

1. La capa de interfaz. Se busca lograr independencia de la plataforma, se aprovecha la *intranet*, se habilita la autoría universal, y se optimiza el *streaming* de audio y video.
2. La capa de acceso y autenticación. Se protegen a los datos, se implementa el control de acceso y el control distribuido.
3. La capa de filtrado e inteligencia colectiva ya sea usando agentes inteligentes o sistemas de filtrado colaborativo.
4. La integración de la capa de aplicación con la capa de inteligencia y la capa de transporte.
5. La capa de transporte, en caso de ser posible se toma ventaja de las redes existentes en la organización.
6. El *"middleware"* y la capa de integración a sistemas heredados.
7. La integración y mejora de la capa de repositorios.

2.2.2. REDES NEURONALES

Una red neuronal es un tipo particular de circuito lógico que recibe ese nombre debido a la presencia de ciertos componentes que sirven como modelo de ciertas propiedades que presentan las neuronas biológicas. Estos componentes se caracterizan por efectuar una suma ponderada de las entradas, compara esta suma con un umbral y, si supera el umbral, produce como salida un 1; en cualquier otro caso, produce un 0. Cada una de las entradas a la red tiene un determinado peso asociado (*w*) [19].

Los circuitos lógicos a los cuales se hace referencia tienen la característica de recibir un cierto conjunto de entradas *(s1, s2, ..., sn)* y producir una acción de acuerdo a la entrada recibida. El cálculo de la acción se realiza en dos fases distintas, como se muestra en la figura II.7.

En la fase de procesamiento se genera un *vector de características* $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$, y en la fase de cálculo de la acción se selecciona una acción teniendo en cuenta

dicho vector de características. Los valores del vector de características pueden ser números reales (*características numéricas*), o bien categorías (*características categorizadas*, es decir, aquellos cuyos valores son nombres o propiedades; por ejemplo, el valor de la característica <<color>> puede ser <<rojo>>, <<azul>> o <<verde>>).

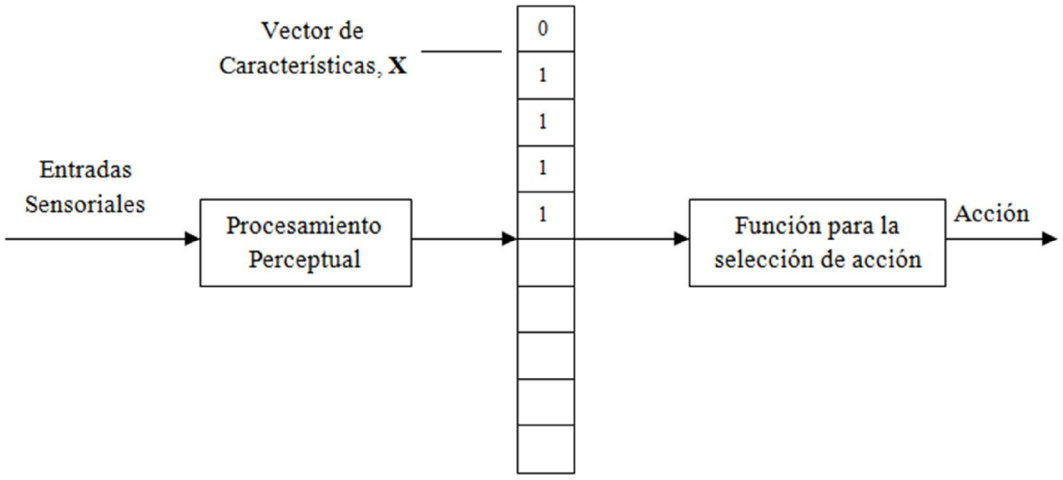


Figura II.7. Componentes de un circuito lógico

La idea es partir de un conjunto E formado por vectores X de dimensión *n* cuyas componentes son x_i, $i = 1, \ldots, n$. Para cada X perteneciente a *E* se conoce la acción *a* que le corresponde. Las acciones que se asocian a los vectores del conjunto *E* suelen denominarse *etiquetas o clases del vector*, y al conjunto *E*, junto con las etiquetas asociadas a sus vectores, se la denomina *conjunto de entrenamiento*. Por tanto, para resolver el problema de aprendizaje, hay que encontrar una función, *f(X)*, que responda "aceptablemente" a los elementos del conjunto de entrenamiento. Normalmente se intenta que las acciones calculadas por la función *f* se correspondan con las etiquetas asociadas a los vectores de entrada en tantos casos como sea posible.

Los pesos de una red pueden ser ajustados o *entrenados* para que la red responda de forma adecuada a determinados conjuntos de entrenamiento. Si la red es usada para el cálculo de acciones, sus entradas han de ser numéricas (para que pueda calcular la suma ponderada de las mismas); por tanto, si el procesamiento produce características categorizadas, éstas deben ser codificadas numéricamente. Una red neuronal compuesta de una sola *"neurona"* recibe el nombre de *Perceptron* o *Adaline* (del inglés, *ada*tive *lin*ear *e*lement, es decir, elemento lineal adaptativo).En la figura II.8 se ilustra un perceptron tipo.

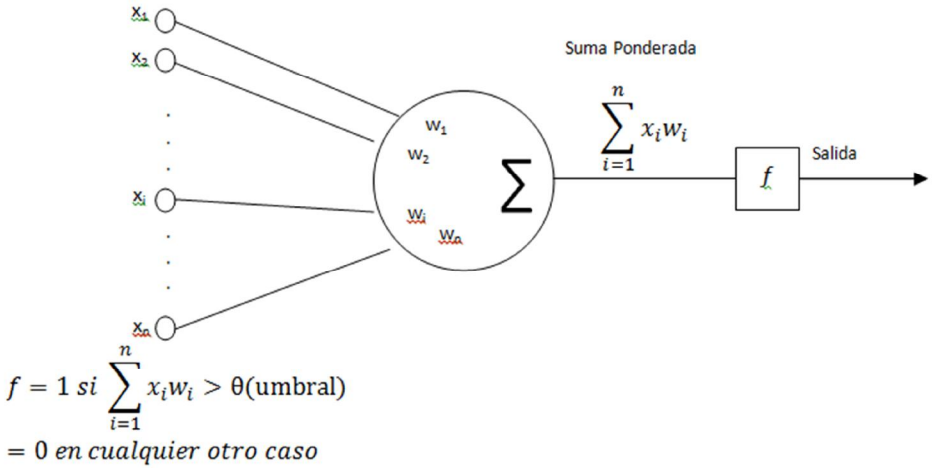


Figura II.8. Unidad lógica con umbral

El entrenamiento de una red se realiza ajustando los pesos variables hasta que se consigan las salidas deseadas. El método más utilizado para efectuar estos ajustes es el del gradiente descendiente. El objetivo es minimizar una función de error, generalmente la más usada es la función del error cuadrático:

$$\varepsilon = \sum_{\epsilon} (d - f)$$

Donde es la respuesta de la Unidad Lógica con Umbral (ULU) al vector de entrada , es la respuesta deseada y el sumatorio se extiende a todos los vectores del conjunto de entrenamiento. Dado un conjunto de entrenamiento E determinado, se puede comprobar que el valor de ε depende de los valores que tengan los pesos (la dependencia aparece a través del cálculo de).

De esto se deriva que el mínimo de ε es:

$$\_\_ \stackrel{\text{def}}{=} \_\_, \ldots, \_\_, \ldots, \_\_$$

Como los ε's dependen de W a través del producto escalar $s = X \cdot W$, aplicando la regla de derivación de la cadena, obtenemos:

$$\_\_ = \_\_$$

Entonces, dado que — = X

— = —

Teniendo en cuenta que — = −2(d − f) —, se puede escribir:

— = −2(−) —

Sin embargo, dado que la salida de *f* es una función de umbral, la derivada de *f* no es continuamente diferenciable respecto a *s*, debido a la presencia de la función umbral. La mayor parte de las pequeñas variaciones que se producen en el producto escalar no afectan a *f*, y cuando *f* cambia, lo hace de manera abrupta, pasando de 0 a 1, o viceversa. Hay dos formas de solventar esta dificultad. En una se ignora la función umbral y se asume que *f* = *s*. En otra, se reemplaza la función umbral por cualquier otra función no lineal que sea derivable. Este segundo método es conocido como "Delta Generalizado" y es el que se empleó en el trabajo, reemplazando a la función *f* por una tangente hiperbólica.

Vale destacar que en el presente trabajo se desarrolló una red neuronal multicapa (múltiples perceptrones) y el método de entrenamiento utilizado fue el de la *retropropagación*, el cual se basa en el método del gradiente descendente.

En una red neuronal multicapa (existe capa de entrada y capa de salida), todas las unidades sigmoidales, exceptuando las de la última capa, se denominan *unidades ocultas*, ya que sus salidas sólo contribuyen indirectamente a la salida final.

2.2.3. COEFICIENTE DE CORRELACIÓN DE PEARSON

El coeficiente de correlación de Pearson, pensado para variables cuantitativas, es un índice que mide el grado de covariación entre distintas variables relacionadas linealmente.

El coeficiente de correlación de Pearson es de fácil ejecución y de fácil interpretación. Los valores del coeficiente oscilan entre -1 y 1. Cuando el coeficiente es -1, la relación es perfecta negativa; cuando es 1, perfecta positiva.

La correlación entre dos variables X e Y es perfecta positiva cuando exactamente en la medida que aumenta una de ellas aumenta la otra. Esto sucede cuando la relación entre ambas variables es funcionalmente exacta. Por ejemplo, la relación entre espacio y tiempo para un móvil que se desplaza a velocidad constante; gráficamente, la relación se muestra en la figura II.9.

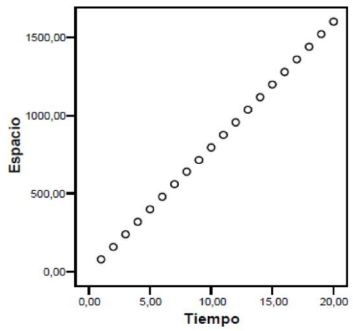


Figura II.9. Ejemplo de una correlación perfecta positiva

La relación es perfecta negativa cuando exactamente en la medida que aumenta una variable disminuye la otra. Sucede para relaciones funcionales exactas. Por ejemplo, la relación entre presión y volumen se ajusta a este caso; en la figura II.10 se ilustra la situación:

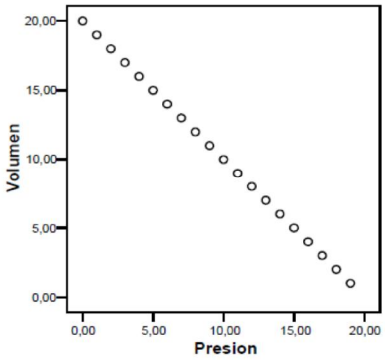


Figura II.10. Ejemplo de una correlación perfecta negativa

El coeficiente de correlación de Pearson viene definido por la siguiente fórmula:

$$= \frac{\sum \; - \frac{\Sigma \; \Sigma}{}}{\sum \; - \frac{(\Sigma \;)}{} \quad \sum \; - \frac{(\Sigma \;)}{}}$$

Donde x e y son las coordenadas de los ejes de las abscisas X e Y respectivamente. N es la cantidad de pares ordenados considerados.

2.2.4. SISTEMAS DE RECOMENDACIÓN

Los sistemas de recomendación son herramientas de ayuda para la toma de decisiones muy utilizados en la actualidad, y de los cuales existe un amplio espectro de aplicaciones [26], sobre todo para comercio electrónico y ocio. Son muy usados en los procesos de descubrimiento de conocimiento y como técnica para resolver el problema de hacer recomendaciones personalizadas de información [25].

Los sistemas de recomendación de tipo colaborativo han sido los que mayor difusión han tenido debido a su simplicidad y buenos resultados [25]. Dichos sistemas usan valoraciones de una comunidad de usuarios sobre una serie de ítems para recomendar a un usuario *u* un ítem *i* que aún no ha valorado, estimando el valor que *u* daría a los ítems candidatos en base al valor que asignaron sobre los *i* candidatos aquellos usuarios con un parecido historial de preferencias.

En la literatura existente se describen los sistemas de recomendación basados en Filtrado Colaborativo (FC) como sistemas que trabajan recogiendo juicios humanos, expresados como votaciones, sobre una serie de ítems en un dominio dado, y tratan de emparejar personas que comparten las mismas necesidades o gustos [27].

Los usuarios de un sistema colaborativo comparten sus valoraciones y opiniones con respecto a los ítems que conocen de forma que otros usuarios puedan decidir qué elección realizar. A cambio de compartir esta información, el sistema proporciona recomendaciones personalizadas para aquellos elementos que pueden resultar interesantes al usuario.

Es de destacar que en el FC son los usuarios, las personas, quienes determinan la relevancia, calidad e interés de los ítems, por lo que se puede realizar el filtrado sobre elementos difíciles de analizar mediante computación. El FC tiene la capacidad de discernir cómo se adapta un ítem a las necesidades o intereses de los usuarios, basándose en la propia capacidad humana de analizar en términos de calidad o gusto, tarea difícilmente realizable por procesos computacionales.

La información manejada en FC consta de una serie de ítems, usuarios y valoraciones proporcionadas por los usuarios sobre esos ítems: el espacio del problema viene definido como una matriz de usuarios frente a ítems, en la que cada celda representa la puntuación de un usuario concreto referida a un ítem específico.

Resolver un problema típico de FC implica predecir qué valores tendría un usuario para aquellos ítems que aún no ha puntuado, basándose para ello en las valoraciones aportadas anteriormente por la comunidad de usuarios [27]. En cuanto a la recomendación que se muestra al usuario, puede estar formada por los valores previstos para una serie de ítems, o bien por una lista formada por aquellos ítems que supuestamente al usuario deberían gustarle más, teniendo para ello en cuenta si deben o no mostrarse ítems ya votados.

Para generar una predicción deben realizarse una serie de tareas [27]: 1) establecer el valor de similitud entre el usuario activo y el resto, 2) seleccionar un conjunto de usuarios para generar la predicción, y 3) generar una predicción en base a combinaciones ponderadas de las valoraciones que realizaron los vecinos [1] seleccionados.

1. Métrica para definir la similitud entre vecinos

Para establecer la similitud entre vecinos debemos definir una medida que permita evaluar el grado de parecido entre unos y otros. Existen varias, por ejemplo, el coeficiente de correlación de Pearson, la distancia Euclidiana, el coeficiente de Jaccard, la distancia de Manhattan, entre otros [27].

En el presente trabajo se decidió emplear el Coeficiente de Correlación de Pearson. Este coeficiente tiende a dar mejores resultados en casos donde los datos no están bien normalizados.

2. Selección del vecindario

Una vez establecidas las similitudes entre el usuario activo y el resto de los usuarios de la comunidad, es necesario elegir cuáles de estos últimos se usarán para computar las predicciones, puesto que en términos de eficiencia y exactitud utilizar todos para el cálculo no sería viable. Por ello se suele escoger un número predeterminado *K* de vecinos, los *K* con mayor valor de similitud [27]. Los dos procesos anteriores se realizan mediante la implementación de un algoritmo K-NN (K Nearest Neighbors).

[1] Vecinos: un conjunto de usuarios con intereses similares a un usuario en particular.

3. Generación de las predicciones

Tras escoger el vecindario resta combinar las valoraciones de éste para producir una predicción. La forma más sencilla de hacerlo es calcular una media ponderada de las predicciones, sin embargo, una de las aproximaciones más utilizadas y que mejores resultados da es calcular una suma media ajustada de dichas puntuaciones utilizando la media de valoración de cada individuo y las correlaciones como pesos [27].

2.2.5. ANÁLISIS DE CLUSTERS

El *análisis de clusters* o *clustering*, es una colección de métodos estadísticos que permiten agrupar casos sobre los cuales se miden diferentes variables o características. Así, casos que presenten características muy similares deberán quedar agrupados en conjuntos llamados *clusters*, de manera que se diferencien respecto a los casos agrupados en otros clusters. Estos clusters deben ser hallados sin información previa y deben ser sugeridos únicamente por la propia esencia de los datos.

Existen dos grandes bloques de métodos de clustering: los jerárquicos y los no jerárquicos o particionales. En los primeros, la pertenencia a un grupo o cluster en un nivel de la jerarquía condiciona la pertenencia a grupos de un nivel superior. Además, se dividen en aglomerativos o divisivos, según que la jerarquía sea construída agrupando casos o bien dividiendo secuencialmente los datos. Los métodos particionales obtienen una única partición de los datos mediante la optimización de alguna función adecuada.

Los métodos particionales utilizan la matriz de datos mientras que los jerárquicos parten de una matriz de distancias o similaridades.

La distancia desde un caso *i* a un caso *j* recogidos de un conjunto Ω se define como:

$$: \times \rightarrow \Re$$
$$(,) \rightarrow (,) =$$

Tal que verifica las siguientes propiedades:

1. $(,) \geq 0, \forall , \in$
2. $(,) = 0, \forall \in$
3. $(,) = (,), \forall , \in$

El concepto dual a la distancia es la similaridad. Una similaridad sobre un conjunto Ω es una función *s*:

$$: \times \rightarrow \Re$$
$$(,) \rightarrow (,) =$$

Tal que:

1. $0 \leq (,) \leq 1, \forall , \in$
2. $1 = (,) \geq (,), \forall , \in$
3. $(,) = (,), \forall , \in$

Cuanto mayor sea la similaridad *s(i,j)*, más parecidos entre sí serán los casos *i* y *j*.

En general, sobre cada caso de Ω se habrán medido *n*variables y por tanto, cada caso puede ser representado como un punto $= \{ , \dots , \} \in \Re$, de manera que cada x_i es el valor que toma la i-ésima variable X_i medida sobre el caso. Dependiendo de la naturaleza de las variables que se hayan considerado (variables continuas, binarias o mixtas), se deben utilizar diferentes tipos de distancias o similaridades. En [4] se pueden consultar sobre los diferentes tipos de distancias y similaridades según los tipos de variables.

En el presente trabajo se aplica el método de clustering jerárquico aglomerativo. En la siguiente sección se explica este método.

2.2.5.1. Clustering Jerárquico

Los métodos jerárquicos establecen una jerarquía entre los clusters, en otras palabras, generan una sucesión de particiones donde cada partición se obtiene uniendo o dividiendo clusters.

Dentro de los métodos jerárquicos se distinguen dos tipos:

1. Métodos aglomerativos
2. Métodos divisivos

Con los métodos aglomerativos los nuevos clusters se crean uniendo clusters. Es decir, en la partición inicial cada caso forma un cluster. Empieza el proceso de aglomeración de manera que se van uniendo los clusters de dos en dos y finaliza cuando todos los casos forman un único cluster.

Con los métodos divisivos, al contrario, los nuevos clusters se crean dividiendo clusters. Es decir, en la partición inicial todos los casos forman un único cluster. Empieza el proceso de división dividiendo los clusters (habitualmente en dos). El proceso puede seguir hasta que cada caso forme un único cluster.

La principal ventaja de los métodos aglomerativos es su rapidez. Por su parte, los métodos divisivos tienen la ventaja que parten de la información global que hay en los datos y que además el proceso de división no tiene por qué seguir hasta que cada elemento forme un cluster. Sin embargo, estos métodos suelen ser muy lentos y en general aplicables sólo para datos con pocos casos.

El análisis de clustering generalmente es visualizado a través de un *dendrograma* el cual muestra cómo ha sido el proceso de unión o división de clusters. En la figura II.11 se muestra la imagen de un dendrograma.

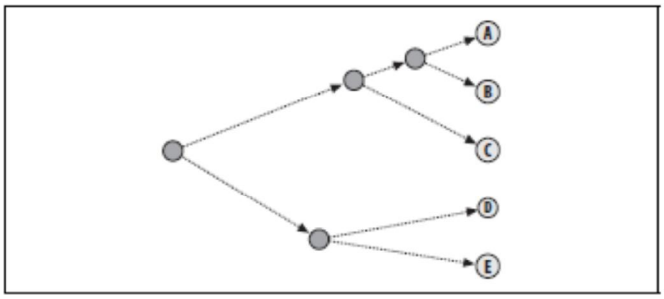


Figura II.11. Un dendrograma es una visualización de un clúster jerárquico

El dendrograma no sólo usa conexiones para mostrar qué ítems pertenecen a qué clúster, sino también se tiene en cuenta la distancia para mostrar cuán distantes los ítems se encuentran. Por ejemplo, en función de la figura II.11, el clúster AB es más cercano a los ítems individuales A y B que el clúster DE lo es a sus ítems individuales D y E.

2.2.5.2. Algoritmo básico de clasificación jerárquica

Dado un conjunto Ω de *m* casos:

1. Inicialmente cada caso formará un clúster. Es decir, la partición inicial es: $P_0 = \{\{1\},\ldots,\{m\}\}$
2. Supongamos que los casos más cercanos son *i* y *j*((,) = í , { (,)}). Donde *u* es la matriz de casos. Entonces, la

unión de estos dos casos formará un único nuevo clúster ($\{\ \} \cup \{\ \} = \{\ ,\ \}$), y se actualizará la matriz *u*: $(\ ,\{\ ,\ \}) = (\ ,\ ) = (\ ,\ )$

3. Considerando la partición obtenida en el paso anterior, $P_1 = \{\{1\},\ldots,\{i,j\},\ldots,\{n\}\}$, se repiten los pasos 2 y 3 del algoritmo hasta que todos los casos de Ω formen un único clúster.

Finalmente, se define el índice $\propto$ de la siguiente manera:

$$\propto(\{\ \}) = 0,\quad = 1,\ldots,\quad ;$$

$$\propto \quad \cup \quad = (\ ,\ )$$

Donde C_i y C_j son dos clústers que se han creado en el proceso de aglomeración. El resultado de este proceso, $(\ ,\propto)$, es una jerarquía indexada.

2.2.6. AJAX

AJAX, acrónimo de ***A**synchronous **J**avaScript **A**nd**X**ML* es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

El modelo clásico de aplicaciones web funciona de la siguiente manera: la mayoría de las acciones del usuario en la interfaz disparan un requerimiento HTTP al servidor web. El servidor efectúa un proceso y devuelve una página HTML al cliente. A la izquierda de la figura II.12 se ilustra el modelo correspondiente.

Una aplicación AJAX elimina la naturaleza "arrancar, frenar, arrancar, frenar" de la interacción en la Web introduciendo un intermediario (un motor AJAX) entre el usuario y el servidor. A la derecha de la figura II.12 se exhibe el modelo correspondiente.

En vez de cargar una pagina Web, al inicio de la sesión, el navegador carga al motor AJAX (escrito en JavaScript). Este motor es el responsable de renderizar la interfaz que el usuario ve y de comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.

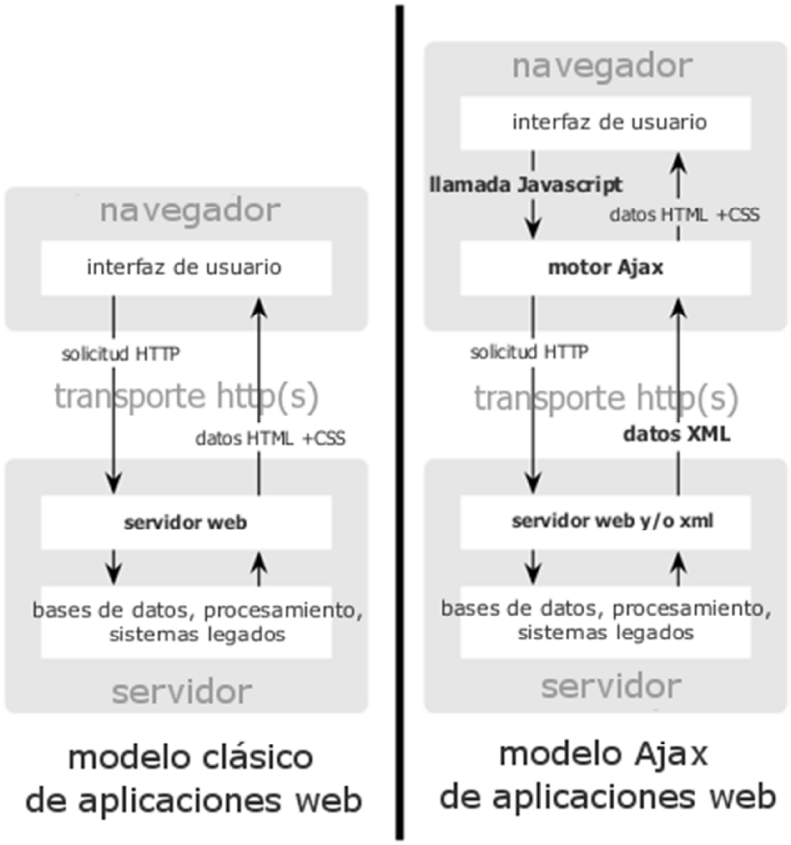


Figura II.12. El modelo tradicional para las aplicaciones Web (izq.) comparado con el modelo de AJAX (der.).

Cada acción de un usuario que normalmente generaría un requerimiento HTTP toma la forma de un llamado JavaScript al motor AJAX. Cualquier respuesta a una acción del usuario que no requiera un viaje de vuelta al servidor es manejado por su cuenta. Si el motor necesita algo del servidor para responder a esos pedidos, asincrónicamente efectúa los pedidos, usualmente usando XML, sin frenar la interacción del usuario con la aplicación.

2.3. MARCO EMPÍRICO

El sistema que se propone en el presente trabajo se enmarcó en el ámbito de la UNSE.

De acuerdo al Estatuto de la UNSE, las facultades son las unidades académicas, administrativas y de gobierno que agrupan, cada una de ellas, Escuelas, Institutos,

Departamentos, Centros u otras subunidades. Las facultades son gestionadas en su nivel superior por un Consejo Directivo, un decano y un vicedecano.

Entre las funciones del Consejo Directivo se encuentran las de hacer cumplir las normas del Estatuto y las que hayan sido establecidas por el Consejo Superior; proyectar planes de estudios y conceder equivalencias; establecer normas reglamentarias sobre docencia e investigación; aprobar los programas de estudio; fijar el calendario de la Facultad y las condiciones de admisibilidad a las aulas; tramitar los concursos de los profesores; entre otras responsabilidades.

El Decano por su parte debe ejercer la representación de la Facultad; convocar al Consejo Directivo a sesión; ejecutar las normas estatutarias y las resoluciones del Consejo Directivo; dirigir, coordinar y supervisar la actividad académica; organizar las secretarias de la Facultad, designar y remover a sus titulares y demás personal no docente; entre otras funciones. El Vicedecano ejerce las funciones que el Decano expresamente le delegue.

Las Escuelas son unidades de administración académica, cuyas funciones son coordinar e integrar las actividades de las distintas unidades docentes que intervienen en el desarrollo de la currícula de una o más carreras afines, y controlar el desempeño académico de los estudiantes.

Los Departamentos son unidades funcionales docentes en los cuales se busca agrupar materias similares o afines y mantener la cooperación científica y de material de enseñanza y de bibliografía entre las cátedras que lo forman. A través de los Departamentos se coordina la enseñanza, se orienta la realización de trabajos de investigación y de seminarios y se organizan cursos de extensión o perfeccionamiento.

Los Institutos son unidades de investigación. Pueden componerse de secciones o laboratorios dedicados a aspectos particulares de su labor. Sus principales tareas en la enseñanza son las de formar investigadores, contribuir a la formación de docentes, dirigir a becarios y dictar cursos de especialización.

Un aspecto a destacar es que en cada una de las Facultades se hayan presentes las siguientes secretarias: Secretaria Académica; Secretaria de Ciencia y Técnica; y la Secretaria de Administración.

La Secretaria Académica tiene como funciones el diseño y desarrollo de alternativas estratégicas para la definición e instrumentación de lineamientos político-académicos relativos a la enseñanza en las carreras de Pre-grado, Grado y Postgrado de la Facultad, como también en la vinculación y articulación con otras instituciones de los distintos niveles educativos tanto públicas como privadas.

La Secretaria de Ciencia y Técnica tiene como propósito contribuir al fortalecimiento de líneas de investigación en diferentes áreas disciplinares y promover la consolidación de nuevos grupos en investigación, principalmente en las áreas de vacancia.

Por último, la Secretaría de Administración tiene la misión y función de brindar apoyo a todas las actividades sustantivas de la Facultad al administrar todo lo relativo a los recursos humanos, financieros y materiales.

Con el sistema desarrollado lo que se busca es proporcionar una herramienta que en principio sirva a cada área para publicar información que pueda ser útil a los demás miembros de la comunidad educativa. El compartir información implica ya sean noticias breves, materiales, enlaces, archivos, datos de perfiles, y en fin cualquier activo digital que se genera como producto de cumplir con las funciones que tiene asignada cada área funcional.

En la figura II.13 se clasifica a las áreas funcionales según contribuyan a la gestión organizacional o la gestión académica. Sólo se muestran las áreas funcionales descritas, las cuales son las más relevantes. Cada una de estas áreas generalmente define secretarias adicionales para poder llevar a cabo sus funciones. El sistema que se propone da soporte fundamentalmente a la Gestión Académica.

Figura II.13. Agrupación de las áreas funcionales según contribuyan a la producción de conocimiento organizacional o académico

CAPÍTULO III
CONSTRUCCIÓN DEL SISTEMA DE GESTIÓN DE CONOCIMIENTO BASADO EN APLICACIONES SOCIALES

3.1. MODELO CONCEPTUAL

El sistema que se propone se basa en la integración de aplicaciones Web 2.0. Las seleccionadas en este caso son: networking, wikis, foros, blogs (Wordpress y Blogger) y aplicaciones de microblogging (Twitter). Las aplicaciones Web 2.0 proveen oportunidades para la eficiente generación de conocimiento, su intercambio, colaboración, aprendizaje y para la toma de decisiones colectivas, debido a su facilidad de uso, portabilidad, desarrollo rápido e implementación sin demasiado consumo de tiempo [24].

La decisión de trabajar con aplicaciones sociales se basa en que la mayoría de las investigaciones sobre gestión del conocimiento consideran que las herramientas de la Web, y muy particularmente de la Web 2.0, son potencialmente adecuadas para mejorar las actividades de gestión del conocimiento. Más aún, desde una perspectiva organizacional, cuando los usuarios colaboran y comparten su trabajo usando estas herramientas, los enfoques subyacentes a la gestión del conocimiento se manifiestan [14].

La elaboración del modelo se basa en la línea de pensamiento de Tiwana [30], quien sostiene que para crear un modelo de gestión del conocimiento, se requiere pensar en términos de una ***infoestructura***, es decir, un entorno informacional a través del cual se facilite la accesibilidad a los flujos de información de una organización.

Si bien el modelo podría ser aplicado en cualquier ámbito, en este caso se lo orienta al ámbito universitario, caracterizado por la alta producción y flujo de conocimientos, y donde existen mayores oportunidades para el desarrollo de tareas

colaborativas que implican la creación y el uso de conocimiento. Concretamente el modelo está diseñado para dar soporte a la gestión de información / conocimiento producto de las tareas de **docencia e investigación**, y para cumplir con este propósito, las aplicaciones están configuradas para almacenar contenidos de esta naturaleza.

Hay dos tipos de conocimientos involucrados en el área de la educación superior según [9]: **conocimiento académico** y **conocimiento organizacional** (específicamente de gestión académica). El conocimiento académico es el elemento vital producido dentro de las universidades y es la clave de su existencia. El conocimiento organizacional está relacionado al conocimiento en general de los "negocios" dentro de una institución, las fortalezas y debilidades de la misma, el mercado al que sirven y los factores críticos que inciden en su éxito. El presente trabajo se enfoca exclusivamente en el conocimiento académico.

Los procesos de conocimiento típicos que son inherentes a la educación superior y que también están implicados en la gestión del conocimiento son: la creación, la captura, el almacenamiento, la diseminación y el compartir el conocimiento. Sin embargo, la tendencia general es que, la mayoría de este proceso es hecho de una manera unidireccional, desde los profesores hacia los alumnos. Por esta razón, los agentes de conocimiento involucrados en el modelo son los docentes y principalmente los alumnos que cumplirán un rol activo en el proceso de gestión del conocimiento.

En el ámbito de la universidad que se tomó como referencia, el sistema que se propone permitirá que estudiantes, egresados, profesores e investigadores puedan tener un acceso ágil a diferentes flujos de información referidos fundamentalmente a lo académico, por ejemplo, horarios de clases, información sobre becas, programas de incentivos docentes, llamados a concursos, contenidos de asignaturas, investigaciones, publicaciones, entre otros. Se aspira a que los productores de estos tipos de contenidos puedan disponer de aplicaciones simples a través de las cuales puedan publicar estos tipos de información. La figura III.1 ilustra lo antes mencionado.

En la figura III.2, mediante un *rich picture*, se representan los distintos aspectos involucrados en el modelo propuesto para dar solución a los problemas de acceso, distribución y esfuerzos replicados durante el desarrollo de las actividades académicas. Se presenta una visión global, en cuanto a integración de aplicaciones sociales que fueron consideradas en el desarrollo de la propuesta.

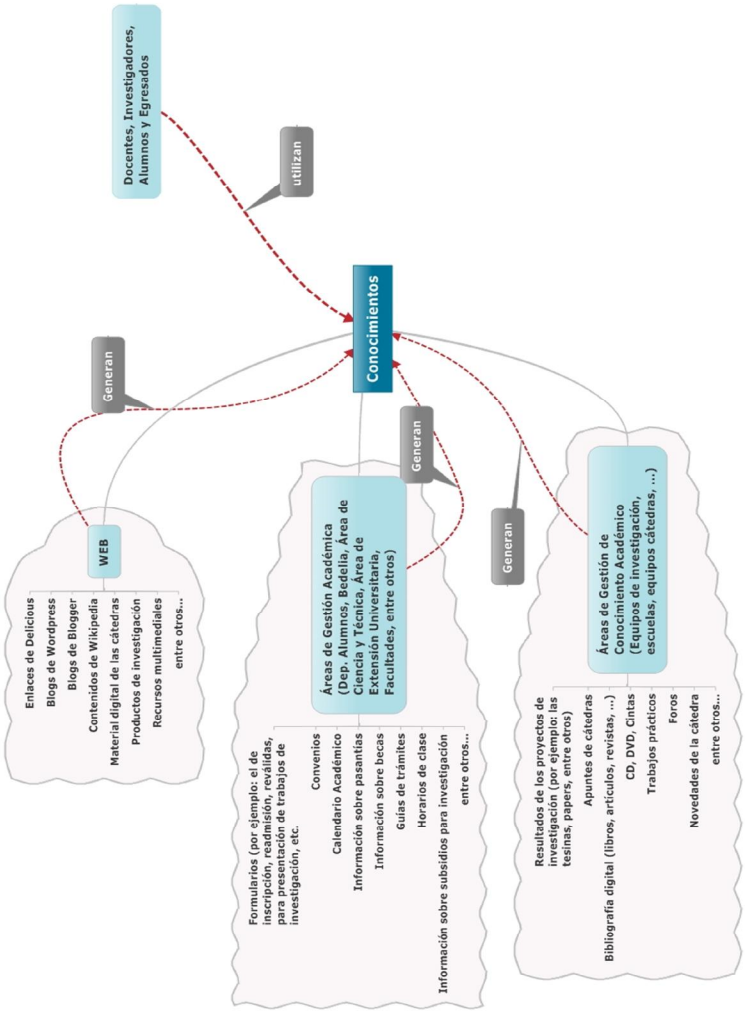


Figura III.1. Esquema de las areas de la UNSE involucradas en el sistema propuesto

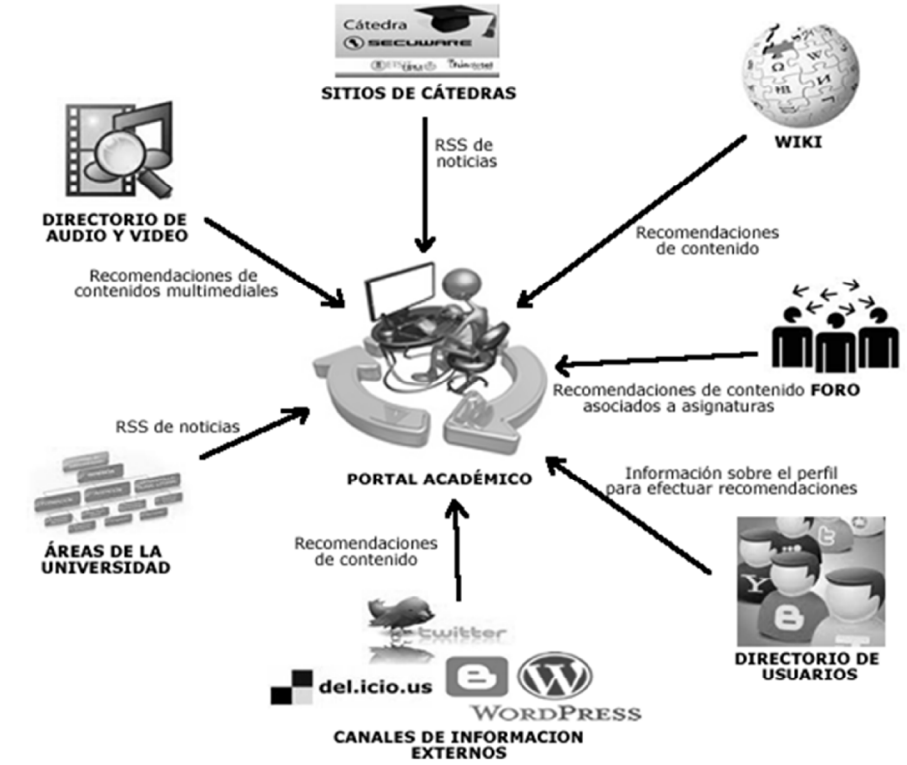


Figura III.2. Rich Picture del modelo integrado propuesto

El modelo propuesto tiende a los siguientes objetivos generales:

- Facilitar el trabajo colaborativo entre pares y el desarrollo de las prácticas académicas,
- Promover la generación colaborativa, la transmisión y el uso de conocimientos científicos y tecnológicos,
- Incrementar la accesibilidad y disponibilidad de los recursos y conocimientos generados a partir de las actividades académicas,
- Disponer de nuevos formatos para la diseminación, adquisición y administración del conocimiento.

Para cumplir con los objetivos, el modelo responde a las características siguientes:

- Interacciones rápidas y fácil acceso a la información y al conocimiento través de la implementación de: canales de información y uso de un directorio de usuarios con características similares a las disponibles en el mercado actual; foros; herramientas de sindicación (RSS); chat; un portal de información; y canales de microblogging. Para obtener un fácil acceso a

la información y al conocimiento generado, se implementaron algunos algoritmos de recomendación que facilitan el enlace a contenidos según los intereses de los usuarios.

- Acceso al conocimiento sobre recursos materiales y recursos humanos, que se logra con la aplicación de un directorio de usuarios realimentado con información sobre los miembros de la institución educativa (por ejemplo intereses, especialidades, experiencias laborales, antecedentes educativos, información para contactarlo, conocer sus aportes en las redes de contenido). Por otro lado, se implementó una wiki, un foro y se diseño una propuesta de sitios para cátedras a través de los cuales se podrá acceder a los productos académicos generados.
- Evita la duplicación de esfuerzos mediante el entorno informacional donde los usuarios tienen facilidades para conocer lo que se tiene disponible y por ende esto evitará a que se vuelva producir algo que ya existe. Con la implementación de este recurso se contribuye al desarrollo de la creatividad, ya que se presupone que los usuarios de la comunidad educativa, al participar en la red intercambiando información, y estableciendo vinculaciones con unidades de conocimiento, implícitamente están influenciados por la misma y activan sus mecanismos creativos para producir nuevos conocimientos o ideas sobre productos/servicios.

3.2. MODELO TECNOLÓGICO

Conforme con la guía propuesta por Amrit Tiwana para el desarrollo de sistemas de gestión del conocimiento, la arquitectura del modelo propuesto se organiza en cinco capas (figura III.3):

- **Capa de interfaz.** Es el punto a través del cual el usuario final interactúa directamente con el sistema para crear, explicar, usar, recuperar y compartir el conocimiento. A través de ésta accede a las distintas aplicaciones disponibles. Esta capa brinda el acceso a los canales de comunicación informales, tales como chat, los cuadros de texto para dejar comentarios, los cuales son importantes para contribuir a la circulación de los conocimientos tácitos.

- **Capa de acceso y autenticación.** Controla los aspectos vinculados a la autenticación (privilegios de acceso), la autorización (verificación de permisos asociados al usuario), seguridad (mediante la configuración de un firewall) y la generación de copias de respaldo (creación de backups y sitios espejos).
- **Capa de filtrado e inteligencia colectiva.** En esta se centra la función de análisis de la información almacenada en las base de datos del sistema, permitiendo, por ejemplo, efectuar recomendaciones de contenido y vinculaciones entre distintas informaciones. Es decir dota de la capacidad de "búsqueda inteligente" a la aplicación. Las herramientas que dan soporte a estas funcionalidades son:
    - Los **mecanismos de recomendación** basados en algoritmos estadísticos (en este trabajo se aplica el coeficiente de correlación de Pearson) que efectúan recomendaciones según las preferencias personales de los usuarios y la calificación que hayan realizado de los contenidos.
    - Procedimiento de **clustering jerárquico** para el análisis de determinadas informaciones existentes en el sistema. Se desarrolló un algoritmo de escalado multidimensional para mostrar los resultados provenientes del análisis de clustering.
    - Medios a través de los cuales se permite que los usuarios puedan "taguear" los elementos de conocimiento (hilos de comentarios, perfil de usuario, contenido del portal, contenido de blogs). Esta funcionalidad es importante para contribuir con la diseminación de contenido, por medio de los algoritmos de recomendación, para facilitar el acceso de contenido que de otra manera pasaría desapercibida y para dar al usuario la posibilidad que organice la información a su manera.
    - Diferentes **mecanismos de búsqueda** de información: por tags, jerárquica y por atributos (como lo hacen la mayoría de los buscadores).
    - **Aplicación de búsqueda inteligente** en el sentido de que va aprendiendo de los usuarios en la selección de los resultados de la búsqueda. Para la propuesta se diseñó una red neuronal de tres capas unidireccional (no presenta ciclos) con un modelo de aprendizaje supervisado.

- El **acceso a las fuentes de información** se realiza a través del método *push* (entrega de contenido sin necesidad de que el usuario la busque por técnicas de sindicación y algoritmos de recomendación), y *pull* (se busca explícitamente el contenido que se necesita con herramientas de búsqueda y análisis de clustering).

- **Capa de aplicación.** Proporciona las distintas aplicaciones a las cuales el usuario final puede tener acceso y con las cuales interactuará a través del uso de un explorador web. Entre las aplicaciones incluidas se tiene a las wikis, blogs, un sistema de directorio de usuarios, chat, mecanismos de tagueo, un portal web que incluye componentes (widget) de información, sitios para desplegar información de cátedras y aplicaciones web que sirvan exclusivamente como repositorios de contenidos multimedia (audio y videos). La mayoría de las aplicaciones en esta capa tienen incorporada las funcionalidades de poder dejar comentarios, calificar y recomendar el contenido.
- **Capa de datos.** Define los repositorios usados por el sistema. Éstos, mantienen los conocimientos informales y los explicitados formalmente. Mayormente, los contenidos de estos repositorios son actualizados por sus mismos usuarios y no por un único "administrador".

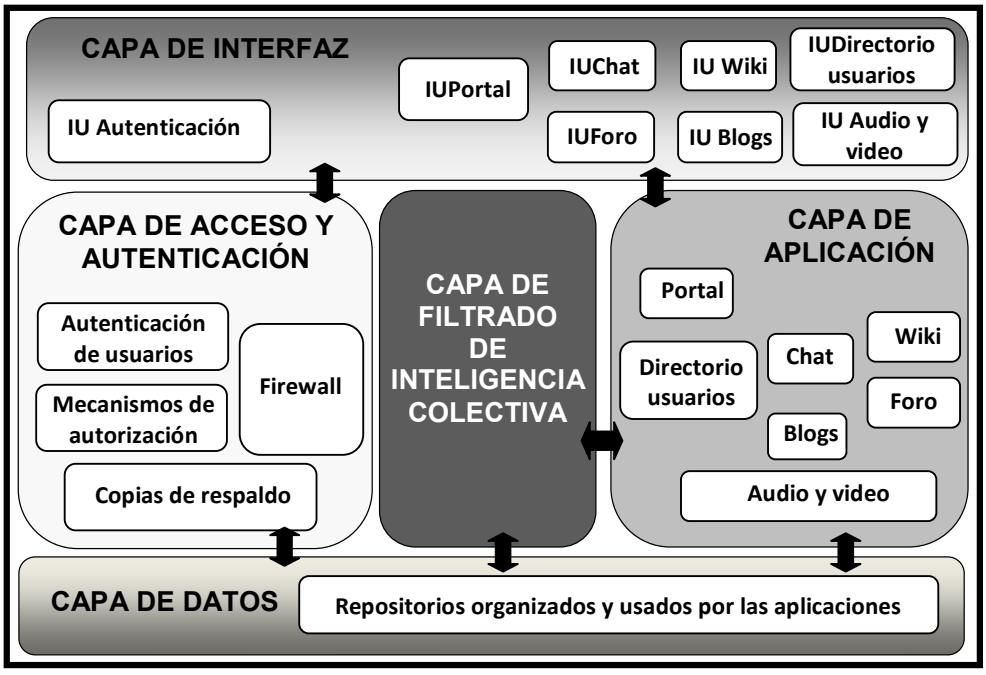


Figura III.3. Modelo de capas del sistema

3.3. DESCRIPCION DE LAS CAPAS

En esta sección se describen cada una de las capas del sistema propuesto. Se hace énfasis en la *capa de aplicación* y en la *capa de filtrado e inteligencia colectiva*, ya que las mismas han sido desarrolladas completamente para este modelo en base al proceso de investigación realizado.

En la figura III.4 se representa mediante un diagrama de clases las interacciones entre las distintas aplicaciones que forman parte de la capa de aplicación. Estas aplicaciones son descriptas en la siguiente sección.

3.3.1. CAPA DE APLICACIÓN

En este apartado se describen las aplicaciones que forman parte de la **capa de aplicación**, mostrando los detalles de diseño efectuados para aquellas que se deben desarrollar en su totalidad, a saber:

a. Portal académico,
b. Directorio de usuarios,
c. Directorio de audio y video, y
d. Sistema para cátedras.

Además se explica cómo se integraron las aplicaciones de terceros:

e. Wiki,
f. Foro,
g. Blogs,
h. Twitter, y
i. Chat.

Los prototipos de las aplicaciones desarrolladas (el portal y el directorio de usuarios) fueron programados en *PHP*, usándose *MySQL* como gestor de base de datos. Tanto *PHP* como *MySQL* son aplicaciones de naturaleza Open Source.

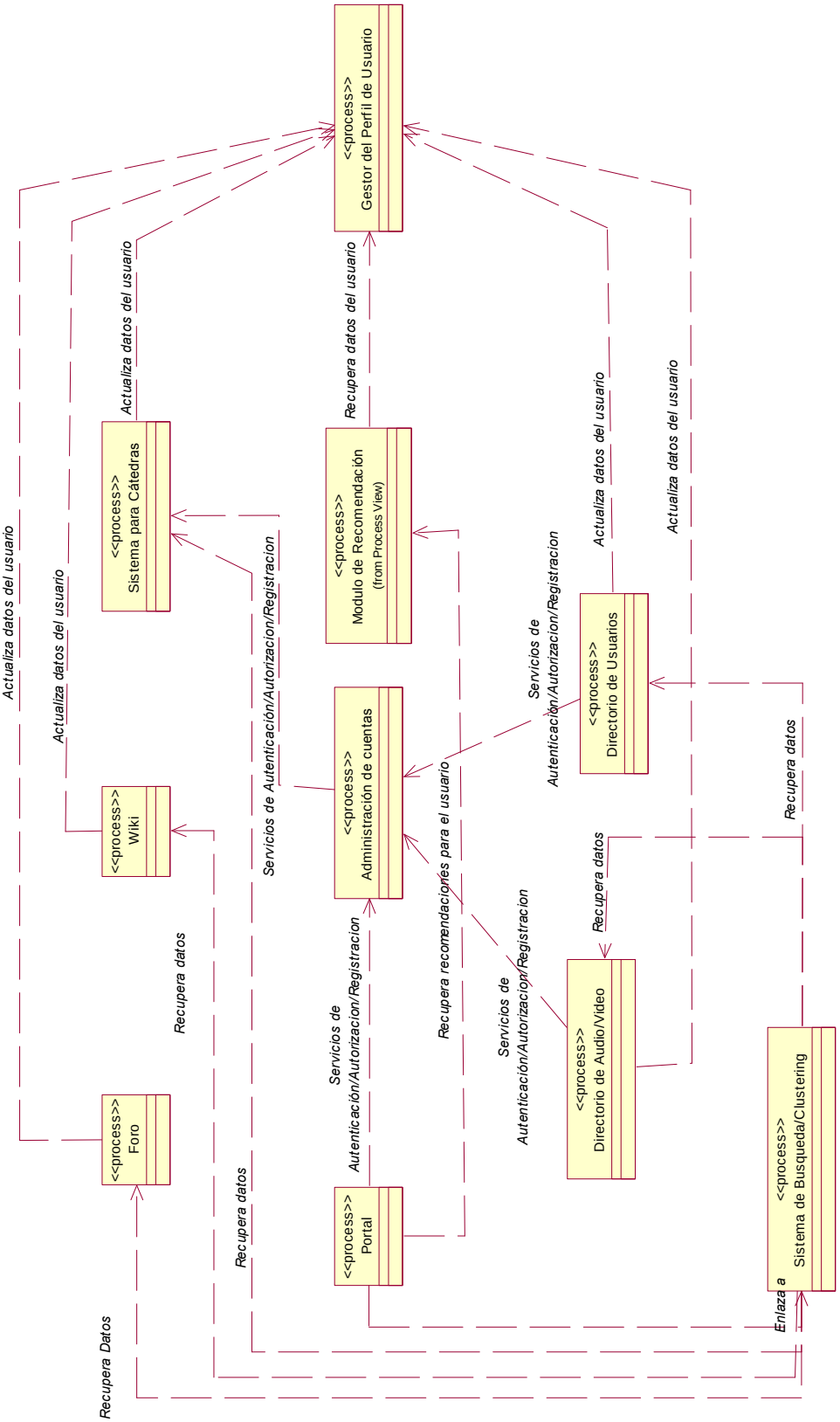


Figura III.4. Diagrama general de las aplicaciones que forman parte de la Capa de Aplicación

a. Portal académico

Esta aplicación concentra la información/conocimiento de interés para los usuarios, la cual estará referida a áreas temáticas facilitándole la búsqueda y minimizando el esfuerzo de memorizar urls de sitios web. Una de las características del portal desarrollado es el de poder ser personalizado (no adaptativo), al permitir al usuario seleccionar qué visualizar en base a sus necesidades y preferencias.

Otro aspecto a destacar, es el hecho de que el portal está orientado a "widgets" lo cual facilita que la aplicación pueda continuar escalando en cuanto a componentes de información, puesto que en caso de necesitarse se procede al desarrollo de un nuevo widget que proporcione la funcionalidad requerida.

En este prototipo se permite que el portal pueda recibir información de las distintas áreas académicas, del directorio de usuario, del directorio de audio y video, de la wiki y del foro a través de RSS, desarrollando previamente widgets que se encargarán de consumir los mismos, interpretarlos y mostrarlos. En la figura III.5 se muestra un prototipo de interfaz de la pestaña "General" del portal, en la cual aparecen distintos widgets cuya información es consumida desde RSS proporcionados por otras fuentes generadoras de contenido (en la imagen, un widget es una ventana en miniatura que aparece embebida en la página). Los widgets mostrados en la figura son sólo a modo ilustrativo.

En la figura III.6 se muestra una pestaña que al activarla despliega widgets que están orientados hacia aspectos académicos. La información que se muestra en estos widgets es recuperada desde una base de datos que es cargada por las diversas áreas de la Universidad a través de una aplicación (*http://betatesting.com.ar/noticias*) que les permite almacenar contenidos/novedades útiles para la comunidad universitaria: como por ejemplo información sobre pasantías, llamados a concursos, fechas de exámenes, horarios de clases, entre otros.

El uso de widgets permite mantener la arquitectura central del portal "limpia y simple", en el sentido de que se mantienen de manera ordenada y estructurada las líneas de código requeridas para ofrecer la funcionalidad principal, facilitando la mantenibilidad y extensibilidad de los servicios que ofrece [2].

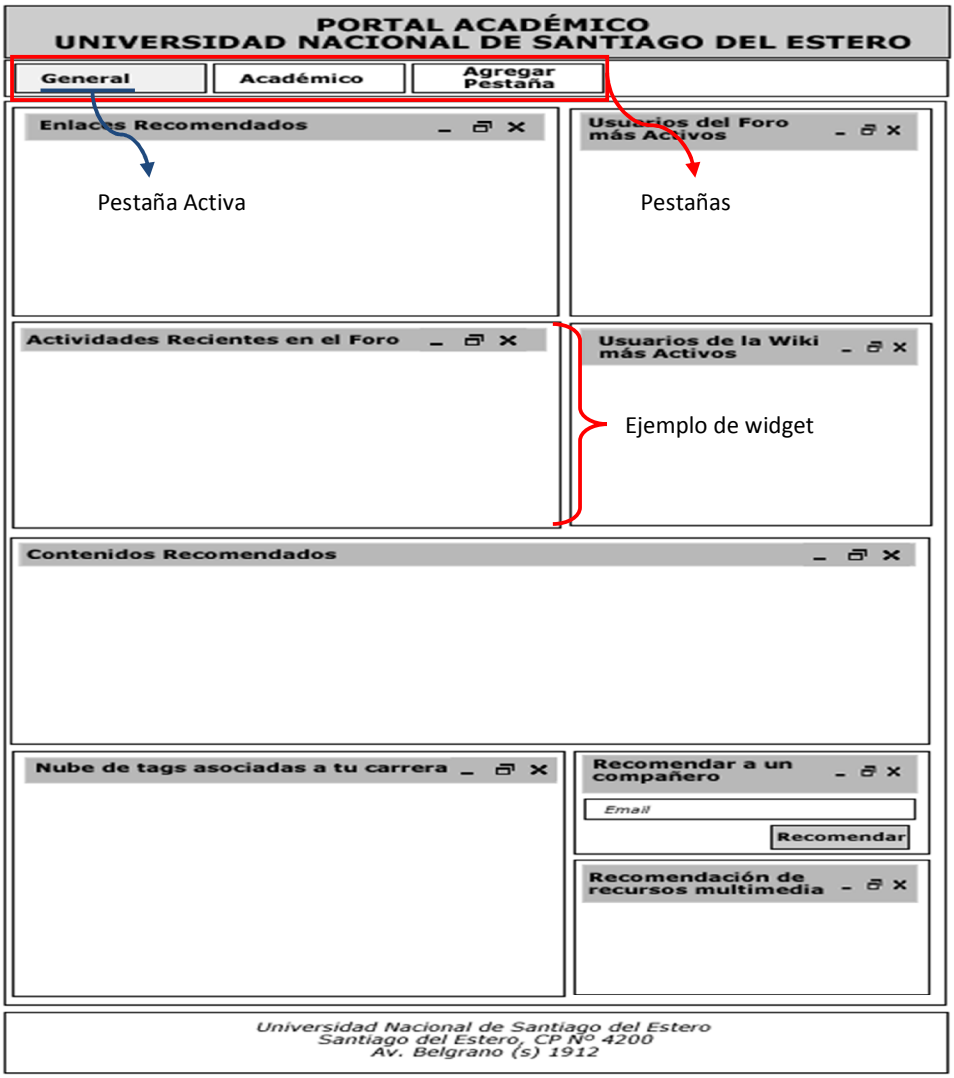


Figura III.5. Prototipo de la interfaz de página principal del portal

De esta manera, con los puntos tratados hasta el momento podemos definir al portal desarrollado como una aplicación web que permite al usuario personalizar las páginas que lo integran, representadas por pestañas en la aplicación, al arrastrar y soltar (*drag-and-drop*, en inglés) widgets sobre ellas. Esto permite que el usuario tenga el control completo de los contenidos que desee visualizar en las páginas, dónde desea visualizarlos, y cómo necesita interactuar con ellos.

El portal desarrollado está potenciado por AJAX lo cual proporciona al usuario una mejor experiencia, al tener menos "refrescos de página" y comportarse de manera similar a una aplicación de escritorio en cuanto a rápidez, y la sensación de estar trabajando localmente y no de forma remota.

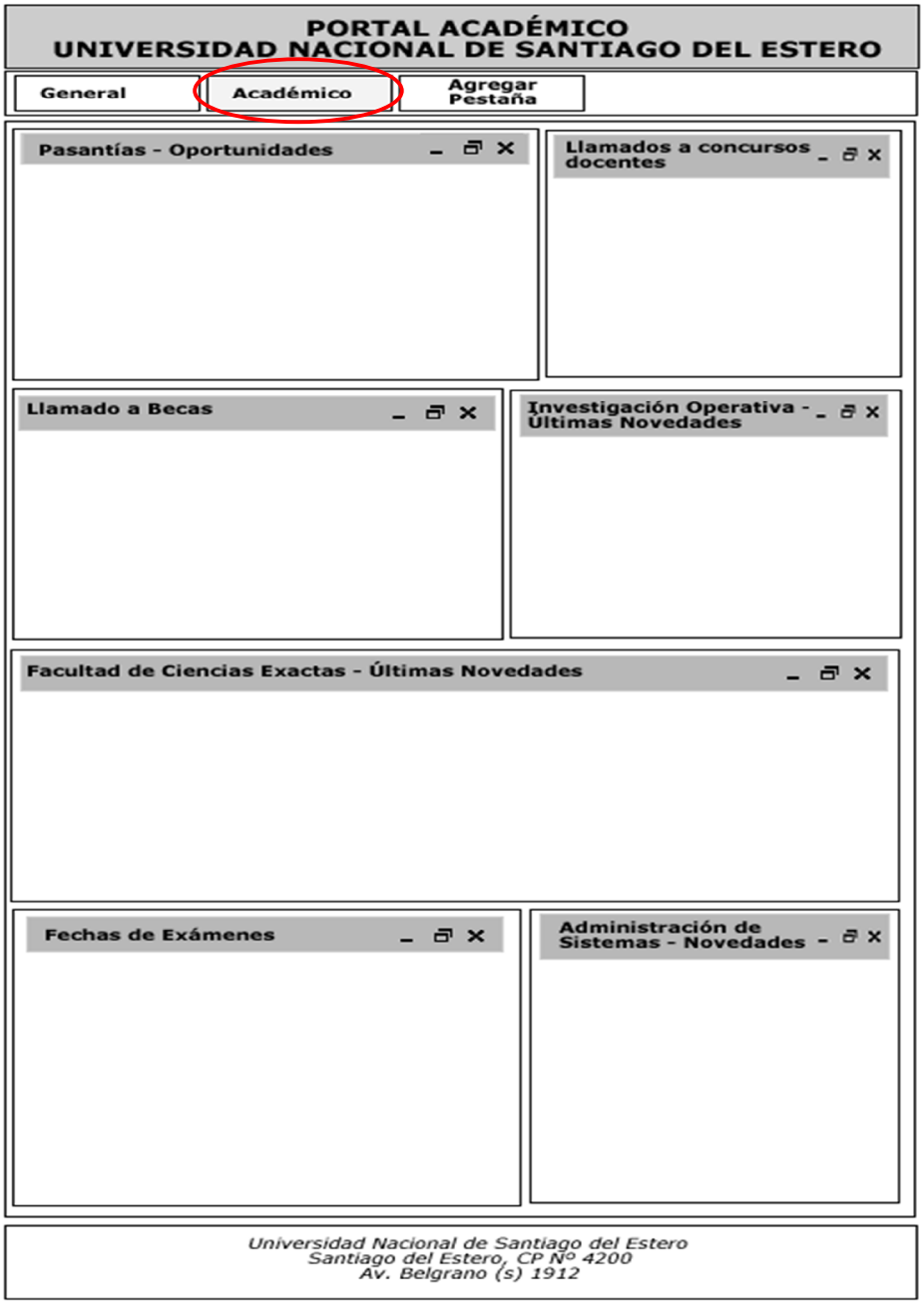


Figura III.6. Prototipo de interfaz de una pestaña del portal orientada hacia aspectos académicos

Por último, en el portal se define una pestaña fija en la cual se ofrecen funcionalidades de búsqueda inteligente que a través de su uso social va "entrenando" a una red neuronal con el objeto de proporcionar mejores resultados. Para la búsqueda inteligente se aplica un "motor de búsqueda" que reside en la capa de "Filtrado e Inteligencia Colectiva"; mayores detalles sobre su desarrollo se presentan más adelante

en la correspondiente sección. En la figura III.7 se muestra el prototipo de interfaz de la página que se despliega al activar la pestaña de búsqueda.

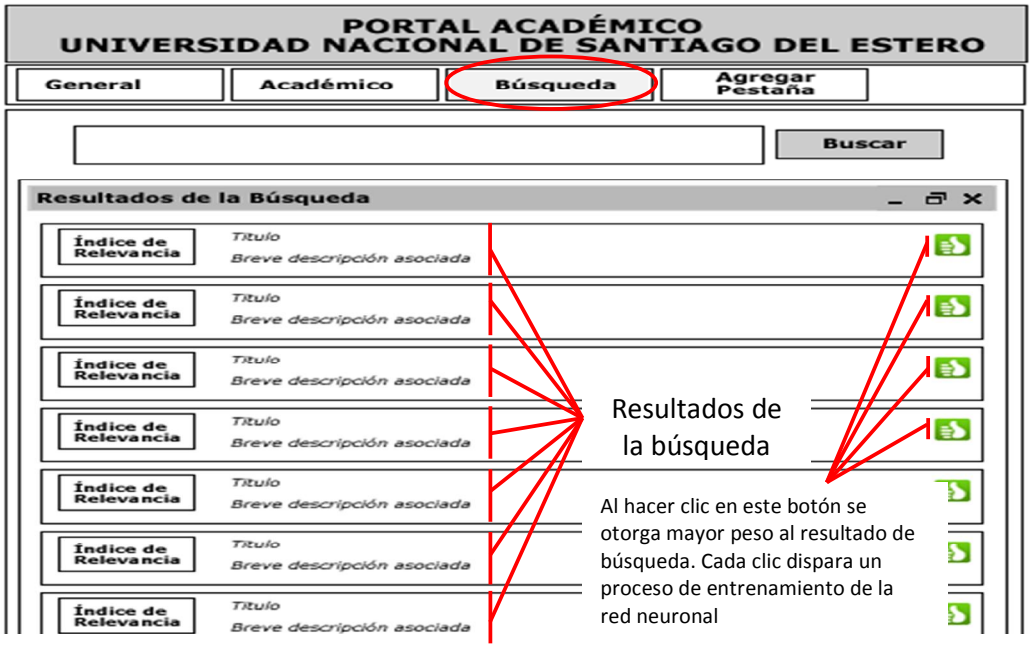


Figura III.7. Prototipo de interfaz para la pestaña que ofrece las prestaciones de búsqueda inteligente

- ***Arquitectura del Portal***

El corazón del portal es el soporte para widgets, mecanismo a través del cual los usuarios pueden "personalizar" sus páginas y el medio del que dispondrán las áreas de la Universidad para difundir sus novedades e información en general al cargar previamente bases de datos que son accedidas por estos widgets.

El archivo *Index.php* es la página de inicio que muestra los widgets y permite agregarlos, removerlos, moverlos y personalizarlos dentro de la página sin causar el refresco de la misma (postback).

Para los usuarios registrados, la aplicación "recuerda" las personalizaciones efectuadas por los mismos de modo que la siguiente vez que accedan al sitio encuentren la interfaz según la última personalización realizada. Para proporcionar esta funcionalidad y dar al mismo tiempo una buena experiencia de usuario, se usó AJAX para transmitir de manera asincrónica las personalizaciones que el usuario va efectuando a fin de ir registrándolas en el servidor.

A nivel portal, un widget es alojado dentro de un marco o contenedor, figura III.8. El contenedor provee la barra de cabecera, con un título; un botón para minimizar y un

botón para cerrar. El widget en sí mismo se carga debajo de la barra de cabecera, dentro del área proporcionada por el contenedor.

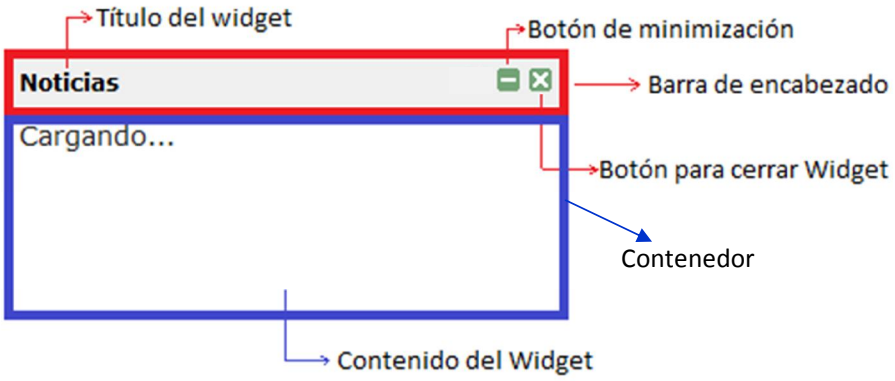


Figura III.8. Partes que componen un Widget

Para poder implementar el comportamiento *"drag-and-drop"* entre múltiples columnas, se dividió la página en tres columnas donde cada una de ellas es un panel en el cual se pueden agregar widgets. En la funcionalidad del *"drag-and-drop"* se utilizó la librería *JQuery* y se contemplaron dos tipos de comportamiento: reordenamiento de widgets sobre la misma columna y movimiento de widgets de una columna a otra. Cuando un widget comienza a ser movido, se efectúan señales de movimiento para indicar dónde el widget puede ser colocado.

- ***Modelo de datos del "Portal"***

En la figura III.9 se muestra el diagrama de entidad-relación en el cual se presentan las entidades de datos relevantes para la aplicación Portal. La página principal del portal, *Index.php*, permite que el usuario pueda definir varias subsecciones, las cuales son representadas por pestañas y que en el modelo de entidad-relación queda representado por la tabla *Tabs*. Cada pestaña (*"tab"*) a su vez puede contener una o más widgets.

Algunos detalles importantes sobre las tablas mostradas, se mencionan a continuación:

- Las tablas *Usuarios* y *CuentasUsuarios* son definidas en la aplicación de "Directorio de Usuarios", por lo cual mayor detalle sobre las mismas serán comentadas en la sección correspondiente a esta última aplicación. A nivel portal, estas tablas están relacionadas con el proceso de autenticación del usuario.

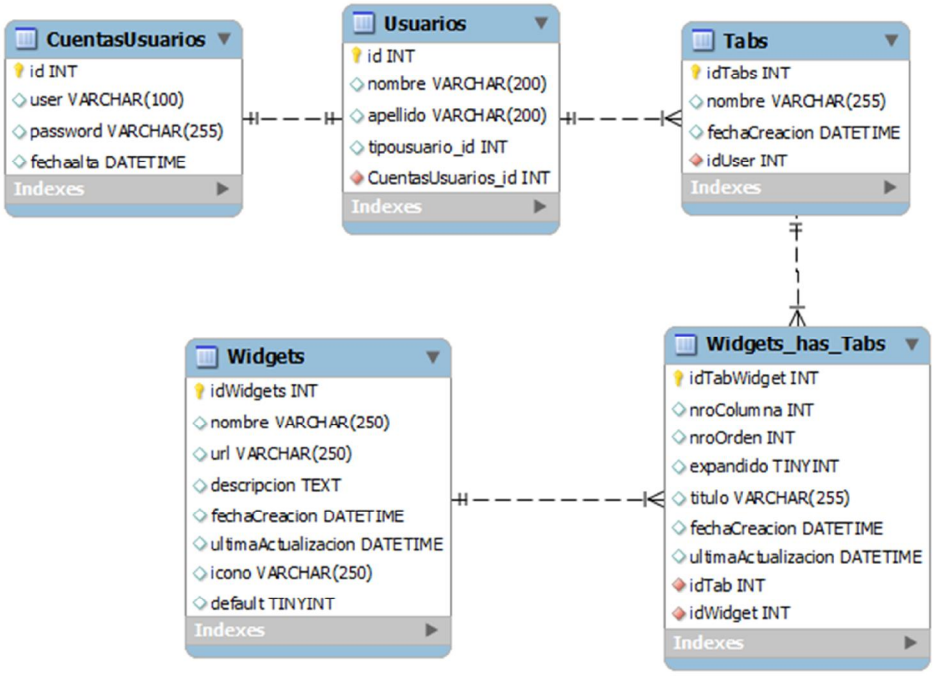


Figura III.9. Modelo de entidad relación definido para el Portal

- La tabla *Tabs* permite almacenar datos sobre las pestañas definidas por un usuario en el portal. Tiene una referencia de clave foránea sobre la columna "idUser" de la tabla *Usuarios* para identificar quién fue el usuario que creó dicho *tab* o pestaña.
- La tabla *Widgets* contiene un inventario de los widgets disponibles en el portal. Almacena el nombre de cada widget, la ubicación (url) desde la cual el widget tiene que ser cargado dinámicamente, su fecha de creación, última fecha de actualización, el ícono vinculado al widget y los widgets por defecto que deben ser creados para los usuarios que visitan por primera vez el portal.
- La tabla *Widgets_has_Tabs* tiene las referencias de clave foráneas sobre las columnas "idTabs" e "idWidgets" de las tablas *Tabs* y *Widgets* respectivamente. Esta tabla almacena la personalización efectuada por el usuario, es decir, qué widgets tendrá una pestaña, su ubicación (en la columna y en qué orden dentro de la misma), el título que tendrá y si aparecerá minimizado o no.

b. Directorio de usuarios

A través de este subsistema se busca dar soporte a la difusión de los recursos humanos existentes en la organización universitaria, a la vez de servir como una herramienta que permita a los miembros de esta institución estar comunicados y poder llevar a cabo procesos de colaboración. Esta aplicación permite a los integrantes de la Institución conocer los demás miembros que forman parte de la organización y llegar a conocer quiénes disponen de conocimiento sobre un tema en particular.

Desde un punto de vista funcional los usuarios del directorio podrán poner a disposición de la comunidad universitaria información relacionada a sus experiencias laborales, experiencias educativas, intereses, especialidades y también agregar una foto de perfil (figura III.10). Esta aplicación permite consumir información sobre sus posteos que realice a través de *Twitter* (se encuentra en la sección "Últimas actividades en Twitter"). El usuario puede controlar qué posteos podrá extraer a través del uso de *hashtags* que los definirá a través de las opciones de configuración de su perfil. Además permite extraer y compartir los enlaces que el usuario haya guardado en su cuenta de *Delicious* sirviendo como base para los algoritmos de recomendación que se emplean en la "capa de filtrado e inteligencia colectiva".

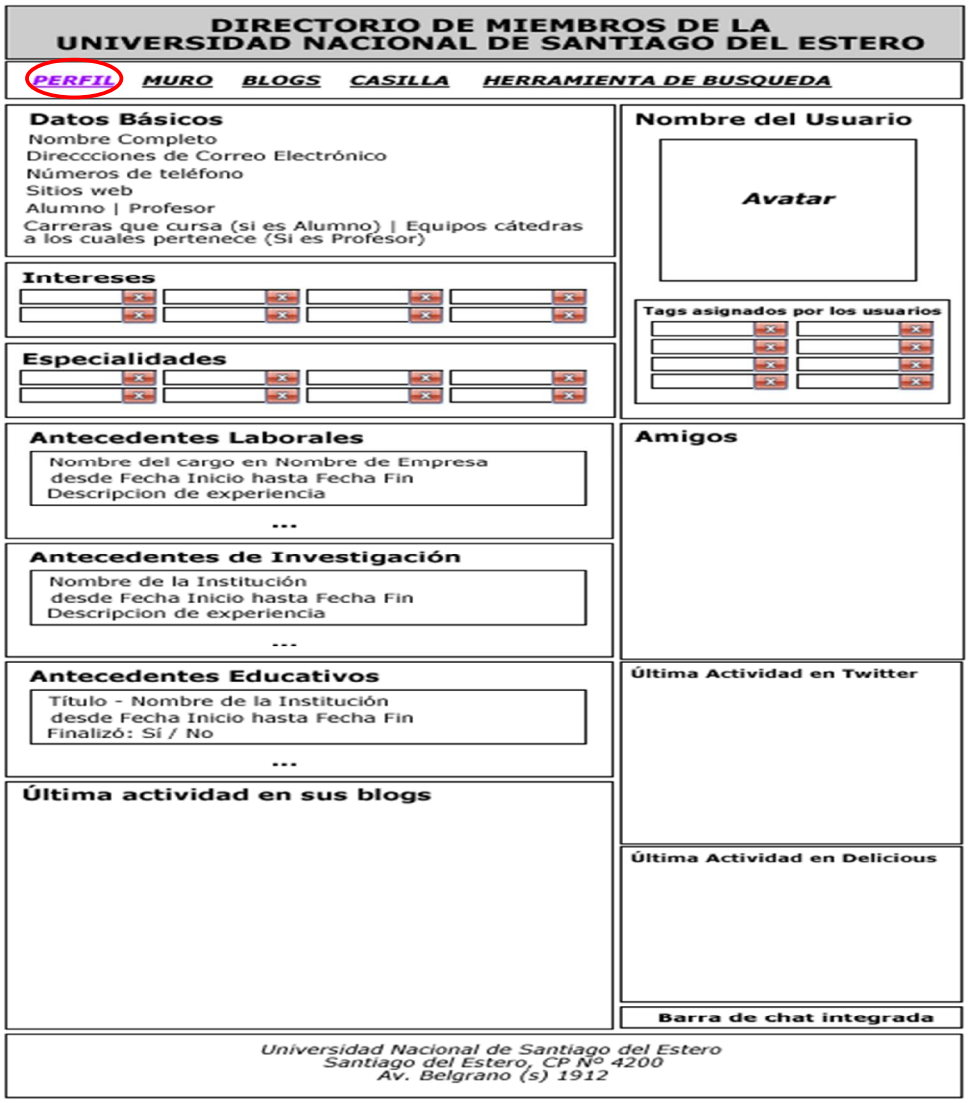


Figura III.10. Boceto de la interfaz de la página Perfil del Directorio de Usuarios.

El directorio dispone de una sección llamada *Muro* a través de la cual el usuario puede compartir información y mantenerse informado de lo que puedan compartir otros usuarios. En la figura III.11 se muestra activa la pestaña Muro con su página asociada. Un aspecto a destacar es que los usuarios pueden dejar comentarios sobre los posteos efectuados en los muros, manifestar si les resulto de interés el aporte efectuado y taguear los contenidos para facilitar la accesibilidad de los mismos en el futuro. Estas funcionalidades también sirven en la capa de filtrado e inteligencia colectiva, particularmente para el subsistema de recomendaciones.

Con respecto a las opciones de tagueo, los perfiles de usuarios también pueden ser clasificados lo cual facilitará el "descubrimiento" de usuarios en el sistema.

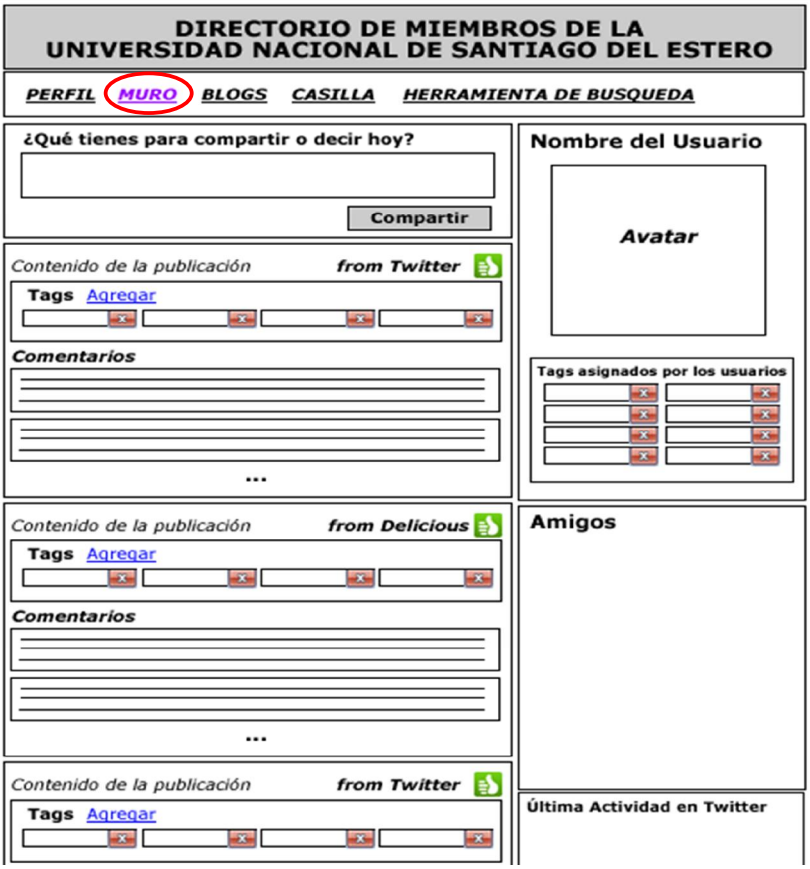


Figura III.11. Boceto de la interfaz de la página Muro del Directorio de Usuarios

Además de la posibilidad de vincular la cuenta del usuario del directorio con las cuentas de *Twitter* y *Delicious*, también se permite vincularla con la cuenta que pudiera tener en *Wordpress* y *Blogger,* con lo cual se da la posibilidad de extraer sus últimos posteos que haya realizado en estas aplicaciones de blogs, permitiendo de esta manera difundir su actividad y compartir información con el resto de la comunidad de usuarios. En la figura III.12 se muestra la pestaña de la página asociada a esta funcionalidad.

En esta aplicación se proporciona también una casilla de correo (mensajes de correos entrantes y salientes, y posibilidad de definir grupos) y el chat para que el usuario pueda enviar mensajes instantáneos a otros que estén conectados, en la figura III.13 se muestra activa la pestaña Casilla y el contenido que despliega.

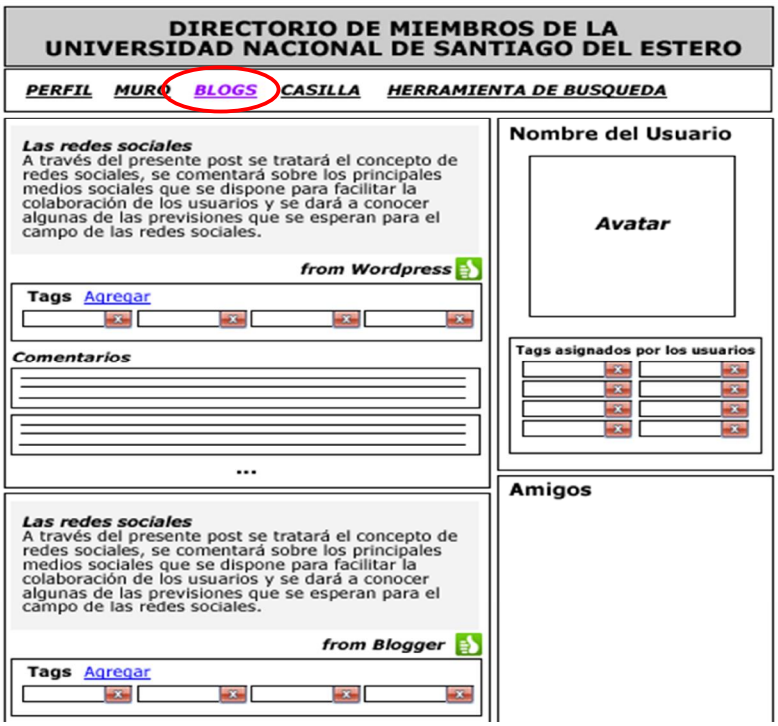


Figura III.12. Boceto de la interfaz de la página Blogs del Directorio de Usuarios

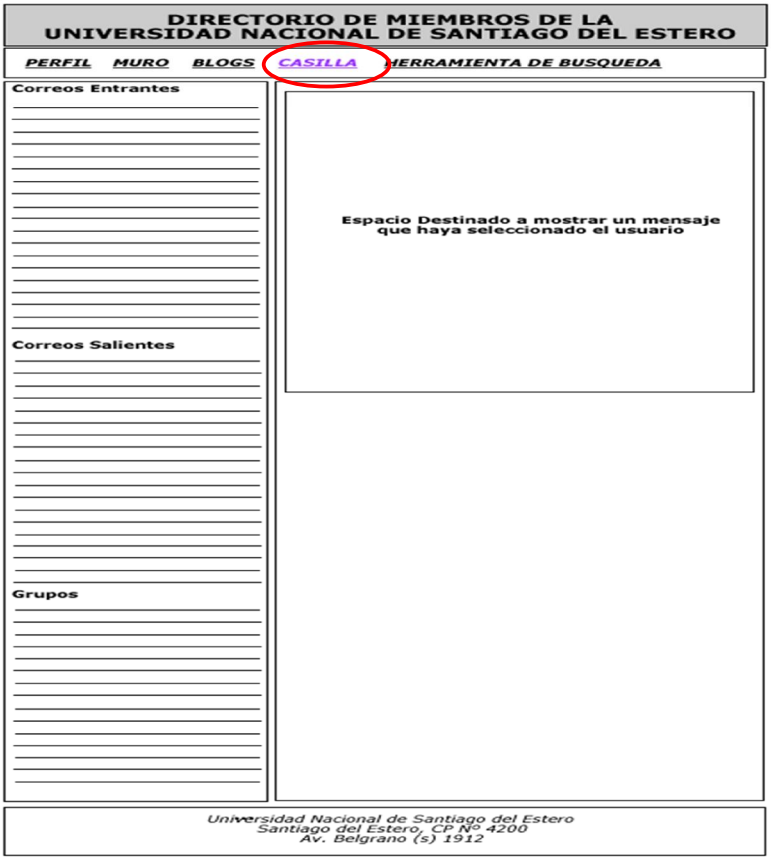


Figura III.13. Boceto de la interfaz de la página Casilla del Directorio de Usuarios

Por último, se proporcionan facilidades de búsqueda de perfiles por diferentes criterios: por facultad, por carrera, por tipo de usuario, por alfabeto, los cuales pueden ser aplicados en combinación y aplicárseles filtros adicionales sobre los resultados arrojados: por intereses, especialidades o aplicación de ambos de manera simultánea. En la figura III.14 se muestra un boceto de interfaz asociada a esta característica.

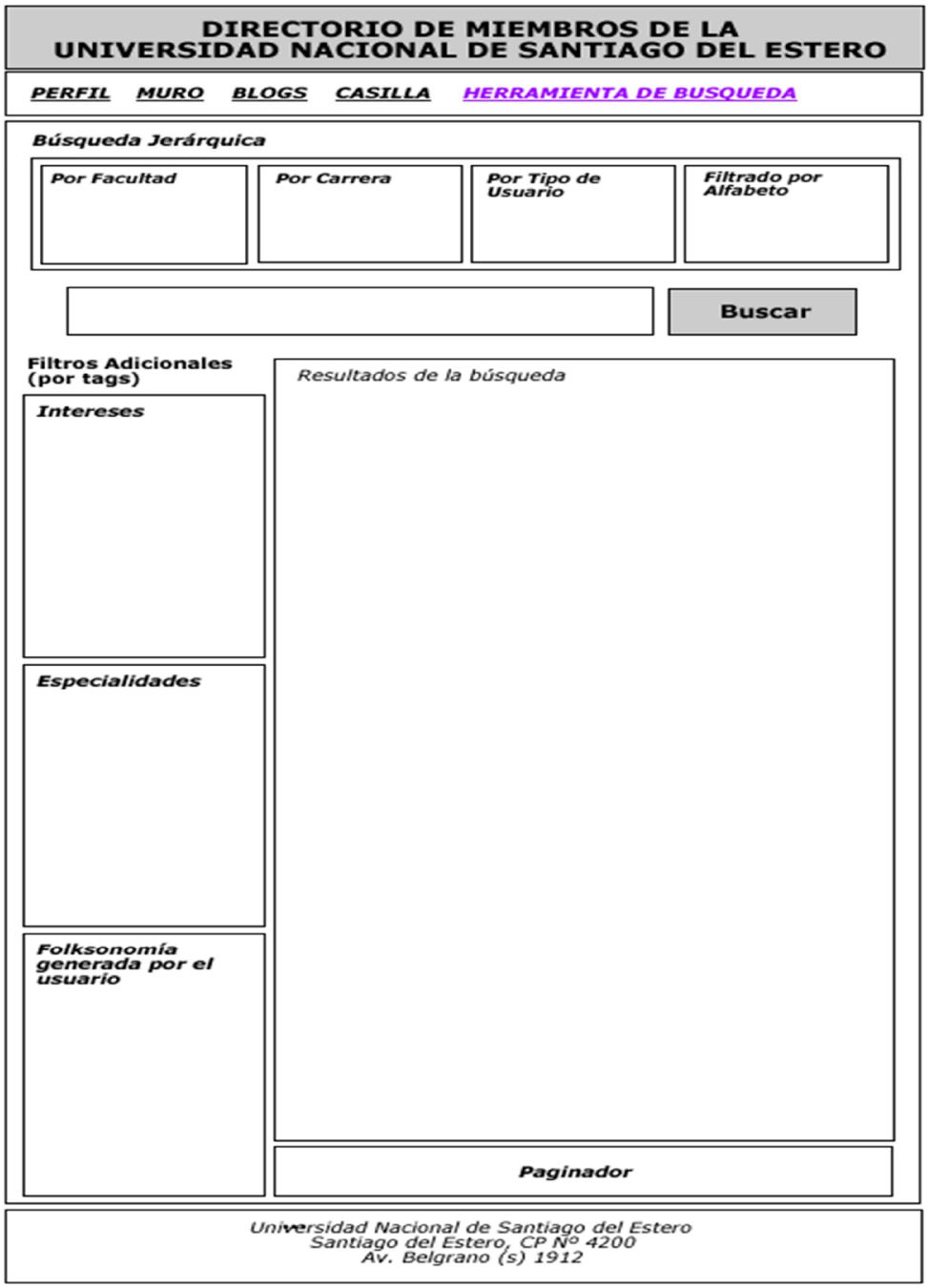


Figura III.14. Boceto de la interfaz de la página Búsqueda del Directorio de Usuarios

Una característica de esta aplicación radica en la posibilidad de extraer información del usuario de una manera "no invasiva" a fin de poder efectuar recomendaciones de contenido al mismo. Con "no invasiva" se hace referencia al hecho de que para otorgar esta funcionalidad no se interfiere en la actividad que el usuario esté efectuando sobre la aplicación para efectuarle preguntas o pidiendo que realice una configuración en particular. El análisis de la información que se colecta del usuario se lleva a cabo en segundo plano de manera ubicua para el mismo.

- ***Modelo de datos del "Directorio de Usuarios"***

En la figura III.15 se representa que un usuario (*usuarios*) dispone de una única cuenta de usuario y pertenece a una categoría (*tipousuarios*) que puede ser alumno, profesor o investigador. En este modelo, cada usuario puede pertenecer a una o más carreras de la Universidad, y las carreras pertenecen a una determinada Facultad. Estos datos se almacenan en las tablas *carreras_usuarios*, *carreras* y *facultades*, respectivamente (figura III.16).

Para cada usuario se almacena información relacionada a sus intereses, especialidades, direcciones de mails, números de teléfono, sitios web, antecedentes laborales, antecedentes educativos y antecedentes en investigación. En relación a los antecedentes, se registra el período del mismo (año de inicio y fin), el lugar en donde hizo dicha experiencia y una breve descripción de la misma. También para cada usuario se almacena el catálogo de contactos a fin de facilitarle un mejor seguimiento de su red de interés.

Para dar soporte a la funcionalidad de poder enviar mails a través del directorio de usuarios, se tomó como iniciativa un enfoque orientado a base de datos en lugar de utilizar los protocolos tradicionales de POP3 y SMTP, por el hecho de que se buscó desarrollar algo de manera rápida, sencilla y fácil de implementar en la etapa de despliegue. Por esta razón se crean las tablas *emails_usuarios* e *emails* de la figura III.15 y las tablas *miembros_grupo* y *grupos_correo* (aparecen en figura III.16). En la tabla *emails* se almacenan todos los mensajes de correo enviados por los usuarios a través de la aplicación; la tabla *emails_usuarios* permite relacionar cada mensaje de la tabla *emails* con sus correspondientes destinatarios y emisores; de esta forma se controla que no cualquiera pueda acceder a un mensaje que no le corresponda. La aplicación para enviar correos también permite armar grupos a fin de poder enviar

mensajes a un mismo grupo de personas, para esto son utilizadas las tablas *miembros_grupos* y *grupos_correo*. Por otro lado, dado que todos los mensajes residen en la tabla *emails*, se implementa un mecanismo para encriptar la información almacenada en dicha tabla.

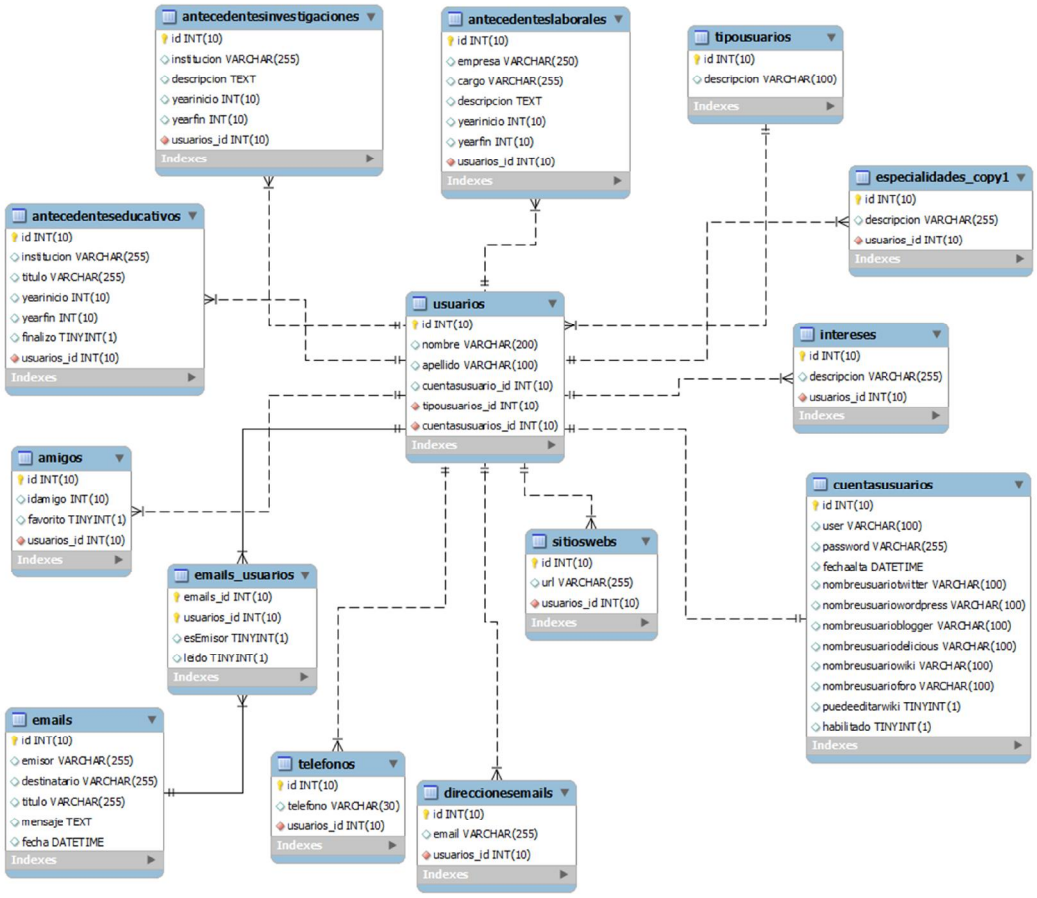


Figura III.15. Diagrama de entidad relación correspondiente al directorio de usuarios. Parte 1

Los usuarios pueden publicar contenidos en el "Muro" y en la sección "Blogs" del directorio de usuarios. Los contenidos que pueden mostrarse en el "Muro" pueden ser mensajes cortos que el usuario publique desde el mismo directorio de usuarios o bien contenidos extraídos desde su cuenta de Twitter y Delicious.

En la sección de "Blogs" se muestran contenidos que los usuarios hayan publicado en sus blogs (para este prototipo, se consumen contenidos desde Blogger y Wordpress). Toda esta información que se genera se almacena en la tabla *Contenidos*, figura III.16. El tipo de contenido de un determinado post se almacena en la tabla *tipoprocedencias*.

Los usuarios pueden manifestar su preferencia con respecto a los contenidos de otros usuarios o de sus propios contenidos. Para hacerlo pueden indicar si el contenido que leyeron les gusto, o bien votando al asignarle un valor numérico (a través de estrellitas). Estos datos se almacenan en la tabla de *Preferencias*, figura III.16. Asimismo, cada contenido publicado puede ser comentado por los usuarios del directorio.

Otro aspecto considerado en la aplicación es la posibilidad de taguear tanto los contenidos como los perfiles de usuarios, información que es almacenada en las tablas *tags_contenidos* y *tags_usuarios*, respectivamente. Ambas tablas se muestran en la figura III.16. La descripción del tag es almacenada en la tabla "tags", la cual se vincula con las dos tablas anteriores para evitar la duplicación de este campo. Esta información generada sirve como materia prima para los algoritmos de recomendación.

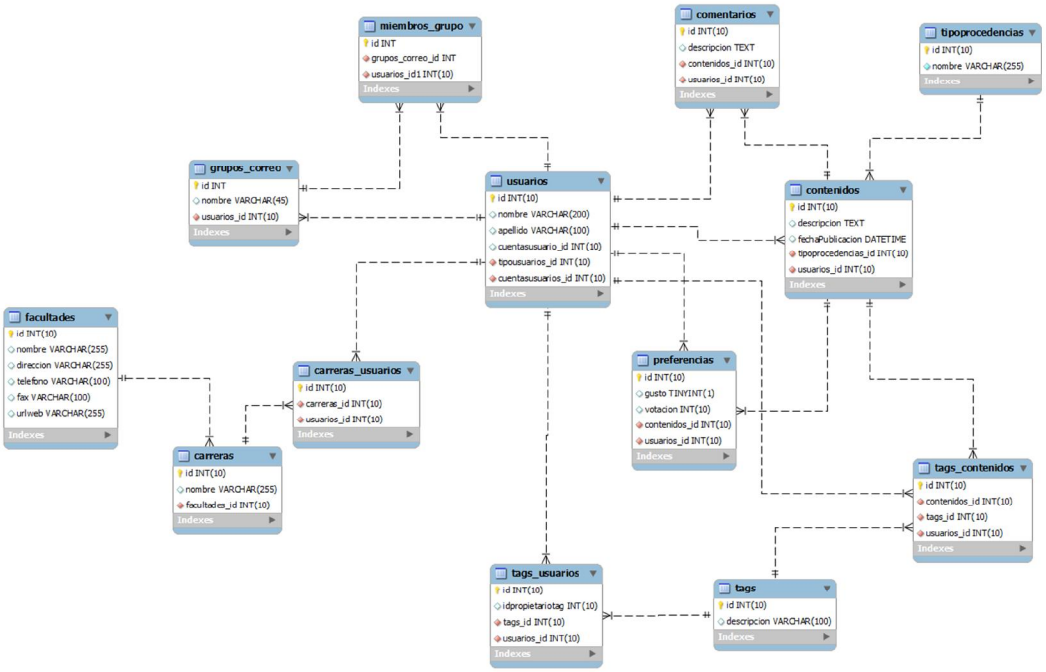


Figura III.16. Diagrama de entidad relación correspondiente al directorio de usuarios. Parte 2

A través de las tablas mostradas en el diagrama de entidad-relación de la figura III.17 se definen los almacenes en los cuales se guarda información resumida de la actividad del usuario en el sistema. Para cada usuario se almacenan los tags comúnmente usados por el mismo para clasificar personas y su frecuencia (*tagspersonas*); para clasificar contenidos y su frecuencia (*tagscontenidos*); clasificar contenidos del directorio de audio y video y su frecuencia (*tagsdirectorioaudiovideos*);

los tópicos del foro en los cuales participa y su frecuencia (*topicosparticipaforos*); y las categorías de la wiki en las cuales participa y su frecuencia (*categoriasintereswikis*). Esta actividad se efectúa para reducir la carga de procesamiento de los algoritmos de recomendación puesto que al ejecutarse ya tendrían disponible la información de base. Estas tablas son pobladas a través de aplicaciones simples que se ejecutan en modo batch cuando el administrador del sistema lo decida; se recomienda sea efectuado cuando la carga del sistema sea baja puesto que estos procesos en batch consumen una cantidad considerable de procesador.

Por otro lado, a los usuarios se les da la posibilidad de configurar los posts que la aplicación puede consumir desde su *Twitter*. Para esto, el usuario debe definir los *hashstags* a fin de que la aplicación solo consuma posts que presenten dichas marcas. Estos datos sobre configuraciones de la cuenta *Twitter* son almacenados en la tabla *configuracionredessociales* mostrada en la figura III.17.

La tabla *usuariosinteresessimilares* registra los resultados de los algoritmos de recomendación referido a la similitud de usuarios con respecto a uno en particular. Esto permitirá que no se tenga que correr frecuentemente las aplicaciones en batch.

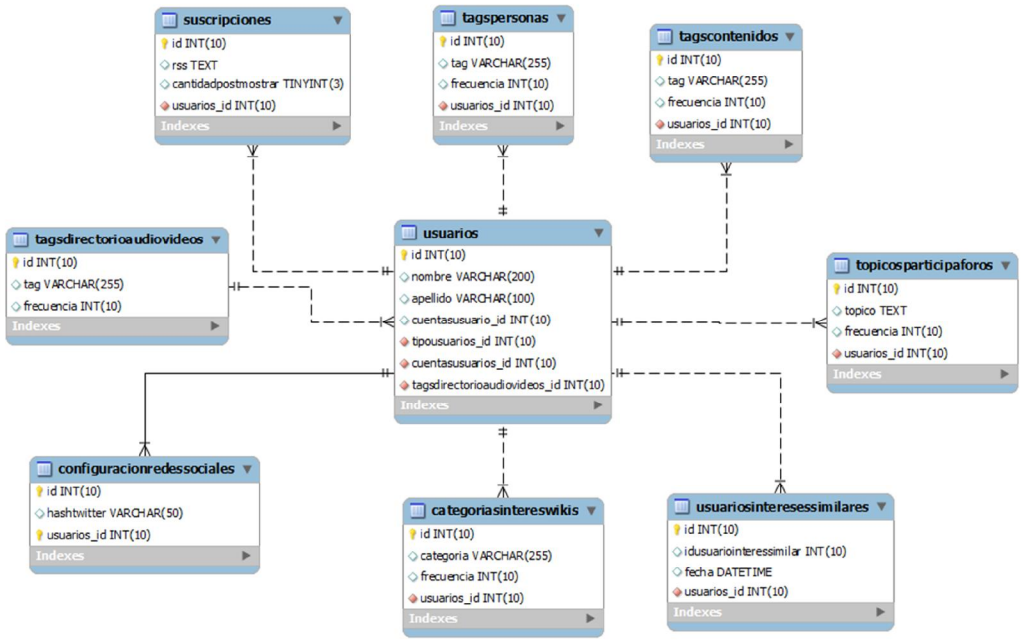


Figura III.17. Diagrama de entidad relación correspondiente al directorio de usuarios. Parte 3

- ***Diagrama de clases del ·"Directorio de Usuarios"***

El diagrama de la figura III.18 muestra las clases que fueron desarrolladas para dar soporte a las funcionalidades ofrecidas por el directorio de usuarios. En el diagrama se aprecia que el usuario puede tener antecedentes laborales, antecedentes de investigación, antecedentes educativos, "amigos", contenidos publicados y tiene asociado una cuenta de usuario y una cuenta de correo la cual contiene emails y en la que se pueden haber definido grupos. Por último, el usuario, sea alumno o profesor, si bien puede pertenecer a varias carreras, mínimamente debe estar asociado a una carrera. Información relacionada a esta relación se almacena en la clase Matricula, básicamente el año de inicio. La carrera pertenece a una determinada facultad, la cual a su vez puede tener varias carreras. Los detalles de cada clase presentada en el diagrama se exponen en el Anexo I.

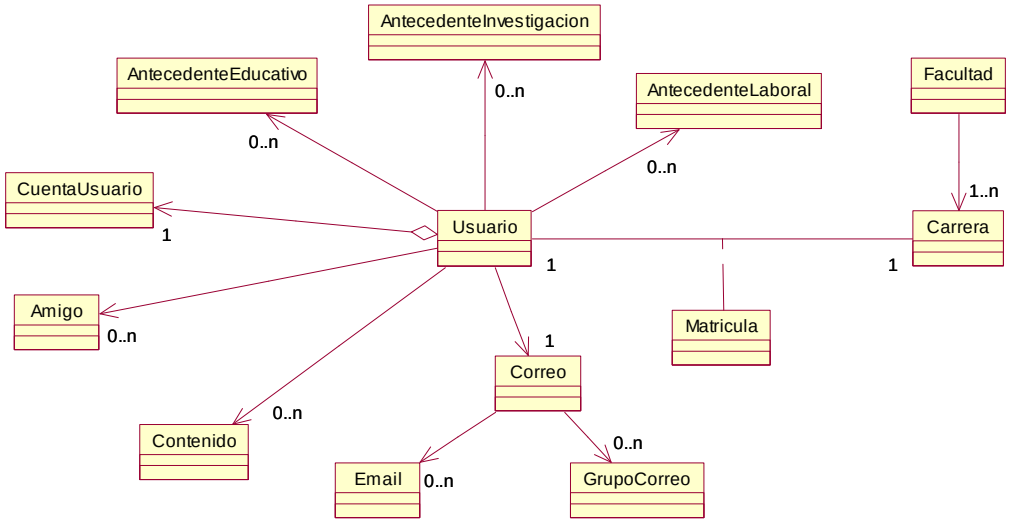


Figura III.18. Diagrama de clases del Directorio de Usuarios

- ***Diagrama de flujo de datos del "Directorio de Usuarios"***

En la figura III.19 y III.20 se muestran los diagramas de flujo de datos que representan las funcionalidades que se proporcionan en el directorio de usuarios, los almacenes de datos que están involucrados y los datos básicos que son requeridos por cada unidad funcional.

Para acceder a las distintas funcionalidades del sistema el usuario debe estar logueado. Por esta razón, cada unidad funcional está vinculada con el proceso "Verificar Sesión Usuario" el cual controla justamente que el usuario haya iniciado una

sesión. En caso de que no lo haya realizado, activa el proceso "Iniciar Sesión" el cual desplegará al usuario una interfaz a través de la cual podrá autenticarse.

Mediante el proceso "Actualizar Perfil" el usuario puede realizar tareas de mantenimiento sobre la información que hace a su perfil (interés, especialidades, antecedentes laborales, entre otros). Por esta razón, el proceso se vincula con los almacenes que albergan estos tipos de datos.

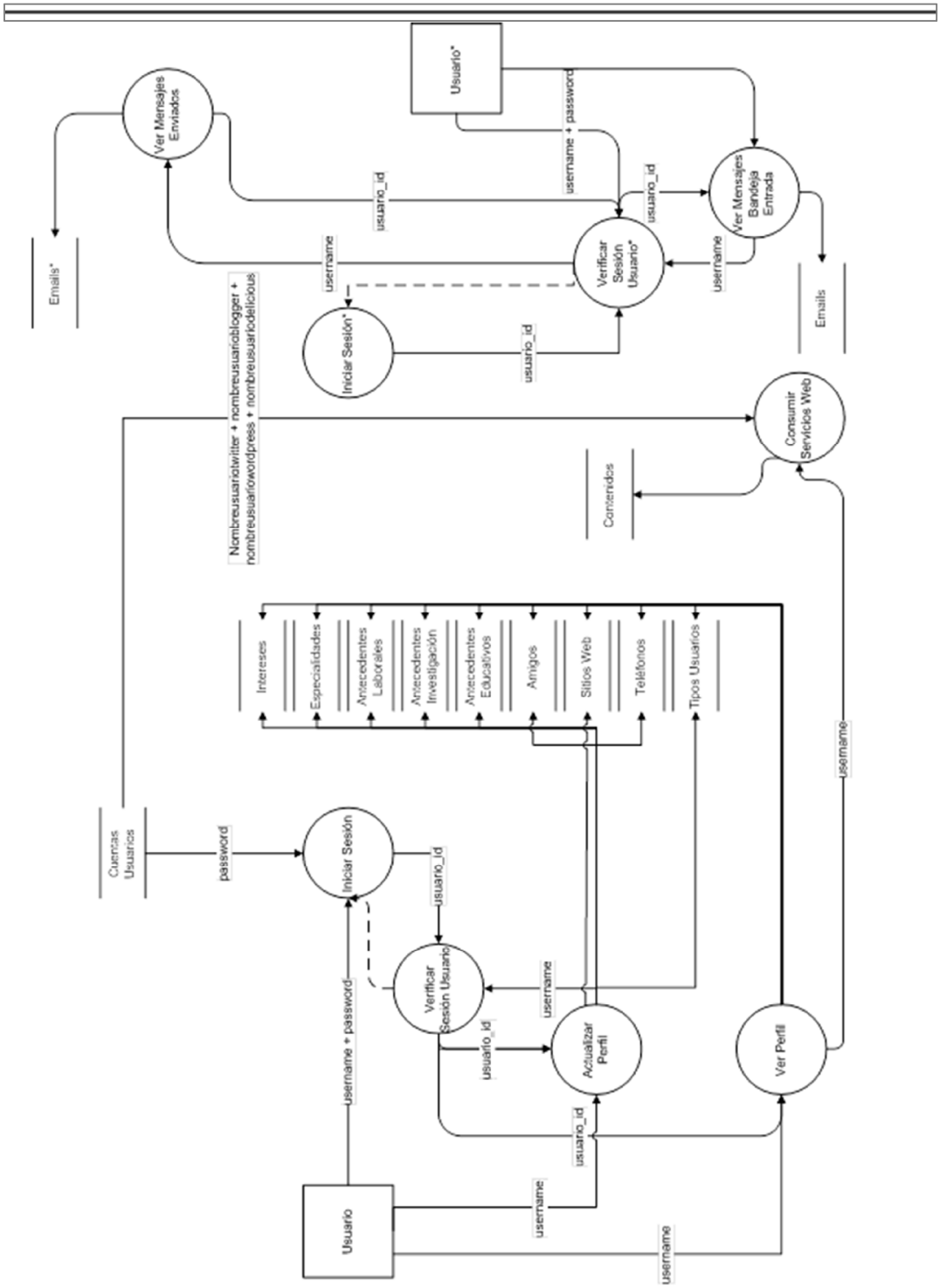


Figura III.19. Diagrama de flujo de datos del Directorio de Usuarios - Parte 1

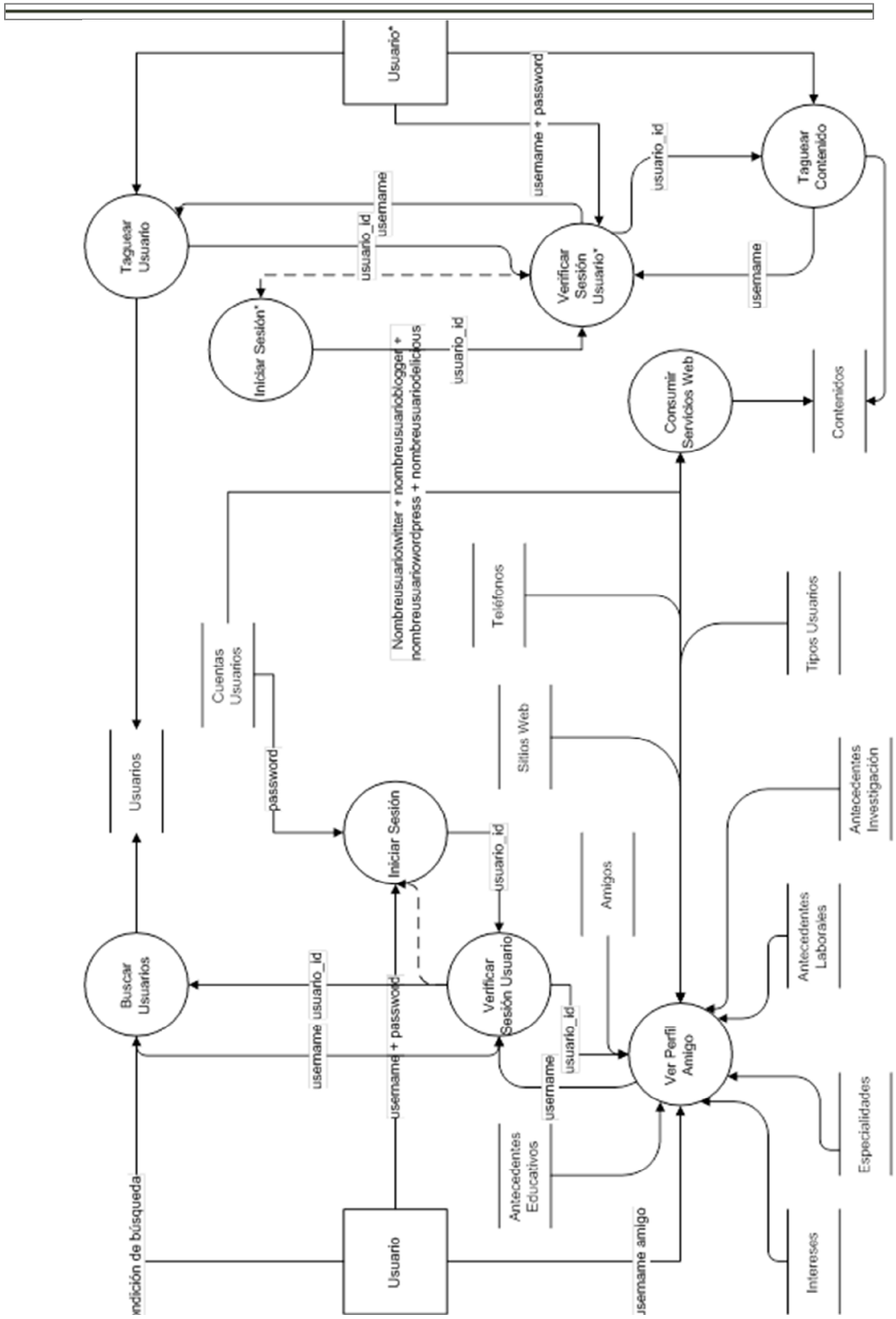


Figura III.20. Diagrama de flujo de datos del Directorio de Usuarios - Parte 2

Al abrir el directorio de usuarios, el proceso por defecto que se inicia es "Ver Perfil", el cual despliega una interfaz que muestra todos los datos que hacen al perfil del usuario que inicia sesión. En la figura III.19 se muestran los almacenes a los que accede para desplegar la interfaz mencionada. El proceso "Ver Perfil" a su vez interacciona con el proceso "Consumir Servicio Web" para extraer contenidos que el usuario haya publicado en otras redes (*Delicious*, *Twitter*, *Wordpress* y *Blogger*). Para conseguirlo, este proceso se conecta con la tabla "Cuentas Usuarios" para obtener los datos de acceso y extraer los contenidos vía RSS. Estos posteos que se consumen desde las redes sociales del usuario posteriormente se almacenan en la tabla *Contenidos* para evitar tener que consumir nuevamente a través de RSS la información. Este proceso de consumir posts desde otras redes se realiza aproximadamente cada una hora.

Con los procesos "Ver Mensajes Bandeja Entrada" y "Ver Mensajes Enviados" se implementa la funcionalidad de correo electrónico en el directorio de usuarios. Estos procesos se vinculan con la tabla *Emails* que es la más importante para esta funcionalidad.

El proceso "Buscar Usuarios" es el encargado de implementar la funcionalidad de búsqueda del directorio de usuarios mostrada en la figura III.14. Básicamente lo que hace es recibir la condición de búsqueda y efectuar una consulta *SQL* sobre la tabla *Usuarios* de tal manera de hacer cumplir la condición solicitada.

El proceso "Taggear Contenido" es el encargado de proporcionar la funcionalidad de tagueo en el directorio de usuarios: taggeo de contenidos de la pestaña muro y de la pestaña blog. El proceso "Taggear Usuario" es el que implementa la funcionalidad de poder taggear perfiles de usuario.

Por último, a través del proceso "Ver Perfil Amigo" se da la posibilidad al usuario de ver los perfiles de otros usuarios del directorio de usuarios. La diferencia de este proceso con el proceso "Ver Perfil" es que éste es más restrictivo en el sentido de que no presenta enlaces para actualizar el perfil, para evitar que el mismo sea actualizado por otra persona.

c. Gestor de perfil de usuarios

El gestor de perfil de usuarios constituye un módulo del subsistema del directorio de usuarios y su función principal es recolectar datos sobre la actividad de los usuarios en los restantes subsistemas, tales como, wiki, foro, directorio de audio y video, twitter, delicious, blogger, wordpress y del sistema de cátedras. El objetivo de este módulo es facilitar el trabajo de los algoritmos de recomendación; en el sentido de que la ejecución de los mismos sea más eficiente, al no tener que salir a buscar información cuando se necesite efectuar recomendaciones, sino que ya tenga disponible con antelación para sólo luego procesarla. El diagrama de entidad-relación mostrado en la figura III.17 es el que soporta fundamentalmente a este módulo. La información generada por este módulo sólo es accedida de manera interna por los restantes subsistemas del sistema general.

d. Directorio de audio y video

El directorio de audio y video tiene por objetivo facilitar al usuario compartir y recuperar información multimedial relacionada a su proceso de formación académica que sirva como complemento a los tradicionales recursos utilizados en su formación. Con esto se pretende contribuir a crear un ambiente de aprendizaje atractivo y el medio apropiado a través del cual registrar múltiples conocimientos tácitos.

En la figura III.21 se muestra el prototipo de la página de inicio de este micrositio en la cual se permite que el usuario pueda efectuar la búsqueda de un archivo multimedia, mostrando previo a la ejecución de cualquier búsqueda, un listado de los últimos archivos subidos. La aplicación permite recomendar recursos multimedia al usuario en base al análisis de la actividad efectuada por el mismo y la existencia de otros usuarios con intereses similares.

Por otro lado, se muestra una nube de palabras claves (tags) que se corresponden con las marcas asignadas a los recursos multimedia. Esto sirve como un mecanismo a través del cual se pueden descubrir nuevos contenidos y efectuar filtros en caso de ser necesaria una exploración más minuciosa.

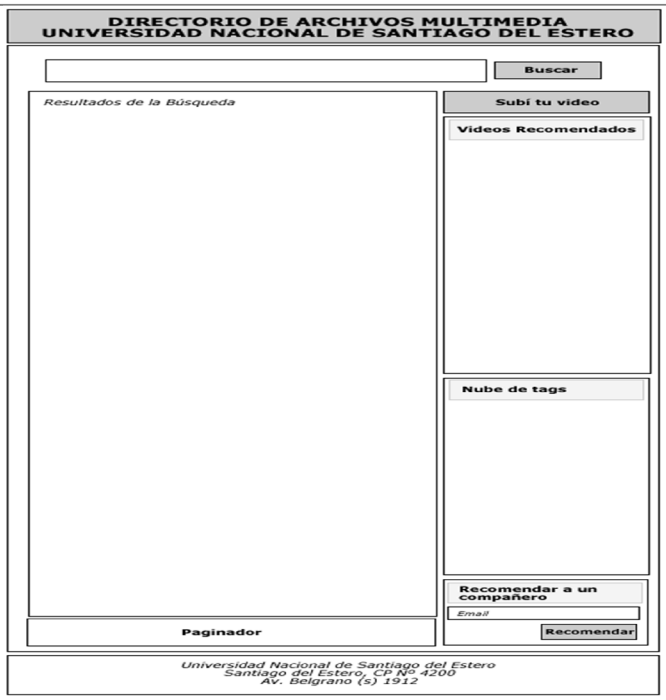


Figura III.21. Prototipo de interfaz de la página de inicio del Directorio de Audio y Video

En la figura III.22 se muestra el prototipo de una página interna tipo de la aplicación. Como se observa, es similar a los principios que direccionan al *Directorio de Usuarios* en el sentido de que el recurso multimedia puede ser tagueado, valorado y comentado por parte de los usuarios.

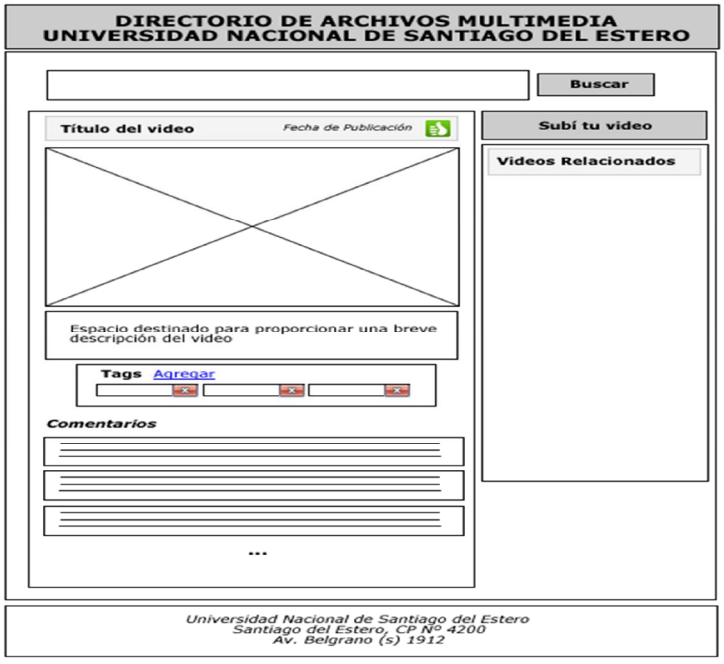


Figura III.22. Prototipo de una página interna tipo del Directorio de Audio y Video

En la figura III.23 se muestra el diagrama de clases en el cual se representan las entidades de software que constituyen el directorio de audio y video, en tanto que, en la figura III.24 se muestra un diagrama de flujo de datos utilizado para modelar el funcionamiento de este subsistema.

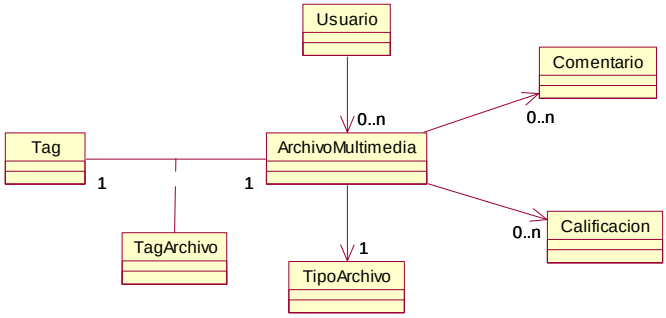


Figura III.23. Diagrama de clases del Directorio de Audio y Video

Al igual que en la aplicación del *Directorio de* Usuarios, para poder usar cualquier funcionalidad de este subsistema el usuario tiene que estar logueado. Por esta razón los procesos "Ver videos", "Gestionar Videos", "Agregar/Eliminar Comentario", "Agregar/Eliminar Tag Video" y "Calificar video" están vinculados con el proceso "Verificar Sesión Usuario" para comprobar que el usuario haya iniciado sesión en el sistema. En caso de que todavía no haya iniciado sesión, este último proceso activa al proceso "Iniciar Sesión"; una vez que el usuario haya realizado esta operación puede tener acceso a la funcionalidad que pretendía acceder.

En el diagrama de flujo se muestran a grandes rasgos las principales funcionalidades que son soportadas por el sistema de audio y video. La más importante es la de ver videos que hayan sido subidos en el sistema. Existen varias formas de recuperar los videos: por rating (valoraciones numéricas asignadas por los usuarios), por usuario, por fecha de publicación, por tag o filtrar por los más comentados.

Otra funcionalidad importante es dar a los usuarios la posibilidad de que puedan gestionar sus videos. Esto abarca el alta, baja y modificación de enlaces de audio y video.

Como características 2.0 se incluyen la posibilidad de taguear a los archivos, calificarlos y comentarlos. Estas funcionalidades son importantes por el hecho de que generan datos que son útiles para los algoritmos de recomendación, y para ir descubriendo el perfil de los usuarios.

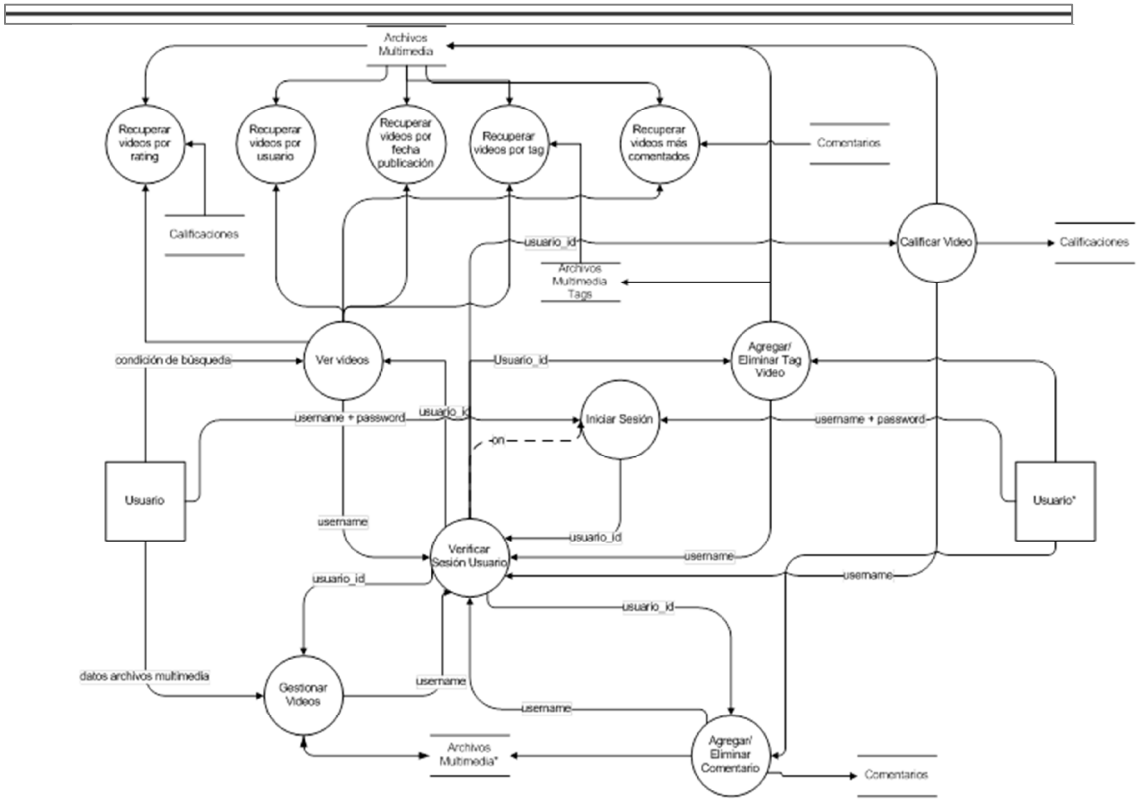


Figura III.24. Diagrama de Flujo de Datos utilizado para el directorio de audio y video

e. Subsistema de cátedras

Con esta aplicación se permite que los equipos cátedras y alumnos tengan los recursos adecuados para una comunicación ágil y sirve como un medio alternativo a la comunicación cara a cara.

El subsistema permite mostrar los datos referidos al equipo cátedra, de los profesores y ayudantes del equipo, junto con sus datos para contactarlos (figura III.25), planificaciones (figura III.26), información sobre asignaturas impartidas por el equipo permitiendo acceder a sus recursos teóricos y trabajos prácticos (figura III.27) e información en general que pueda ser de interés para el alumno (figura III.28). El subsistema provee un canal RSS el cual es usado por el Portal para extraer información sobre equipos cátedra que el usuario haya elegido.

Figura III.25. Prototipo de interfaz de la pestaña Miembros de los sitios de cátedra

Figura III.26. Prototipo de interfaz de la pestaña Planificaciones de los sitios de cátedra

Figura III.27. Prototipo de interfaz de la pestaña "Trabajos Prácticos" de los sitios de cátedra

Figura III.28. Prototipo de interfaz de la pestaña "Novedades" de los sitios de cátedra

En la figura III.29 se muestra el diagrama de entidad-relación que da soporte fundamentalmente al sistema de cátedras. Para cada cátedra se almacena información sobre los profesores y ayudantes, que se almacenan en las tablas *Profesores* y *Ayudantes* respectivamente, y se relacionan con la tabla *Usuarios* a fin de reutilizar sus campos. Para cada profesor se almacena título/s en la tabla *Profesiones* y la relación de pertenencia a uno o más equipos cátedras y su cargo en cada una, información almacenada en la tabla *Profesores_equiposcatedras*.

Cada equipo cátedra a su vez está a cargo de una o varias asignaturas, de ahí la relación entre la tabla *Equiposcatedras* y *Asignaturas*. Las asignaturas a su vez disponen de uno o varios ayudantes estudiantiles que brindan actividad de soporte en el proceso de formación de los alumnos. En el diagrama también se modela el hecho de que las asignaturas disponen de una planificación anual, genera novedades y produce trabajos prácticos; información que es almacenada en las tablas *Planificaciones*, *Novedades* y *Trabajospracticos*, respectivamente. La tabla *Novedades* sirve para constituir los RSS que son consumidos desde el portal. Con respecto a los trabajos prácticos, lo que se hace es almacenar las urls hacia los archivos de los mismos, los cuales son subidos a través de la aplicación del sistema de cátedras. Las planificaciones por su parte son subidas directamente a través de un editor de tipo WYSIWYG, por lo cual al momento de mostrar esta información en el sistema de cátedras se lo despliega como una simple página web.

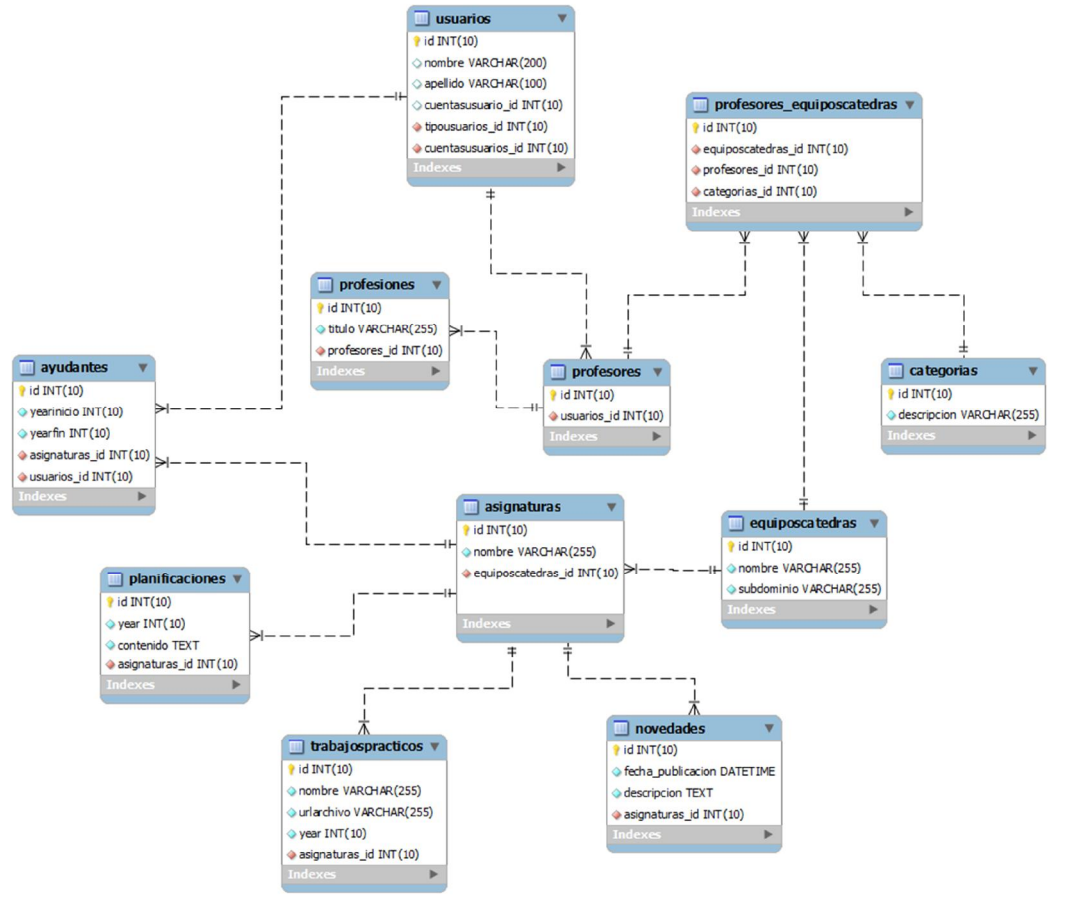


Figura III.29. Diagrama de entidad-relación que soporta el sistema de cátedras

En la figura III.30 se muestra el diagrama de clases que soporta al presente subsistema y en la figura III.31 se muestra el diagrama de flujos de datos. El detalle de cada clase del diagrama de la figura III.30 se pone a disposición en el Anexo I.

A los usuarios del sistema de cátedras se los puede clasificar en tres categorías: administrador (docentes), administrador general (persona/equipo que se encargará de gestionar el sistema completo) y alumno. En el diagrama de la figura III.31 se muestran las funcionalidades básicas que proporciona el sistema a cada tipo de usuario.

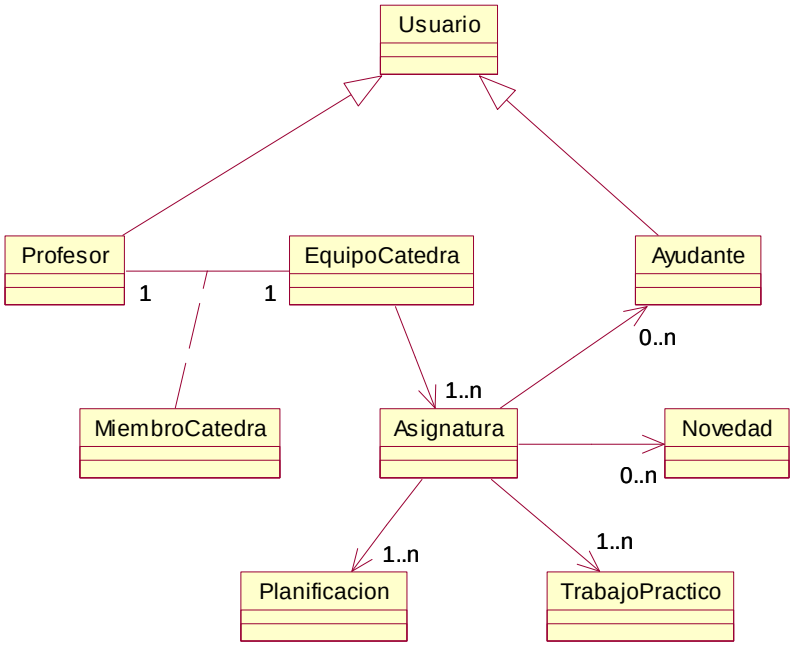


Figura III.30. Diagrama de clases del Sistema para Cátedras

El alumno y en general cualquier tipo de usuario puede ver información sobre el equipo cátedra (profesores que lo integran, ayudantes, asignaturas a cargo, horarios de consulta), información sobre planificaciones, novedades que informe el equipo y la posibilidad de descargar los trabajos prácticos.

La función del administrador general es gestionar los equipos cátedras. Cuando un nuevo equipo cátedra necesita ser dado de alta, esta persona es la encargada de hacerlo. Dar de alta una cátedra consiste en completar un formulario con los datos propios de la cátedra (como por ejemplo su nombre y demás datos definidos en la tabla *equiposcatedras*) y asociar un profesor responsable a la misma. Una vez creado el equipo cátedra, el profesor asociado como responsable podrá llevar a cabo las tareas administrativas necesarias.

Entre las tareas que pueden realizar los profesores en el sistema de cátedras están las posibilidades de mantener actualizada la información del equipo, sus ayudantes, las asignaturas a cargo del equipo, las planificaciones de las asignaturas, los trabajos prácticos, y proporcionar las novedades que a su criterio sean necesarias.

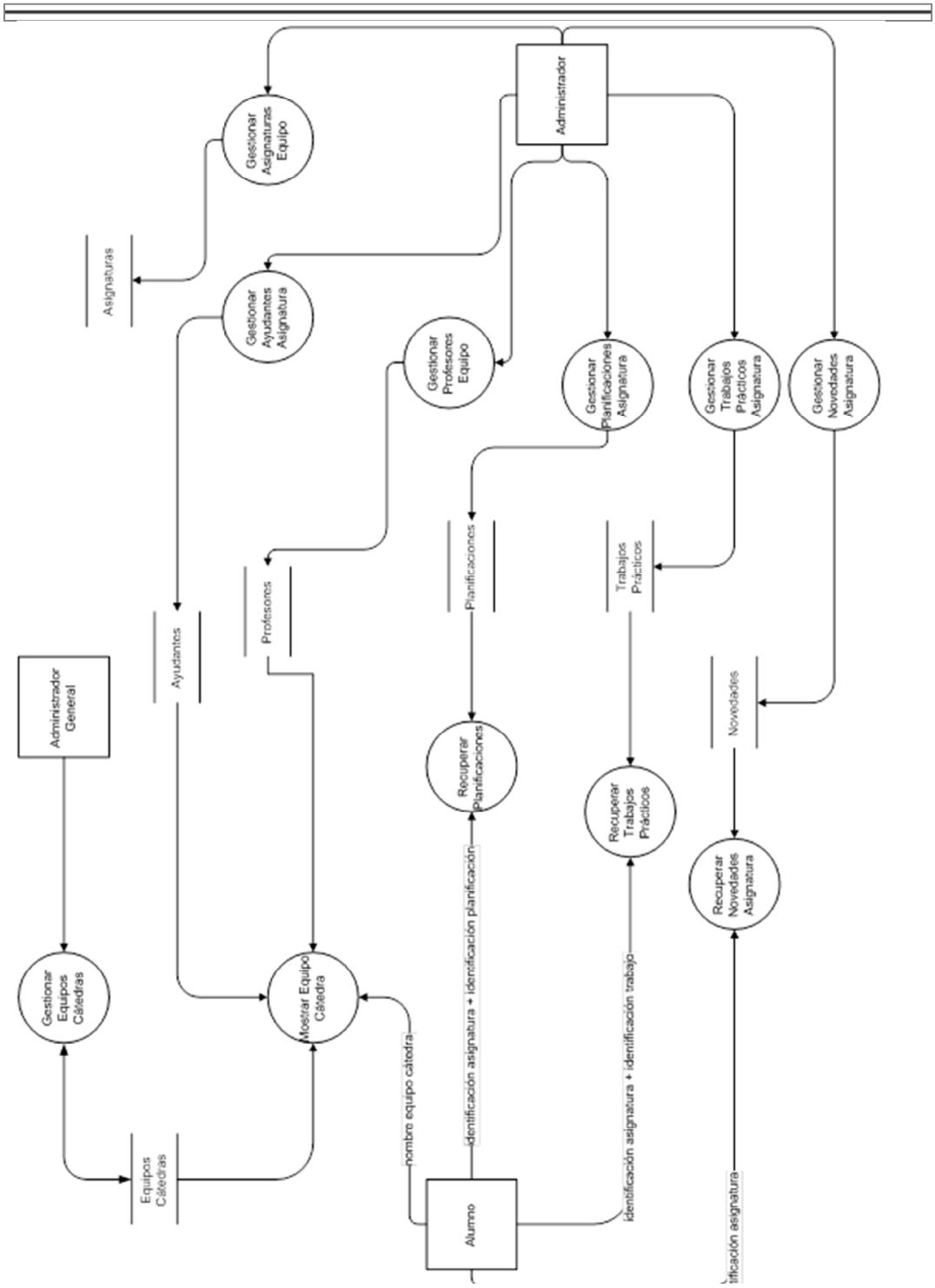


Figura III.31. Diagrama de Flujo de Datos del sistema de cátedras

f. Wiki

El objetivo de la integración de una wiki en el sistema es proporcionar una herramienta que permita la colaboración entre los miembros de la comunidad en la producción de contenidos académicos. En el prototipo que se proporciona, se decidió trabajar con la wiki desarrollada por MediaWiki [15] dado el soporte que ofrece y por el hecho de que es usada como plataforma de Wikipedia, la enciclopedia reconocida a nivel mundial y generada por la colaboración de usuarios de todo el mundo.

Fundamentalmente con esta aplicación se da soporte para que los equipos cátedras proporcionen los recursos pedagógicos-didácticos (notas de clases, diapositivas, bibliografía, entre otros). En general también se aspira a que sea fuertemente utilizada para divulgar información sobre proyectos de investigación y los resultados alcanzados.

Una característica interesante que se implementó es el hecho de recomendar contenido de la wiki al usuario en función a los tipos de temáticas en las cuales él colabora. Para lograrlo, lo que se hace es analizar en qué páginas el usuario trabaja y, como cada página tiene una determinada categoría, se arma una folksonomía del usuario y con esto se efectúan las recomendaciones pertinentes.

Para desarrollar la funcionalidad mencionada, se analizó la wiki y fundamentalmente su modelo de datos. En la figura III.32 se muestra parte del diagrama que sirve para efectuar la integración con la funcionalidad programada.

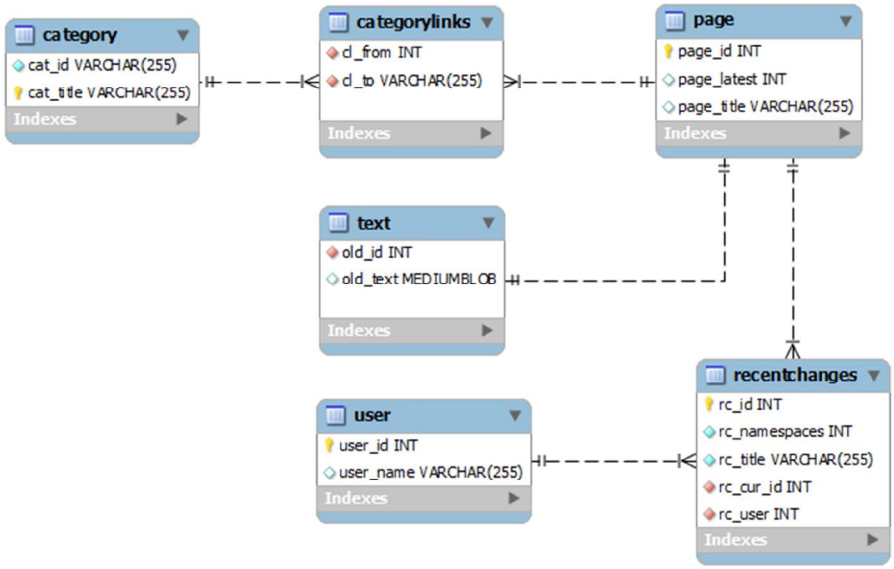


Figura III.32. Diagrama de entidad relación de parte de la Wiki

La tabla *Page* almacena información sobre las páginas que fueron creadas en la wiki. El texto de cada página es almacenada en la tabla *Text* debido a que se realiza control de versiones. La vinculación entre las tablas *Page* y *Text* se da entre las columnas page_id y old_id respectivamente. Cada página a su vez es tagueada con una determinada categoría, la cual es almacenada en la tabla *categorylinks*. A su vez, se lleva un control sobre los cambios que cada usuario va efectuando en las páginas, información que se almacena en la tabla *recentchanges*.

g. Foro

A través de esta herramienta se proporciona un ámbito propicio para la externalización de los conocimientos tácitos de los miembros de la comunidad. Al integrarlo en el marco del proyecto global se intenta fomentar la utilización del mismo para informar a través del portal las últimas actividades efectuadas mediante "posts" relacionados a tópicos en los cuales realmente participa el usuario y que le pueden resultar de interés.

El foro que se utiliza en el proyecto es conocido como *PHPBB* [21] el cual fue elegido por ser el más conocido, por tener buen soporte y ser software libre.

Para lograr el objetivo de proporcionar información de interés al usuario, extraída desde el foro, se analizó la base de datos del mismo. En la figura III.33 se muestra un fragmento del diagrama de entidad relación que representa el modelo de datos y resulta relevante a nivel de integración.

Los temas generados en el foro son almacenados en la tabla *phpbb_forums*, a su vez cada tema tiene asociados uno o más tópicos (*phpbb_topics*), los cuales a su vez tienen uno o más post asociados (*phpbb_posts*). Los post son generados por un usuario determinado (*phpbb_users*). En la figura III.33 se muestra sólo una parte del diagrama de entidad relación y de manera parcial las tablas que son de interés para lograr la integración con el foro.

A través de la vinculación que existe entre el usuario y los posts, se puede conocer en qué temas y tópicos participa, información que es utilizada para poder efectuar recomendaciones de contenido.

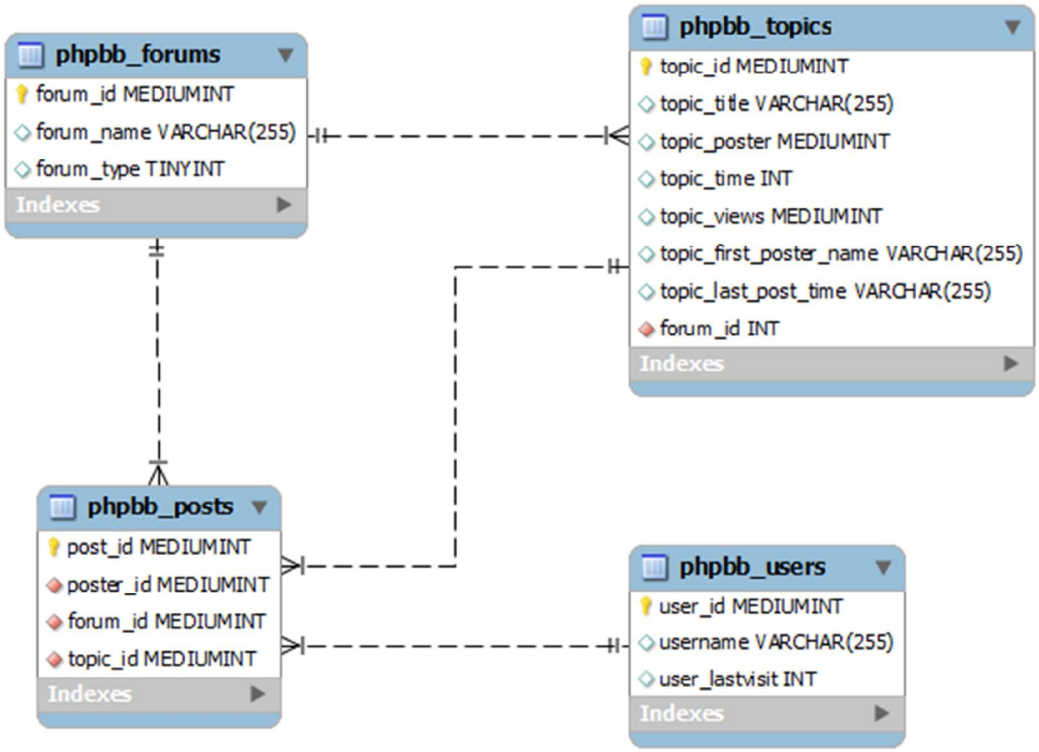


Figura III.33. Fragmento del diagrama de entidad relación del foro

h. Chat

Es un componente integrado en el directorio de usuarios para facilitar una comunicación interactiva y fluida entre los usuarios que se encuentren conectados en un determinado momento; constituyendo una alternativa de comunicación. En el contexto del sistema general es una parte importante porque facilita el proceso de socialización al permitir la circulación de pensamientos, ideas, valores, experiencias, etc., difíciles de externalizar de modo más formal; de esta forma contribuye en gran medida a la circulación del componente tácito del conocimiento.

La implementación que se realiza en el trabajo es muy simple y similar al que se utiliza en *Facebook* en donde por cada sesión de chat que se establezca con un contacto, se abre una pequeña ventana que se posiciona en la parte inferior cerca de la barra de estado del navegador.

Las tablas mostradas en la figura III.34 dan soporte a la funcionalidad del prototipo de chat. En la tabla *usuarios_online* se muestran los usuarios que actualmente se encuentran conectados. Esta tabla es actualizada de manera asincrónica cada cierto

intervalo de tiempo, generalmente cada dos segundos. A través de la tabla *chat* se almacena temporalmente la conversación entre dos usuarios mientras dure la sesión de los mismos. La funcionalidad del chat hace un uso importante de AJAX para poder implementar la interacción entre los usuarios y no producir refrescos de página.

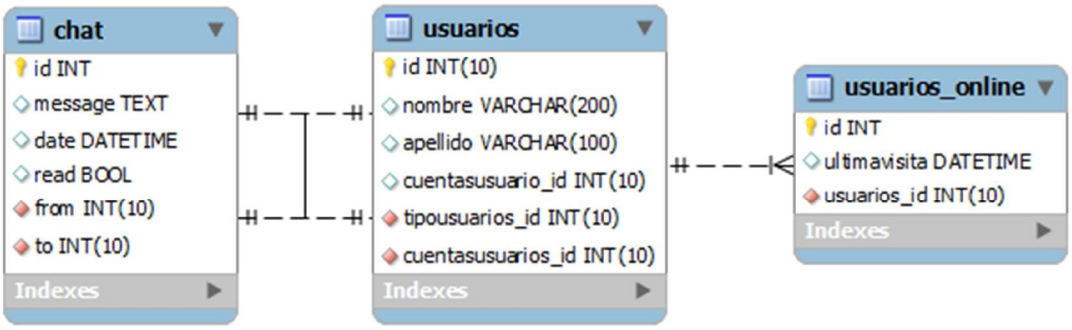


Figura III.34. Diagrama de entidad-relación que da soporte al chat

3.3.2. CAPA DE FILTRADO E INTELIGENCIA COLECTIVA

En esta capa se desarrollan las funcionalidades de:

a) búsqueda y clasificación;

b) búsqueda mejorada mediante una red neuronal,

c) clustering y,

d) recomendación.

Es una capa importante en el modelo propuesto ya que otorga funciones que permiten analizar la información, proporcionar información relativamente relevante a un determinado usuario y de esta forma se busca hacer frente a la sobrecarga de información.

a. Búsqueda y clasificación

Se desarrollan e implementan tres opciones a través de las cuales los usuarios pueden llevar a cabo actividades de clasificación y búsqueda de información:

- Opción de búsqueda jerárquica
- Opción de búsqueda por atributos
- Opción de tagueo

A través de cada una de estas opciones se permite que el usuario tenga acceso a la información bajo demanda, es decir, mediante un tradicional enfoque *pull*.

Sin embargo, además de este método básico de acceso a la información, también se proporciona un enfoque *push*, el cual ofrece la posibilidad de entregar contenido sin necesidad de que el usuario la tenga que buscar de una manera activa. Esta funcionalidad se proporciona a través de las herramientas de sindicación (RSS) y los algoritmos de recomendación.

- ***Opción de búsqueda jerárquica***

Este tipo permite efectuar la búsqueda de alumnos, docentes o investigadores que se encuentren registrados (figura III.35).

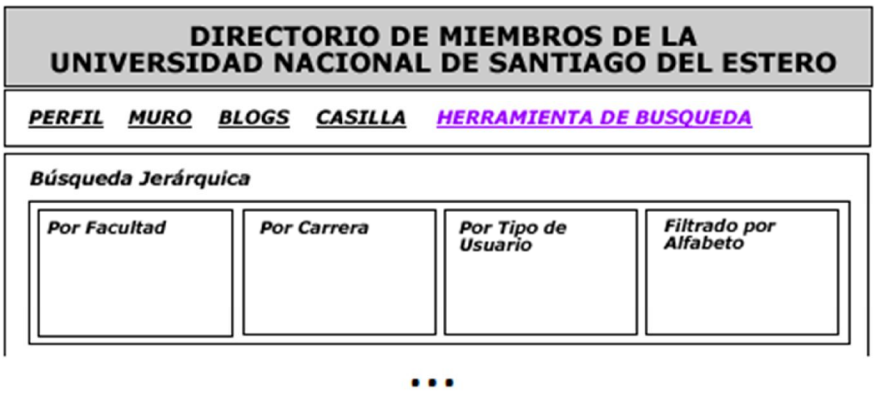


Figura III.35. Opción de búsqueda jerárquica en el directorio de usuarios

Esta opción permite la organización del directorio de usuarios en forma jerarquica según se muestra en lafigura III.36.

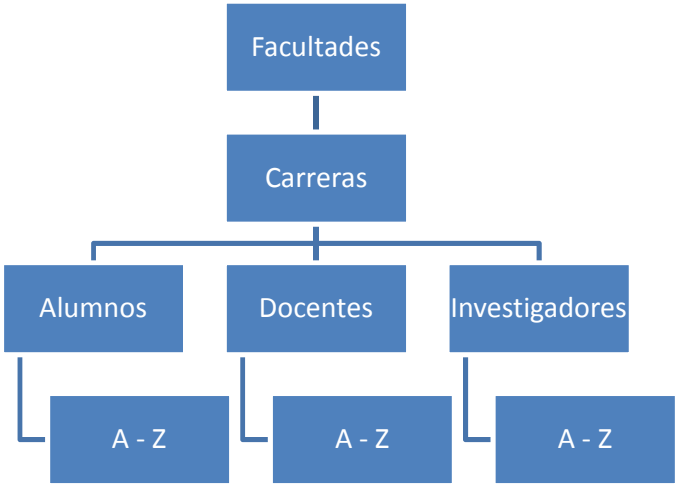


Figura III.36. Jerarquía impuesta en el directorio de usuario por la opción de búsqueda

Esta estructura jerárquica es muy sencilla y permite a los usuarios ir segmentando los resultados de la búsqueda.

- ***Opción de búsqueda por atributos***

Esta opción es la que se implementa en la mayoría de los buscadores, permitiendo a las personas buscar sobre un gran conjunto de documentos por una serie de palabras, clasificando los resultados de acuerdo con la relevancia de las palabras en los documentos.

Para proporcionar esta funcionalidad se diseña y programa un motor de búsqueda que implementa diferentes métricas para valorar la relevancia de los resultados devueltos. Esta funcionalidad se encuentra en el portal (figura III.37).

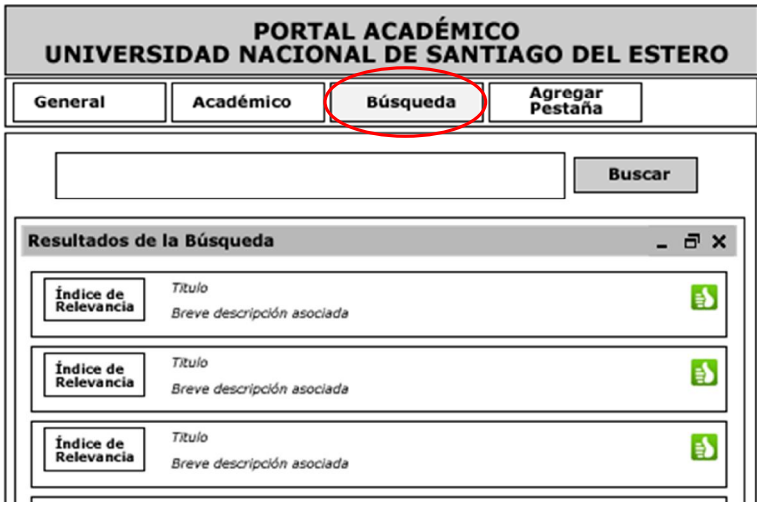


Figura III.37. Prototipo de la interfaz en la cual se proporciona opciones de búsqueda por atributos

La figura III.37 muestra el contenido que se despliega al activar la pestaña que representa la funcionalidad de búsqueda. En esta imagen también se percibe el dibujo de la "manito", con lo cual se permite que el usuario influya sobre los futuros resultados de búsqueda y aprovechar de esta manera la potencia de la colaboración. El usuario puede influir sobre los futuros resultados de búsquedas en base a una red neuronal que clasifica las consultas que se van realizando. La red neuronal aprende a asociar búsquedas con sus resultados de acuerdo con los links sobre los cuales las personas hayan manifestado su preferencia. La red neuronal usa esta información para cambiar el orden de los resultados a fin de reflejar de una mejor manera lo que las personas más utilizan o acceden.

El desarrollo del motor de búsqueda involucró tres pasos:

Paso 1. Desarrollo de un *crawler*. Permite recorrer una gran colección de páginas web a partir de un proceso recursivo que consiste en visitar a cada página web a partir

de un enlace definido en una o más páginas web que se hayan establecido como base. A fin de optimizar el proceso, cada vez que el *crawler* accede a una nueva página invoca al indexador para que este efectúe su tarea.

En las figuras III.38 y III.39 se muestra el fragmento de código desarrollado para el *crawler*.

```
// $pages: Listado de páginas a indexar. Es un arreglo unidimensional
// $depth: Nivel de profundidad de indexación por página
// Por ejemplo, $depth = 2 significa que por cada link que se encuentra
// en la página que se está indexando, se sigue dichos links, y
// los links que se definan en estas últimas páginas.
public function Crawl($pages, $depth = 2)
{
    set_time_limit(0 ); // Se indica que no haya límite de ejecución
    for($i=0; $i <$depth; $i++)
    {
      $newpages = array();
      foreach($pages as $indice =>$page)
      {
            $contents = @file_get_contents($page );
            // Extrae el contenido del archivo especificado en $page a
            // la cadena $contents
            if(!$contents)
            {
            echo"No se pudo escanear la p&aacute;gina $page.<br/>";
            continue;
            }
            // Se crea la representación DOM de la página
            $dom = new DOMDocument();
            // DOMDocument es un objeto usado para representar
            // documentos HTML y XML
            @$dom->loadhtml($contents );
            $xpath = new DOMXPath($dom );
            // DOMXPath es un objeto que proporciona un conjunto de
            // funcionalidades para recorrer el árbol DOM
            $this->AddToIndex($page, $xpath ); // Llamada al indexador

            $links = $xpath->evaluate("/html/body//a");
```

Figura III.38. Script en PHP del crawler desarrollado

Paso 2. Desarrollo del *indexador*. Tiene la función de armar una tabla con los documentos analizados y la ubicación de las diferentes palabras que se detecta en el mismo, esto es conocido como índice. Debido a que en la base de datos no se guarda todo el documento completo, se deja almacenada una referencia a dicho archivo: la url.

```
                for($j=0; $j <$links->length; $j++)
                {
                        $url = $links->item($j)->getAttribute('href');
                        // Se verifica que el anchor haya definido una url
                        if(strlen($url) >0){
                                //Genera una dirección absoluta
                                $url = $this->PIPHP_RelToAbsURL($page, $url);
                                if(strpos($url, "'")) continue;
                                if ((substr($url,0,4) == "http")&&
                                        (!($this->IsIndexed($url))))
                                                $newpages[] = $url;
                                // Extrae el texto del link
                                $this->AddLinkRef($page, $url,
                                        $this->getInnerHTML($links->item($j)) );


                        }
                }
                $this->Dbcommit();
        }
        $pages = $newpages;
    }
}
```

Figura III.39. Continuación de la Figura III.39

Dado que las páginas web poseen muchas "*marcas html*", propiedades y otra información que no es necesario indexar, previo a realizar este proceso, el contenido de la página es tratado de tal manera de extraer sólo el contenido tipo texto.

En la figura III.40 se muestra el esquema de base de datos, el cual es principalmente poblado durante este paso.

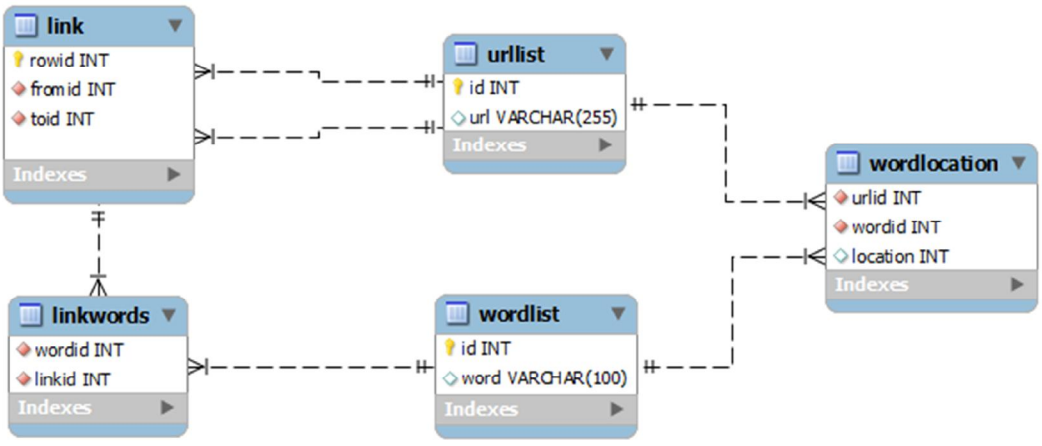


Figura III.40. Esquema de base de datos del buscador

En la tabla *urllist* se almacenan las referencias (url) de todas las páginas web que han sido indexadas; en *worldist* se mantienen todas las palabras que fueron detectadas en las distintas páginas que fueron analizadas; en *wordlocation* se almacena la ubicación de las palabras en los documentos, esta tabla sería el índice; en la tabla *link* se mantienen las referencias entre páginas detectadas en las páginas *crawleadas*, es decir, hipervínculos que hayan existido; y, en la tabla *linkwords* se guardan las palabras usadas en cada una de las urls o enlaces.

Paso 3. Programación de métricas de valoración. Permite devolver una lista clasificada de documentos a partir de una consulta determinada. Las métricas programadas fueron las siguientes:

Frecuencia de palabras: calcula la relevancia de una página en función de la cantidad de veces que las palabras de la consulta aparecen en el contenido. La Figura III.41 muestra el código que implementa esta métrica.

```
//Parámetros de entrada: $rows ver definición en la función GetMatchRows
public function FrequencyScore($rows){
    $counts = array();
    foreach($rows as $row)$counts[$row[0]] = 0;
    foreach($rows as $row)$counts[$row[0]] += 1;
    return$this->NormalizeScores($counts);
}
```

Figura III.41. Implementación de la metrica de valoración por frecuencia de palabras

Ubicación de palabras en el documento: se basa en la hipótesis de que el asunto principal de un documento aparecerá probablemente al inicio de este. Teniendo en cuenta esta suposición, si una página es relevante, el término de búsqueda tiene que aparecer cerca del inicio del documento, incluso probablemente en el título. En la Figura III.42 se muestra el código a través del cual se implementa este método de valoración.

```
//Parámetros de entrada: $rows ver definición en la función GetMatchRows.
public function LocationScore($rows){
    $location = array();
    //Una posición muy alta por defecto
    foreach($rows as $row)$location[$row[0]] = 1000000;
    foreach($rows as $row){
      $loc = array_sum($row) - $row[0];
      if($loc <$location[$row[0]]) $location[$row[0]] = $loc;
    }
    return$this->NormalizeScores($location, true);
}
```

Figura III.42. Implementación de la metrica de valoración por ubicación de palabras en el documento

Distancia entre palabras: Se valora una página en función a la cercanía existente entre los términos utilizados para una determinada consulta. En la Figura III.43 se muestra el código desarrollado para implementar esta opción de valoración.

```
// Parámetros de entrada: $rows ver definición en función GetMatchRows.
public function DistanceScore($rows){
	if(count($rows[0]) <= 2){
		$mindistance = array();
		foreach($rows as $row)$mindistance[$row[0]] = 1;
		return$mindistance;
	}
	else{
	$mindistance = array();
	foreach($rows as $row)$mindistance[$row[0]] = 1000000;
	foreach($rows as $row){
	//Se comienza desde la segunda posición
		$dist = 0;
		for($i = 1; $i < count($row); $i++)
			$dist += abs($row[$i] - $row[$i -1]);
		if($dist <$mindistance[$row[0]])
				$mindistance[$row[0]] = $dist;
		}
	return$this->NormalizeScores($mindistance, true);
	}
}
```

Figura III.43. Implementación de la metrica de valoración por distancia entre palabras

PageRank: es un algoritmo que permite realizar la valoración de páginas. Fue inventado por el fundador de Google y variaciones sobre su idea original son actualmente usados por todas las máquinas de búsqueda. El algoritmo asigna a cada página un puntaje que indica cuán importante una página es. La importancia de la página se calcula a partir de la importancia de todas las otras páginas que hacen referencia a esta y del número de enlaces que esas páginas definan.

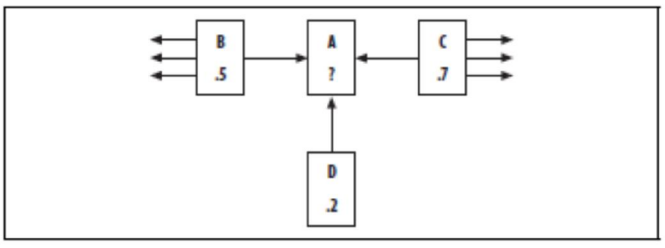


Figura III.44. Cálculo del PageRank de la página A

Por ejemplo, de acuerdo a la figura III.44, el *PageRank* de la página A se calcula de la siguiente manera:

$$PR(A) = 0.15 + 0.85 + (PR(B) / links(B) + PR(C) / links(C) + PR(D) / links(D))$$

Donde *0.15* es un valor mínimo que se le asigna a la página y *0.85* es un *factor de amortiguación* ante el posible efecto que puede causar el hecho de que el usuario continúe haciendo clics en los diferentes enlaces que se definen en las páginas.

En las figuras III.45 y III.46 se muestran las dos funciones principales que implementan el algoritmo para calcular el *PageRank* de una página:

```
public function CalculatePageRank($iterations = 20){
    // Inicializa cada url con un PageRank de 1
    $query = "INSERT INTO pagerank SELECT id, 1.0 FROM urllist";
    $result = $this->dbSearchEngine->query($query);
    set_time_limit(0);
    for($i = 0; $i <$iterations; $i++){
      echo"Iteración N° $i<br/>";
      $query = "SELECT id FROM urllist";
      $urlids = $this->dbSearchEngine->
                query($query, MYSQLI_STORE_RESULT);
      while($row = $urlids->fetch_row()){
            $pr = 0.15;
            //Se efectúa un loop sobre todas las
            // páginas que apuntan a esta
            $query = "SELECT DISTINCT fromid FROM link WHERE toid = " .
                     $row[0];
            $linkers = $this->dbSearchEngine->
                       query($query, MYSQLI_STORE_RESULT);
            while($linker = $linkers->fetch_row()){
                  // Se obtiene el PageRank del linkeador
                  $query = "SELECT score FROM pagerank WHERE urlid =".
                           $linker[0];
                  $resultAux = $this->dbSearchEngine->
                           query($query, MYSQLI_STORE_RESULT);
                  $rowAux = $resultAux->fetch_row();
                  if($rowAux) $linkingpr = $rowAux[0];
                  //Obtiene el número total de enlaces
                  // desde el linkeador
                  $query = "SELECT count(*) as cantidad FROM
                           link WHERE fromid = " . $linker[0];
                  $resultAux = $this->dbSearchEngine->
                           query($query, MYSQLI_STORE_RESULT);
                  $rowAux = $resultAux->fetch_row();
                  $linkingcount = $rowAux[0];
                  $pr += 0.85 * ($linkingpr / $linkingcount);
                  $query = "UPDATE pagerank SET score = $pr
                           WHERE urlid = " .$row[0];
                  $this->dbSearchEngine->query($query);
            }
      }
    }
}
```

Figura III.45. Fragmento de código en el que se implementa la función del PageRank

```
// Parámetros de entrada: $rows (ver definición en la función GetMatchRows.
public function pagerankscore($rows){
     $pageranks = array();
     foreach($rows as $row){
     $query = "SELECT score FROM pagerank WHERE urlid = " .$row[0];
     $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
     $rowAux = $result->fetch_row();
     $pageranks[$row[0]] = $rowAux[0];
     }
     $maxrank = max($pageranks);
     $scoresNormalize = array();
     foreach($pageranks as $urlid =>$score)
       $scoresNormalize[$urlid] = $score / $maxrank;
     return$scoresNormalize;
}
```

Figura III.46. Continuación de la figura III.46

Uso de las palabras de los enlaces: es otra métrica potente para rankear los resultados de una búsqueda. La idea es tener en cuenta las palabras de los links que apuntan a la página que se desea valorar. Se realiza un ciclo a través de todas las palabras que aparecen en el término de búsqueda, buscando todos los enlaces que tengan esas palabras. Si las palabras de los enlaces se corresponden con los términos de búsqueda, el *PageRank* de la página con ese link se le suma al puntaje final de la página que se está evaluando. En la figura III.47 se muestra el código del algoritmo.

Las diferentes métricas, después de ejecutar el cálculo de valoración, aplican la función, *NormalizeScores*, que permite normalizar los resultados. Esta función permite poder comparar los resultados arrojados por las diferentes métricas de búsqueda puesto que en las mismas, la valoración o puntaje que se asigna a cada página tiene distinta interpretación, en algunas casos un puntaje mayor es mejor (caso de la métrica que considera la frecuencia de palabras) en tanto que en otras, el mejor es el menor puntaje (caso de la métrica que considera la posición de las palabras en el documento).

La función de normalización toma como parámetro un arreglo de identificadores de páginas con sus correspondientes puntajes asignados por la métrica y devuelve el mismo arreglo de identificadores pero con puntajes entre 0 y 1. Cada puntaje es escalado de acuerdo a cuán cerca esté del mejor resultado, el cual siempre tomará el valor 1. En la figura III.48 se muestra el código de la función *NormalizeScores.*

En la figura III.49 se muestra el código fuente de una función que permite combinar las distintas métricas que se programaron para valorar los resultados de una búsqueda:

Por último, las dos funciones mostradas en las figuras III.50 y III.51 son usadas para efectuar la búsqueda. Los resultados generados por la función *Query* mostrada en la figura III.50 son usados como parámetros de entrada de la función *GetScoredList* mostrada en la figura III.49.

```
// Parámetros de entrada: $rows y $wordids (ver definición de ambos en la
función GetMatchRows definida más adelante.
public function linktextscore( $rows, $wordids ){
    $linkscores = array();
    foreach($rows as $row)$linkscores[$row[0]] = 0;
    foreach($wordids as $indice =>$wordid){
      $query = "SELECT link.fromid, link.toid FROM linkwords, link
            WHERE wordid = $wordid AND linkwords.linkid = link.rowid";
      $result = $this->dbSearchEngine->
                        query($query, MYSQLI_STORE_RESULT);
      while($row = $result->fetch_row()){
            $fromid = $row[0];
            $toid = $row[1];
            if(array_key_exists($toid, $linkscores)){
                  $query = "SELECT score FROM pagerank
                              WHERE urlid = ". $fromid;
                  $resultAux = $this->dbSearchEngine->
                              query($query, MYSQLI_STORE_RESULT);
                  $rowAux = $resultAux->fetch_row();
                  $linkscores[$toid] += $rowAux[0];
            }
      }
    }
    $maxscore = max($linkscores);
    $scoresNormalize = array();
    foreach($linkscores as $urlid =>$score)
          $scoresNormalize[$urlid] = $score / $maxscore;
    return$scoresNormalize;
}
```

Figura III.47. Algoritmo que permite rankear páginas en función a los textos de su url con respecto a los términos de búsqueda.

```
// $scores es un arreglo asociativo en el cual el índice representa a una
// url y el valor de su celda el puntaje de relevancia otorgado por la
// función getscoredlist
public function NormalizeScores($scores, $smallIsBetter = false){
    $vsmall = 0.00001; // Evita la división por cero
    if($smallIsBetter){
      $minscore = min($scores);
      $scoresNormalize = array();
      foreach($scores as $urlid =>$score){
            $scoresNormalize[$urlid] = $minscore / max($vsmall, $score);
      }
    }
    else{
      $maxscore = max($scores);
      if($maxscore == 0) $maxscore = $vsmall;
      $scoresNormalize = array();
      foreach($scores as $urlid =>$score){
            $scoresNormalize[$urlid] = $score / $maxscore;
      }
    }
    return$scoresNormalize;
}
```

Figura III.48. Código que implementa la función NormalizeScores

```
// Parámetros de entrada: $rows y $wordids (ver definición de ambos en la
función GetMatchRows definida más adelante.
public function getscoredlist($rows, $wordids){
    $totalscores = array();
    foreach($rows as $row)$totalscores[$row[0]] = 0;
    //El primer elemento del siguiente arrego es el peso el cual indica
    // el grado de importancia que se le otorga a una métrica
    $weights[] = array(1, $this->FrequencyScore($rows));
    $weights[] = array(1.5, $this->LocationScore($rows));
    $weights[] = array(1.3, $this->DistanceScore($rows));
    $weights[] = array(1.8, $this->pagerankscore($rows));
    $weights[] = array(1, $this->linktextscore($rows, $wordids));

    foreach($weightsas $weight =>$scores){
      foreach($totalscores as $urlid =>$score){
            $totalscores[$urlid] += $scores[0] * $scores[1][$urlid];
      }
    }
    return$totalscores;
}
```

Figura III.49. Código de una función que permite combinar los resultados de distintas métricas de valoración

```
public function Query($q){
     $result = $this->GetMatchRows($q);
     // Valoriza los resultados devueltos por la consulta
     $scores = $this->getscoredlist($result[0], $result[1]);
     arsort( $scores );// Ordena un arreglo en orden inverso manteniendo
     // las asociaciones de índices
     foreach($scores as $urlid =>$score){
       echo$score . "  " . $this->GetUrlName($urlid) . "<br/>";
     }
}

//Permite efectuar una búsqueda por correspondencia exacta de palabras
// $q es el término o frase a buscar
public function GetMatchRows($q){
     set_time_limit(0);
     //String para construir la consulta
     $fieldlist = 'w0.urlid'; $tablelist = ''; $clauselist = '';
     $wordids = array();
     //Divide las palabras por espacios
     $words = explode(' ', $q); $tablenumber = 0;
     foreach($words as $word){
       //Se obtiene el ID de la palabra
       $query = "SELECT id FROM wordlist WHERE word = '$word'";
       $result = $this->dbSearchEngine->
                    query($query, MYSQLI_STORE_RESULT);
       $row = $result->fetch_row();
       if($row){
             $wordid = $row[0];
             $wordids[] = $wordid;
```

Figura III.50. Algoritmos utilizados para implementar la función de búsqueda

```
            if($tablenumber >0){
                  $tablelist .=", ";
                  $clauselist .=" AND ";
                  $clauselist .="w" . ($tablenumber-1) .".urlid = w" .
                        ($tablenumber) . ".urlid AND ";
            }
            $fieldlist .=", w$tablenumber.location";
            $tablelist .="wordlocation AS w$tablenumber";
            $clauselist .="w$tablenumber.wordid = $wordid";
            $tablenumber += 1;
      }
    }
    //Se crea la consulta
    $fullquery = "SELECT $fieldlist FROM $tablelist WHERE $clauselist";
    $result = $this->dbSearchEngine->
                  query($fullquery, MYSQLI_STORE_RESULT);
    $rows = array();
    while($row = $result->fetch_row()){
      $rows[] = $row;
    }
    $resultado[] = $rows;
    // $rows es un arreglo en el cual cada celda es un arreglo
    // unidimensional de m componentes que indican para cada m (donde m
    // es la palabra que se busca) en qué documento se encuentra y en qué
    // posición dentro del mismo.
    $resultado[] = $wordids;
    // $wordids: es un arreglo unidimensional en el cual cada celda es un
    // valor entero que representa al identificador de una palabra.
    return$resultado;
}
```

Figura III.51. Continuación del código fuente mostrado en la figura III.51

- ***Opción de tagueo***

Las funcionalidades de *"tagueo"* se proporcionan sobre todo en el directorio de usuarios para taguear perfiles, hilos de comentarios y posteos realizados a través del muro (figura III.52, figura III.53). A nivel wiki y directorio de audio y video (figura III.54) también se proporciona la posibilidad de que los usuarios puedan clasificar las entradas con palabras claves.

Con los diferentes tags generados, posteriormente se arman folksonomias (nubes de palabras), dando de esta manera al usuario un medio para realizar filtros o descubrir nuevos contenidos que le puedan resultar de interés.

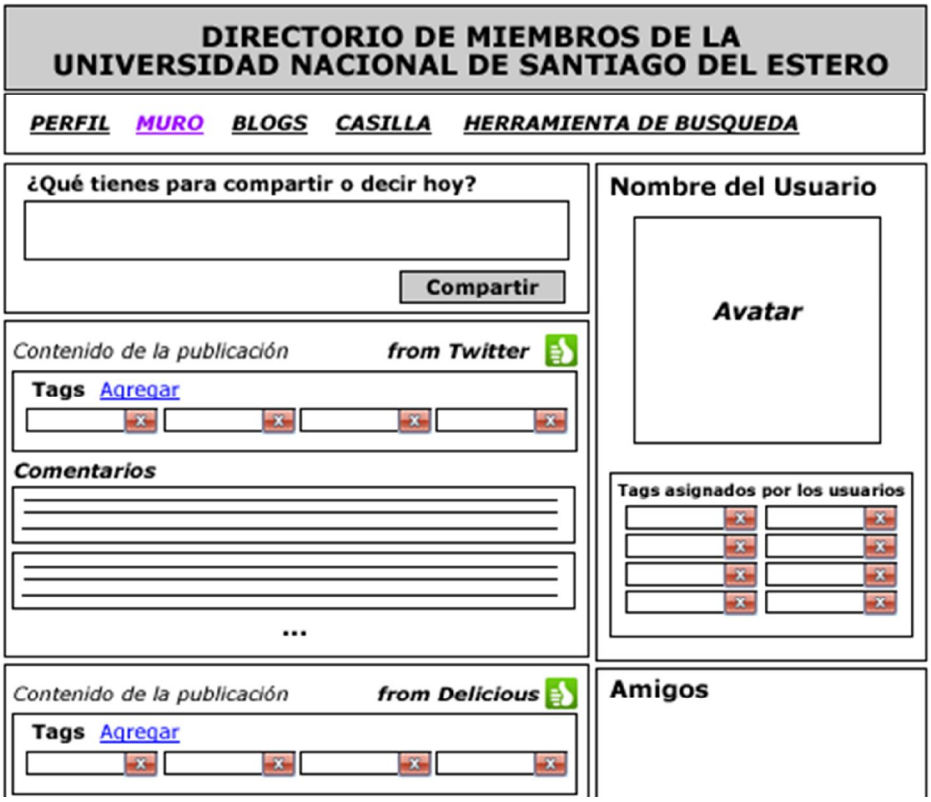


Figura III.52. Prototipo de interfaz del directorio de usuarios en la cual se proporcionan los mecanismos para realizar marcados (tagueos)

Figura III.53. Prototipo de interfaz del directorio de usuarios en la cual se proporcionan los mecanismos para realizar marcados (tagueos)

Figura III.54. Prototipo de interfaz del directorio de auio y video en la cual se proporcionan los mecanismos para realizar marcados (tagueos)

b. Búsqueda mejorada mediante una red neuronal

- ***Esquema de la red neuronal desarrollada***

Como se mencionó anteriormente, para aprovechar el efecto de la colaboración y del potencial uso que le puedan dar los usuarios a la herramienta de búsqueda, se programó una red neuronal que permite sacar ventaja de las dos oportunidades señaladas. El objetivo es que la red neuronal vaya aprendiendo a asociar búsquedas con sus resultados a partir de analizar las actividades de los usuarios con respecto a esta operación.

En el presente trabajo, se programó una red neuronal con una capa de nodos ocultos. Para pedirle a la red que devuelva los mejores resultados para una determinada consulta, los nodos de la entrada de las correspondientes palabras de la consulta se los establece a 1. Las salidas de estos nodos se activarán y ellos intentarán activar los nodos de la capa oculta. A su vez, los nodos de la capa oculta que reciban una entrada lo suficientemente "fuerte", se activarán e intentarán activar los nodos de la capa de salida.

Los nodos en la capa de salida se vuelven activos en diferentes grados y su nivel de actividad puede ser usado para determinar la relevancia de un enlace con respecto a las palabras que integran la consulta original.

Estas cadenas de activaciones dependen de que las conexiones sean correctas, lo cual se logra al entrenar la red cada vez que alguien realiza una consulta y valore los resultados que fueron arrojados (ver figura III.37).

La potencia de usar una red neuronal radica en el hecho de que se pueden hacer predicciones interesantes sobre resultados de consultas que nunca hayan sido efectuadas con anterioridad, basándose en la similaridad con otras consultas.

- ***Configuración de la base de datos para la red neuronal***

Debido a que la red neuronal será entrenada a través del tiempo a medida que los usuarios realicen consultas, se almacenó una representación de la red en una base de datos. La figura III.55 ilustra el diagrama de entidad-relación que define a esta base de datos. Se definió una tabla para la capa oculta (*hiddennode*) y dos tablas de conexiones (una desde la capa de palabras a la capa oculta *wordhidden* y otra desde la capa oculta hacia la capa de salida *hiddenurl*).

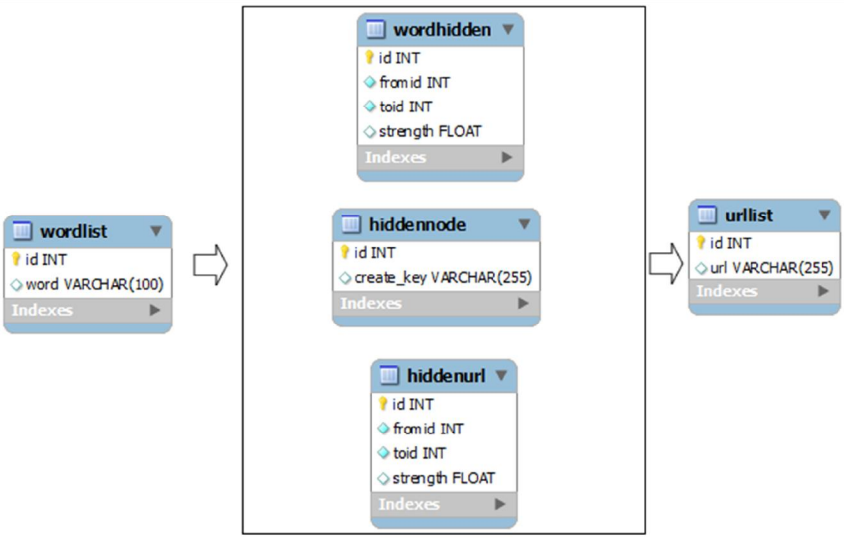


Figura III.55. Esquema de ER que representa la red neuronal en la BD

En este momento, conviene mencionar sobre tres métodos importantes que se encargan de acceder y mantener actualizadas estas tablas de la capa oculta:

GetStrength. Determina el peso actual de una conexión. Debido a que las nuevas conexiones son creadas cuando es necesario, este método devuelve un valor por defecto si no hay conexiones. Para los enlaces desde las palabras a la capa oculta, el valor por defecto es de (*-0.2*) de modo que, por defecto, las palabras extras tendrán un efecto ligeramente negativo sobre el nivel de activación de un nodo oculto. Para los enlaces desde la capa oculta hacia las "urls", el valor por defecto es 0.

```
public function GetStrength($fromid, $toid, $layer){
     if($layer == 0) $table = "wordhidden";else $table = "hiddenurl";
     $query = "SELECT strength FROM $table
               WHERE fromid = $fromid AND toid = $toid";
     $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
     $row = $result->fetch_row();
     if(!$row){
        if($layer == 0) return -0.2;
        if($layer == 1) return 0;
     }
     return$row[0];
}
```

Figura III.56. Código fuente que implementa la función GetStrength

SetStrength. Determina si una conexión ya existe y, actualiza o crea la nueva conexión con un nuevo peso. Este método es usado por el algoritmo de entrenamiento de la red.

```
public function SetStrength($fromid, $toid, $layer, $strength){
    if($layer == 0) $table = "wordhidden";else $table = "hiddenurl";
    $query = "SELECT id FROM $table WHERE fromid = $fromid AND toid =
$toid";
    $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
    $row = $result->fetch_row();
    if(!$row){
      $query = "INSERT INTO $table (fromid, toid, strength)
            VALUES ($fromid, $toid, $strength)";
      $this->dbSearchEngine->query($query);
    }
    else{
      $rowid = $row[0];
      $query = "UPDATE $table SET strength = $strength WHERE id =
$rowid";
      $this->dbSearchEngine->query($query );
    }
}
```

Figura III.57. Código fuente que implementa la función SetStrength

GenerateHiddenNode. Se encarga de crear un nuevo nodo en la capa oculta cada vez que se le pasa una combinación de palabras que nunca ha sido visto antes. La función crea luego enlaces con ponderaciones por defecto entre las palabras y los nodos ocultos y, entre los nodos de la consulta y los "urls" resultantes devueltos por la consulta.

```
public function GenerateHiddenNode($wordids, $urls){
    // Chequea si se creó un nodo para esta lista de palabras
    if(count($wordids) >1) sort($wordids);
    $createkey = "";
    $count_wordids = count($wordids); $cont = 0;
    foreach($wordids as $indice =>$wi){
      if($cont <$count_wordids -1)
            $createkey .= (string)$wi . '_';
      else
            $createkey .= (string)$wi;
      $cont += 1;
    }
    $query = "SELECT id FROM hiddennode WHERE create_key = '$createkey'";
    $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
    $row = $result->fetch_row();
    if(!$row){
      //Se crea el nodo oculto puesto que no existe
      $query = "INSERT INTO hiddennode (create_key) VALUES
('$createkey')";
      $this->dbSearchEngine->query($query);
      $hiddenid = $this->dbSearchEngine->insert_id;
      //Coloca el peso por defecto
      foreach($wordids as $indice =>$wordid)
            $this->SetStrength($wordid, $hiddenid, 0,
      1/count($wordids));
      foreach($urls as $indice =>$url)
            $this->SetStrength($hiddenid, $url, 1, 0.1);
    }
}
```

Figura III.58. Código fuente que implementa la función GenerateHiddenNode

- ***Definición de la función de transferencia***

Continuando con el proceso de configuración de la red neuronal, para definir el mecanismo de activación de nodos en las diferentes capas, se eligió como función de transferencia una del tipo sigmoidal, más precisamente, la tangente hiperbólica (*tanh*). En la figura III.59 se muestra la gráfica de la función tangente hiperbólica.

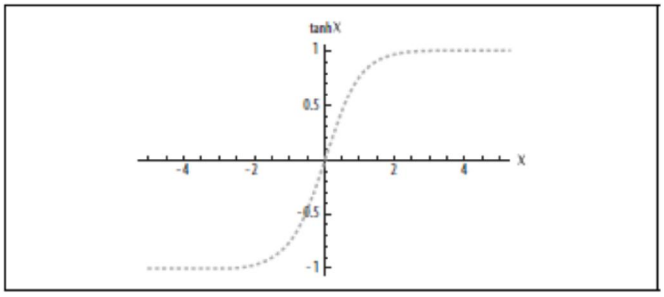


Figura III.59. Gráfica de la función tangente hiperbólica

A continuación se muestra el algoritmo que se usó para el proceso de activación hacia adelante.

```
public function Feedforward(){
     // Activación de los nodos de la capa oculta
     foreach($this->hiddenids as $indice =>$hiddenid){
       $sum = 0;
       foreach($this->wordids as $indiceAux =>$wordid){
              $sum = $sum + $this->ai[$wordid] *
                     $this->wi[$wordid][$hiddenid];
       }
       $this->ah[$hiddenid] = tanh($sum);
     }
     // Activación de los nodos de la capa de salida
     foreach($this->urlids as $indice =>$urlid){
       $sum = 0;
       foreach($this->hiddenids as $indiceAux =>$hiddenid){
              $sum = $sum + $this->ah[$hiddenid] *
                     $this->wo[$hiddenid][$urlid];
       }
       $this->ao[$urlid] = tanh($sum);
     }
     return$this->ao;
}
```

Figura III.60. Código fuente que implementa la función Feedforward

El método de entrenamiento que se empleó es el "método de retropropagación" o "backpropagation" [19]. A continuación se muestra el algoritmo desarrollado para implementar el método indicado:

```
public function BackPropagate($targets, $n = 0.5){
    // Calcula los errores para la salida
    $output_deltas = array();
    foreach($this->urlids as $indice =>$urlid){
      $output_deltas[$urlid] = 0.0;
    }
    foreach($this->urlids as $indice =>$urlid){
      $error = $targets[$urlid] - $this->ao[$urlid];
      $output_deltas[$urlid] = $this->dtanh($this->ao[$urlid]) * $error;
    }

    //Calcula los errores para los nodos de la capa oculta
    $hidden_deltas = array();
    foreach($this->hiddenids as $indice =>$hiddenid){
      $hidden_deltas[$hiddenid] = 0.0;
    }
    foreach($this->hiddenids as $indice =>$hiddenid){
      $error = 0.0;
      foreach($this->urlids as $indiceAux =>$urlid){
            $error = $error + $output_deltas[$urlid ] *
                  $this->wo[$hiddenid][$urlid];
      }
```

```
      $hidden_deltas[$hiddenid] = $this->dtanh($this->ah[$hiddenid]) *
            $error;
    }

    //Actualiza los pesos de las salidas
    foreach($this->hiddenids as $indice =>$hiddenid){
      foreach($this->urlids as $indiceAux =>$urlid){
            $change = $output_deltas[$urlid] * $this->ah[$hiddenid];
            $this->wo[$hiddenid][$urlid] =
                  $this->wo[$hiddenid][$urlid] + $n * $change;
      }
    }

    //Actualiza los pesos de las entradas
    foreach($this->wordids as $indice =>$wordid){
      foreach($this->hiddenids as $indiceAux =>$hiddenid){
            $change = $hidden_deltas[$hiddenid] * $this->ai[$wordid];
            $this->wi[$wordid][$hiddenid] =
                  $this->wi[$wordid][$hiddenid] + $n * $change;
      }
    }
}
```

Figura III.62. Continuación del código mostrado en la figura III.62

En esta sección se mostraron los algoritmos más importantes que se usaron para proporcionar las funcionalidades de búsqueda. El criterio que se tuvo en cuenta para decidir cuál mostrar, fue el que tenían que tener presente aspectos de innovación o ser poco comunes. Por esta razón, el resto de los procedimientos que se usaron pueden ser analizados desde el código fuente que se proporciona con este trabajo.

c. Clustering

En el presente trabajo se proporciona la funcionalidad de efectuar análisis de clustering dado un determinado tag de una folksonomía. Con esto se permitirá descubrir la posible relación que puede existir entre los recursos que hayan sido marcados con tal tag.

Se desarrolló también el correspondiente algoritmo que permite efectuar el gráfico del *dendrograma* a partir del análisis de clúster efectuado. En el código fuente que se proporciona se puede observar los algoritmos desarrollados para generar este tipo de gráfico.

Ademas de poder visualizar el análisis de clustering a través del dendrograma se tiene la posibilidad de ver las relaciones existentes a través de un *escalado multidimensional*, en el cual se muestra una nube de palabras, en donde la cercanía entre las palabras señala la vinculación existente.

El método de clustering que se utilizó es el jerárquico.

- ***Funciones desarrolladas para implementar el clustering jerárquico***

GetMatriz

Esta función lee un archivo el cual tiene almacenado un listado de los contenidos que han sido marcados con un determinado *tag* junto con la frecuencia con la cual aparecen las palabras indexadas desde dichos documentos. La primera línea del archivo contiene las palabras indexadas desde los documentos que serán extraídas hacia el vector "*colnames*"; en tanto que la primera columna almacena una lista de nombres de los documentos bajo análisis que se extraen hacia el vector "*rownames*". Finalmente, la frecuencia con la que se da cada palabra en un determinado documento es almacenada en una matriz bidimensional llamada"*data*", en la cual cada celda se puede referenciar a través de los índices de los vectores"*rownames*" y "*colnames*".

```
functionGetMatriz(&$colnames, &$rownames, &$data)
{
     $linesFile = file("blogdata.txt");
     $arrayLines = "";
     foreach($linesFile as $line)
     {
        $arrayLines[] = $line;
     }
     //La primera línea contiene los títulos de las columnas
     $colnames = explode("\t", trim($arrayLines[0]));
     array_shift($colnames);
     $rownames = "";
     $data = "";
     for($i=1; $i < count($arrayLines); $i++)
     {
        $vAux = explode("\t", trim($arrayLines[$i]));
        //La primera columna de cada fila es el nombre de la fila
        $rownames[] = $vAux[0];
        //Los datos para esta fila es el resto de la fila
        $dataAux = "";
        for($j = 1; $j < count($vAux); $j++)
        {
               $dataAux[] = $vAux[$j];
        }
        $data[] = $dataAux;
     }
}
```

Figura III.63. Código fuente en el que se implementa la función GetMatriz

Lo siguiente que se tiene que definir es la similaridad entre los clusters. Para calcularlo se emplea el *coeficiente de correlación de Pearson*.

```
Function coeficientePearson($v1, $v2)
{
     //Suma todas las preferencias
     //Suma también el cuadrado del valor de las preferencias
     //Suma el producto de las preferencias
     $sum1 = 0; $sum1Sq = 0;
     $sum2 = 0; $sum2Sq = 0;
     $pSum = 0;
     for($i = 0; $i < count($v1); $i++)
     {
        $sum1 += $v1[$i];
        $sum1Sq += pow($v1[$i], 2);
        $sum2 += $v2[$i];
        $sum2Sq += pow($v2[$i], 2);
        $pSum += $v1[$i ] * $v2[ $i ];
     }
     //Calcula el coeficiente de Pearson
     $n = count($v1);
     $num = $pSum - ($sum1 * $sum2 / $n);
     $den = sqrt(($sum1Sq - pow($sum1,2)/$n)*
               ($sum2Sq-pow($sum2,2)/$n));
     if($den == 0) return 0;
     $r = 1 - ($num/$den);
     returnabs( $r );
}
```

Figura III.64. Código fuente en el que se implementa el coeficiente de correlación de Pearson

Cada clúster en el algoritmo de clustering jerárquico es ya sea un nodo con dos ramas, o una terminación asociada con una fila real del conjunto de datos (en este caso, un documento que este siendo analizado). Cada clúster contiene también datos sobre su ubicación, el cual es ya sea el vector "dato" para las terminaciones o un vector producto del mezclado de clúster para los otros tipos de nodos. Para representar el árbol jerárquico, se definió la siguiente clase: "bicluster", la cual tiene todas estas propiedades que se comentaron.

```
classBicluster
{
	public$left;
	public$right;
	public$distance;
	public$vec;
	public$id;
	function__construct($vec, $id = "", $left = "",
		$right = "", $distance = 0.0)
	{
		$this->left = $left;
		$this->right = $right;
		$this->distance = $distance;
		$this->vec = $vec;
		$this->id = $id;
	}
}
```

Figura III.65. Código de la clase que representa a una instancia de un objeto de tipo Bicluster

En la figura III.66 se muestra la codificación del algoritmo de clustering jerárquico.

```
functionHCluster($rows, $distance = "coeficientePearson")
{
    $distances = array();
    $currentClusterId = -1;
    //Los clusters inicialmente son las distintas filas
    $clusters = array();
    $i = 0;
    foreach($rows as $idRow =>$valueRow)
    {
      $clusters[] = new Bicluster($valueRow, $i);
      $i += 1;
    }
```

Figura III.66. Código fuente en el que se implementa la funcionalidad de clustering jerárquico

```
    while( count( $clusters ) >1 )
    {
      $primerElementoPar = 0;
      $segundoElementoPar = 1;
      $clusterAux1 = $clusters[$primerElementoPar];
      $clusterAux2 = $clusters[$segundoElementoPar];
      $closest = $distance($clusterAux1->vec, $clusterAux2->vec );
      //Hace un ciclo por cada par buscando la distancia más
pequeña
      for($i=0; $i < count($clusters); $i++)
      {
            for($j = $i + 1; $j < count($clusters); $j++)
            {
                  //$distances almacena el cálculo de las
            distancias
                  $clusterAux1 = $clusters[$i];
                  $clusterAux2 = $clusters[$j];
                  $id = $clusterAux1->id ."-" . $clusterAux2->id;
                  if(array_key_exists($id, $distances) != 1)
                  {
                        $distances[$id] =
                              $distance($clusterAux1->vec,
                              $clusterAux2->vec);
                  }
                  $d = $distances[$id];
                  if($d <$closest)
                  {
                        $closest = $d;
                        $primerElementoPar = $i;
                        $segundoElementoPar = $j;
                  }
            }
      }
      //Calcula el promedio de los dos clusters
      //$clusterAux= $clusters[0];
      $clusterAux1 = $clusters[$primerElementoPar];
      $clusterAux2 = $clusters[$segundoElementoPar];
      $mergevec = array();
      for($i=0; $i < count($clusterAux1->vec); $i++)
      {
          $mergevec[] = ((float)$clusterAux1->vec[$i] +
                        (float)$clusterAux2->vec[$i])/2;
      }
      //Se crea el nuevo cluster
      $newcluster = new Bicluster($mergevec, $currentClusterId,
                  $clusterAux1, $clusterAux2, $closest);
      $currentClusterId = -1;
      deleteItemArreglo($segundoElementoPar, $clusters);
      deleteItemArreglo($primerElementoPar, $clusters);

      array_push($clusters,$newcluster);
    }
    return$clusters[0];
}
```

Figura III.67. Continuación del código de la figura III.66

Debido a que cada clúster referencia a los dos clústeres que fueron mezclados para crearlo, el clúster final que devuelve esta función puede ser recorrido recursivamente para recrear todos los clústeres y sus nodos.

- ***Visualización de datos en dos dimensiones – Escalado multidimensional***

Para comprender las relaciones entre varios ítems suele ser útil verlos graficados sobre una página con distancias reducidas indicando la similaridad existente entre los datos.

El escalado multidimensional es una técnica que permite encontrar una representación bi-dimensional de un conjunto de datos. El algoritmo toma la diferencia entre cada par de ítems e intenta hacer un gráfico en el cual las distancias entre los ítems se correspondan con aquellas diferencias. Para lograrlo, el algoritmo primero calcula las distancias objetivo entre todos los ítems. En el presente desarrollo, para efectuar estos cálculos se emplea el *coeficiente de correlación de Pearson*.

Al inicio todos los ítems (en este caso blogs) son colocados aleatoriamente en un espacio bidimensional.

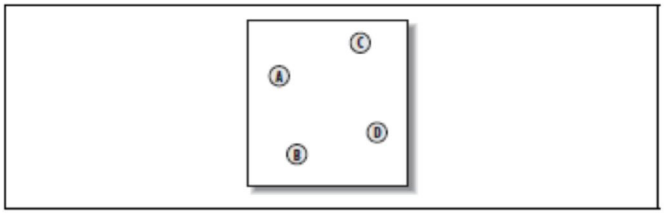


Figura III.67. Ubicación inicial de la proyección 2D

La distancia real entre todos los ítems se calcula usando la distancia actual (suma de la diferencia de los cuadrados).

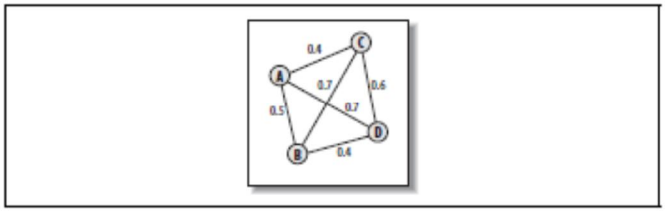


Figura III.68. Distancias entre items

Para cada par de ítems, la distancia objetivo es comparada a la distancia actual y un término de error se calcula. Cada ítem se mueve en una pequeña proporción alejándose o acercándose en proporción al error entre los dos ítems. La figura III.70 muestra las fuerzas que actúan sobre el ítem A. La distancia entre A y B en el gráfico es

0.5, pero la distancia objetivo es sólo 0.2, de modo que A tiene que ser movida de tal manera de acercarla a B. Al mismo tiempo, A también está siendo atraída a C y D porque están muy relacionados.

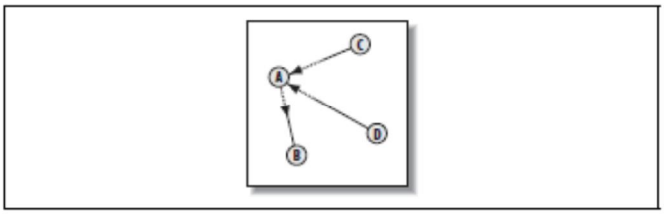


Figura III.69. Fuerzas que actúan sobre el item A

Cada nodo es movido de acuerdo a la combinación de todos los otros nodos según estos estén efectuando fuerzas de atracción o repulsión. Cada vez que esto ocurre, las diferencias entre las distancias actuales y las deseadas se hacen más pequeñas. El proceso se repite tantas veces hasta que el error total sea reducido.

La función que nos permite lograr estos resultados toma como argumento el vector de datos y devuelve como resultado dos columnas: X e Y, coordenadas de los ítems en el gráfico bidimensional.

```
Function scaledown($data, $distance = "coeficientePearson", $rate = 0.1)
{
     $n = count($data);
     //Distancia real entre cada par de item
     for($i = 0; $i <$n; $i++)
       for($j = 0; $j <$n; $j++)
              $realdist[$i][$j] = $distance($data[$i], $data[$j]);
     $outersum = 0;
     // sistema de coordenadas cartesianas XY
     for( $i = 0; $i <$n; $i++ )
       $loc[$i] = array(random(0, 1), random(0, 1));
```

Figura III.70. Código fuente en el que se implementa la funcionalidad de escalado multidimensional

```
    for($i = 0; $i <$n; $i++)
      for($j = 0; $j <$n; $j++)
            $fakedist[$i][$j] = 0.0;

    $lasterror = "";
    for($m = 0; $m <1000; $m++)
    {
      //Busca las distancias proyectadas o esperadas
      for($i = 0; $i <$n; $i++)
      {
            for($j = 0; $j <$n; $j++)
            {
                  for($x = 0; $x < count($loc[$i]); $x++)
                  {
                        $aux += pow($loc[$i][$x] -
                              $loc[$j][$x], 2);
                  }
                  $fakedist[$i][$j] = sqrt($aux);
            }
      }
      //Mueve los puntos
      for($i = 0; $i <$n; $i++)
            $grad[$i] = array(0.0, 0.0);
      $totalerror = 0;
      for($k = 0; $k <$n; $k++)
      {
            for ( $j = 0; $j <$n; $j++ )
            {
                  if ( $j == $k ) continue;
                  // El error es el porcentaje de diferencia
                  // entre las distancias
                  $errorterm = ( $fakedist[ $j ][ $k ] -
                        $realdist[$j][$k])/$fakedist[$j][$k];
                  // Cada punto debe ser movido más cerca o más lejos
                  // a otro a punto en proporción al error existente
                  $grad[$k][0] += (($loc[$k][0] -
                        $loc[$j][0])/$fakedist[$j][$k]) *
                        $errorterm;
                  $grad[$k][1] += (($loc[$k][1] -
                        $loc[$j][1])/$fakedist[$j][$k]) *
                        $errorterm;
                  // Se tiene en cuenta el error total
                  $totalerror += abs($errorterm);
            }
      }

      //Si la respuesta obtenida empeora el movimiento del punto, se
finalizó
      if(($lasterror!="")&&($lasterror <$totalerror)) break;
      $lasterror = $totalerror;
      //Mueve cada punto al aprender el gradiente de la tasa de error
      for($k=0; $k <$n; $k++)
      {
            $loc[$k][0] -= $rate * $grad[$k][0];
            $loc[$k][1] -= $rate * $grad[$k][1];
      }
    }
    return$loc;
}
```

Figura III.71. Continuación del código mostrado en la figura III.72

d. Mecanismos de recomendación

En esta sección se detalla la particularización de cómo fueron implementados cada unos de los pasos necesarios (señalados en el marco metodológico, sección 2.2.4) para proporcionar un mecanismo de recomendación en el sistema.

1. Métrica seleccionada para definir la similitud entre vecinos: Coeficiente de Correlación de Pearson. Codificación:

```
function coeficientePearson($prefs, $person1, $person2)
{
    $si = array();
    //Mantiene la lista de intereses compartidos "shared-items"
    $itemsPerson1 = $prefs[$person1];
    if(is_array($itemsPerson1))
    {
      while($keyValue = each($itemsPerson1))
      {
            //Item a comparar: $keyValue[0]
            if(array_key_exists($keyValue[0],
                  $prefs[$person2 ] ) )
            {
                  $si[$keyValue[0]] = 1;
            }
      }
      //Determina el número de elementos del arreglo "si"
      $n = count($si);

      //Sin no hay puntos de comparación se devuelve 0
      if($n == 0) return 0;
      //Suma todas las preferencias
      //Suma también el cuadrado del valor de las preferencias
      //Suma el producto de las preferencias
      $sum1 = 0;
      $sum1Sq = 0;
      $sum2 = 0;
      $sum2Sq = 0;
      $pSum = 0;
      while($key = key($si))
      {
            $sum1 += $prefs[$person1][$key];
            $sum1Sq += pow($prefs[$person1][$key], 2);
            $sum2 += $prefs[$person2][$key];
            $sum2Sq += pow($prefs[$person2][$key], 2);
            $pSum += $prefs[$person1][$key] *
                  $prefs[$person2][$key];
            next( $si );
      }
```

Figura III.72. Código fuente en el que se implementa la función de correlación de Pearson

```
      //Calcula el coeficiente de Pearson
      $num = $pSum - ( $sum1 * $sum2 / $n );
      $den = sqrt(($sum1Sq - pow($sum1,2)/$n)*
            ($sum2Sq-pow($sum2,2)/$n));
      if($den == 0) return 0;
      $r = $num/$den;
      return$r;
    }
```

Figura III.73. Cóntinuación del código fuente mostrado en la figura III.74

2. Selección del vecindario. Función implementada:

```
function TopMatches($prefs, $person, $n = 5,
      $similarity = "coeficientePearson")
{
    $allPersons = array_keys($prefs);
    $scores = "";
    foreach($allPersons as $personToCompare)
    {
      if($person != $personToCompare)
          $scores[$personToCompare] = $similarity($prefs,
                  $person, $personToCompare) . "<br/>";
    }
    arsort($scores);
    $scoresAux = "";
    $cont = 0;
    while($key = key($scores))
    {
      $cont += 1;
      $scoresAux[$key] = $scores[$key];
      if($cont == $n-1) break;
      elsenext($scores);
    }
    return$scoresAux;
}
```

Figura III.74. Código fuente de la función utilizada para la selección del vecindario

3. Generación de las predicciones. Algoritmo implementado:

```
function Recomendar( $prefs, $person,
      $similarity = "coeficientePearson")
{
    $totals = "";
    $simSums = "";
    $allPersons = array_keys($prefs);
    foreach($allPersons as $personToCompare )
    {
      // No compara con respecto a la persona a quien se
      // le quiere hacer la recomendación
      if( $person == $personToCompare ) continue;
      $sim = $similarity($prefs, $person, $personToCompare) .
            "<br/>";
      //Ignora puntuajes menores o iguales a 0
      if($sim <= 0) continue;
      $itemsOtherPerson = $prefs[$personToCompare];
      while( $keyValue = each( $itemsOtherPerson ) )
      {
            //Item a comparar: $keyValue[0]
            //Sólo se considera items que no hayan sido puntuados
            //por la persona a quien se le quiere hacer
      recomendaciones
            if(!(array_key_exists($keyValue[0],
                  $prefs[$personToCompare]))||
                  ($prefs[$person][$keyValue[0]] == 0))
            {
                  // Similaridad * Puntuaje
                  $totals[$keyValue[ 0 ] ] +=
                        $prefs[$personToCompare ][ $keyValue[ 0 ] ]
            *
                        $sim;
                  // Suma de similaridades
                  $simSums[$keyValue[0]] += $sim;
            }
      }
    }
    //Crea la lista normalizada
    $items = array_keys($totals );
    //Lista de posibles items a recomendar
    $rankings = "";
    foreach($items as $item)
    {
      $rankings[$item] = $totals[$item]/$simSums[$item];
    }
    arsort($rankings);
    return$rankings;
}
```

Figura III.75. Código fuente del algoritmo de recomendación propiamente dicho

3.3.3. CAPA DE ACCESO Y AUTENTICACIÓN

En esta sección se explica a grandes rasgos los aspectos que se contemplan en la capa de acceso y autenticación:

a) administración de cuentas,

b) copias de respaldo y,

c) firewall.

a. Administración de cuentas

La administración de cuentas es un subsistema que tiene por objetivo almacenar información relacionada a los datos de acceso de los usuarios al sistema y también los datos mínimos para la interacción de los usuarios con otras redes en la que participa (Wiki, Foros, Twitter, Delicious, Blogger y Wordpress). El dato mínimo es el nombre de usuario.

Este módulo es el encargado de proporcionar el servicio de autenticación y autorización que es usado por el resto de los subsistemas como el de Cátedras, Directorio de Audio y Video, Directorio de Usuarios y el Portal. El módulo de autorización se lo implementó como una *lista de control de acceso*, en la cual se definieron roles a los que los usuarios deben ser asignados. Se definieron tres tipos de roles: administrador, usuario registrado e invitado. Para cada rol se definieron las acciones del sistema a las cuales puede tener acceso.

Por último, a través de este módulo los usuarios pueden configurar los tipos de posts que el sistema puede consumir desde Twitter. Para configurar estos filtros el usuario debe definir los hashtags.

b. Copias de respaldo

Es de fundamental importancia generar periódicamente, en base al nivel del uso del sistema, copias de respaldo de las bases de datos de cada subsistema que compone al sistema propuesto. Por otro lado, se debe mantener una copia del sistema (código fuente) y tener registrada toda la configuración del mismo en caso de que sea necesario tener que volver a ponerlo en operación después de algún problema.

c. Firewall

Es importante configurar un firewall a fin de restringir tráfico malicioso desde Internet como por ejemplo tráfico no deseado, intentos de conexión por fuera de la red privada y datos maliciosos. La arquitectura de configuración de despliegue de los

firewalls debe ser decidido por el departamento de IT de la organización en base a las configuraciones de red ya existentes. En este caso sólo se recomienda el uso de al menos un firewall.

3.4. PRUEBA Y EVALUACIÓN DEL PROTOTIPO

Una vez concluido el diseño, se construyó el prototipo (implementado en http://betatesting.com.ar) el cual ha sido evaluado en su funcionalidad y comportamiento.

Se han aplicado distintos tipos de pruebas a las distintas aplicaciones desarrolladas. En los casos donde se detectaron fallos o errores se realizaron las modificaciones y correcciones pertinentes hasta obtener los resultados esperados.

En el Anexo III se presentan algunas de las pruebas efectuadas al prototipo y los resultados obtenidos.

CAPÍTULO IV
EVALUACIÓN DEL SISTEMA PROPUESTO

En este capítulo se realiza una validación de tipo informal del sistema desarrollado con relación a las características propias de un sistema de gestión del conocimiento.

Se toma de referencia el modelo de producción/conversión del conocimiento propuesto por Nonaka-Tackeuchi (descripto en el apartado 2.1.5 del capítulo 2) cuyas etapas se representan en cada cuadrante de la figura IV.1, y en donde se indican las aplicaciones que se implementan y que actuan como *habilitadoras* para cada uno de los procesos de conversión del conocimiento: socialización, externalización, combinación e internalización. Ademas se tienen en cuenta las actividades incluidas en el proceso genérico de gestión del conocimiento explicadas en el apartado 2.1.4, capitulo 2.

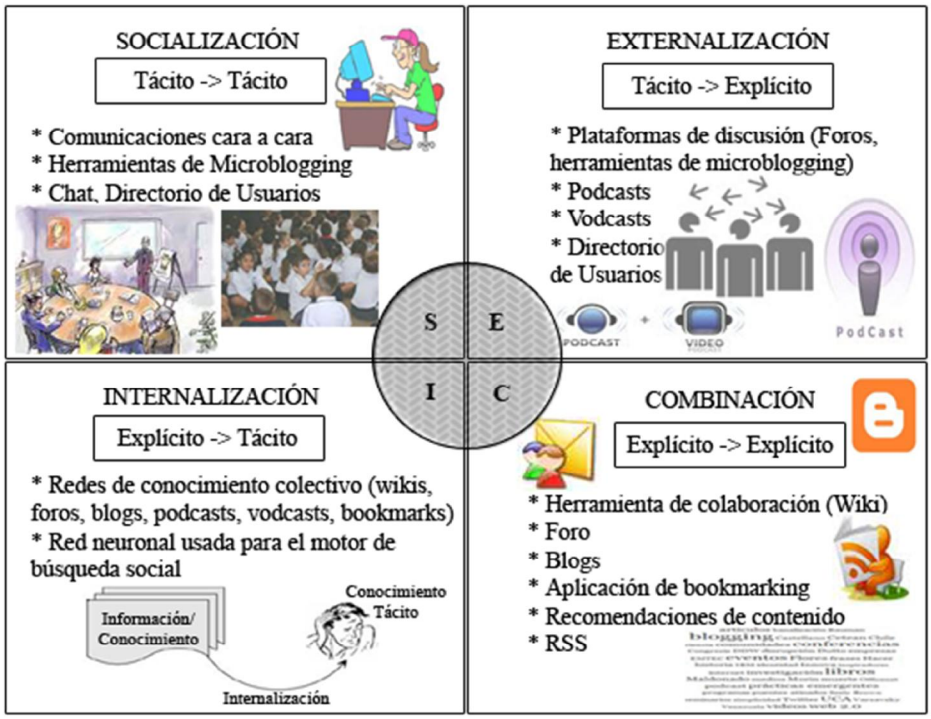


Figura IV.1. Aplicaciones sociales ubicadas en el ciclo del proceso de gestión de conocimiento

La validación realizada consistió en definir la trazabilidad correspondiente entre los principales elementos del modelo desarrollado: las **etapas del proceso de conversión del conocimiento**, las **actividades del proceso de gestión del conocimiento**, las **herramientas o aplicaciones tecnológicas** que dan el soporte correspondiente y las **actividades académicas** en el ambito universitario como instancias del proceso.

Se aclara que es difícil determinar una correspondencia estrictamente líneal entre los elementos del sistema, debido a que por ejemplo, en algunos casos los recursos tecnológicos pueden servir en los diferentes procesos, sin embargo se define una trazabilidad lo mas aproximada posible con el fin de visualizar los aspectos clave de la gestión del conocimiento.

ETAPA DE CONVERSIÓN 1: **Socialización**, en este proceso se produce la conversión de conocimiento *tácito a tácito*, y comunmente se corresponden con las comunicaciones de tipo informal, como un medio para promover la actividad social.

Las actividades académicas que se presentan en el ámbito de la universidad, donde se maneja el conocimiento tácito de tipo académico son por ejemplo:

- El desarrollo de clases teóricas-prácticas de las asignaturas;
- Trabajo en grupos o equipos entre los alumnos;
- Atención de consultas sobre diferentes asignaturas entre alumno-docente, alumno-alumno;
- Desarrollo de seminarios, talleres, congresos, jornadas internas;
- Atención de consultas para llevar a cabo un determinado trámite (por ejemplo inscripción en carreras, asignaturas; solicitud de reválidas, etc.);
- Desarrollo de planes de estudio, trabajos prácticos, recursos educativos, etc.
- Consultas sobre aspectos de gestión académica de alumnos y docentes, por ejemplo calendario académico, correlatividades de asignaturas, etc.

Para dar soporte tecnológico a estas actividades el sistema proporciona distintas aplicaciones: si bien la modalidad de enseñanza en esta universidad es presencial, las clases tanto teóricas como prácticas pueden incorporar alternativas tecnológicas para socializar el *producto* de la clase, a través del **muro** o del **twitter.** Además el muro en el directorio de usuario facilita compartir y socializar diversas informaciones o noticias, por ejemplo, cuestiones relacionadas a trámites académicos. Para apoyar el trabajo

grupal entre alumnos o la atención de consultas entre alumno-docente, los usuarios cuentan con un **servicio de chat y twitter** desde el cual la aplicación consume sus últimos posteos y los **comentarios** que se encuentran en los muros de otros usuarios.

ETAPA DE CONVERSIÓN 2: **Externalización**, etapa donde el conocimiento *tácito se convierte en explícito*, por ejemplo son instancias que se presentan en los casos de:

- Los trabajos prácticos que se desarrollan para las asignaturas;
- Los resultados de los trabajos de investigación (por ejemplo, papers, tesis, posters, entre otros);
- Las notas de clases;
- Los diferentes documentos en donde se plasman los procedimientos de gestión académica de la universidad;
- Los comunicados que se realizan en los transparentes y las distintas publicaciones en boletines etc.

Para cubrir estas necesidades se disponen de distintas herramientas: **blogs** y **wiki**s donde se *contienen* distintas publicaciones, artículos, etc.; el **directorio de usuarios**, que posibilita el conocimiento sobre intereses e ideas de otros usuarios, el **directorio de audio y video** facilita la registración de una clase que se puede documentar con el video; el **sistema para cátedras** que permite externalizar y publicar los trabajos prácticos, notas de clase; y el **portal** en el cual se pueden publicar distintas informaciones. A través de estas herramientas, se proporciona soporte necesario para la externalización, codificar y compartir conocimiento. Cualquiera de estas herramientas combinadas con RSS facilitarían el acceso a información – conocimiento externalizado en las distintas aplicaciones (blog, wikis, portal).

También se externalizan las respuestas obtenidas de las consultas que quedan registradas en los foros implementados.

Adicionalmente el uso de estas herramientas, por ejemplo, los foros, wikis y blogs, contribuirían a que los estudiantes mejoren los modos de comunicación y colaboración.

ETAPA DE CONVERSIÓN 3: **Combinación**, es la etapa en la cual se produce nuevo conocimiento a partir de la combinación de *conocimientos explícitos*. Algunos ejemplos de actividades que se relacionan con la combinación de conocimientos son:

- Desarrollo de nuevos trabajos de investigación cuando se tienen en cuenta los resultados de investigaciones ya realizadas;
- Desarrollo de nuevos contenidos a partir de la combinación de contenidos de otros materiales (libros, apuntes, entradas de blogs, entre otros);
- En la organización y creación de planes de estudio teniendo en cuenta planes o modelos de años anteriores;
- En la definición u optimización de procedimientos académicos en base a modelos existentes en otras áreas, etcétera.

Las tecnologías que soportan las actividades de los procesos de socialización y externalización también sirven para dar soporte a este proceso de combinación, puesto que los usuarios pueden utilizar la información y los conocimientos disponibles en la **wiki**, el **foro**, el **directorio de usuarios**, el **sistema de cátedras**, el **portal**, los **RSS** y **contenidos recomendados** para formar nuevas unidades de conocimiento.

ETAPA DE CONVERSIÓN 4: **Internalización**, proceso donde el conocimiento *explícito de convierte nuevamente en tácito*. Los usuarios al interactuar con el sistema potencialmente irán configurando y enriqueciendo su conocimiento tácito mediante la asimilación de los conocimientos que van consiguiendo de las múltiples fuentes que dispone el sistema. De esta manera, los usuarios irán adquiriendo conceptos, ideas y conocimientos en los procesos de socialización, enriqueciéndose con las aportaciones de otras personas en los procesos de externalización llevados a cabo, por cuanto el esfuerzo de explicitar lo que sabemos en informes, esquemas o palabras modifica y refuerza nuestro propio conocimiento, y en los procesos de combinación, en la medida en que estamos involucrados y verificando el conocimiento explícito resultante.

Como ejemplo de internalización en el plano académico se puede mencionar:

- La comprensión de un trabajo de investigación, un apunte teórico, o cualquier otro recurso educativo disponible;
- Al hacer propias las reglas que rigen el desenvolvimiento en la universidad ya sea como docente o alumno;
- Al adquirir los conceptos impartidos en una clase, seminario o curso;

- Cuando se estudia para los parciales y finales, a partir de la información/conocimiento explicitada en las distintas aplicaciones que integran el modelo.

La tecnología que dará soporte a este proceso son en general todas las herramientas implementadas en este sistema que permiten externalizar conocimiento y distribuirlo, tal el caso de los wiki, blogs chat, los sistemas de cátedra, directorio de audio y video, etc., a los cuales el usuario podrá acceder para obtener información y conocimiento que hayan sido externalizados.

Con respecto a las **actividades del proceso de gestión del conocimiento**: descubrimiento, captura, clasificación y almacenamiento, distribución y diseminación, compartimiento y colaboración, y utilización; las herramientas proporcionadas en el sistema tienen el potencial de poder dar el soporte requerido por las mismas. En la figura IV.2 se representa la correspodencia entre actividades de la gestión de conocimiento y soporte tecnológico.

Teniendo en cuenta la cantidad potencial de usuarios que harían uso del sistema y de la cantidad de contenido que fluiría, los *mecanismos de recomendación*, de *análisis de clustering* y *búsqueda inteligente*, implementados en el sistema, resultan esenciales para contribuir al **descubrimiento** de conocimiento por parte del usuario. Otro aspecto que ayuda de manera indirecta en este sentido son las *"nubes de palabras"* generadas como consecuencia de la actividad de clasificación que los mismos usuarios realizan.

La **captura** de unidades de conocimiento se habilita a través de los distintos mecanismos ofrecidos en el sistema: wiki, foro, directorio de usuario, sistema de cátedra, directorio de audio y video. Con estas herramientas los usuarios pueden externalizar cualquier tipo de conocimiento y/o información que se necesite registrar y compartir.

Con respecto a la **clasificación**, se adoptó como estrategia el uso de "*tags*" que es un sistema de clasificación social debido que los mismos usuarios son los que crean las categorías necesarias contra las cuales harán corresponder las unidades de conocimientos que deseen registrar. Esta estrategia de clasificación colaborativa permite que la clasificación sea realizada por el mismo sistema en base a la contribución de los usuarios, sin necesidad de contar con un administrador de indices.

El **almacenamiento** de las unidades de conocimiento se realiza en las bases de datos correspondientes a cada subsistema que integra la solución planteada.

La **distribución** y **diseminación** de conocimiento se logra a través de los mecanismos de recomendación y el empleo de la tecnología RSS, habilitando con este último que cualquier usuario que se encuentre interesado pueda consumir información del sistema. Una gran ventaja del empleo de RSS para facilitar la diseminación de la información es su baja demanda de ancho de banda de la red.

El **compartimiento** es algo que ocurre de manera implícita puesto que los usuarios que deciden publicar contenido/información/conocimiento en el sistema lo hacen con intenciones de compartirlo.

En tanto que, el **uso** de la información/conocimiento existente en el sistema está abierto para cualquiera de los usuarios e indudablemente será utilizada cuando estos la necesiten.

Es importante recalcar que el sistema permite y soporta comunicaciones "ricas" ya que dispone de medios a través de los cuales el contexto, significados y opiniones puedan circular; se adhiere a estándares industriales (MVC, XML, RSS); es integrativo e interactivo; da soporte a la colaboración y existe la posibilidad de que tenga un buen recibimiento por parte de la comunidad usuaria por las características sociales que presenta.

De esta manera, queda reflejado que el sistema propuesto tiene el potencial de servir como un habilitador para los procesos de gestión del conocimiento en el ámbito de la Educación Superior.

La tabla IV-1 contiene una síntesis de la trazabilidad dada entre los distintos elementos mencionados anteriormente. Se aclara una vez más que las actividades de gestión del conocimiento se pueden establecer en cualquiera de las etapas de conversión de conocimiento por tal razón no se explícitan en la tabla.

Tabla IV-1. Trazabilidad entre etapas de Conversión del Conocimiento, Actividades académicas instanciadas y aplicaciones sociales que dan el soporte tecnológico.

ETAPA DE CONVERSIÓN DE CONOCIMIENTO	INSTANCIAS DE TAREAS ACADÉMICAS	SOPORTE TECNOLÓGICO/ APLICACIÓN
Socialización	~ Aprendizaje de procesos internos relativos a la vida universitaria ~ Clases teóricas prácticas ~ Desarrollo de trabajos en equipo; ~ Seminarios, talleres, congresos ~ Planificación y desarrollo de planes de estudio, trabajos prácticos, recursos educativos, etc.	~ Muro ~ Chat ~ Twitter ~ Sistema de Audio y Video ~ Comentarios
Externalización	~ Desarrollo de trabajos prácticos ~ Presentación de los resultados de las investigaciones (papers, tesinas, posters, etc.) ~ Apuntes teóricos ~ Documentos sobre procsos administrativos ~ Comunicados realizados a través de los transparentes ~ Otros	~ Foro ~ Blogs ~ Wiki ~ Directorio de usuarios ~ Directorio de audio y video ~ Sistema para cátedras ~ Portal
Combinación	~ Producción de trabajos de investigación (utilización de otras fuentes) ~ Creación de planes de estudio (tomando como referencia otros existentes) ~ Definición de procesos administrativos en base a existentes en otras áreas ~ Otros	~ Wiki ~ Foro ~ Directorio de usuarios ~ Sistema de cátedras ~ Portal ~ RSS ~ Recomendación de contenidos

Tabla IV-1. Trazabilidad entre etapas de Conversión del Conocimiento, Actividades académicas instanciadas y aplicaciones sociales que dan el soporte tecnológico (Cont.).

ETAPA DE CONVERSIÓN DE CONOCIMIENTO	INSTANCIAS DE TAREAS ACADÉMICAS	SOPORTE TECNOLÓGICO/ APLICACIÓN
Internalización	~ Comprensión de un trabajo de investigación, un apunte teórico, o cualquier otro recurso disponible ~ Interiorización de las reglas que rigen el desenvolvimiento en la universidad ya sea como docente o alumno ~ Comprensión de los coneptos impartidos en una clase, seminario o curso ~ Procesos de estudio para los parciales y finales ~ Entre otros.	~ Muro ~ Chat ~ Twitter ~ Sistema de Audio y Video ~ Comentarios ~ Wiki ~ Foro ~ Directorio de usuarios ~ Sistema de cátedras ~ Portal ~ RSS ~ Recomendación de contenidos

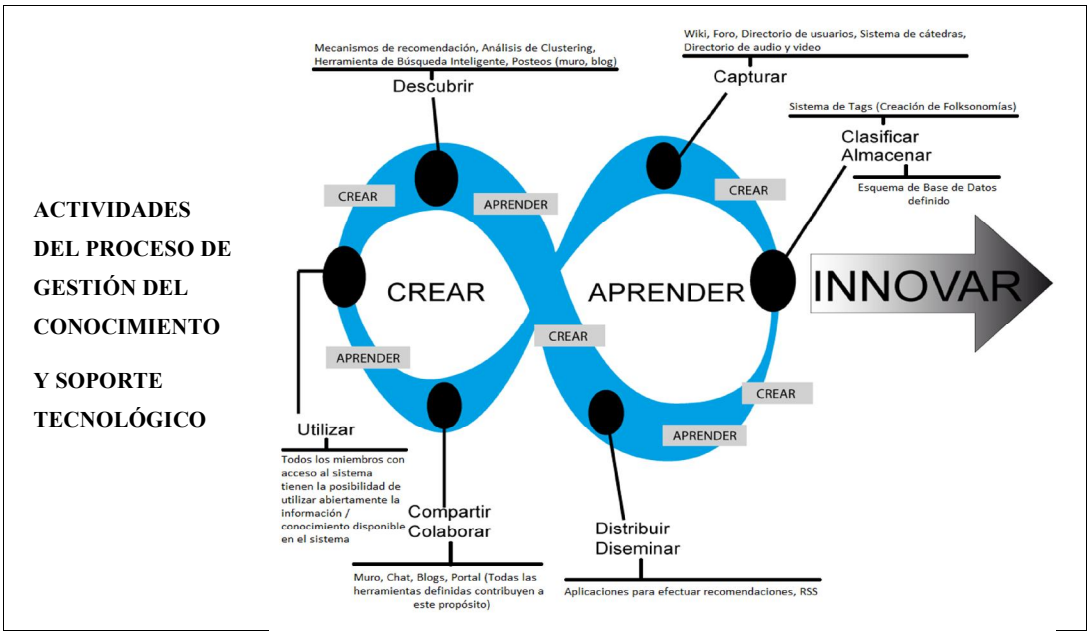


Figura IV-2. Correspondencia entre actividades de la gestión de conocimiento y aplicaciones sociales que dan el soporte tecnológico

CAPÍTULO V
CONCLUSIONES Y TRABAJO FUTURO

Las conclusiones arribadas en el presente trabajo pueden ser desarrolladas en distintos aspectos:

- ✓ *Con respecto al trabajo global*: en general, la realización del trabajo permitió el logro de los objetivos planteados en la propuesta, ya que principalmente se obtiene un sistema web para el ámbito de la educación superior con el cual es posible dar algunas respuestas a los principales desafíos que actualmente plantea la sociedad del conocimiento, relacionados principalmente con la aplicación de enfoques innovadores de creación y transmisión de conocimientos en la comunidad educativa de la UNSE.

- ✓ *En relación al análisis y diseño del sistema*: en el desarrollo de esta etapa se efectuó un análisis del contexto con el fin de encontrar y lograr una mejor comprensión de las relaciones entre la gestión del conocimiento, las aplicaciones sociales y su implementación en la educación superior. Esto permitió posteriomente el diseño del sistema web con características 2.0, y la selección de las aplicaciones tecnológicas a implementarse para que den un soporte adecuado a los procesos y actividades de gestión del conocimiento, y responder a los requisitos y necesidades analizadas previamente. Como resultado se obtiene un modelo de sistema organizado en capas, lo cual permitió reducir la complejidad que implica su construcción y facilitar la mantenibilidad futura del mismo. Finalmente se obtuvieron los productos entregables que representan la visión lógica, estructural y funcional del sistema, tales como el modelo de clases, modelo de entidad-relación de datos y modelo de flujo de datos, lo que implica contar con una documentación completa para facilitar cambios ante nuevos requerimientos.

✓ *Desde el punto de vista de la construcción del prototipo*: la construcción ha abarcado las principales funciones del sistema mediante la codificación de las aplicaciones: *portal* y la *wiki* orientados a poder difundir novedades, noticias y recursos; *directorio de usuarios*, proporciona un catálogo en línea de los recursos humanos disponibles en la Universidad e información relacionada a los mismos desde sus datos académicos hasta sus intereses; *chat, correos,* brindan la posibilidad de generar comunicaciones informales; *foro*, medio a través del cual es posible manifestar dudas, inquietudes; *clustering para las* recomendaciones de contenidos, de usuarios, clasificación de los mismos por intereses; y una aplicación para efectuar búsquedas "inteligentes" mediante una red neuronal sencilla.

Una vez finalizado el prototipo, ha sido evaluado para comprobar que el sistema efectivamente brinda soporte a los procesos y actividades propias de la gestión del conocimiento.

En general se infiere que la herramienta global permite:

- Disponer de nuevos formatos para la adquisición, diseminación, y administración del conocimiento producido en la institución.
- Mejorar la accesibilidad y disponibilidad de los recursos y conocimientos generados a partir de las actividades académicas.
- Potenciar en cierta medida los procesos de colaboración entre docentes y alumnos.
- Disponer de un ambiente de intercambio *atractivo* de distintos tipos de información y conocimiento provenientes de la función de docencia e investigación, al soportar modos personalizados de recuperar, administrar y transformar la información.

✓ *Desde la perspectiva de los autores del trabajo*: la investigación de tipo exploratoria permitió la comprensión de los diferentes aspectos vinculados a la aplicación práctica de conceptos de gestión del conocimiento. El desarrollo de un caso práctico implicó por un lado el afianzamiento de los conceptos investigados y por otro, la aplicación de los conocimientos relacionados a la programación de aplicaciones web, tales como manejo de servicios web, XML, RSS, PHP, el modelo MVC, MySQL, CSS, HTML, entre otras tecnologías propias de la industria del software. Específicamente, con el desarrollo e implementación de una red neuronal, un sistema de clustering y un

sistema de recomendaciones para completar el prototipo del sistema, consideramos haber realizado un paso más allá de lo teórico.

Finalmente como trabajo futuro queda planteado avanzar en el desarrollo del sistema completo, optimizar los algoritmos de la capa de inteligencia colectiva a fin de proporcionar mejores prestaciones en cuanto a rendimiento y eficiencia; mejorar la integración con las redes sociales (es decir, incorporar nuevas funcionalidades que permitan una mayor interacción entre el sistema propuesto y aplicaciones como Twitter, Facebook y Delicious) y, refinar aspectos del sistema relacionados fundamentalmente con la seguridad como ser *falsas solicitudes*, *inyecciones*, *ataques de denegación de servicios* y *seguridad en passwords*.

REFERENCIAS

[1] **ALAVI, M.; LEIDNER, D. E.** *Knowledge Management Systems: issues, challenges, and benefits.* Communications of the AIS, 1 (2es), 1999

[2] A**L ZABIR, O.** *Building a 2.0 Portal with ASP.NET 3.5.* O'Reilly. Estados Unidos (2008)

[3] **ANDERSON, P.***What is Web 2.0? Ideas, technologies and implications for education.*JISC Technology and Standards Watch. 2-64. 2007

[4] **ARAUJO, B.***Aprendizaje Automático: Conceptos básicos y avanzados. Aspectos prácticos utilizando el software WEKA*. Madrid: Prentice Hall, 2006

[5] **BRAWN, J.S.; DUGUID, P.** *Organising Knowledge*. California Management Review. 40(3), 90-111. 1998

[6] **CARR, N**. *Is Web 2.0 Enterprise Ready?.*Rough Type, 2006. Disponible en: http://www.roughtype.com/archives/2006/04/is_web_20_enter.php (Consultado marzo 2010)

[7] **DOCTOROW, C.** *Essential Blogging.* O'Reilly Media Inc. Beijing, 2002

[8] **DUFFY, P.; BRUNS, A.** *The Use of Blogs, Wikis and RSS in Education: A Conversation of Possibilities.*Proceedings Online Learning and Teaching Conference 2006. Brisbane, pp 31-38. 2006

[9] **EDUCATION, I.T.H.** *The Implementation of Knowledge Management System in Taiwan's Higher Education.*Journal of College Teaching & Learning.Septiembre 2(5). 2005

[10] **FRANKLIN, T.; HARMELEN, M.** *Web 2.0 for Content for Learning and Teaching in Higher Education*. Innovation in the use of ICT for Education and Research.2007. Disponible en: http://www.jisc.ac.uk/publications/reports/2007/web2andpolicyreport.aspx. (Consultado marzo 2010)

[11] **JONES, B.** *Web 2.0 Heroes: Interviews with 2.0 Influencers.* Indiana: Wiley Publishing, 2008

[12] **KIDWELL, J.; VANDER LINDE, K.; JOHNSON, S.** *Applying Corporate Knowledge Management Practices in Higher Education.* Educause Quarterly, 23(4), 28-33. 2000

[13] **MATHES, A.** *Folksonomies – Cooperative Classification and Communication through Shared Metadata.* 2004

[14] **McAFEE, A.** *Enterprise 2.0: The Dawn of Emergent Collaboration.* MIT Sloan Management Review, 2006

[15] **MediaWiki.** Página principal de la empresa desarrolladora de la plataforma de MediaWiki. URL: http://www.mediawiki.org. (Consultado marzo 2010)

[16] **MONTANO, Bonnie.** *Innovations of Knowledge Management.*Estados Unidos: IRM Press, 2005

[17] **MORGAN, Milagros; SULMONT, Lea.** *Un enfoque estratégico para la gestión del conocimiento en la universidad: La experiencia de la Universidad Peruana de Ciencias Aplicadas*. Revista Digital en Docencia Universitaria (RIDU), 2006. Disponible en: http://beta.upc.edu.pe/calidadeducativa/ridu/2006/ridu2_4MM_LS.pdf. (Consultado junio 2010)

[18] **NARANJO, María.** *Millenials: La generación del futuro*. La Nación, 2008. Disponible en: http://www.lanacion.com.ar/nota.asp?nota_id=1020257. (Consultado marzo 2011)

[19] **NILS, Nilsson.** *Inteligencia Artificial: Una nueva síntesis.* Editorial: Mc Graw Hill, 2001

[20] **OPTIONS, A.; LATEST, T.** *Participative Web and User-Created Content: Web 2.0, Wikis and Social Networking.* Complete Edition-ISBN 9264037462. Source OECD Science & Information Technology, 2007(15)

[21] **PHPBB.** Página principal de la empresa desarrolladora de la plataforma de PHPBB. URL: http://www.phpbb.com/. (Consultado marzo 2011)

[22] **O'REILLY, T.** *Design Patterns and Business Models for the Next Generation of Software*. O'Reilly Media, 2005. Disponible en: http://oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1 (Consultado abril 2011)

[23] **REDECKER, Christine; FERRARI, Anusca; et. al**. *Learning 2.0: The Impact of Web 2.0 Innovations on Education and Training in Europe*. JRC Scientific and Technical Reports, 2009

[24] **SAEED, N.; YANG, *Y***: *Incorporating Blogs, Social Bookmarks, and Podcasts into Unit Teaching. In Proc. Tenth Australasian Computing Education Conference (ACE 2008), Wollongong, NSW, Australia (2008).*

[25] **SARWAR, B.; KARYPIS, J.; et. al.***Item-Based Collaborative Filtering Recommendation Algorithms.* Hong Kong (2001). ACM PRESS

[26] **SCHAFFERT, S.; BISCHOF, D; BUERGER, T.; GRUBER, W.** *Learning with Semantic Wikis.*http://www.eswc2006.org/technologies/usb/proceedings-workshops/eswc2006-workshop-semantic-wikis.pdf. 2006. (Consultado mayo 2011)

[27] **SEGARAN, Toby.** *Programming Collective Intelligence: Building Smart Web 2.0 Applications.*Estados Unidos: O`REILLY, 2007.

[28] **SIMÖES, Luis; GOUVEIA, Luis.** *Web 2.0 and Higher Education: Pedagogical Implications*. Knowledge Technologies for Social Transformation, Global University Network for Innovation. Volumen 2. 2008

[29] **SOSA, Mabel; VELÁSQUEZ, Isabel; SILVA, Carmen; ZARCO, Raquel.** *Enfoque sistémico cultural de la Gestión Académica: Propuesta de cambios para promover ventajas competitivas*. IV Jornadas de Ciencia y Tecnología de las Facultades de Ingeniería del NOA. 2008.

[30] **TIWANA, Amrit.** *The Knowledge Management Toolkit: Practical Techniques for Building a Knowledge Management System. Orchestrating IT, Strategy and Knowledge Plataforms*. Segunda Edición. Estados Unidos: Prentice Hall, 2000

[31] **VALHONDO SOLANO, Domingo.** *Gestión del conocimiento: Del mito a la realidad*. Primera edición. Madrid: Diaz Santos, 2002

[32] **UNESCO Publishing.** *WorldReport: Towards Knowledge Societies*. Estados Unidos, 2005. Disponible en: http://unesdoc.unesco.org/images/0014/001418/141843e.pdf. (Consultado marzo 2011)

[33] **Universidad Abierta de Inglaterra** (Open University). Portal de información académica: http://openlearn.open.ac.uk/. (Consultado junio 2011)

[34] **Universidad de Berkeley.** Portal de información: http://webcast.berkeley.edu/

[35] **Universidad de Massachusetts.** Portal de información académica: http://ocw.mit.edu/OcwWeb/web/home/home/index.htm. (Consultado marzo 2011)

[36] **VOSS, J.** *Tagging, Folksonomy & Co-Renaissance of Manual Indexing.* URL: http://arxiv.org/abs/cs.IR/0701072. 2007. (Consultado marzo 2011)

[37] **Web Site de DevianArt.** Url: http://www.deviantart.com/

[38] **Web Site de Facebook.** Url: http://www.facebook.com/

[39] **Web Site de Flickr.** Url: http://www.flickr.com/

[40] **Web Site de iTunes.** Url: http://www.apple.com/itunes/?cid=OAS-US-DOMAINS-itunes.com

[41] **Web Site de Linkedin.** Url: http://www.linkedin.com/

[42] **Web Site de MySpace.** Url: http://www.myspace.com/

[43] **Web Site de Scribd.** Url: http://es.scribd.com/

[44] **Web Site de Second Life.** Url: http://secondlife.com/

[45] **Web Site de Slideshare.** Url: http://www.slideshare.net/

[46] **Web Site de YouTube.** Url: http://www.youtube.com

BIBLIOGRAFÍA COMPLEMENTARIA

BELL, Gavin.*Building Social Web Applications: Establishing Community at the Heart of Your Site.*Estados Unidos: O`REILLY, 2009.

CHOATE, Mark.*Proffesional Wikis.* Indiana: Wrox, 2008.

CRUMLISB, Christian; MALONE, ERIN.*Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience.* Estados Unidos: O`REILLY, 2009.

GILMORE, Jason.*Beginning PHP 5 and MySQL 5: From Novice to Professional.*Estados Unidos: Apress, 2006, 2da edición.

LIEBOWITZ, Jay.*Knowledge Retention: Strategies and Solutions.* Estados Unidos: CRC Press, 2009.

RICHARDS, Robert.*Pro PHP XML and Web Services: Master Working with XML and Web Services using PHP.*Estados Unidos: Apress, 2006.

ZERVAAS, Quentin.*Practical Web 2.0 Applications with PHP.*Estados Unidos: Apress, 2008.

ANEXO I

DOCUMENTACIÓN DE CLASES DEFINIDAS

1. CLASES DEL DIRECTORIO DE USUARIOS

Clase Usuario

Heredada por las clases *Profesor* y *Ayudante*.

Descripción
Clase usada para representar a la entidad Usuario.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del usuario.	Integer	Privado
$nombre	Nombre del usuario.	String	Privado
$apellido	Apellido del usuario.	String	Privado
$tipousuario	Identificador del tipo de usuario. $tipousuario = 1 => Alumno $tipousuario = 2 => Profesor	Integer	Privado
$intereses	Intereses manifestados por el usuario. Es una cadena en la cual los intereses vienen separados unos de otros por una coma.	String	Privado
$especialidades	Especialidades del usuario. Es una cadena en la cual las especialidades vienen separadas unas de otra por una coma.	String	Privado
$direcciones_emails	Cuentas de correo electrónico del usuario. Es una cadena en la cual cada cuenta se separa de otra por una coma.	Text	Privado
$sitios_web	Urls de sitios del usuario.	Text	Privado

	Cada url se separa de otra por una coma.		
$telefonos	Números de teléfonos del usuario. Cada número de teléfono se separa de otro por una coma.	String	Privado

Métodos

Crear ($nombre, $apellido, $tipousuario, $intereses, $especialidades, $direcciones_emails, $telefonos, $sitios_web)
Permite crear un nuevo usuario en el sistema. **Parámetros de entrada:** *string* $nombre *string* $apellido *integer* $tipousuario *string* $intereses *string* $especialidades *string* $direcciones_emails *string* $telefonos *string* $sitios_web **Salida** Usuario Objeto de tipo *Usuario*

Recuperar ($usuario_id)
Recupera los datos de un determinado usuario. **Parámetros de entrada:** *integer* $usuario_id **Salida** Usuario Objeto de tipo *Usuario*

RecuperarPorCuentaUsuario ($id_cuenta)
Recupera los datos de un determinado usuario en base al identificador de su cuenta. **Parámetros de entrada:** *integer* $id_cuenta **Salida** Usuario Objeto de tipo *Usuario*

ActualizarAvatar ($usuario_id, $foto_perfil)
Actualiza la imagen de perfil de un determinado usuario. **Parámetros de entrada:** *integer* $usuario_id *string* $foto_perfil **Salida** True | False

Actualizar ($nombre, $apellido, $tipousuario, $usuario_id)
Permite actualizar el nombre, apellido y tipo de usuario de un determinado usuario. **Parámetros de entrada:** *string* $nombre

string $apellido
integer $tipousuario
integer $usuario_id
Salida
True | False

ActualizarIntereses ($intereses)

Método de instancia que permite actualizar los intereses de un determinado usuario.
Parámetros de entrada:
string[] $intereses
Salida
True | False

ActualizarEspecialidades ($intereses)

Método de instancia que permite actualizar las especialidades de un determinado usuario.
Parámetros de entrada:
s*tring[]* $especialidaes
Salida
True | False

ActualizarEmails ($emails)

Método de instancia que permite actualizar las direcciones de correo de un determinado usuario.
Parámetros de entrada:
s*tring[]* $emails
Salida
True | False

ActualizarSitiosWeb ($sitios_web)

Método de instancia que permite actualizar las direcciones de páginas web de un determinado usuario.
Parámetros de entrada:
s*tring[]* $sitios_web
Salida
True | False

ActualizarTelefonos ($telefonos)

Método de instancia que permite actualizar los números de teléfono de un determinado usuario.
Parámetros de entrada:
s*tring[]* $telefonos
Salida
True | False

Eliminar ($usuario_id)

Elimina del sistema un determinado usuario y toda la información asociada al mismo.
Parámetros de entrada:

integer$usuario_id
Salida
True | False

ExisteUsuario ($usuario_id)
Verifica si existe el usuario con identificador igual al valor del parámetro que recibe.
Parámetros de entrada:
integer $usuario_id
Salida
True | False

RecuperarCuentaUsuario ($id_cuenta)
Recupera la cuenta de usuario en base al identificador que recibe como parámetro.
Parámetros de entrada:
integer $id_cuenta
Salida
CuentaUsuario Objeto de tipo *CuentaUsuario*

RecuperarTodos ()
Recupera todos los usuarios disponibles en el sistema.
Parámetros de entrada:
Ninguno
Salida
Usuario[]

RecuperarUsuariosPorIdFacultad ($id_facultad)
Recupera los usuarios que pertenecen a una determinada facultad.
Parámetros de entrada:
integer $id_facultad
Salida
Usuario[]

RecuperarUsuariosIdCarrera ($id_carrera)
Recupera los usuarios que pertenecen a una determinada carrera.
Parámetros de entrada:
integer $id_carrera
Salida
Usuario[]

RecuperarIntereses ($usuario_id)
Recupera los intereses señalados por un determinado usuario.
Parámetros de entrada:
integer $usuario_id
Salida
string[]

RecuperarCorreos ($usuario_id)
Recupera las cuentas de correo de un determinado usuario.

Parámetros de entrada:
integer $usuario_id
Salida
string[]

RecuperarTelefonos ($usuario_id)
Recupera los números de teléfonos de un determinado usuario.
Parámetros de entrada:
integer $usuario_id
Salida
string[]

RecuperarSitiosWeb ($usuario_id)
Recupera los sitios web de un determinado usuario.
Parámetros de entrada:
integer $usuario_id
Salida
string[]

RecuperarEspecialidades ($usuario_id)
Recupera las especialidades señalados por un determinado usuario.
Parámetros de entrada:
integer $usuario_id
Salida
string[]

Clase AntecedenteLaboral

Descripción
Clase usada para representar a la entidad Antecedente Laboral.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del antecedente laboral.	Integer	Privado
$empresa	Nombre de la empresa.	String	Privado
$cargo	Cargo que el usuario desempeña en la empresa.	String	Privado
$descripcion	Descripción del antecedente laboral.	Text	Privado
$year_inicio	Año en que inicio la experiencia.	Integer	Privado
$year_fin	Año en que finaliza la experiencia.	Integer	Privado
$usuario_id	Identificador único del usuario.	Integer	Privado

Métodos

Crear ($empresa, $cargo, $descripcion, $year_inicio, $year_fin, $usuario_id)
Crea un antecedente laboral para un determinado usuario.
Parámetros de entrada:

string $empresa
string $cargo
text$descripcion
integer $year_inicio
integer $year_fin
integer $usuario_id
Salida
AntecedenteLaboral Objeto de tipo *AntecedenteLaboral*

Recuperar ($id_antecedente)

Recupera un determinado antecedente laboral.
Parámetros de entrada:
integer $id_antecedente
Salida
AntecedenteLaboral Objeto de tipo *AntecedenteLaboral*

RecuperarAntecedentesUsuario ($id_usuario)

Recupera los antecedentes laborales de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
AntecedenteLaboral[]

Actualizar ($id_antecedente, $empresa, $cargo, $descripcion, $year_inicio, $year_fin)

Actualiza los datos de un determinado antecedente laboral.
Parámetros de entrada:
integer $id_antecedente
string $empresa
string $cargo
text $descripcion
integer $year_inicio
integer $year_fin
Salida
True | False

Eliminar ($id_antecedente)

Elimina un determinado antecedente laboral.
Parámetros de entrada:
integer $id_antecedente
Salida
True | False

EliminarAntecedentesUsuario ($id_usuario)

Elimina los antecedentes laborales de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
True | False

Clase AntecedenteInvestigacion

Descripción
Clase usada para representar a la entidad Antecedente de Investigación.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del antecedente de investigación.	Integer	Privado
$institucion	Nombre de la institución.	String	Privado
$descripcion	Descripción del antecedente de investigación.	Text	Privado
$year_inicio	Año en que inicio la experiencia.	Integer	Privado
$year_fin	Año en que finaliza la experiencia.	Integer	Privado
$usuario_id	Identificador único del usuario.	Integer	Privado

Métodos

Crear ($institucion, $descripcion, $year_inicio, $year_fin, $usuario_id)

Crea un antecedente de investigación para un determinado usuario.
Parámetros de entrada:
string $institucion
text $descripcion
integer $year_inicio
integer $year_fin
integer $usuario_id
Salida
AntecedenteInvestigacion Objeto de tipo *AntecedenteInvestigacion*

Recuperar ($id_antecedente)

Recupera un determinado antecedente de investigación.
Parámetros de entrada:
integer $id_antecedente
Salida
AntecedenteInvestigacion Objeto de tipo *AntecedenteInvestigacion*

RecuperarAntecedentesUsuario ($id_usuario)

Recupera los antecedentes de investigación de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
AntecedenteInvestigacion[]

Actualizar ($id_antecedente, $institucion, $descripcion, $year_inicio, $year_fin)

Actualiza los datos de un determinado antecedente de investigación.

Parámetros de entrada:
integer $id_antecedente
string $institucion
text $descripcion
integer $year_inicio
integer $year_fin
Salida
True | False

Eliminar ($id_antecedente)

Elimina un determinado antecedente de investigación.
Parámetros de entrada:
integer $id_antecedente
Salida
True | False

EliminarAntecedentesUsuario ($id_usuario)

Elimina los antecedentes de investigación de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
True | False

Clase AntecedenteEducativo

Descripción
Clase usada para representar a la entidad Antecedente Educativo.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del antecedente educativo.	Integer	Privado
$institucion	Nombre de la institución.	String	Privado
$titulo	Titulo otorgado por la institución.		
$year_inicio	Año en que inicio la experiencia.	Integer	Privado
$year_fin	Año en que finaliza la experiencia.	Integer	Privado
$finalizo	Variable booleana que indica si el usuario finalizo el antecedente educativo que señala.	Boolean	Privado
$usuario_id	Identificador único del usuario.	Integer	Privado

Métodos

Crear ($institucion, $titulo, $year_inicio, $year_fin, $finalizo, $usuario_id)

Crea un antecedente educativo para un determinado usuario.
Parámetros de entrada:

string $institucion
string $titulo
integer $year_inicio
integer $year_fin
boolean $finalizo
integer $usuario_id
Salida
AntecedenteEducativo Objeto de tipo *AntecedenteEducativo*

Recuperar ($id_antecedente)
Recupera un determinado antecedente educativo.
Parámetros de entrada:
integer $id_antecedente
Salida
AntecedenteEducativo Objeto de tipo *AntecedenteEducativo*

RecuperarAntecedentesUsuario ($id_usuario)
Recupera los antecedentes educativos de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
AntecedenteEducativo[]

Actualizar ($id_antecedente, $institucion, $titulo, $year_inicio, $year_fin, $finalizo)
Actualiza los datos de un determinado antecedente educativo.
Parámetros de entrada:
integer $id_antecedente
string $institucion
string $titulo
integer $year_inicio
integer $year_fin
boolean $finalizo
Salida
True | False

Eliminar ($id_antecedente)
Elimina un determinado antecedente educativo.
Parámetros de entrada:
integer $id_antecedente
Salida
True | False

EliminarAntecedentesUsuario ($id_usuario)
Elimina los antecedentes educativos de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
True | False

Clase Amigo

Descripción
Clase usada para representar a la entidad Amigo.

Atributos

Atributo	Comentario	Tipo	Alcance
$id_amigo	Identificador único del usuario que representa al amigo.	Integer	Privado
$favorito	Indica si el amigo es marcado como favorito para el usuario de quien es amigo.	Boolean	Privado
$usuario_id	Identificador único del usuario quien señala la relación de amistad.	Integer	Privado

Métodos

Crear ($id_amigo, $favorito, $id_usuario)

Asocia el usuario con identificador "id_amigo" como amigo del usuario con identificador "id_usuario".
Parámetros de entrada:
integer $id_amigo
boolean $favorito
integer $id_usuario
Salida
integerIndica si se pudo establecer o no la relación. 1 en caso afirmativo, 0 en caso contrario.

Recuperar ($id_amigo)

Recupera los datos de un determinado amigo de un usuario.
Parámetros de entrada:
integer $id_amigo
Salida
Amigo.

RecuperarAmigosUsuario ($id_usuario)

Recupera los amigos de un determinado usuario.
Parámetros de entrada:
integer $id_usuario
Salida
Amigo[]

RecuperarAmigosPorNombre ($consulta, $id_usuario)

Recupera los amigos de un determinado usuario. Realiza la búsqueda por nombre. El nombre del usuario viene en el parámetro $consulta e internamente se realiza una búsqueda aproximada.
Parámetros de entrada:
string $consulta

integer $id_usuario
Salida
Usuario[]

Actualizar ($id_registro, $id_amigo, $favorito)
Actualiza los datos relacionados a un determinado amigo de un usuario.
Parámetros de entrada:
integer $id_registro Identificador del registro que almacena la información de relación de amistad de un usuario con otro.
integer $id_amigo
boolean $favorito
Salida
True | False

Eliminar ($id_amigo)
Elimina un amigo de un determinado usuario.
Parámetros de entrada:
integer $id_amigo
Salida
True | False

ExisteAmigo ($id_amigo, $id_usuario)
Verifica si el usuario con identificador "id_amigo" es amigo del usuario con identificador "id_usuario".
Parámetros de entrada:
integer $id_amigo
integer $id_usuario
Salida
True | False

Clase Contenido

Descripción

Clase usada para representar los contenidos subidos por un usuario. Por contenido se hace referencia a publicaciones en Twitter, Delicious, Wordpress, Blogger y posteos efectuados en el muro del directorio de usuarios.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del contenido.	Integer	Privado
$descripcion	Contenido de la publicación.	Text	Privado
$fecha_publicacion	Fecha en que el contenido fue publicado.	Datetime	Privado
$procedencia	Indica el tipo de contenido.	Integer	Privado
$usuario_id	Identificador del usuario que publico el contenido.	Integer	Privado
$votacion_promedio	Cantidad de votos	Double	Privado

	promedio que recibió el contenido.		
$cantidad_votos_positivos	Cantidad de votos "Me gusta" que recibió el contenido.	Integer	Privado
$cantidad_votos_negativos	Cantidad de votos "No me gusta" que recibió el contenido.	Integer	Privado
$tags	Tags asociados al contenido.	String[]	Privado

Métodos

Crear ($descripcion, $fecha_publicacion, $procedencia, $id_usuario)

Crea un contenido asociado a un determinado usuario.
Parámetros de entrada:
string $descripcion
datetime$fecha_publicacion
integer $procedencia
integer $id_usuario
Salida
integer Indica si se pudo efectuar el alta del contenido. 1 en caso afirmativo, 0 en caso contrario.

Recuperar ($condiciones)

Recupera los contenidos que cumplan las condiciones indicadas por el parámetro de entrada.
Parámetros de entrada:
string[] $condiciones Cada elemento del arreglo es una condición que sirve al procedimiento para efectuar las filtraciones.
Salida
Contenido[]

Eliminar ($condiciones)

Elimina los contenidos que cumplan las condiciones indicadas por el parámetro de entrada.
Parámetros de entrada:
string[] $condiciones Cada elemento del arreglo es una condición que sirve al procedimiento para armar la condición que indica los contenidos que tendrán que ser eliminados.
Salida
True | False

CalificarContenido ($id_contenido, $id_usuario, $votos)

Asigna la cantidad de votos indicado por $votos al contenido cuyo identificador es $id_contenido.
Parámetros de entrada:
integer $id_contenido
integer $id_usuario
integer $votos
Salida

True | False

IndicarPreferencia ($id_contenido, $id_usuario, $gusta)

Permite establecer la preferencia de un determinado con respecto a un contenido.

Parámetros de entrada:

integer $id_contenido

integer $id_usuario

tinyint $gusta $gusta = 1 => Establece un voto positivo; $gusta = 2 => Establece un voto negativo

Salida

True | False

Comentar ($id_contenido, $id_usuario, $comentario)

Asigna un comentario a un contenido por parte de un determinado usuario.

Parámetros de entrada:

integer $id_contenido

integer $id_usuario

text $comentario

Salida

True | False

AsignarTag ($id_contenido, $id_usuario, $tag)

Asigna un tag a un contenido por parte de un determinado usuario.

Parámetros de entrada:

integer $id_contenido

integer $id_usuario

string $tag

Salida

True | False

EliminarTag ($id_contenido, $id_usuario, $tag)

Elimina un determinado tag asociado a un contenido por parte de un usuario.

Parámetros de entrada:

integer $id_contenido

integer $id_usuario

string $tag

Salida

True | False

RecuperarComentarios ($id_contenido)

Recupera los comentarios realizados a un determinado contenido.

Parámetros de entrada:

integer $id_contenido

Salida

Text[]

RecuperarTags ($id_contenido, $id_usuario, $tag)

Recupera los tags asociados a un determinado contenido.

Parámetros de entrada:
integer $id_contenido
Salida
string[]

RecuperarTodosPorFiltro ($arreglo_parametros)
Recupera los contenidos que cumplan las condiciones de filtro establecidas por el arreglo de parámetros que recibe como entrada. **Parámetros de entrada:** *string[]* $arreglo_parametros Arreglo en el cual cada elemento del arreglo es una condición que es usado para definir la cláusula *where* de la consulta SQL. **Salida** Contenido[]

RecuperarTodosTagsPorFiltro ($tag)
Recupera todos los contenidos a los cuales se les haya aplicado un determinado tag. **Parámetros de entrada:** *string* $tag **Salida** Contenido[]

Clase CuentaUsuario

Descripción
Clase usada para representar y gestionar la Cuenta de un determinado usuario.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único dela cuenta de usuario.	Integer	Privado
$user	Nombre de usuario	String	Privado
$password	Password usado para poder iniciar sesión en el sistema.	String	Privado
$fecha_alta	Fecha en que fue creada la cuenta de usuario.	Datetime	Privado
$cuenta_twitter	Nombre de usuario en Twitter.	String	Privado
$cuenta_wordpress	Nombre de usuario en Wordpress	String	Privado
$cuenta_blogger	Nombre de usuario en Blogger.	String	Privado
$cuenta_delicious	Nombre de usuario en Delicious.	String	Privado
$cuenta_wiki	Nombre de usuario en la wiki.	String	Privado
$cuenta_foro	Nombre de usuario en el foro.	String	Privado
$habilitado	Indica si la cuenta está activa.	Boolean	Privado

Métodos

Crear ($user, $password)
Permite crear una cuenta de usuario al indicar el nombre de usuario y password que usará el usuario para iniciar sesión en el sistema. **Parámetros de entrada:** *string* $user *string*$password **Salida** CuentaUsuario Objeto de tipo *CuentaUsuario*.

Recuperar ($user)
Permite recuperar una cuenta de usuario por nombre de usuario o por identificador de la cuenta. **Parámetros de entrada:** *string* | *integer* $user Si es string busca la cuenta de usuario por nombre de usuario. Si es entero busca por identificador. **Salida** CuentaUsuario Objeto de tipo *CuentaUsuario*.

RecuperarTodas ( )
Recupera todas las cuentas de usuario creadas. **Parámetros de entrada:** Ninguno **Salida** CuentaUsuario[]

Actualizar ($id_cuenta, $cu_twitter, $cu_wordpress, $cu_blogger, $cu_delicious, $cu_wiki, $cu_foro)
Actualiza los datos de una cuenta de usuario. **Parámetros de entrada:** *integer* $id_cuenta *string* $cu_twitter *string* $cu_wordpress *string* $cu_blogger *string* $cu_delicious *string* $cu_wiki *string* $cu_foro **Salida** True | False

Eliminar ($id_cuenta)
Elimina una determinada cuenta de usuario. **Parámetros de entrada:** *integer* $id_cuenta **Salida** True | False

ActivarCuenta ($id_cuenta)
Permite activar una determinada cuenta de usuario. **Parámetros de entrada:**

integer $id_cuenta
Salida
True | False

ExisteCuentaUsuario ($username)
Verifica si existe una determinada cuenta de usuario.
Parámetros de entrada:
string $username
Salida
True | False

EstaHabilitado ($username)
Verifica si una cuenta de usuario está habilitada.
Parámetros de entrada:
string $username
Salida
True | False

Clase Correo

Descripción
Clase usada para representar y gestionar la Casilla de Correo de un determinado usuario.

Atributos

Atributo	Comentario	Tipo	Alcance
$email_id	Identificador único del email.	Integer	Privado
$usuario_id	Identificador del usuario a quien le pertenece el email.	Integer	Privado
$es_emisor	Indica si en el mensaje el usuario es emisor o receptor. En caso de ser verdadero es emisor.	Boolean	Privado

Métodos

Crear ($usuario_id, $es_emisor, $leido)
Permite crear una cuenta de usuario al indicar el nombre de usuario y password que usará el usuario para iniciar sesión en el sistema.
Parámetros de entrada:
integer $usuario_id
boolean$es_emisor
boolean$leido Esta variable toma el valor FALSE por defecto.
Salida
integer Indica si pudo ser creado o no el mensaje. 1 en caso afirmativo, 0 en caso contrario.

Recuperar ($id_correo)
Permite recuperar todos los mensajes de la casilla de correo.

Parámetros de entrada:
integer $id_correo
Salida
Correo[]

Eliminar ($id_correo)

Elimina todos los mensajes de una casilla de correo.
Parámetros de entrada:
integer $id_correo
Salida
True | False

RecuperarMensajesNoLeidoUsuario ($usuario_id)

Recupera todos los mensajes no leídos de un determinado usuario.
Parámetros de entrada:
integer $usuario_id
Salida
Correo[]

Clase Email

Descripción
Clase usada para representar y gestionar emails enviados por un usuario.

Atributos

Atributo	Comentario	Tipo	Alcance
$emisor	Nombre de la persona que envía el mensaje.	String	Privado
$destinatario	Nombre de la persona que recibe el mensaje.	String	Privado
$titulo	Asunto del mensaje	String	Privado
$mensaje	Contenido del mensaje enviado por el usuario.	Text	Privado
$fecha	Fecha en la que se envió el mensaje	Datetime	Privado
$leido	Indica si el mensaje fue leído. Si es True indica que fue leído.	Boolean	Privado

Métodos

Crear ($emisor, $destinatario, $titulo, $mensaje)

Crea un nuevo mensaje.
Parámetros de entrada:
string $emisor
string$destinatario
string$titulo
text $mensaje
Salida
integer Indica si pudo ser creado o no el mensaje. 1 en caso afirmativo, 0 en caso contrario.

Recuperar ($id_mail)
Recupera un determinado mensaje. **Parámetros de entrada:** *integer* $id_mail **Salida** Mensaje Objeto de tipo *Mensaje*

MarcarLeido ($id_email)
Marca un determinado mensaje como leído. **Parámetros de entrada:** *integer* $id_mail **Salida** True | False

Eliminar ($id_mail)
Elimina un determinado mensaje. **Parámetros de entrada:** *integer* $id_mail **Salida** True | False

RecuperarDestinatarios ($id_mail)
Recupera todos los destinatarios de un determinado e-mail. **Parámetros de entrada:** *integer* $id_mail **Salida** string[]

RecuperarMensajesEnviados ($usuario_id)
Recupera todos los mensajes enviados por un determinado usuario. **Parámetros de entrada:** *integer* $usuario_id **Salida** Email[]

RecuperarMensajesRecibidos ($usuario_id)
Recupera todos los mensajes recibidos por un determinado usuario. **Parámetros de entrada:** *integer* $usuario_id **Salida** Email[]

Clase Carrera

Descripción
Clase usada para representar y gestionar carreras de la institución educativa.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único de	Integer	Privado

	la carrera.		
$nombre	Nombre de la carrera	String	Privado
$id_facultad	Identificador de la facultad a la cual pertenece la carrera	Integer	Privado

Métodos

Crear ($nombre, $id_facultad)
Crea una carrera en una determinada facultad. **Parámetros de entrada:** *string* $nombre *integer*$id_facultad **Salida** integer Indica si pudo ser creado o no la carrera. 1 en caso afirmativo, 0 en caso contrario.

Recuperar ($id_carrera)
Recupera los datos de una carrera. **Parámetros de entrada:** *integer* $id_carrera **Salida** Carrera Objeto de tipo *Carrera*

RecuperarTodoIdFacultad ($id_facultad)
Recupera todas las carreras de una determinada facultad **Parámetros de entrada:** *integer* $id_facultad **Salida** Carrera[]

Actualizar ($id_carrera, $nombre)
Permite actualizar el nombre de una determinada carrera. **Parámetros de entrada:** *integer* $id_carrera *string* $nombre **Salida** True | False

Eliminar ($id_carrera)
Elimina una carrera cuyo identificador coincida con el que se recibe como parámetro.. **Parámetros de entrada:** *integer* $id_carrera **Salida** True | False

Clase Matricula

Descripción

Clase usada para representar y gestionar la relación existente entre la clase Usuario y la clase Carrera.

Atributos

Atributo	Comentario	Tipo	Alcance
$year_ingreso	Año en que el usuario empieza a cursar la carrera	Integer	Privado

Métodos

Asociar ($id_usuario, $id_carrera, $year_ingreso)
Asocia un determinado usuario a una carrera. **Parámetros de entrada:** *integer* $id_usuario *integer* $id_carrera *integer* $year_ingreso **Salida** True | False

Desvincular ($id_usuario, $id_carrera)
Elimina la relación existente entre un usuario y una carrera. **Parámetros de entrada:** *integer* $id_usuario *integer* $id_carrera **Salida** True | False

RecuperarCarrerasInscripto ($id_usuario)
Recupera todas las carreras en las cuales este inscripto un determinado usuario. **Parámetros de entrada:** *integer* $id_usuario **Salida** Carrera[]

Clase Facultad

Descripción
Clase usada para representar y gestionar las facultades de la institución educativa.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único de la facultad.	Integer	Privado
$nombre	Nombre de la facultad.	String	Privado
$direccion	Dirección de la facultad	String	Privado
$telefono	Teléfonos de la facultad	String	Privado
$fax	Fax de la facultad	String	Privado
$url_web	Url del sitio web de la facultad.	String	Privado

Métodos

Crear ($nombre, $direccion, $telefono, $fax, $url_web)
Crea una facultad. **Parámetros de entrada:** *string* $nombre *string*$direccion *string* $telefono *string* $fax *string* $url_web **Salida** integer Indica si pudo ser creado o no la facultad. 1 en caso afirmativo, 0 en caso contrario.

Recuperar ($id_facultad)
Recupera los datos de una determinada facultad. **Parámetros de entrada:** *integer* $id_facultad **Salida** Facultad Objeto de tipo *Facultad.*

RecuperarTodos ()
Recupera todas las facultades de la institución. **Parámetros de entrada:** Ninguno **Salida** Facultad[]

Actualizar ($id_facultad, $nombre, $direccion, $telefono, $fax, $url_web)
Actualiza los datos de una facultad. **Parámetros de entrada:** *integer* $id_facultad *string* $nombre *string* $direccion *string* $telefono *string* $fax *string* $url_web **Salida** True | False

Eliminar ($id_facultad)
Elimina una determinada facultad. **Parámetros de entrada:** *integer* $id_facultad **Salida** True | False

Clase GrupoCorreo

Descripción

Clase usada para representar y gestionar los grupos creados en la aplicación de correo del sistema.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del grupo.	Integer	Privado
$nombre	Nombre del grupo de correo.	String	Privado
$usuario_id	Identificador del usuario quien creó el grupo.	Integer	Privado

Métodos

Crear ($nombre, $usuario_id)

Permite a un usuario crear un nuevo grupo.
Parámetros de entrada:
string $nombre
integer$usuario_id
Salida
integer Indica si pudo ser creado o no el grupo. 1 en caso afirmativo, 0 en caso contrario.

RecuperarGruposPorUsuario ($usuario_id)

Recupera todos los grupos de correo creados por un determinado usuario.
Parámetros de entrada:
integer $usuario_id
Salida
GrupoCorreo[] Arreglo unidimensional en el cual cada elemento es del tipo *GrupoCorreo*.

RecuperarGruposPorUsuarioConsulta ($consulta, $usuario_id)

Recupera todos los grupos de correo creados por un determinado usuario que cumplan la condición establecida por el parámetro $consulta.
Parámetros de entrada:
string $consulta Condición usada para crear la clausula *where* de la sentencia SQL.
integer $usuario_id
Salida
GrupoCorreo[] Arreglo unidimensional en el cual cada elemento es del tipo *GrupoCorreo*.

RecuperarMiembrosPorGrupo ($grupo_id)

Recupera los miembros de un determinado grupo.
Parámetros de entrada:
integer $grupo_id
Salida
Usuario[]

AgregarMiembrosAlGrupo ($grupos_correo_id, $usuario_id)

Agrega un usuario a un grupo.

Parámetros de entrada:
integer $grupos_correo_id
integer $usuario_id
Salida
integer Indica si pudo agregar el miembro al grupo. 1 en caso afirmativo, 0 en caso contrario.

EliminarMiembrosGrupo ($grupos_correo_id, $usuario_id)
Elimina un usuario de un grupo. **Parámetros de entrada:** *integer* $grupos_correo_id *integer* $usuario_id **Salida** integer Indica si pudo eliminar el usuario del grupo. 1 en caso afirmativo, 0 en caso contrario.

2. CLASES DEL DIRECTORIO DE AUDIO Y VIDEO

Clase ArchivoMultimedia

Descripción
Clase usada para representar a los recursos multimedia. Estos recursos son fragmentos de código de otros sistemas de alojamiento de archivos multimedia como por ejemplo YouTube.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del recurso multimedia.	Integer	Privado
$titulo	Titulo asignado al recurso multimedia.	String	Privado
$descripcion	Descripción adicional asignada al recurso.	Text	Privado
$script	Almacena el fragmento de código del recurso multimedia.	Text	Privado
$fechapublicacion	Fecha en que fue dado de alta el recurso multimedia.	Datetime	Privado
$usuario_id	Identificador del usuario que comparte el recurso multimedia.	Integer	Privado
$tipoarchivo_id	Identificador del tipo de archivo	Integer	Privado

Métodos

Crear($titulo, $descripcion, $script, $usuario_id, $tipoarchivo_id)
Permite dar de alta un nuevo recurso multimedia. **Parámetros de entrada:** *string* $titulo *text* $descripcion *text* $script

integer $usuario_id
integer $tipoarchivo_id
Salida
ArchivoMultimedia

Recuperar ($archivomultimedia_id)

Permite dar de alta un nuevo recurso multimedia.
Parámetros de entrada:
integer $archivomultimedia_id Identificador del archivo multimedia
Salida
ArchivoMultimedia

Actualizar ($objetoarchivomultimedia)

Permite actualizar un objeto de tipo ArchivoMultimedia
Parámetros de entrada:
ArchivoMultimedia$objetoarchivomultimedia
Salida
True | False

Eliminar ($archivomultimedia_id)

Elimina el recurso multimedia que coincide con el identificador recibido como parámetro.
Parámetros de entrada:
integer $archivomultimedia_id Identificador del archivo multimedia
Salida
True | False

RecuperarComentarios ($archivomultimedia_id)

Recupera los comentarios realizado a un determinado recurso multimedia.
Parámetros de entrada:
integer $archivomultimedia_id Identificador del archivo multimedia
Salida
Comentario[] Arreglo unidimensional en el cual cada componente es un objeto del tipo Comentario.

RecuperarUltimosArchivosCompartidos ($periodo_en_semana)

Recupera los últimos recursos multimedia compartidos en el período indicado.
Parámetros de entrada:
integer$periodo _en_semana Indica la cantidad de semana anteriores a la actual desde la cual se deben empezar a recuperar los recursos multimedia.
Salida
ArchivoMultimedia[]

RecuperarPorTag ($tag)

Recupera archivos multimedia que hayan sido marcados con un determinado tag.
Parámetros de entrada:
string$tag
Salida

ArchivoMultimedia[]

RecuperarPorCalificacion ($calificacion)
Recupera recursos multimedia que hayan recibido una determinada clasificación. **Parámetros de entrada:** *integer* $calificacion **Salida** ArchivoMultimedia[]

RecuperarPorTipoArchivo ($nombretipo)
Recupera recursos multimedia que sean de un determinado tipo de archivo. **Parámetros de entrada:** *string*$nombretipoNombre del tipo de archivo **Salida** ArchivoMultimedia[]

RecuperarTodos ( )
Recupera todos los recursos multimedia dados de alta. **Parámetros de entrada:** Ninguno **Salida** ArchivoMultimedia[]

RecuperarPorUsuario ($usuario_id)
Recupera los recursos multimedia agregados por un determinado usuario. **Parámetros de entrada:** *integer* $usuario_id Identificador único del usuario **Salida** ArchivoMultimedia[]

RecuperarTags ( )
Recupera los tags asociados a un recurso multimedia. Es un método de instancia, por lo cual recupera los tags del objeto que hace el llamado. **Parámetros de entrada:** Ninguno **Salida** TagArchivo[] Arreglo unidimensional en el cual cada elemento es del tipo TagArchivo

Clase TagArchivo

Descripción
Esta clase se usa gestionar los tags que se asignan a un determinado recurso multimedia.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador del tag asociado a un archivo.	Integer	Privado

$archivomultimedia_id	Identificador asociado al recurso multimedia.	Integer	Privado
$usuario_id	Identificador del usuario que asocio el tag al recurso multimedia.	Integer	Privado
$tag_id	Identificador del tag asociado al recurso multimedia.	Integer	Privado
$nombretag	Tag asociado al archivo.	String	Privado
$usuario_id	Identificador del usuario que comparte el recurso multimedia.	Integer	Privado
$apellidonombre_usuario	Nombre y apellido del usuario que asocia el tag al recurso multimedia.	String	Privado

Métodos

AsociarTagArchivo ($tag, $archivomultimedia_id, $usuario_id)

Asocia un tag a un recurso multimedia.
Parámetros de entrada:
string $tag Tag a asignar al recurso multimedia. Dentro del proceso se hace una búsqueda del tag en la base de datos para obtener su identificador en caso de que exista.
integer $archivomultimedia_id
integer $usuario_id
Salida
True | False

DevincularTagArchivo ($tagarchjvo_id, $archivomultimedia_id, $usuario_id)

Desvincula un tag asociado a un recurso multimedia por parte de un determinado usuario.
Parámetros de entrada:
integer $tagarchivo_idIdentificador del tag.
integer $archivomultimedia_id
integer $usuario_id
Salida
True | False

RecuperarTagsArchivo ($archivomultimedia_id)

Recupera los tags asociados a un determinado recurso multimedia.
Parámetros de entrada:
integer $archivomultimedia_id
Salida
TagArchivo[]

RecuperarPorTag ($tag)

Recupera los recursos multimedia que tengan asociados un determinado tag.
Parámetros de entrada:
string $tag

Salida
TagArchivo[]

Clase Tag

Descripción
Clase usada para representar a la entidad Tag.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador del tag.	Integer	Privado
$descripcion	Tag.	String	Privado

Métodos

Crear ($descripcion)
Permite dar de alta un tag.
Parámetros de entrada:
string $descripcion
Salida
Tag Objeto de tipo *Tag*

Recuperar ($tag_id)
Recupera un tag dado su identificador.
Parámetros de entrada:
integer $tag_id
Salida
Tag Objeto de tipo *Tag*

RecuperarPorNombre ($tag)
Recupera un objeto de tipo tag dado su nombre.
Parámetros de entrada:
string $tag
Salida
Tag Objeto de tipo *Tag*

Actualizar ()
Método de instancia que actualiza al objeto que lo invoca.
Parámetros de entrada:
Ninguno
Salida
True | False

Eliminar ($tag_id)
Da de baja un tag dado su identificador.
Parámetros de entrada:
integer $tag_id
Salida
True | False

RecuperarTodos ()
Recupera todos los tags creados. **Parámetros de entrada:** Ninguno **Salida** Tag[]

Clase TipoArchivo

Descripción
Clase usada para representar a los tipos de archivos de los cuales pueden ser los recursos multimedia.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador del tipo de archivo.	Integer	Privado
$nombre	Nombre asignado al tipo de archivo.	String	Privado

Métodos

Crear ($nombre)
Permite dar de alta un tipo de archivo. **Parámetros de entrada:** *string* $nombre Nombre del tipo de archivo **Salida** TipoArchivo Objeto de tipo *TipoArchivo*

Recuperar ($tipoarchivo_id)
Recupera un tipo de archivo dado su identificador. **Parámetros de entrada:** *integer* $tipoarchivo_idIdentificador del tipo de archivo **Salida** TipoArchivo Objeto de tipo *TipoArchivo*

Actualizar ($objetoArchivo)
Actualiza el objeto tipo TipoArchivo que recibe como parámetro. **Parámetros de entrada:** *TipoArchivo* $objetoArchivo **Salida** True | False

Eliminar ($tipoarchivo_id)
Elimina un tipo de archivo dado su identificador. **Parámetros de entrada:** *integer* $tipoarchivo_id Identificador del tipo de archivo **Salida** True | False

RecuperarTodos ()
Recupera todos los tipos de archivos dados de alta. **Parámetros de entrada:** Ninguno **Salida** TipoArchivo[]

Clase Calificacion

Descripción
Clase usada manejar las calificaciones asignadas a los recursos multimedia.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador delacalificación asignada a un recurso multimedia.	Integer	Privado
$calificacion	Calificación asignada por un usuario a un determinado recurso multimedia en la escala de 0 a 10.	Double	Privado
$usuario_id	Identificador del usuario que realiza la calificación	Integer	Privado
$archivomultimedia_id	Identificador del recurso multimedia que recibe la calificación	Integer	Privado
$apellidonombre_usuario	Apellido y nombre del usuario que realiza la calificación.	String	Privado

Métodos

Crear ($calificacion, $usuario_id, $archivomultimedia_id)
Asigna una calificación a un determinado recurso multimedia por parte de un determinado usuario. **Parámetros de entrada:** *double* $calificacion *integer* $usuario_id *integer* $archivomultimedia_id **Salida** Calificacion Objeto de tipo *Calificacion*

Recuperar ($calificacion_id)
Recupera un objeto de tipo Clasificacion dado su identificador. **Parámetros de entrada:** *integer* $calificacion_id **Salida**

Calificacion Objeto de tipo *Calificacion*

Actualizar ($objcalificacion)
Actualiza el objeto de tipo calificación que recibe como parámetro. **Parámetros de entrada:** *Calificacion* $objcalificacion **Salida** True | False

Eliminar ($calificacion_id)
Elimina un objeto de tipo calificación dado su identificador. **Parámetros de entrada:** *integer* $calificacion_id **Salida** True | False

RecuperarPorCalificacion ($calificacion)
Recupera objetos de tipo Calificacion que hayan sido calificados con un determinado valor. **Parámetros de entrada:** *double* $calificacion **Salida** Calificacion[]

Clase Comentario

Descripción
Clase usada manejar los comentarios asignados a los recursos multimedia.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador del comentario asignado al recurso multimedia.	Integer	Privado
$descripcion	Contenido del comentario asignado al recurso.	Text	Privado
$usuario_id	Identificador del usuario que realiza el comentario	Integer	Privado
$archivomultimedia_id	Identificador del recurso multimedia que recibe el comentario	Integer	Privado
$apellidonombre_usuario	Apellido y nombre del usuario que realiza el comentario.	String	Privado

Métodos

Crear ($descripcion, $usuario_id, $archivomultimedia_id)
Asigna una calificación a un determinado recurso multimedia por parte de un

determinado usuario.
Parámetros de entrada:
string $comentario
integer $usuario_id
integer $archivomultimedia_id
Salida
Comentario Objeto de tipo *Comentario*

Recuperar ($comentario_id)
Recupera un objeto de tipo Comentario dado su identificador. **Parámetros de entrada:** *integer* $comentario_id **Salida** Comentario Objeto de tipo *Comentario*

Actualizar ($objcomentario)
Actualiza el objeto de tipo comentario que recibe como parámetro. **Parámetros de entrada:** *Comentario* $objcomentario **Salida** True | False

Eliminar ($comentario_id)
Elimina un objeto de tipo comentario dado su identificador. **Parámetros de entrada:** *integer* $comentario_id **Salida** True | False

RecuperarComentariosArchivo ($archivomultimedia_id)
Recupera los comentarios asignados a un determinado recurso multimedia. **Parámetros de entrada:** *integer* $archivomultimedia_id **Salida** Comentario[]

3. CLASES DEL SISTEMA DE CÁTEDRAS

Clase Profesor

Hereda de la clase usuario. Ver atributos y métodos heredados.

Descripción
Clase usada para representar a la entidad Profesor.

Atributos

Atributo	Comentario	Tipo	Alcance
$yearinicio	Fecha en que el profesor inicio sus actividades como	Datetime	Privado

	docente en la institución educativa.		
$usuario_id	Identificador del usuario	Integer	Privado

Métodos

Crear_Profesor ($nombre, $apellido, $tipousuario_id, $cuentausuario_id, $telefonos, $sitiosweb, $yearinicio, $usuario_id)
Crea un nuevo objeto de tipo Profesor. En caso de que el usuario no exista, lo da de alta; en caso contrario, sólo se registra que el usuario se desempeña como profesor. **Parámetros de entrada:** *string* $nombre *string*$apellido *integer* $tipousuario_id *integer* $cuentausuario_id *string* $telefonos *text* $sitiosweb *datetime* $yearinicio *integer* $usuario_id **Salida** Profesor Objeto de tipo *Profesor*

Recuperar_Profesor ($usuario_id)
Recupera los datos asociados al profesor cuyo identificador se recibe como parámetro. **Parámetros de entrada:** *integer* $usuario_id **Salida** Profesor Objeto de tipo *Profesor*

Actualizar_Profesor ($objprofesor)
Actualiza en la base de datos los datos asociados en el objeto Profesor que recibe como parámetro. **Parámetros de entrada:** *Profesor* $objprofesor **Salida** True | False

Eliminar_Profesor ($usuario_id)
Elimina del sistema la información asociada a un determinado profesor. **Parámetros de entrada:** *integer* $usuario_id **Salida** True | False

Clase Ayudante

Hereda de la clase usuario. Ver atributos y métodos heredados.

Descripción
Clase usada para representar a la entidad Ayudante.

Atributos

Atributo	Comentario	Tipo	Alcance
$yearinicio	Fecha en que el ayudante inicio sus actividades como tal en la institución educativa.	Datetime	Privado
$yearfin	Fecha en que finaliza el cargo como ayudante	Datetime	Privado
$asignaturas_id	Identificador de la asignatura en la cual es ayudante.	Integer	Privado
$usuario_id	Identificador del usuario	Integer	Privado

Métodos

Crear_Ayudante ($nombre, $apellido, $tipousuario_id, $cuentausuario_id, $telefonos, $sitiosweb, $yearinicio, $yearfin, $asignatura_id, $usuario_id)

Crea un nuevo objeto de tipo Ayudante. En caso de que el usuario no exista, lo da de alta; en caso contrario, sólo se registra que el usuario se desempeña como ayudante de cátedra.
Parámetros de entrada:
string $nombre
string $apellido
integer $tipousuario_id
integer $cuentausuario_id
string $telefonos
text $sitiosweb
datetime $yearinicio
datetime $yearfin
integer $asignatura_id
integer $usuario_id
Salida
Ayudante Objeto de tipo *Ayudante*

Recuperar_Ayudante ($usuario_id)

Recupera los datos asociados al ayudante cuyo identificador se recibe como parámetro.
Parámetros de entrada:
integer $usuario_id
Salida
Ayudante Objeto de tipo *Ayudante*

Actualizar_Ayudante ($objayudante)

Actualiza en la base de datos los datos asociados en el objeto Ayudante que recibe como parámetro.

Parámetros de entrada:
Ayudante $objayudante
Salida
True | False

Eliminar_Ayudante ($usuario_id)
Elimina del sistema la información asociada a un determinado ayudante. **Parámetros de entrada:** *integer* $usuario_id **Salida** True | False

RecuperarAyudantesAsignatura ($asignatura_id)
Recupera todos los ayudantes asociados a una determinada asignatura. **Parámetros de entrada:** *integer* $asignatura_id **Salida** Ayudante[]

Clase MiembroCatedra

Descripción
Clase usada para almacenar información de la relación entre la clase *EquipoCatedra* y *Profesor*. No dispone de métodos puesto que más bien es usada como un tipo de datos por parte de la clase *EquipoCatedra*.

Atributos

Atributo	Comentario	Tipo	Alcance
$profesores_id	Identificador del profesor	Integer	Privado
$categoria_id	Categoría del profesor en una determinada asignatura. Valores que puede tomar: "JTP", "Profesor Adjunto", "Profesor Titular".	String	Privado

Clase EquipoCatedra

Descripción
Clase usada para representar a la entidad EquipoCatedra y gestionar información relacionada a la misma.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del equipo cátedra	Integer	Privado
$nombre	Nombre del equipo cátedra	String	Privado
$subdominio	Subdominio a través del	String	Privado

	cual se podrá acceder al sitio del equipo cátedra.		
$miembros	Profesores que integran el equipo cátedra.	MiembroCatedra[]	Privado

Métodos

Crear ($nombre, $subdominio)
Permite dar de alta un equipo cátedra. **Parámetros de entrada:** *string* $nombre *string* $subdominio **Salida** EquipoCatedra Objeto de tipo *EquipoCatedra*

Recuperar ($equipocatedra_id)
Recupera información asociada a un equipo cátedra determinado. **Parámetros de entrada:** *integer* $equipocatedra_id **Salida** EquipoCatedra Objeto de tipo *EquipoCatedra*

Actualizar ($objcatedra)
Actualiza la información contenida en el objeto $objcatedra en la base de datos. **Parámetros de entrada:** *EquipoCatedra* $objcatedra **Salida** True | False

Eliminar ($equipocatedra_id)
Elimina un equipo cátedra de la base de datos. **Parámetros de entrada:** *integer* $equipocatedra_id **Salida** True | False

RecuperarTodos ()
Recupera los equipos cátedras existentes. **Parámetros de entrada:** Ninguno **Salida** EquipoCatedra[]

RecuperarMiembros()
Es un método de instancia que recupera los miembros del equipo cátedra del objeto que invoca el método. **Parámetros de entrada:** Ninguno **Salida** MiembrosCatedra[]

ActualizarMiembros($miembros)
Actualiza la información relacionada a los profesores que integran el equipo cátedra. La actualización puede implicar modificación de datos de los profesores, eliminación de profesores o sumar nuevos profesores al equipo. **Parámetros de entrada:** *MiembrosCatedra[]* $miembros Arreglo unidimensional en el cual cada elemento es un objeto del tipo *MiembrosCatedra.* **Salida** void

RecuperarAsignaturas ($equipocatedra_id)
Recuperar las asignaturas que están a cargo del equipo cátedra. **Parámetros de entrada:** *integer* $equipocatedra_idIdentificador del equipo cátedra. **Salida** Asignatura[] Arreglo unidimensional en el cual cada elemento es del tipo *Asignatura*

Clase Asignatura

Descripción
Clase usada para representar a la entidad Asignatura y gestionar información relacionada a la misma.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único de la asignatura.	Integer	Privado
$nombre	Nombre de la asignatura.	String	Privado
$equipocatedra_id	Identificador del equipo cátedra.	Integer	Privado

Métodos

Crear ($nombre, $equipocatedra_id)
Permite dar de alta una asignatura en el sistema. **Parámetros de entrada:** *string* $nombre *integer*$equipocatedra_id **Salida** Asignatura Objeto de tipo *Asignatura*

Recuperar ($asignatura_id)
Recupera información asociada a una determinada asignatura. **Parámetros de entrada:** *integer* $asignatura_id **Salida** Asignatura Objeto de tipo *Asignatura*

Actualizar ($objasignatura)

Actualiza los datos de una asignatura.
Parámetros de entrada:
Asignatura $objasignatura
Salida
True | False

Eliminar ($asignatura_id)

Elimina una determinada asignatura del sistema.
Parámetros de entrada:
integer $asignatura_id
Salida
True | False

RecuperarAsignaturasCatedra ($equipocatedra_id)

Recupera las asignaturas a cargo de un determinado equipo cátedra.
Parámetros de entrada:
integer$equipocatedra_id
Salida
Asignatura[]

RecuperarPlanificacionesDisponibles ($asignatura_id)

Recupera las planificaciones asociadas a una asignatura.
Parámetros de entrada:
integer$asignatura_id
Salida
Planificacion[] Arreglo unidimensional en el cual cada elemento es del tipo *Planificacion*

RecuperarNovedades ($asignatura_id)

Recupera las novedades asociadas a una asignatura.
Parámetros de entrada:
integer $asignatura_id
Salida
Novedad[] Arreglo unidimensional en el cual cada elemento es del tipo *Novedad*

RecuperarTrabajosPracticos ($asignatura_id)

Recupera los trabajos prácticos asociadas a una asignatura.
Parámetros de entrada:
integer $asignatura_id
Salida
TrabajoPractico[] Arreglo unidimensional en el cual cada elemento es del tipo *TrabajoPractico*

RecuperarAyudantes ($asignatura_id)

Recupera los ayudantes de una asignatura.
Parámetros de entrada:
integer $asignatura_id
Salida

Ayudante[] Arreglo unidimensional en el cual cada elemento es del tipo *Ayudante*

Clase Novedad

Descripción
Clase usada para representar y gestionar las novedades generadas por una asignatura.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único de la novedad.	Integer	Privado
$fechapublicacion	Fecha en que fue publicada la novedad.	Datetime	Privado
$descripcion	Descripción de la novedad.	Text	Privado
$asignaturas_id	Identificador de la asignatura.	Integer	Privado

Métodos

Crear ($fechapublicacion, $descripcion, $asignatura_id)
Crea una novedad para una determinada asignatura. **Parámetros de entrada:** *datetime* $fechapublicacion *string*$descripcion *integer* $asignatura_id **Salida** Novedad Objeto de tipo *Novedad.*

Recuperar ($ novedad_id)
Recupera una determinada novedad dado su identificador. **Parámetros de entrada:** *integer* $novedad_id **Salida** Novedad Objeto de tipo *Novedad.*

Actualizar ($objnovedad)
Actualiza los datos de una novedad. **Parámetros de entrada:** *Novedad* $objnovedad **Salida** True | False

Eliminar ($ novedad_id)
Elimina la novedad cuyo identificador se recibe como parámetro. **Parámetros de entrada:** *integer* $novedad_id **Salida** True | False

RecuperarNovedadesAsignatura ($ asignatura_id)
Recupera las novedades de una determinada asignatua. **Parámetros de entrada:** *integer* $asignatura_id **Salida** Novedad[]

Clase TrabajoPractico

Descripción
Clase usada para representar y gestionar los trabajos prácticos asociados a una asignatura.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único del trabajo práctico.	Integer	Privado
$nombre	Título del trabajo práctico.	String	Privado
$urlarchivo	Dirección en la cual se encuentra el archivo. Los trabajos prácticos son subidos al sistema. No hay restricciones en cuanta a extensiones.	String	Privado
$year	Año en que fue cargado el trabajo práctico.	Integer	Privado
$asignaturas_id	Identificador de la asignatura a la cual pertenece el trabajo.	Integer	Privado

Métodos

Crear ($nombre, $urlarchivo, $year, $asignatura_id)
Permite cargar un trabajo práctico para una determinada asignatura. **Parámetros de entrada:** *string* $nombre *string* $urlarchivo *integer* $year *integer* $asignatura_id **Salida** TrabajoPractico Objeto de tipo *TrabajoPractico.*

Recuperar ($trabajopractico_id)
Recupera un determinado trabajo práctico. **Parámetros de entrada:** *integer* $trabajopractico_id **Salida** TrabajoPractico Objeto de tipo *TrabajoPractico.*

Actualizar ($objtrabajopractico)
Actualiza los datos asociados a un trabajo práctico.

Parámetros de entrada:
TrabajoPractico $objtrabajopractico
Salida
True | False

Eliminar ($trabajopractico_id)
Elimina un determinado trabajo práctico.
Parámetros de entrada:
integer $trabajopractico_id
Salida
True | False

RecuperarTrabajosPracticosAsignatura ($asignatura_id)
Recupera los trabajos prácticos creados por una determinada asignatura.
Parámetros de entrada:
integer $asignatura_id
Salida
TrabajoPractico[]

Clase Planificacion

Descripción
Clase usada para representar y gestionar los trabajos prácticos asociados a una asignatura.

Atributos

Atributo	Comentario	Tipo	Alcance
$id	Identificador único de la planificación.	Integer	Privado
$year	Año en el cual se aplica la planificación.	Integer	Privado
$planificacion	Contenido de la planificación propiamente dicha.	Text	Privado
$asignaturas_id	Identificador de la asignatura a la cual pertenece la planificación.	Integer	Privado

Métodos

Crear ($year, $planificacion, $asignatura_id)
Permite cargar una planificación para una determinada asignatura.
Parámetros de entrada:
integer $year
text $planificacion
integer $asignatura_id
Salida
Planificacion Objeto de tipo *Planificacion.*

Recuperar ($planificacion_id)

Recupera una determinada planificación de una asignatura.
Parámetros de entrada:
integer $planificacion_id
Salida
Planificacion Objeto de tipo *Planificacion.*

Actualizar ($objplanificacion)

Actualiza los datos de una determinada planificación.
Parámetros de entrada:
Planificacion $objplanificacion
Salida
True | False

Eliminar ($planificacion_id)

Elimina del sistema una determinada planificación.
Parámetros de entrada:
integer $planificacion_id
Salida
True | False

RecuperarPlanificacionesAsignatura ($asignatura_id)

Recupera las planificaciones cargadas en una determinada asignatura.
Parámetros de entrada:
integer $asignatura_id
Salida
Planificacion[]

ANEXO II

PRINCIPALES INTERFACES DEL PROTOTIPO PLANTEADO

A partir de la solución de diseño del software desarrollada en este trabajo, se crearon prototipos sobre las características más importantes y relevantes que se plantearon en el diseño. Se eligió aquellos subsistemas y funcionalidades esenciales para destacar las diferentes características del software diseñado. Los prototipos creados fueron codificados en el lenguaje *PHP*(software libre), empleándose los Framework *Zend* y *CodeIgniter* como plataformas de desarrollo, así como la librería *JQuery*, y el motor de base de datos *MySql*.

Entre los subsistemas elegidos para crear los prototipos se encuentran el Directorio de Usuarios, el Portal, y la Aplicación para carga de novedades por parte de las diferentes áreas de la institución. El foro y la wiki fueron implementadosusando los software libre phpBB y Mediawiki respectivamente.

En las siguientes secciones se detallarán las diferentes interfaces de los prototipos:

DIRECTORIO DE USUARIOS:

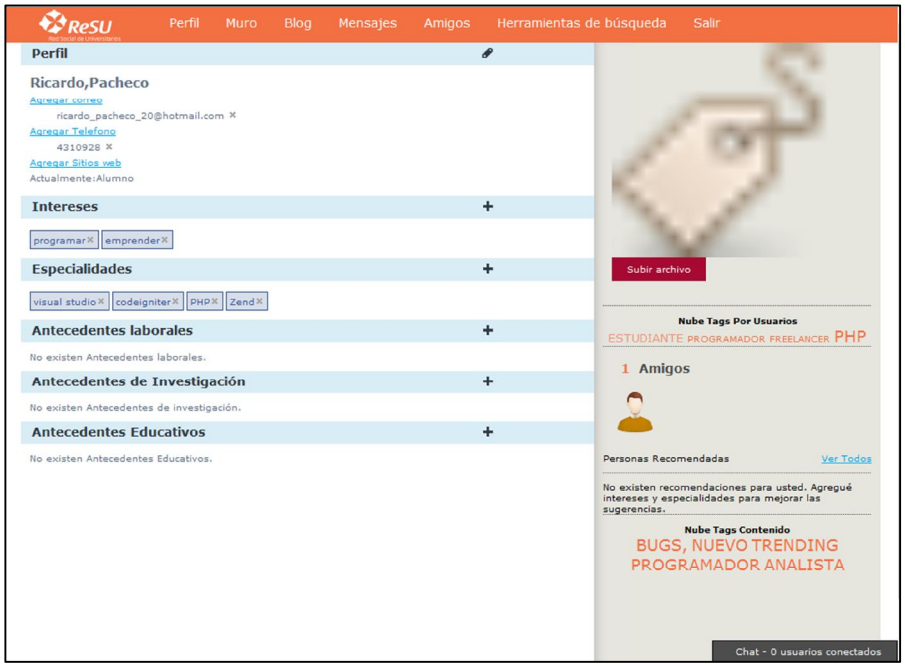
La aplicación se carga a partir de la url *http://betatesting.com.ar.* La misma despliega una pantalla para permitir inicio de sesión del usuario al Directorio de Usuarios, y un enlace para crear una nueva cuenta de usuario (Anexo III. Figura I).

Anexo III. Figura 1. Interfaz correspondiente al Directorio de Usuario para el inicio de sesión

El paso inicial para acceder al Directorio de Usuarios es crear una cuenta (Anexo III. Figura 2). En la misma se indican datos básicos del usuario, tales como: nombre, apellido, usuario, contraseña e indicar la facultad y carrera a la cual pertenece el mismo. Una vez creada la cuenta, la aplicación envía un e-mail con un enlace para activarla. Habilitada la cuenta, el usuario tiene disponible la aplicación para utilizarla.

Anexo III. Figura 2. Interfaz correspondiente al formulario para la creación de cuentas

Luego de haber iniciado sesión en la aplicación, se muestra el perfil del usuario (Anexo III. Figura 3). Aquí el usuario puede cargar toda la información relevante a su persona como datos básicos, sus intereses, sus especialidades, sus antecedentes laborales, educativos, y de investigación y cargar una imagen para su perfil. El proceso de carga de datos, es interactivo y rápido por lo que el usuario no debería esperar a que la página web recargue para que observe los cambios que va realizando sobre su perfil.



Anexo III. Figura 3. Interface correspondiente a un perfil del directorio de usuarios

Perfil

Para cargar datos básicos, tales como dirección de correo electrónica,teléfono/s o sitios web se debe hacer clic según corresponda en *"Agregar correo"*, *"Agregar teléfono"* ó *"Agregar sitios web"* para que se despliegue unos controles de entrada que le permitirán al usuario cargar los datos correspondientes. Para remover algunos de estos datos básicos basta con presionar sobre la "x" que se ubica al lado derecho del correspondiente dato.

Anexo III. Figura 4. Interfaz del directorio de usuarios para la edición de correo electrónico, teléfonos y sitios web

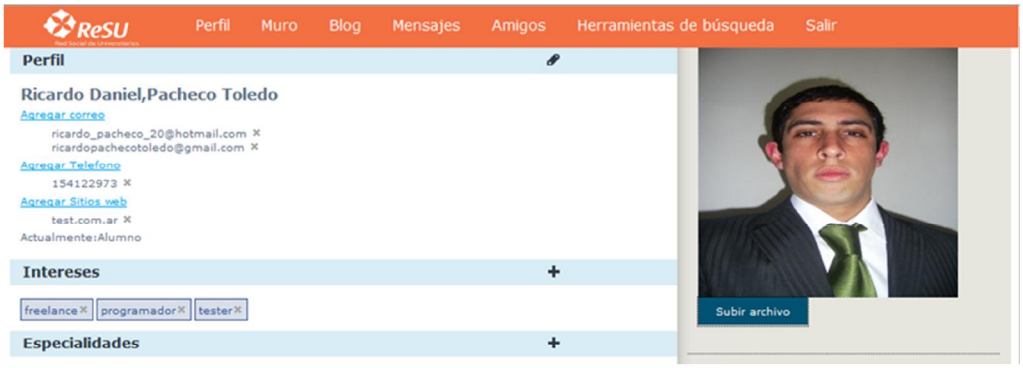
El procedimiento para cargar intereses y especialidades es similar al anterior (Anexo III. Figura 5).Para lograr que se desplieguen los controles de carga de datos es necesario en presionar en el símbolo "+". Para remover un interés o especialidad basta con presionar sobre la "x".

Anexo III. Figura 5. Interfaz del directorio de usuarios para la edición de intereses y especialidades

La carga de los antecedentes laborales, educativos y de investigación siguen el mismo proceso de carga. Para agregar un nuevo antecedente laboral, se debe hacer click sobre el símbolo +. Posteriormente se mostrará una pantalla que permitirá la carga del correspondiente antecedente. Una vez cargado el antecedente, el mismo puede ser removido haciendo click en el texto *eliminar* o también puede ser modificado haciendo click en el texto *editar*(Anexo III. Figura 6).

Anexo III. Figura 6. Interfaz para la edición de los antecedente de investigación y de educación

A través del perfil se permite al usuario cargar una imagen para su perfil brindándole de esta manera una forma de transmitir su identidad.



Anexo III. Figura 7. Interfaz correspondiente al perfil de un usuario con fotografía proporcionada

A la derecha de la carga de datos del perfil, se presentan la nube de tags que le fue asignada al usuario, los contactos del mismo, los usuarios recomendados y la nube de tags generada a partir de los diversos tags que fueron creados por el usuario. Esta información también se muestra en otras interfaces como la del Muro, Blog, Mensajes,Amigos y Búsquedas.

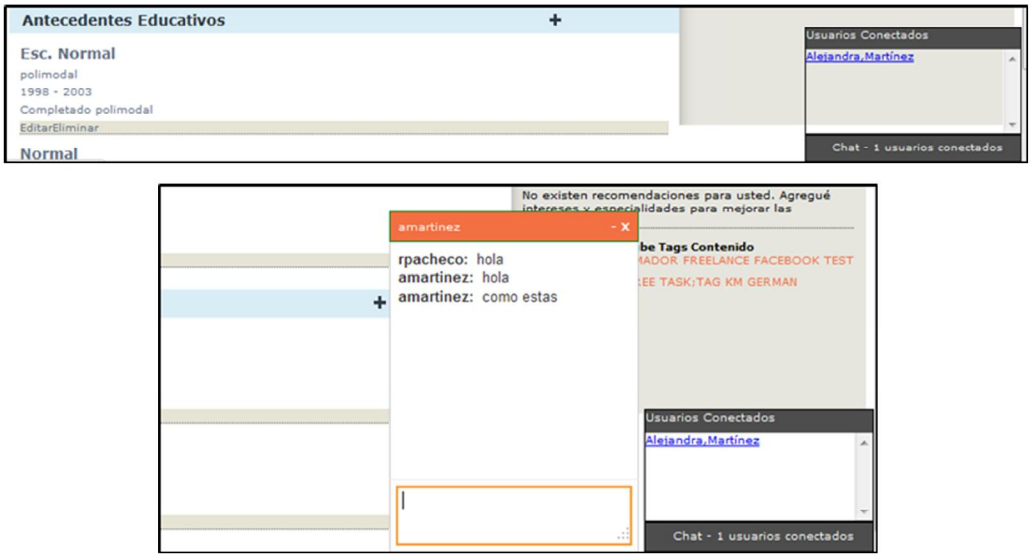
La nube de tags asignadas al usuario: Muestra una nube de tag (etiquetas) generada a partir del proceso de etiquetado realizado por otros usuarios cuando visitan su perfil.

Contactos: Muestra los amigos / contactos del usuario.

Usuarios recomendados: Muestra los usuarios recomendados por el algoritmo de recomendación según intereses y especialidades de los usuarios.

Nube de tags de contenidos: Muestra una nube de tag(etiquetas) generada a partir del proceso de etiquetadoque realiza el usuario sobre los contenidos publicados en el Directorio de Usuario.

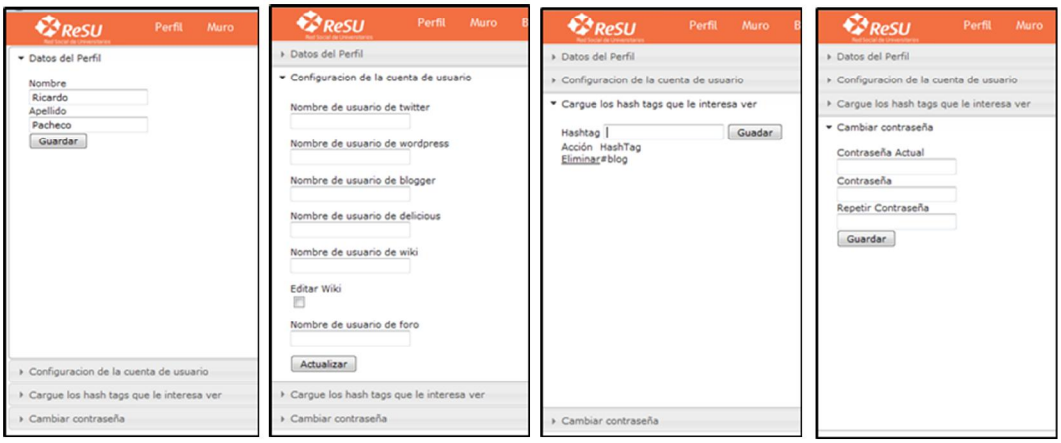
El sistema de chat se encuentra disponible en las interfaces del Perfil, Muro, Blog y Búsquedas. Este sistema muestra los usuarios que se encuentran en línea en la parte inferior del sitio. Para establecer una sesión de chat con otro usuario es necesario hacer click sobre el nombre del mismo, lo cual generará que se cargue una ventana en la parte inferior de la interface del sitio, a la izquierda de la lista de usuarios en línea (Anexo III. Figura 8). Esta ventana es la que hace posible el envío de mensajes entre dos usuarios.



Anexo III. Figura 8. Interface del directorio de usuarios correspondiente al Chat

Desde el perfil se puede acceder a la configuración de la cuenta del usuario haciendo click en el icono. En la configuración de la cuenta es posible cambiar el nombre y el apellido del usuario. También se puede agregar o modificar los nombre de usuario de las cuentas de Twitter, Blogger, Delicious, Wiki, y del Foro. Estos nombres de usuarios se utilizan para recuperar la información que tiene el usuario publicada en otras aplicaciones web (Anexo III. Figura 9).

Con respecto a la cuenta de twitter, es posible indicar qué hashtag le interesa al usuario que sean recuperados. Es decir, sólo se recuperan aquellas publicaciones de interés (Anexo III. Figura 9).



Anexo III. Figura 9. Interfaz del Directorio de Usuarios para configurar diversos aspectos de la cuenta

Cambiar la contraseña de usuario es una funcionalidad que también se ofrece desde esta sección del Directorio de Usuarios (Anexo III. Figura 9).

Muro

A través del Muro el usuario puede compartir información y mantenerse informado de lo que puedan compartir otros usuarios (Anexo III. Figura 10). La característica de la información que se comparte, debe ser breve, ya que para las publicaciones extensas se tiene disponible el foro y la wiki. Lo que se comparte puede ser etiquetado(tagging), buscando con ello generar una taxonomía a partir de la colaboración de los mismos usuarios.

En el muro se muestra los twitts recuperados desde las cuentas de twitter y delicious que el usuario tenga asociado en el sistema.

Anexo III. Figura 10. Interface del Directorio de Usuarios correspondiente al Muro

Las publicaciones del muro pueden ser etiquetadas como se mencionó anteriormente, también pueden ser comentadas por los diferentes usuarios del directorio (Anexo III. Figura 11). Para agregar comentarios es necesario hacer click sobre el texto *"Agregar comentario"*. Los comentarios se actualizan en el momento en que se hace almacena el mismo.

Anexo III. Figura 11. Interface del Directorio de Usuarios en la cual se ejemplifica que una entrada del muro puede ser comentada y taggeada

Las publicaciones realizadas en el muro pueden ser removidas haciendo click sobre el icono "x" (Anexo III. Figura 12).

Anexo III. Figura 12. Interface del Directorio de Usuarios en la que se muestra que una entrada puede ser borrada al hacer clic sobre el botón "x".

Blog

Muestra información que es recuperada de los blogs públicos (Blogger y Wordpress) en los cuales el usuario haya realizado la configuración(Ver configuración de cuenta de usuarios). Esta información es almacenada en el directorio de Usuarios con el propósito que otros usuarios pueden efectuar comentarios sobre el mismo, y marcar el

contenido con etiquetas(tagging). En la figura Anexo III. Figura 13 se muestra la interface correspondiete a la pestaña Blog del Directorio de Usuarios.

Anexo III. Figura 13. Interface del Directorio de usuario correspondiente a la pestaña Blog

Mensajes

Es una aplicación de correo entre usuarios del Directorio de Usuarios. Esta aplicación emplea la base de datos para almacenar los mensajes entre los usuarios, es decir, no se emplean protocolos como POP3 ySMTP.

La aplicación dispone de una bandeja de entrada (Anexo III. Figura 14), donde se muestran los mensajes que van ingresando a la cuenta del usuario. Para visualizar el mensaje es necesario hacer click sobre el *asunto del mensaje*.

Anexo III. Figura 14. Interface del directorio de usuario en la cual se muestra activa la pestaña Bandeja de Entrada

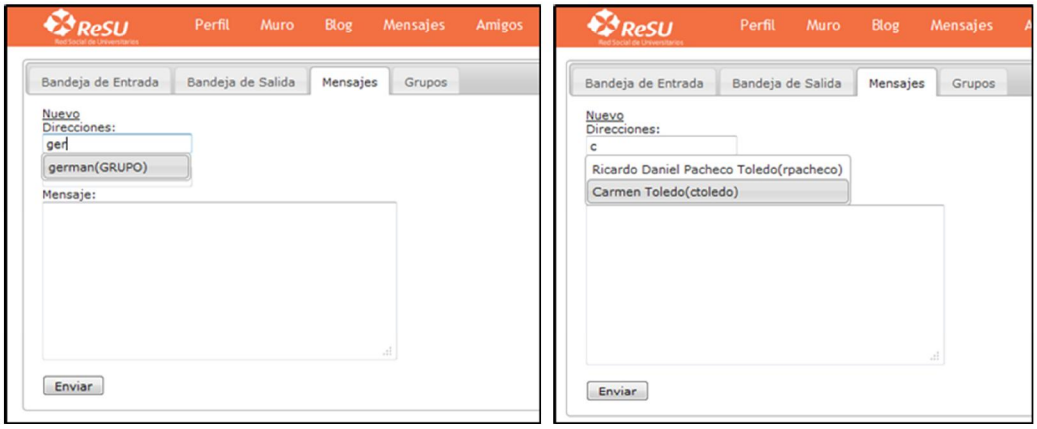
También se dispone de una bandeja de salida en la cual se registran los mensajes que envío el usuario a otros usuarios del Directorio de Usuarios. Para ver el mensaje que fue enviado, y los destinatarios a los cuales se les envío es necesario hacer click sobre el *asunto del mensaje.*

Anexo III. Figura 15. Interface del Directorio de Usuario en la cual se muestra activa la Bandeja de Salida

Haciendo click tanto en los mensajes de entrada como en los de salida se visualiza una interface similar a la mostrada en la figura Anexo III.Figura 16.

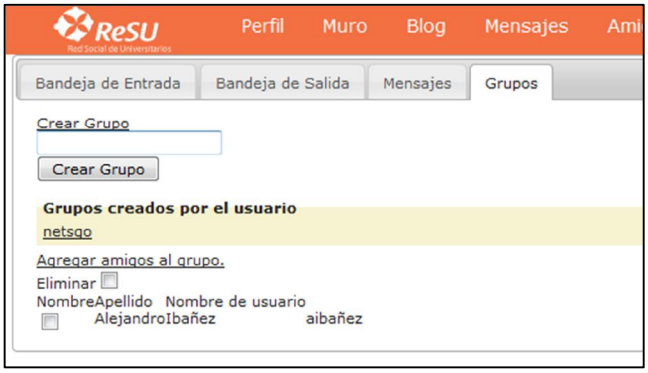
Anexo III. Figura 16. Interface del Directorio de Usuario en la cual se presenta un ejemplo de un mensaje tipo

Para enviar mensajes es necesario, hacer click sobre la palabra "Nuevo". Allí se cargará un formulario que permitirá el ingreso de los datos necesarios para enviar un mensaje. Los mensajes pueden ser enviados seleccionado grupos o usuarios amigos. En el campo direcciones, al escribir como mínimo dos letras se filtraran todos aquellos usuarios y/o grupos que coincidan con las letras ingresadas (Anexo III. Figura 17). Luego de ingresar el título y la descripción del mensaje se puede presionar el botón "Enviar" para mandar el mensaje al destinatario que se selecionó.



Anexo III. Figura 17. Interface del Directorio de Usuario en la cual se visualiza la selección del usuario a quien se le enviará un mensaje

En esta sección de mensajes es posible crear grupos de usuarios. Para ello es necesario primeramente crear el grupo y posteriormente asignarles los usuarios. Sólo es posible asociar usuarios amigos al grupo. Para crear un grupo se presiona sobre el link"Crear Grupo" (Anexo III. Figura 18).



Anexo III. Figura 18. Interface del Directorio de Usuario en la cual se muestra la pestaña para la creación de gripos

Para asociar usuarios amigos al grupo sedebe presionr sobre el *nombre del grupo*, y luego hacer clic sobre*"Agregar amigos al grupo"*. En ese momento se mostrarán todos los amigos del usuario. Posteriormente se deben marcar a todos aquellos usuarios que se deseen agregar al grupo. Por último se requiere hacer clic sobre*"Asociar al grupo"* para almacenar los cambios (Anexo III. Figura 19).

Anexo III. Figura 19. Interface del Directorio de Usuario en la cual se muestra la selección de usuarios para agregar a un grupo

Es posible también eliminar usuarios de un grupo. Para ello se debe marcar aquellos usuarios que se desean eliminar y luego se hace clic sobre el link*"Eliminar"* (Anexo III. Figura 20).

Anexo III. Figura 20. Interface del Directorio de Usuarios en la cual se muestra la manera de proceder para eliminar un usuario de un grupo

Amigos

Los amigos de un usuario son listados en la página que se despliega al hacer clic sobre la pestaña *"Amigos"* (Anexo III. Figura 21). Cada usuario es destacado con su nombre, apellido y una de las carreras que tenga asociada. Es posible remover usuarios presionando sobre el texto del enlace *"Eliminar"*.

Anexo III. Figura 21. Interface del Directorio de Usuarios en la cual se muestran los amigos de un usuario (Pestaña Amigos)

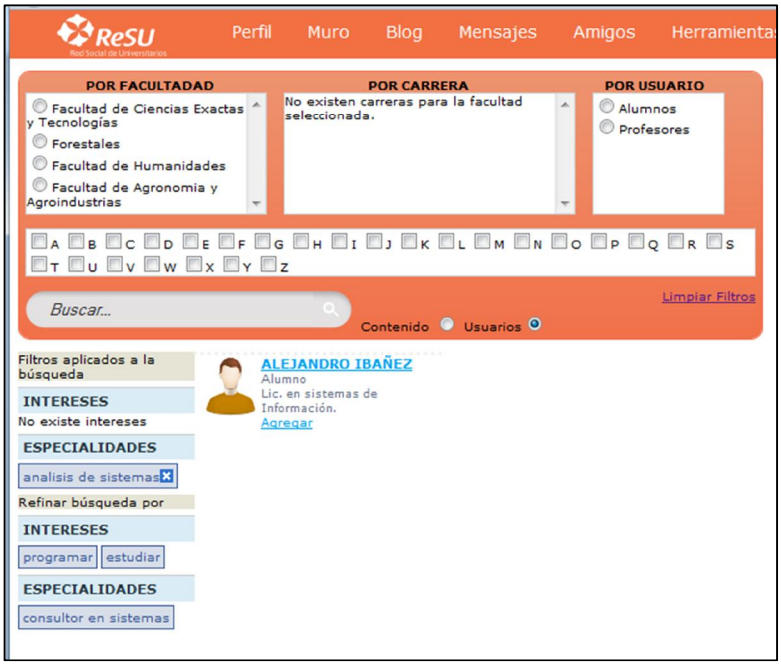
Búsquedas

La interface de búsquedas permite efectuar la búsqueda de contenidos y de usuarios que existen en el Directorio de Usuarios (Anexo III. Figura 22). Se encuentra disponible un conjunto de filtros que permiten encontrar contenidos con un mayor número de coincidencias. Entre los filtros disponibles se encuentran: *por Facultad*, *por carrera*, *por profesor o alumno* y todos aquellos contenidos/usuarios que inicien con una determinada *letra*.

Para buscar contenido es necesario marcar la opción contenido. Si se desea encontrar usuarios es necesario marcar la opción usuarios. También es posible limpiar todos los filtros presionando sobre el enlace *"Limpiar filtros"*.

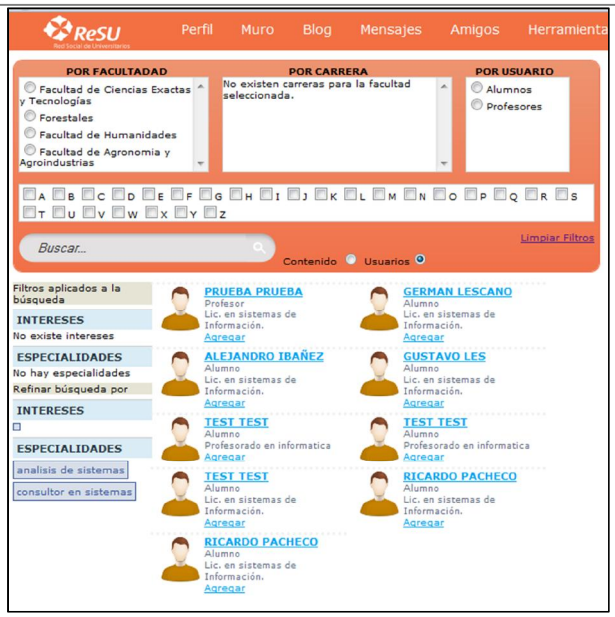
Anexo III. Figura 22. Interface del Directorio de Usuarios correspondiente a la pestaña "Herramientas de Búsqueda"

Cuando se obtiene resultados de usuarios en las búsquedas, se presenta un nuevo filtro según los intereses y especialidades de los usuarios. Para filtrar por un interés o especialidad es necesario hacer click sobre el enlace correspondiente. Estos filtros también pueden removerse presionando sobre la "x".



Anexo III. Figura 23. Interface del Directorio de Usuarios en la cual se muestran los filtros adicionales cuando se efectúan búsquedas por usuario

Cuando se realiza una búsqueda de usuarios, allí se muestran como resultados todos aquellos usuarios que aún no sean amigos del usuario. Es posible agregarlos como amigos presionado sobre el enlace *"Agregar"* (Anexo III. Figura 24).

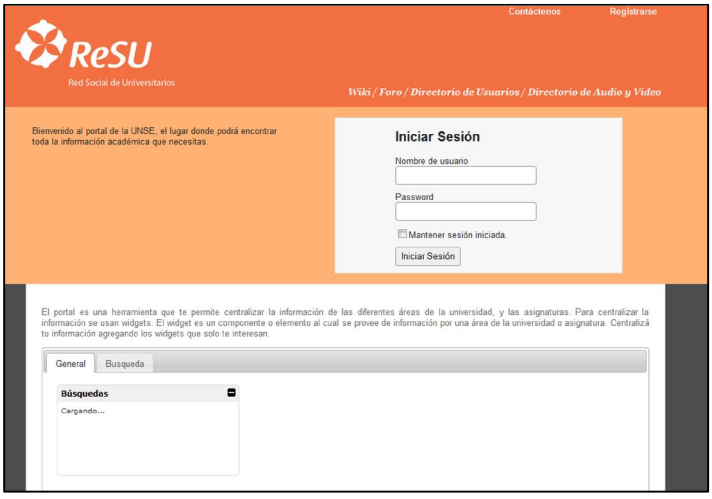


Anexo III. Figura 24. Interface del Directorio de Usuario en la cual se muestra el enlace "Agregar" cuando un usuario no es amigo del que realiza la búsqueda

En las búsquedas de contenido, los resultados pueden ser etiquetados y comentados de igual forma como se realiza con el contenido del muro.

Portal:

Esta aplicación se carga a partir de la url *http://betatesting.com.ar/portal*, la misma presenta en su pantalla inicial (Anexo III. Figura 25) las interfaces de inicio de sesión del usuario, el enlace para crear una nueva cuenta de usuario, las pestañas que destacan los widgets por defecto, la pestaña de búsqueda potenciada por una red neuronal y las recomendaciones de contenidos en caso de que existan para un usuario en particular.



Anexo III. Figura 25. Página de inicio del portal

Una vez iniciada la sesión se mantiene información asociada al usuario.En este punto el usuario puede realizar personalizaciones de lo que ve en su portal, ya sea crear pestañas así como cargar determinados widgets sobre cada pestaña.

Para crear una nueva pestaña basta con pulsar sobre el botón *"Añadir pestaña"*(Anexo III. Figura 26). El número máximo de pestañas que puede tener el usuario son seis incluyendo la pestaña de búsqueda (pestaña fija, no removible).

Anexo III. Figura 26. Interface del portal en la cual se muestran dos nuevas pestañas agregadas

Es posible también remover las pestañas creadas, presionando el botón secundario en la pestaña activa (Anexo III. Figura 27). Al remover la pestaña son removidos los diferentes widgets que la misma tuviese. El nombre de la pestaña puede ser cambiado siguiendo el mismo procedimiento.

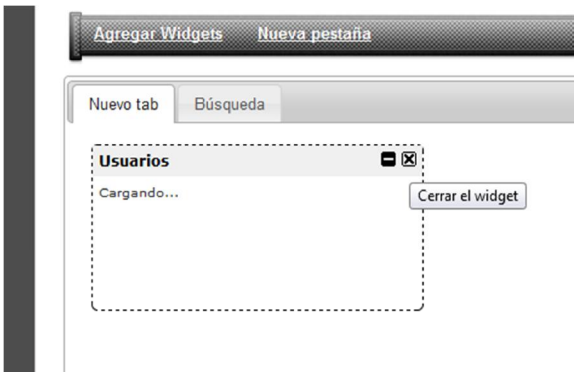
Anexo III. Figura 27. Interface del potal en las cuales se muestran las opciones para eliminar una pestaña o bien cambiarle el nombre de la misma

A las pestañas asociadas al usuario (excepto la de "Búsqueda") es posible añadirle widgets del conjunto de widgets disponibles. Un widget sólo puede estar asociado a una

pestaña. Para añadir un widget se presiona el botón *"Agregar widget"*, y luego se mostrará un panel con el conjunto de widgets disponibles para ser asociados.

Anexo III. Figura 28. Interface del portal en la cual se muestra la agregación de widgets en una pestaña

El widget puede ser removido, presionando en el boton cerrar del mismo "x" (Anexo III. Figura 29).

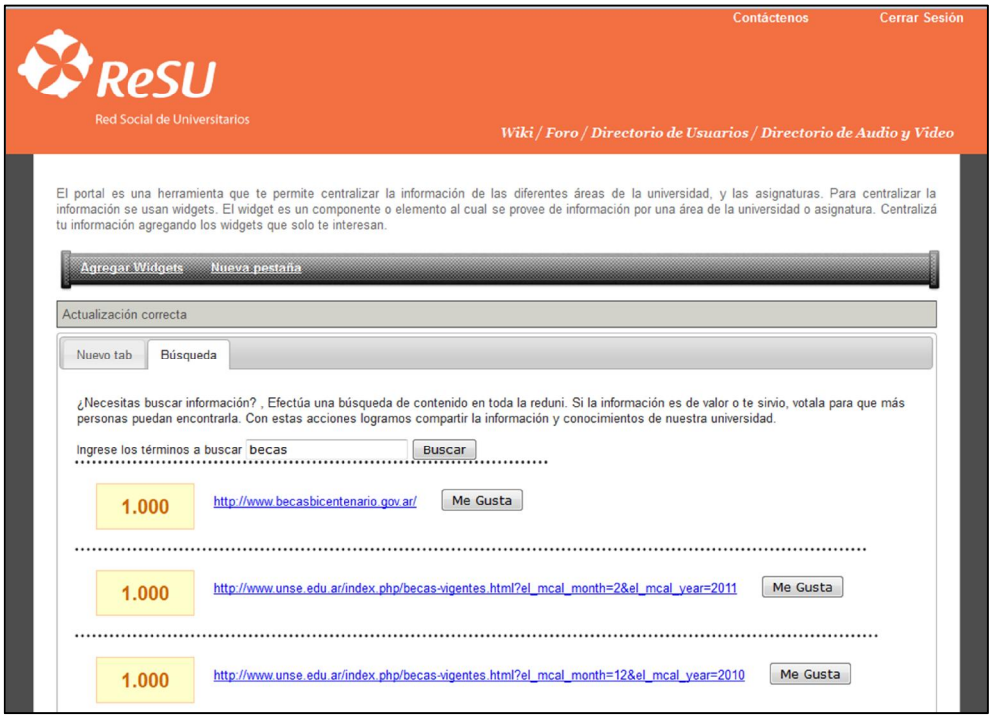


Anexo III. Figura 29. Botón cerrar de un widget del portal

Los widgets pueden ser reorganizados arrastrándolos y soltándos dentro de la pestaña que los contiene. En la figura Anexo III. Figura 30 se muestra una esquematización del movimiento del widget de una posición a otra.

Anexo III. Figura 30. Re-localización de un widget sobre una pestaña del portal

La pestaña "Búsqueda" (Anexo III. Figura 31) dispone de una caja de texto que permite introducir el texto a buscar en todo los contenidos creadosa través de las diferentes aplicaciones. Los resultados entregados mostrarán un enlace al contenido, la valoración estimada por las métricas de búsqueda programadas y un botón que permite votar al usuario permitiéndole de esta manera colaborar en el entrenamiento de una red neuronal a fin de que la misma aprenda constantemente a partir de estas interacciones.



Anexo III. Figura 31. Interface de la página que se despliega al hacer clic sobre la pestaña "Búsqueda" del portal

Noticias

La aplicación de noticias es una aplicación que permite gestionar la carga de novedades para una determinada área de la universidad. Estas novedades, son mostradas en los diferentes widgets que se ponen a disposición a través del portal.

La pantalla inicial de esta aplicación para la gestión de novedades es la de inicio de sesión (Anexo III. Figura 32). Las funciones se habilitarán según sea el usuario un administrador o no.

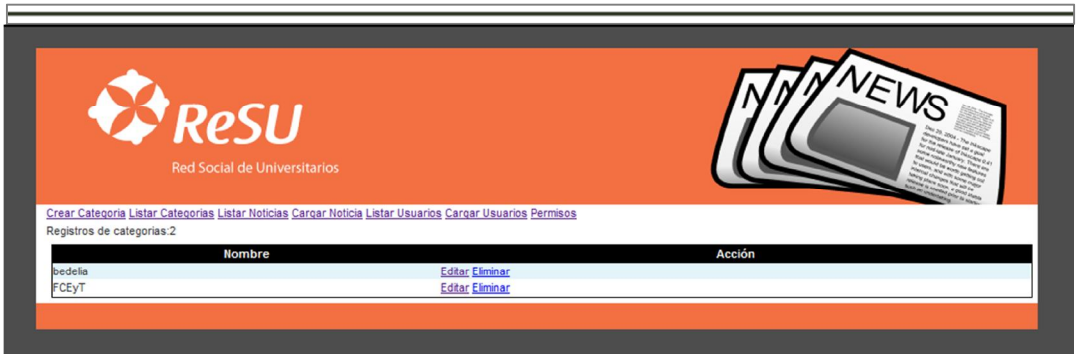
Anexo III. Figura 32. Página de inicio de la aplicación de gestión de novedades

- ***Categorías***

En las categorías se cargan las diferentes áreas que desean publicar novedades. Para ello se ingresa el texto de la categoría y se guarda en el sistema. Las categorías son gestionadas por el administrador del sistema (Anexo III. Figura 33).

Anexo III. Figura 33. Carga de categorias sobre las cuales se puede cargar novedades

También es posible visualizar todas aquellas categorías existentes, a las cuales se les puede efectuar acciones de modificación o eliminación de registros. Para modificar o eliminar registros es necesario hacer click sobre la acción correspondiente (Anexo III. Figura 34).

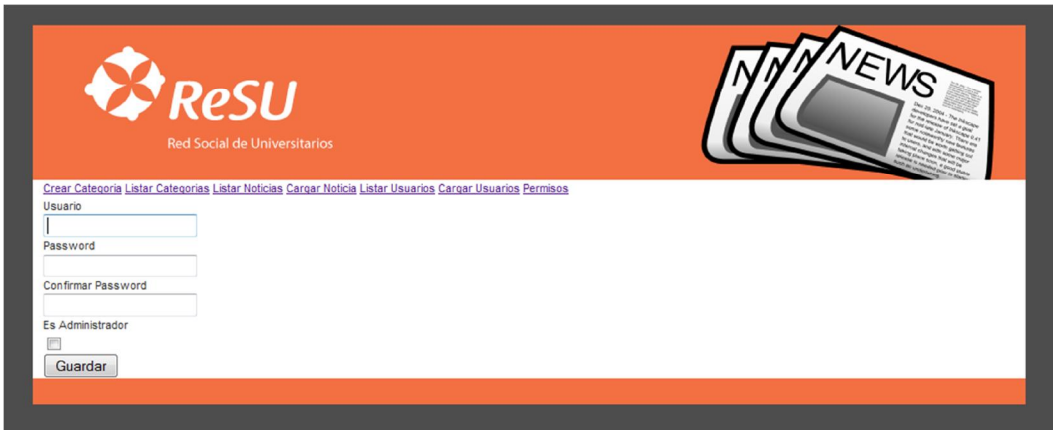


Anexo III. Figura 34. Interface para la edición o eliminación de categorías

- ***Usuarios***

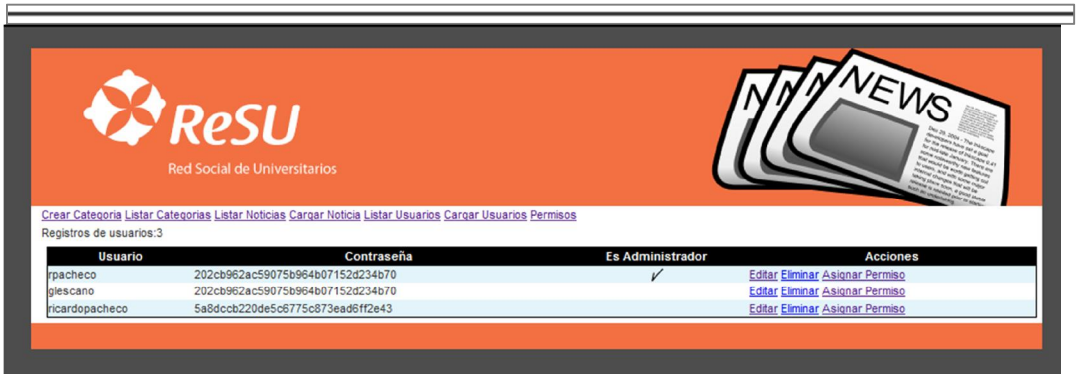
A través de esta sección de la aplicación para la gestión de novedades se permite cargar, modificar y eliminar usuarios con el propósito de poder operar el sistema de noticias.

Para cargar usuarios (Anexo III. Figura 35) es necesarios agregar datos básicos como el nombre y apellido, el nombre de usuario de la aplicación, la contraseña y si el mismo tendrá permisos de administrador.



Anexo III. Figura 35. Creación de cuentas de usuario en la aplicación de gestión de novedades

También es posible visualizar todos aquellos usuarios existentes, a las cuales se les puede efectuar acciones de modificación o eliminación de registros. Para modificar o eliminar registros es necesario hacer click sobre la acción que según corresponda (Anexo III. Figura 36). El procedimiento de modificación es similar al de ingreso.



Anexo III. Figura 36. Edición y eliminación de cuentas de usuario

- ***Permisos***

La interfaz de permisos permite agregar permisos a usuarios, es decir, indicarles a los mismos sobre qué categorías o áreas podrán realizar la carga de noticias.

Para visualizar los permisos que tienen los usuarios es necesario presionar sobre el texto *"Permiso"* en el menú. Los permisos pueden ser removidos presionando sobre el enlace*"Eliminar"*.

Anexo III. Figura 37. Permisos asignados a usuarios en la aplicación de gestión de novedades

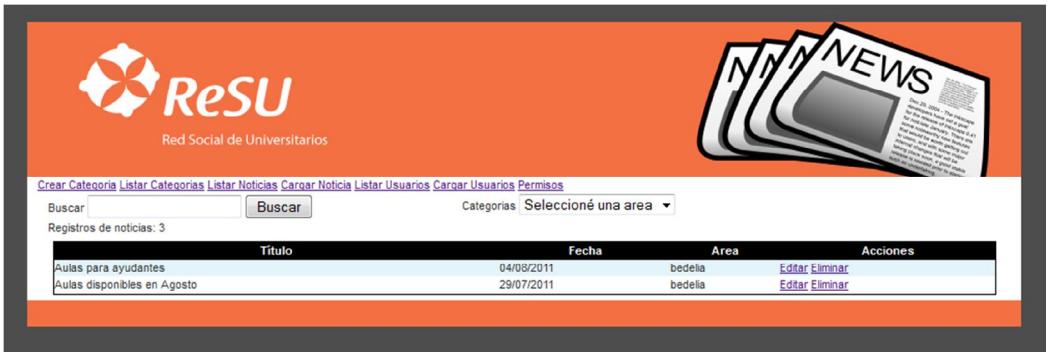
Para crear un permiso se requiere presionar sobre el enlace*"Nuevo permiso"*(Anexo III. Figura 38), allí se muestra un formulario con las categorías y los usuarios disponibles de los cuales se deben seleccionar los que se desean, y presionar en guardar para generar el permiso. En caso de que el permiso ya exista, el sistema informarásobre ese evento.

Anexo III. Figura 38. Asigación de permisos a usuarios en el portal

- ***Noticia***

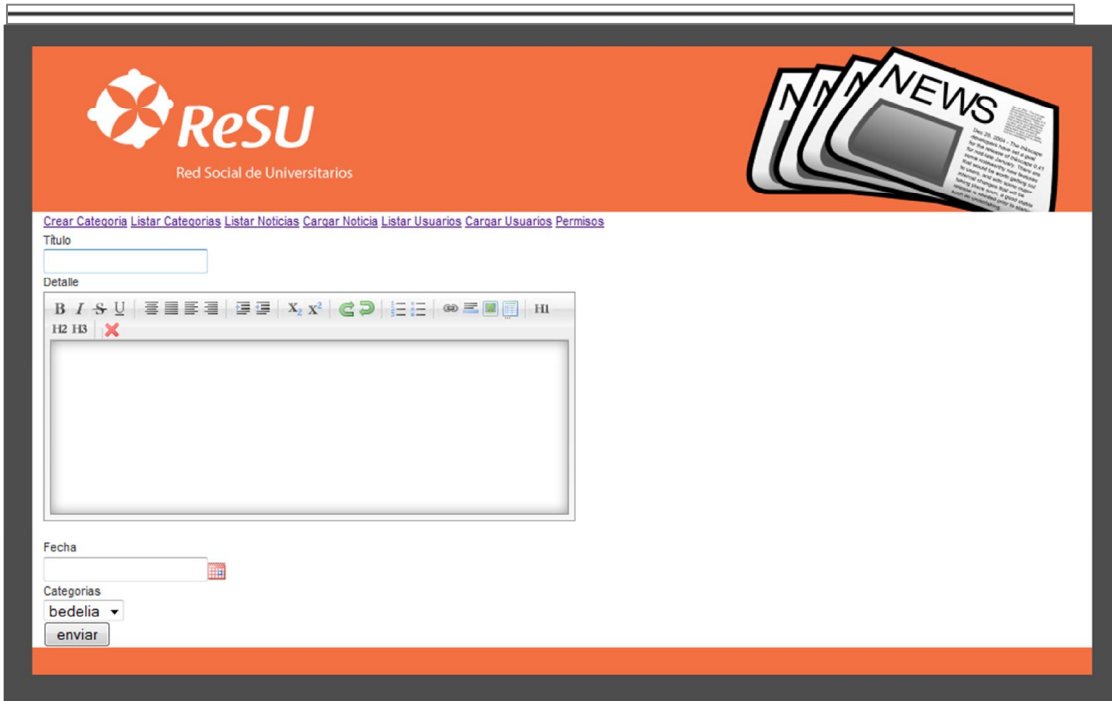
En la interfaz de noticia, se cargan las noticias de las diferentes áreas o categorías. Esta función es administrada por usuarios que no necesariamente sean administradores.

Para visualizar todas las novedades que el usuario cargo se requiere presionar sobre el enlace *"Listar noticias"*(Anexo III. Figura 39). Para modificar o eliminar registros es necesario hacer click sobre la acción que según corresponda.



Anexo III. Figura 39. Interfaz en la que se muestra noticias cargadas en la aplicación

Para cargar una nueva novedad es necesario, presionar sobre el texto del enlace *"Menú crear noticia"*. Se cargará la interfaz, la cual ofrece campos para el ingreso de datos (Anexo III. Figura 40). Entre ellos es necesario ingresar el título de la noticia, la descripción, la fecha, y la categoría en la que se desea cargar la noticia. Si el usuario sólo puede cargar en una categoría, tendrá una única opción en la categoría.



Anexo III. Figura 40. Interfaz para la carga de noticias

ANEXO III

CASOS DE PRUEBA

1. PORTAL

Propósito	Tipo de Prueba	Criterios / Entradas	Salida/s Deseada/s	Salida/s Obtenida/s	Observaciones
Prueba de la página de inicio	Prueba de Interface	No se aplica	Se cargan los widgets por defecto	Carga los widgets por defecto., pero no su contenido.	Analizando el por qué no se cargaba el contenido de los widgets por defecto se detecto que habían problemas de direccionamiento. Los enlaces definidos en la interface no estaban direccionando correctamente al momento de ejecutar la prueba (Problema corregido). En la pestaña de búsqueda, el formulario está direccionando mal en el *action*. Luego de haber corregido el problema anterior se detectó que el buscador no estaba funcionando porque las cadenas de conexión no eran

					las correctas.
Prueba de inicio de sesión (reutilizado en el directorio de usuarios)	Partición Equivalente	**Nombre de usuario y contraseña.** Conjunto: • No se proporciona estos datos e intenta entrar sin ningún usuario (entrada no válida); • Proporciona un nombre de usuario y password válido • Proporciona un nombre de usuario y password no válido • Se intenta iniciar con datos de una cuenta valida pero inactiva.	En caso de que el usuario no proporcione credenciales correctas no debería poder ingresar al sistema, caso contrario el usuario obtiene el acceso.	Cuando no se le proporciona nombre de usuario ni password no permite el ingreso. Al proporcionar un nombre de usuario correcto pero un password no válido no permite el ingreso. Al proporcionar un nombre de usuario y password válido permite el ingreso al portal. Cuando se proporciona un usuario válido con su password pero estando su cuenta inactiva permite el ingreso al sistema. (Problema detectada)	El problema detectado con la prueba fue debidamente solucionado. No se muestra un mensaje cuando el usuario no pudo iniciar sesión. En este sentido la interfaz puede ser mejorada.
Prueba del formulario de registración	Partición Equivalente	**Nombre** Conjunto: Sólo caracteres alfanuméricos **Apellido**	En caso de que los datos proporcionados en el formulario no sean correctos debe mostrar el mensaje de	***Análisis para el caso erróneo:*** No identifica caracteres no permitidos en el nombre y apellido.	

		Conjunto: Sólo caracteres alfanuméricos **Nombre de usuario:** Rango: mínimo de seis caracteres Conjunto: caracteres alfanuméricos incluidos el punto (.) y el guión bajo (_) Lógico. El nombre de usuario no debe estar utilizado por otra persona. **Password:** Rango: mínimo de seis caracteres Conjunto: caracteres alfanuméricos más los siguientes caracteres especiales ($, %, #, &,.,_). Se permite los espacios en blanco **Confirmación de password:** Lógica: Tiene que ser igual al password ingresado anteriormente **Email:**	error correspondiente. Si todos los datos son correctos, se debe crear un registro en la base de datos correspondiente al usuario y se le debe enviar un mensaje de correo informándole que su cuenta fue creada y necesita activarla al hacer clic en el enlace que se le proporciona en el mensaje. Además del envío del correo se deben asociar los widgets por defectos del portal con la cuenta del usuario.	En el caso del mail, lo reconoce como no valido pero cuando se lo altera a megatron@tron lo reconoce como valido faltándole el dominio. En los passwords reconoce que no son iguales. Al mandar las dos iguales (124df) detecta que no cumple con la cantidad mínima de caracteres. Detecta que el nombre de usuario no está disponible. Detecta cuando no se selecciona una carrera. ***Análisis para el caso correcto:*** Se obtiene los resultados esperados: se manda el mensaje, se registra al usuario en espera de activación de su cuenta y se le crean	

		Conjunto: caracteres alfanuméricos más los siguientes caracteres especiales (@,.,_). Lógica: La secuencia de caracteres debe ser de la forma [w+]@[w+].w{3}. Donde *w* es un carácter alfanumérico. **Selección de la Facultad:** Conjunto: {Fac. de Cs. Exáctas, Fac. de Humanidades, Fac. de Agronomía o Fac. de Forestales} ***Caso Erróneo:*** Nombre: Far@g (no esta permitido el carácter "@") Apellido: mega_tron (no está permitido el carácter "_") Usuario: german.lescano (usuario repetido) Pasword: 124df		los widgets por defecto en el portal.	

		Confirmación password: 012df (no coincide con el password anterior) Email: mega@@tron (Doble arroba y no tiene el dominio) Facultad: no elige ***Caso Correcto:*** Nombre: Emanuel Osvaldo Apellido:Perez Usuario: eperez Pasword: prueba@123 Confirmación password: prueba@123 Email: eperez@gmail.com Facultad: Ciencias Exactas y Tecnologías			
Prueba del portal una vez iniciada la sesión.	Prueba de interface	-	Se cargan las pestañas definidas por el usuario. Por cada pestaña definida por el usuario se cargan los widgets	Carga las pestañas definidas por el usuario. Respeta las personalizaciones realizadas por el usuario.	

			que haya elegido y se los muestra como los había configurado (según personalizaciones efectuadas) Se realizan recomendaciones de links (en caso de que haya) Se realizan recomendaciones de entradas de la wiki Se realizan recomendaciones de lectura de tópicos del foro.	Al momento de realizar la prueba no estaba implementado el módulo de recomendaciones.	
Prueba para agregar una pestaña	Prueba de interface	**Nombre de la pestaña:** Conjunto: solo caracteres alfanuméricos **Prueba de un caso no válido:** Nombre de pestaña: mi@pestaña (no se debe permitir	Se abre un cuadro de dialogo solicitando el nombre de la pestaña. Se crea la pestaña y se da a elegir los widgets que se desean agregar	No controla el hecho de que no se permiten caracteres especiales en el nombre de la pestaña. No da a elegir los widgets a agregar. Es poco intuitivo. A pesar de esto, al hacer clic en el botón para agregar	Cuando se hace clic para agregar una nueva pestaña, recarga la página y no lo agrega. Esto paso al querer agregar la cuarta pestaña, siempre estando posicionado en la primera pestaña. Permite cerrar la pestaña de Busqueda cuando no debería porque es una pestaña por

		caracteres especiales) **Prueba de un caso válido:** Nombre de pestaña: Noticias Académicas		widgets muestra los disponibles, y cuando se elige uno si agrega correctamente.	defecto.

2. DIRECTORIO DE USUARIOS

Propósito	Tipo de Prueba	Criterios / Entradas	Salida/s Deseada/s	Salida/s Obtenida/s	Observaciones
Prueba de la pestaña relacionada al perfil	Prueba de Interface	-	Se carga la información relacionada a correo electrónico, teléfono y sitio web (*). Se cargan los intereses y especialidades del usuario (*). Se cargan los antecedentes laborales (*). Se cargan los antecedentes educativos (*). Se cargan los antecedentes en investigación (*). Se recuperan las marcas que	Se obtuvieron las salidas esperadas. Se controló bien de no mostrar los enlaces para edición en caso de que no se fuese dueño del perfil.	Al momento de ejecutar la prueba no había la cantidad suficientes de usuarios para que el algoritmo de recomendaciones puediera hacer una. Este algoritmo se lo probó de manera independiente funcioando correctamente. Máas adelante se detalla esta prueba.

			le fueron asignadas al usuario por parte de la comunidad (*). Se muestra recomendaciones de usuarios. Se muestra una nube de palabras definida en función a los tags que usó el usuario para marcar contenidos. (*) En caso de ser definidos Nota: Si el usuario está viendo su propio perfil deberían mostrarse los enlaces para poder editar la información que se hace disponible. El usuario no debería poder editar los datos de otra persona.		
Prueba de la pestaña relacionada al muro	Prueba de interface	-	Recupera los posteos efectuados por el usuario (del propio directorio, de Twitter y de Delicious). Recupera los comentarios asociados a cada posteo (en caso de tener comentarios).	Se muestran los resultados deseados: se recuperan los posts señalados, los comentarios asociados a cada post y los tags que tenga asignado.	No se muestra de manera correcta los comentarios (hay un problema de estilos). Al momento de realizar la prueba se detecta que no se estaban mostrando

			Recupera los tags de cada posteo (en caso de habérsele asignado).		los posteos que se traen desde deliciious.
Prueba de la pestaña relacionada a Blogs.	Prueba de Interface	-	Se muestran fragmentos de entradas realizadas por el usuario en sus cuentas de Blogger o Wordpress. Se recuperan los comentarios que le hayan realizado los usuarios. Se recuperan los tags asignados a dichas entradas.	Se obtiene la salida deseada: recupera las entradas desde los blogs, los comentarios que los usuarios le hayan realizado y los tags que le fueron asignados.	Se observan problemas de estilo (la ficha del "ítem") no se muestra claramente. Por otro lado, se observan caracteres "extraños". La entrada correspondiente al blog no se le entiende porque lo que recupera parece sin sentido.
Prueba de envío/recepción de mensajes	Prueba de interface	-	Se recuperan los mensajes que haya recibido un usuario. Se recuperan los grupos que haya creado el usuario. En caso de enviar un mensaje, comprobar que el mensaje fue efectivamente recibido.	Envia correctamente los mensajes (llegan al destinario). Se reciben de manera satisfactoria los mensajes. No carga los grupos creados por un usuario (problema detectado).	Se detecta que es necesario desarrollar una función que permita responder a mensajes.
Prueba de la pestaña	Prueba de	-	Se recuperan los contactos	Se obtiene los	

amigos	interface		del usuario que fueron agregados como amigos.	resultados esperados: mostrar la lista de amigos del usaurio.	
Prueba de la herramienta de búsqueda del directorio de usuarios.	Prueba de interface		Buscar contenidos o usuarios por facultad. Buscar contenidos o usuarios por facultad y carrera. Buscar contenidos o usuarios por facultad, carrera y tipo de usuario. Buscar contenidos o usuarios por facultad, carrera, tipo de usuario y filtro alfabético. Sobre los resultados arrojados por las búsquedas anteriores aplicarle filtro sólo por intereses, luego sólo por especialidades y finalmente aplicar ambos filtros.	Busqueda de contenidos o usuarios por facultad: sin problemas Busqueda de contenidos o usuarios por facultad, carrera y tipo de usuario: sin problemas. Busqueda de contenidos o usuarios por facultad, carrera, tipo de usuario y filtro alfabetico: funciona correctamente solo que permite tildar dos o más letras lo cual posteriormente hace que lance un error.	

3. PRUEBA DEL MÓDULO DE RECOMENDACIÓN

Propósito	Tipo de Prueba	Criterios / Entradas	Salida/s Deseada/s	Salida/s Obtenida/s	Observaciones
Prueba del módulo de recomendación como componente independiente	Partición de Equivalencia	Prefs Es un arreglo asociativo. La clave de este arreglo sólo acepta caracteres alfanuméricos. El valor asociada a esta clave es otro arreglo asociativo. Este segundo arreglo asociativo se compone de una clave que sólo puede contener caracteres alfanuméricos y el valor asociado a esta clave tiene que ser un valor entero. **Person** Conjunto: Sólo caracteres alfanuméricos.	En caso de que haya recomendaciones un arreglo asociativo en donde la clave sólo acepta caracteres alfanuméricos (identifica a un usuario) y el valor asociado a esta clave debe ser un valor entero que representa el grado de semejanza.	Sedetectó que en la recomendación se recomendaba asimismo a un usuario. A pesar de las mejoras detectadas, el algoritmo realiza las recomendaciones como se espera del análisis del algoritmo codificado.	Al empezar la prueba se observó que a pesar de que existían usuarios con los mismos intereses que un usuario que se uso para la prueba, no se lo mostraba en el arreglo como potenciales usuarios a recomendar. Se podría mejorar el mecanismo de recomendación puesto que se detecta algunas cuestiones semáticas que se pueden tener en cuenta, por ejemplo, si a un usuario le gusta programar, no se detecta a usuarios especialistas en el tema porque fueron tagueados como

					programadores por ejemplo. Se debería hacer que palabras como “programar”, “programador”, “programadores” los reconozca como similares. Otra mejora a introducir es que no recomiende usuarios de los cuales uno ya es amigo.

4. PRUEBA DEL MÓDULO DE CLUSTERING

Propósito	Tipo de Prueba	Criterios / Entradas	Salida/s Deseada/s	Salida/s Obtenida/s	Observaciones
Prueba del módulo de clustering como componente independiente	Prueba de Comparación	El archivo matriz.txt proporcionado en el CD-ROM que se entrega con el trabajo.	Un dendrograma similar al que se muestra en la figura Anexo IV.1	Se obtiene un dendrograma similar. Se adjunta en el CD que acompaña a este trabajo la imagen del dendrograma obtenido.	Para el desarrollo de este módulo nos basamos en un algoritmo de recomendación que estaba realizado en Python. Dicho modulo se encuentra debidamente probado con el archivo matriz.txt. Por esta razón

					decidimos hacer una prueba de comparación, puesto que si bien en nuestro trabajo programamos el módulo en PHP ambos algoritmos deberían arrojar los mismos resultados. De ahí que al realizar esta prueba buscamos lograr los mismos resultados.

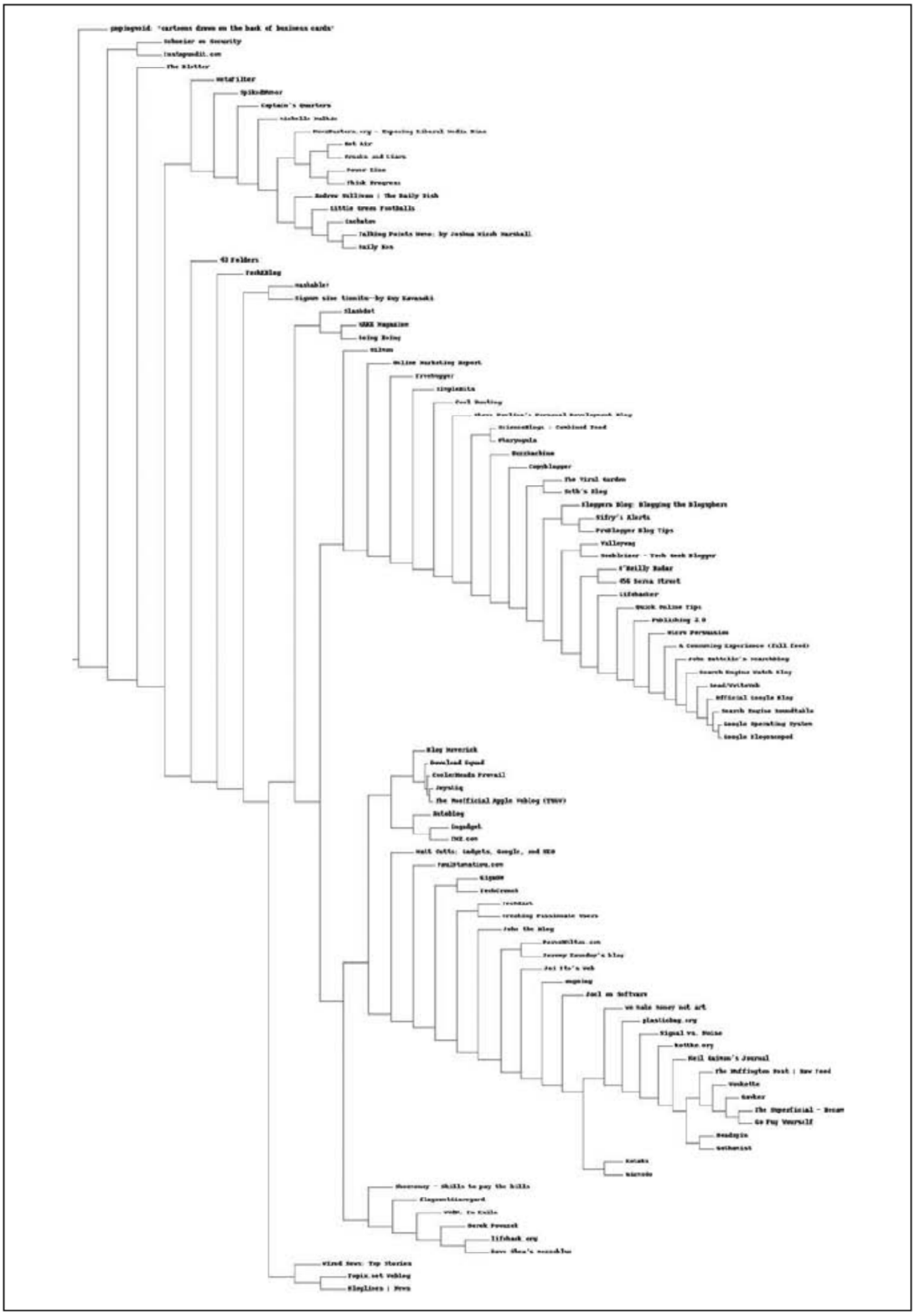


Figura Anexo IV.1. Imagen extraída de [27] pág. 41. Para una mejor visualización de la imagen aconsejamos remitirse al archivo digital del libro en su versión ebook.

5. PRUEBA DEL MÓDULO DE BÚSQUEDA INTELIGENTE

Propósito	Tipo de Prueba	Criterios / Entradas	Salida/s Deseada/s	Salida/s Obtenida/s	Observaciones
Prueba del módulo de búsqueda básico	Prueba de interface	Buscar alguna palabra que se sepa que existe. Se buscó por becas y decanato.	Se obtiene resultados de búsqueda en donde se sugieren páginas con los términos de búsqueda.	Se obtienen resultados de búsqueda para términos que se encuentran indexados. Se detectó que el algoritmo arrojaba error cuando no encontraba resultados.	Se indexó la página de la UNSE.
Prueba del módulo de búsqueda con red neuronal	Prueba de interface	Se busca por becas y al resultado que se considera más relevante se lo vota. Aplicar esto varias veces.	El ítem de búsqueda debe subir a las primeras posiciones del resultado de búsqueda.	Al momento de probar se encontró un error de javascript que producía que nose efectúe la valoración. Luego de corregido el problema detectado, al entrar en la base de datos se pudo constatar que efectivamente el peso de las url	Con esto se hace entrenar la red neuronal para aumentar el peso de una salida.

				votadas aumentaban su valor.	
	Prueba de interface	1º Búsqueda por área de bienestar estudiantil 2º Bésqueda por área de Bienestar Estudiantil y Becas 3º. Entrenamiento para dar peso al mejor resultado.	El mejor resultado de la búsqueda de becas + área de bienestar estudiantil debería figurar primero. Cuando se busque sólo por algunos de estos términos, ese mejor resultado entrenado debería aparecer primero en la lista.	Se consiguen los resultados esperados aunque luego de haber corregido un error que se daba entre la vinculación de los nodos de entrada y salida a través de los nodos ocultos.	Objetivo es hacer relacionar los términos becas con área de bienestar estudiantil. Con esto se ejercita el entrenamiento de la red.

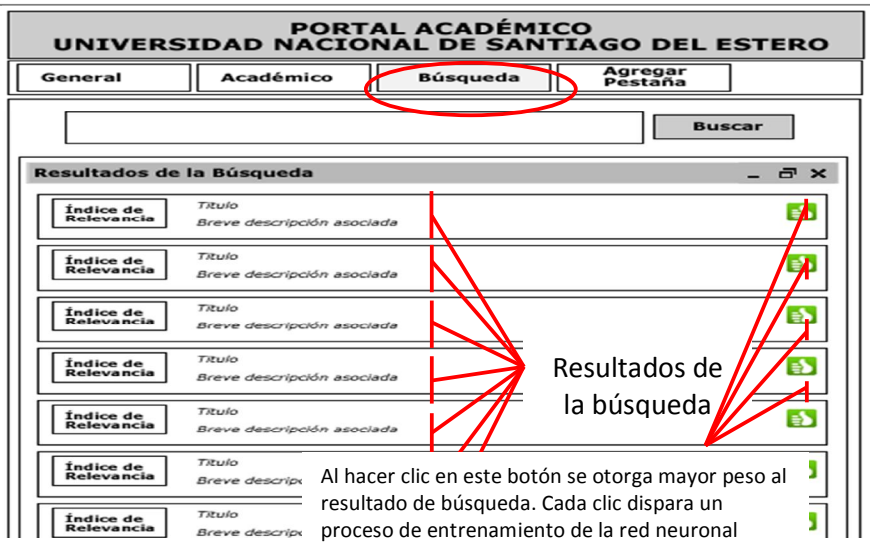
Nota: Los problemas detectados en los casos de pruebas diseñados fueron debidamente corregidos y se comprobó que efectivamente estuviesen funcionando como se esperaba.

ANEXO IV

EXPLICACIÓN PASO A PASO DEL FUNCIONAMIENTO DE LA RED NEURONAL USADA EN LA FUNCIONALIDAD DE BÚSQUEDA

El objetivo de incorporar una red neuronal en el motor de búsqueda desarrollado es ofrecer a los usuarios mejor calidad en los resultados de búsquedas en cuanto a relevancia de los mismos.

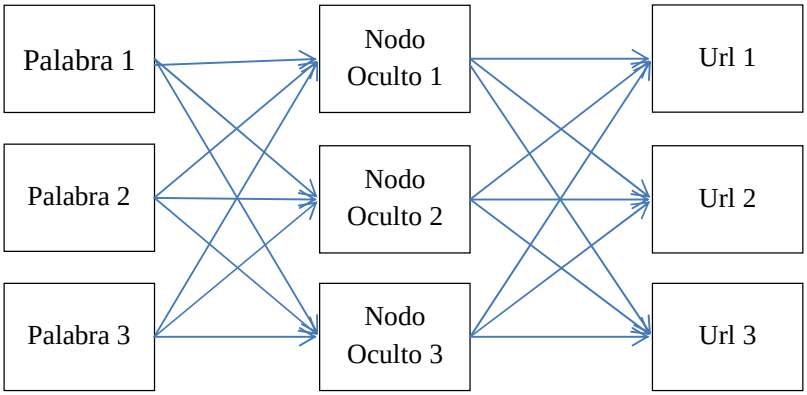
Para lograrlo, en los resultados de la búsqueda se agrega un botón por cada ítem del listado a través del cual el usuario puede manifestar si ese ítem particular le fue de utilidad (Anexo IV. Figura 1). Con este dato se buscó indagar en cómo mejorar la clasificación de los resultados de futuras búsquedas relacionadas, encontrándose que una solución está en el empleo de una red neuronal. Cada vez que el usuario interacciona haciendo clic en el botón mencionado, se dispara por detrás un proceso de entrenamiento de la red.



Anexo IV. Figura 1. Prototipo de la interfaz en la cual se proporciona la funcionalidad de búsqueda potenciada por una red neuronal

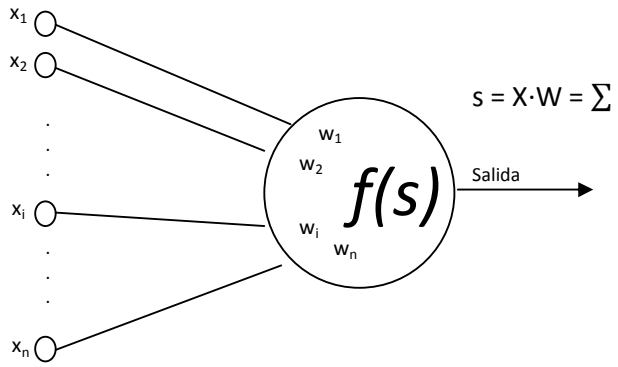
CONFIGURACIÓN DE LA RED DE SEGUIMIENTO DE LAS INTERACCIONES DEL USUARIO

La red neuronal programada consiste de tres capas: la capa de entrada, la capa oculta y la capa de salida. Cada vez que se realiza una consulta, la frase de la consulta es separada en palabras y, cada una de estas palabras actúan como nodos de la capa de entrada. Todos los nodos en la capa de entrada están conectados con todos los nodos de la capa oculta, y todos los nodos de esta última capa están a su vez conectados con todos los nodos de la capa de salida (en este caso particular representarían direcciones web – urls –). En Anexo IV.Figura 2 se ilustra la estructura de la red.



Anexo IV. Figura 2. Estructura de la red neuronal desarrollada

Una representación esquemática de una neurona tipo viene dada en Anexo IV. Figura 3.



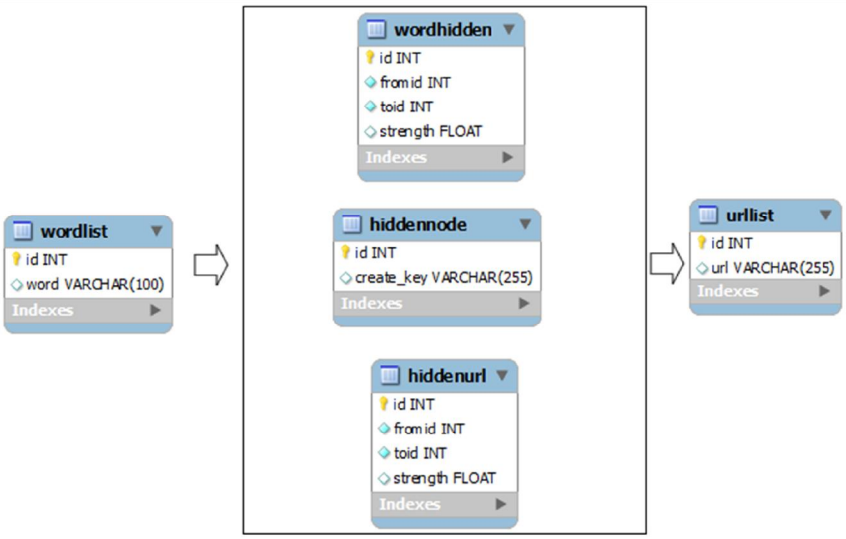
Anexo IV. Figura 3. Ilustración de una neurona tipo

Donde *x1, x2, ..., xi ,..., xn* son las entradas, a lo que se denomina vector de entrada *X*. Por su parte, *w1, w2, ..., wi, ..., wn* representan los pesos de la entrada y se define como vector de pesos *W*. La función *f* que se utilizó en este trabajo es la tangente hiperbólica por el hecho de que comúnmente las redes neuronales utilizan funciones sigmoidales para calcular la salida y además en este caso particular se necesita que la salida este comprendida en el intervalo [0; 1], debiéndose interpretar que en cuanto a la salida esté más cerca de 1, el enlace que corresponde a la misma se encuentra más activo.

Para pedir a la red que devuelva los mejores resultados para una consulta, los nodos de entrada para las palabras en la frase de búsqueda son establecidos en 1 (uno). Las salidas de estos nodos se encienden e intentan activar a los nodos necesarios de la capa oculta. A su vez, los nodos en la capa oculta que reciban entradas lo suficientemente fuertes activarán sus enlaces con los nodos de la capa de salida.

REPRESENTACIÓN DE LA RED COMO BASE DE DATOS

Se definió una tabla para la capa oculta (*hiddennode*) y dos tablas de conexiones (una desde la capa de palabras a la capa oculta (*wordhidden*)y otra desde la capa oculta hacia la capa de salida (*hiddenurl*).



Anexo IV. Figura 4. Esquema de ER que representa la red neuronal en la BD

A continuación se detallan las tablas mostradas en la Anexo IV. Figura 4.

wordlist

Campo	Tipo	Comentario
id	Autonumérico	Es el identificador de una palabra determinada.
word	Alfanumérico	Representa una palabra indexada por el motor de búsqueda

hiddennode

Campo	Tipo	Comentario
id	Autonumérico	Es el identificador de un determinado nodo de la capa oculta.
create_key	Alfanumérico	Representa el nombre de la capa oculta. Son nodos que se crean por cada consulta que se realiza. Para constituir el nombre se enlazan los identificadores de las palabras de la consulta en orden ascendente, unidas por un guión bajo.

wordhidden

Campo	Tipo	Comentario
id	Autonumérico	Es el identificador de una determinada asociación entre la capa de entrada y la capa oculta.
fromid	Entero	Representa el identificador de un determinado registro de la tabla de palabras (*wordlist*).
toid	Entero	Representa el identificador de un determinado registro de la capa oculta (*hiddennode*).
strength	Real	Indica la fortaleza de la conexión entre la capa de entrada y la capa oculta.

hiddenurl

Campo	Tipo	Comentario
id	Autonumérico	Es el identificador de una determinada asociación entre la capa oculta y la capa de salida.
fromid	Entero	Hace referencia al identificador de un determinado registro de la capa oculta (*hiddennode*).

toid	Entero	Representa el identificador de un determinado registro de la capa de salida (*urllist*).
strength	Real	Indica la fortaleza de la conexión entre la capa oculta y la capa de salida.

urllist

Campo	Tipo	Comentario
id	Autonumérico	Es el identificador de un determinado enlace indexado por el motor de búsqueda.
url	Alfanumérico	Representa un enlace o dirección web indexado por el motor de búsqueda.

Para manejar el acceso a la base de datos de la red neuronal se hizo uso de los siguientes métodos. El primero, llamado *getstrength* determina el peso actual de una conexión. Puesto que las nuevas conexiones se crean cuando es necesario, este método devuelve un valor por defecto si no hay conexiones de acuerdo a las siguientes políticas.

- Para asociaciones entre la capa de entrada y la capa oculta, el valor por defecto es -0,2, de modo que, por defecto, las palabras extras tendrán un efecto ligeramente negativo sobre el nivel de activación de un nodo oculto.
- Para enlaces de la capa oculta hacia la capa de salida, el método devolverá por defecto el valor 0 (cero).

En Anexo IV. Figura 5 se muestra el código fuente correspondiente a la función *getstrength*.

```
public function GetStrength($fromid, $toid, $layer){
     if($layer == 0)
       $table = "wordhidden";
     else
       $table = "hiddenurl";
     $query = "SELECT strength FROM $table
               WHERE fromid = $fromid AND toid = $toid";
     $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
     $row = $result->fetch_row();
     if( !$row ){
       if( $layer == 0 ) return -0.2;
       if( $layer == 1 ) return 0;
     }
     return $row[0];
}
```

Anexo IV. Figura 5. Código fuente de la función getstrength

Otro método utilizado es el *setstrength* el cual determina si una conexión ya existe y actualiza o crea la conexión con la ponderación dada. Este método es usado por el procedimiento que se encarga del entrenamiento de la red. El código de la función *setstrength* se muestra en Anexo IV. Figura 6.

```
public function SetStrength($fromid, $toid, $layer, $strength){
     if($layer == 0)
       $table = "wordhidden";
     else
       $table = "hiddenurl";
     $query = "SELECT id FROM $table WHERE fromid = $fromid AND toid =
$toid";
     $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
     $row = $result->fetch_row();
     if(!$row){
       // Si no existe la entrada la da de alta
       $query = "INSERT INTO $table (fromid, toid, strength)
              VALUES ($fromid, $toid, $strength)";
       $this->dbSearchEngine->query($query);
     }
     else{
       $rowid = $row[0];
       $query = "UPDATE $table SET strength = $strength WHERE id = $rowid";
       $this->dbSearchEngine->query($query );
     }
}
```

Anexo IV. Figura 6. Código fuente de la función setstrength

Como se detalló en la tabla correspondiente a la capa oculta (*hiddennode*), para crear nuevos nodos en la misma, se hace uso de la función *generatehiddennode* que crea una conexión con una ponderación por defecto entre la capa de entrada y la capa oculta, y entre la capa oculta y la capa de salida. En Anexo IV. Figura 7 se muestra el código de esta función.

```
public function GenerateHiddenNode($wordids, $urls){
     // $wordids: arreglo de identificadores de las palabras usadas en
     // la frase de búsqueda.
     if( count( $wordids ) > 1 )
        sort( $wordids );   // Ordena de manera ascendente
                            // los identificadores de
                            // las palabras usadas en la frase de búsqueda.

     $createkey = "";
     $count_wordids = count($wordids);
     $cont = 0;
     foreach($wordids as $indice => $wi){
        if($cont < $count_wordids -1)
               $createkey .= (string)$wi . '_';
        else
               $createkey .= (string)$wi;
        $cont += 1;
     }
     $query = "SELECT id FROM hiddennode WHERE create_key = '$createkey'";
     $result = $this->dbSearchEngine->query($query, MYSQLI_STORE_RESULT);
     $row = $result->fetch_row();
     if(!$row){
        //Se crea el nodo oculto puesto que no existe
        $query = "INSERT INTO hiddennode (create_key) VALUES ('$createkey')";
        $this->dbSearchEngine->query($query);
        //Se obtiene el identificador del último registro agregado
        //en la tabla hiddennode
        $hiddenid = $this->dbSearchEngine->insert_id;
        //Coloca el peso por defecto
        foreach($wordids as $indice =>$wordid)
               $this->SetStrength($wordid, $hiddenid, 0, 1/count($wordids));
        foreach($urls as $indice =>$url)
               $this->SetStrength($hiddenid, $url, 1, 0.1);
     }
}
```

Anexo IV. Figura 7. Código fuente de la función generatehiddennode

Dado que la base de datos que representa la red puede ser extremadamente grande, cada vez que se realiza una búsqueda se carga en memoria una porción representativa de la red en función a la consulta efectuada. Para realizar esto se usa una función que busca todos los nodos de la capa oculta que son relevantes a la consulta realizada (los nodos deben estar conectados con algunas de las palabras usadas en la consulta o bien con algunas de las direcciones web mostradas en los resultados). Todos los demás nodos no serán usados para determinar la salida ni en el entrenamiento de la red.

En Anexo IV. Figura 8 se muestra el código de la función que se encarga de seleccionar los nodos ocultos necesarios.

```
public function GetAllHiddenIds($wordids, $urlids){
     $li = array();
     foreach($wordids as $indice => $wordid){
             $query = "SELECT toid FROM wordhidden
                 WHERE fromid = $wordid";
             $result = $this->dbSearchEngine->query($query,
                 MYSQLI_STORE_RESULT);
             while($row = $result->fetch row()){
                     $li[$row[0]] = 1;
             }
     }
     foreach($urlids as $indice => $urlid){
             $query = "SELECT fromid FROM hiddenurl WHERE toid = $urlid";
             $result = $this->dbSearchEngine->query($query,
                 MYSQLI_STORE_RESULT);
             while($row = $result->fetch_row()){
                     $li[$row[0]] = 1;
             }
     }
     return array_keys($li);
}
```

Anexo IV. Figura 8. Código fuente de la función GetAllHiddenIds

Para armar la porción de la red representativa a la búsqueda se usa la función *setupnetwork*. Esta función inicializa las variables referidas a la lista de palabras, los nodos ocultos, las direcciones web, la salida correspondiente a cada nodo, y los pesos de los enlaces entre los nodos. Los pesos se recuperan desde la base de datos usando las funciones definidas previamente. En Anexo IV. Figura 9 se muestra el código fuente de esta función.

```
public function SetupNetwork($wordids, $urlids){
    // Lista de valores
    $this->wordids = $wordids;
    $this->hiddenids = $this->GetAllHiddenIds($wordids, $urlids);
    $this->urlids = $urlids;

    foreach($this->wordids as $indice => $wordid){
        $this->ai[$wordid] = 1; // Nodos de la capa de entrada
    }
    foreach($this->hiddenids as $indice => $hiddenid){
        $this->ah[$hiddenid] = 1; // Nodos de la capa oculta
    }
    foreach($this->urlids as $indice => $urlid){
        $this->ao[$urlid] = 1; // Nodos de la capa de salida
    }

    // Crea la matriz de pesos
    foreach($this->wordids as $indice => $wordid){
        foreach($this->hiddenids as $indice_hidden => $hiddenid){
            // Matriz que almacen información de conexiones entre la capa
de
            // entrada y la capa oculta
            $this->wi[$wordid][$hiddenid] = $this->GetStrength($wordid,
                $hiddenid, 0);
        }
    }
    foreach($this->hiddenids as $indice => $hiddenid){
        foreach($this->urlids as $indice_url => $urlid){
            // Matriz que almacen información de conexiones entre la capa
            //  oculta y la capa de salida
            $this->wo[$hiddenid][$urlid] = $this->GetStrength($hiddenid,
                $urlid, 1);
        }
    }
}
```

Anexo IV. Figura 9. Código fuente de la funcion SetupNetwork

Para calcular las salidas de los nodos de la red (activación de enlaces) en función de las entradas recibidas en la capa de entradas, se desarrolló la función *feedforward*. Una vez que ésta función se ejecuta, en cada enlace de la red se tiene un número real que indica su nivel de activación.

Para calcular el nivel de activación de los enlaces, el algoritmo realiza primero un ciclo sobre todos los nodos en la capa oculta, en el cual por cada nodo, se acumula la sumatoria de las salidas de cada nodo de la capa de entrada por la ponderación de la conexiones que unen estos nodos de la capa de entrada con este nodo de la capa oculta que se encuentra bajo el ciclo principal. La salida de estos nodos es igual a la tangente hiperbólica de la sumatoria calculada. Para calcular la salida de los nodos de la capa de salida se procede de manera similar al cálculo de la salida de los nodos de la capa oculta pero teniendo en cuenta que esta vez la sumatoria se realiza con las salidas de los nodos

de la capa previa, en este caso, la capa oculta. En Anexo IV. Figura 10 se muestra el código de la función *feedforward*.

```
public function Feedforward(){
    // Activación de los nodos de la capa oculta
    Foreach ($this->hiddenids as $indice => $hiddenid ){
      $sum = 0;
      foreach( $this->wordids as $indiceAux => $wordid ){
            $sum = $sum + $this->ai[$wordid] *
                  $this->wi[$wordid][$hiddenid];
      }
      $this->ah[$hiddenid] = tanh($sum);
    }
    // Activación de los nodos de la capa de salida
    foreach( $this->urlids as $indice => $urlid ){
      $sum = 0;
      foreach( $this->hiddenids as $indiceAux =>$hiddenid ){

            $sum = $sum + $this->ah[$hiddenid] *
                  $this->wo[$hiddenid][$urlid];
      }
      $this->ao[$urlid] = tanh($sum);
    }
    return $this->ao;
}
```

Anexo IV. Figura 10. Código fuente de la función Feedforward

ENTRENAMIENTO DE LA RED

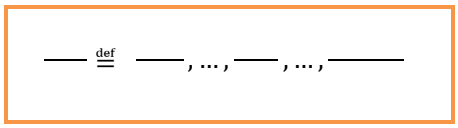
El entrenamiento de la red para el contexto planteado en el trabajo consiste en mostrar a la misma algunos ejemplos reales de lo que las personas buscan, los resultados que se devuelven, y lo que los usuarios manifiestan haberle sido de interés.

Técnicamente durante el entrenamiento de una red se ajustan los pesos variables asociados a las entradas hasta que se consigan las salidas deseadas. El método más utilizado para efectuar estos ajustes en una red neuronal compuesto por una única neurona es el del gradiente descendiente. El objetivo es minimizar una función de error, generalmente la más usada es la del error cuadrático:

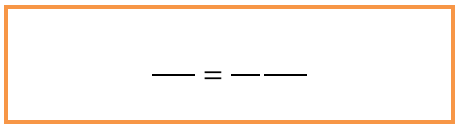
$$\varepsilon = \sum_{\epsilon} (d - f)$$

Donde es la respuesta de la neurona al vector de entrada , es la respuesta deseada y el sumatorio se extiende a todos los vectores del conjunto de entrenamiento. Dado un conjunto de entrenamiento E determinado, se puede comprobar que el valor de ε depende de los valores que tengan los pesos (la dependencia aparece a través del cálculo de).

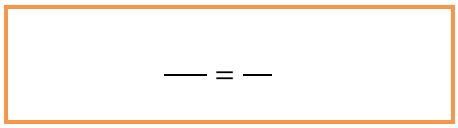
De esto se deriva que el mínimo de ε es:



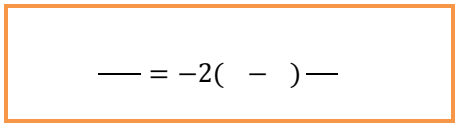
Como los ε′s dependen de W a través del producto escalar $s = X \cdot W$ (para guiarse ver Anexo IV. Figura 3), aplicando la regla de derivación de la cadena, obtenemos:



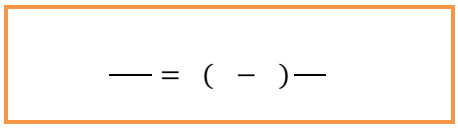
Entonces, dado que $= X$



Teniendo en cuenta que $= -2(d - f)$, se puede escribir:



Si se incluye el factor 2 en un parámetro llamado *factor de aprendizaje, c*, la formula anterior queda de la siguiente manera:



Para el caso de redes multicapa, el algoritmo usado para el entrenamiento de la red es conocido como *"backpropagation"* o de *"retropropagación"*. Este trabaja alterando los pesos de los enlaces entre los nodos para reflejar de una mejor manera lo que a la red se le está enseñando sobre cuál es la respuesta correcta. Los pesos se ajustan lentamente puesto que no se puede asumir que lo que le "sirva" a un usuario resulte apropiado para todos.

Las fórmulas del cálculo de gradiente mostradas para redes neuronales de una única neurona serán utilizadas por el algoritmo de retropropagación, haciendo previamente modificaciones mínimas.

Cuando se entrena la red, siempre se conoce la salida deseada de cada nodo en la capa de salida. En este caso, debería ser llevado a 1 si el usuario presiona el botón indicando que le interesó un ítem en particular y, llevado a 0 (cero) si el usuario no realizó ninguna interacción sobre un ítem en particular. El único modo de cambiar el nivel de activación de un nodo es modificando la entrada total recibida por dicho nodo.

Para determinar en cuánto debería cambiar la entrada total a un nodo, el algoritmo de entrenamiento necesita conocer la pendiente de la función tangente hiperbólica (tanh) en el nivel de salida actual del nodo. En el punto medio de la función, cuando la salida es cero la pendiente es muy pronunciada, de modo que un cambio mínimo en la entrada produce un gran impacto. A medida que la activación se aproxima a 1 ó -1, un cambio en la entrada tiene un efecto menor sobre la salida. La pendiente de la función para cualquier valor de salida viene dada por la siguiente función:

dtanh(y) = 1 – y * y

Demostración:

(1) $\frac{\partial}{\partial} = \frac{}{(\ )}$

(2) $h(\ ) - h(\ ) = 1 \rightarrow 1 - \frac{(\ )}{(\ )} = \frac{}{(\ )} \rightarrow 1 - h(\ ) = \frac{}{(\ )} \rightarrow$

$$h(\ ) = \frac{1}{1 - h(\ )}$$

(3) Luego, $\frac{\partial}{\partial} = \frac{}{(\ )} = 1 - h(\ )$

Por lo que, la derivada de la función en el punto *y* sea igual a *1 – y * y*

Antes de ejecutar el algoritmo de retropropagación es necesario ejecutar la función *feedforward* para así disponer de la salida de cada nodo en la red. El algoritmo de retropropagación realiza los siguientes pasos:

Para cada nodo en la capa de salida:

1. Calcula la diferencia entre la salida actual del nodo y el valor deseado.
2. Aplica la derivada (*dtanh*) sobre la salida actual para determinar en cuánto la entrada del nodo actual tiene que cambiar.

3. Cambia el peso de cada enlace entrante en proporción al peso actual del enlace y la tasa de aprendizaje.

Esto puede resumirse bajo la siguiente formula = + (−) —— , donde *k* representa a la capa de salida y el segundo miembro de la suma corresponde al gradiente para el nodo bajo análisis.

Para cada nodo en la capa oculta:

1. Cambia la salida del nodo por la suma de los pesos de cada enlace saliente multiplicado por una cantidad que le permita aproximar a su nodo objetivo (error).
2. Aplica la derivada (*dtanh*) sobre la salida actual para determinar en cuánto la entrada del nodo actual tiene que cambiar.
3. Cambia el peso de cada enlace entrante en proporción al peso actual del enlace y la tasa de aprendizaje.

Estos pasos reflejados bajo la siguiente fórmula $W = W + c\,\delta\,X$, donde $\delta = —\sum$ w δ . Se tiene que tener en cuenta que m_{j+1} representa el número de neuronas en la capa *(j + 1)*; *l*, la *l*-ésima neurona de la dicha capa y, δ , el gradiente de la *l*-ésima neurona de la capa *(j + 1)*.

La implementación del algoritmo al principio calcula todos los errores a medida que avanza en la red y luego efectúa el ajuste de los pesos. En Anexo IV. Figura 11 se muestra el código a través del cual se implementa el algoritmo de retropropagación.

ANEXO V

GLOSARIO

Covariación: Relación existente entre dos magnitudes o series estadísticas, de manera que todo aumento o disminución de una de ellas se traduce en un aumento o disminución de la otra.

CSS: Del inglés, Cascading Style Sheets, es un lenguaje usado para definir la presentación de un documento escrito en HTML o XML. El objetivo es separar la estructura de un documento de su presentación.

Folksonomía: Es el resultado de la clasificación colaborativa por medio de etiquetas simples en un espacio de nombres llano, sin jerarquías ni relaciones de parentesco predeterminadas.

Hashtag: En servicios web tales como Twitter, es una cadena de caracteres formada por una o varias palabras concatenadas y precedidas por una almohadilla (#).

HTML: Del inglés, HiperText Markup Language (Lenguaje de Marcado de Hipertexto, es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. Este lenguaje también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script, el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTTP: Del inglés, HiperText Transfer Protocol (Protocolo de Transferencia de Hipertexto), es el protocolo usado en cada transacción de la World Wide Web (www). Es orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores.

Javascript: Es un lenguaje de programación interpretado usado fundamentalmente en el lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

JQuery: Es una biblioteca de javascript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (conjunto estándar de

objetos para representar documentos HTML o XML), manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

MVC: MVC (Modelo Vista Controlador) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se usa frecuentemente en páginas web, donde la vista es la página HTML y el código provee de datos dinámicos a la página. El modelo es el sistema de gestión de base de datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

POP3: Del inglés, Post Office Protocol (Protocolo de Oficina de Correo), es un protocolo diseñado para recibir mensajes de correo electrónico almacenados en un servidor remoto. No es utilizado para enviar correos. Permite a los usuarios con conexiones intermitentes o muy lentas, descargar su correo electrónico mientras tienen conexión y revisarlo posteriormente incluso estando desconectados.

Renderización: Es un término usado en la jerga informática para referirse al proceso de generar una imagen desde un modelo. También hace referencia al despliegue o visualización de una página web dinámica.

Sitio Espejo: En relación a un sitio computacional, es una copia exacta del mismo. Generalmente están ubicados geográficamente en distintos lugares y son utilizados como mecanismo para recuperar un sistema ante una situación inesperada.

SMTP: Del inglés, Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo), es un protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDAs, teléfonos móviles, etc.).

Umbral: Valor mínimo de un agente físico o estímulo capaz de producir un efecto.

WYSIWYG: Es el acrónimo de What You See Is What You Get (en inglés, "lo que ves es lo que obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso.

XML: Siglas en ingles de eXtensible Markup Language (Lenguaje de Marcas Extensibles), es un metalenguaje extensible de etiquetas desarrollado por la World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos y generalmente se lo usa como estándar para el intercambio de información estructurada entre diferentes plataformas.