UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
Facultad de Ciencias Exactas y Tecnologías
Licenciatura en Sistemas de Información

Trabajo Final de Graduación

Metabuscador basado en agentes para grupos de estudiantes colaborativos

Autores:

Carlos Francisco PEREZ CRESPO
María Martha PEREZ CRESPO

Profesor Guía:

Dra. Rosanna COSTAGUTA

SEPTIEMBRE de 2017

TRABAJO FINAL DE GRADUACIÓN DE LA LICENCIATURA EN SISTEMAS DE INFORMACIÓN

METABUSCADOR BASADO EN AGENTES PARA GRUPOS DE ESTUDIANTES COLABORATIVOS

Autores:

....

María Martha Pérez Crespo Carlos Francisco Pérez Crespo

Profesor Guía:

.....

Dra. Rosanna Costaguta

---------- ----------*

Aprobado el día ……. del mes de ……. del año 20…… por el tribunal integrado por

....

....

Dedicado a Mamá y Papá

Agradecimientos

A mi madre Martha, a mi padre Carlos, y a mis hermanos Charly y Lau, por motivarme a concretar mis metas, por su infinito amor y por nunca soltarme la mano. Por fomentar los valores de la perseverancia, responsabilidad y honestidad, y así crecer tanto en lo personal como en lo profesional. Especialmente a Charly, por empujarme constantemente y construir nuestra meta, juntos.

A mi pareja Santiago, por toda la fuerza, apoyo y amor incondicional.

A mis amigos Debi, Lou, Val, Carli, Luji, Mile, Vane, Eze, Bruno, Gonza, Gabi, Tincho, Dani, Patri, Tato, Fede, Feru, Andy, Bel, Lucho, Kari, Mary, por todo su cariño y aguante. Gracias a todos por acompañarme en este viaje.

María Martha Pérez Crespo

A mi vieja, porque gracias a ella llegué a cumplir este sueño, y a mi viejo por enseñarme a dar los primeros pasos en el mundo de las redes y la informática.

A mi compañera de Trabajo Final, mi hermana Marthy, por todas las locuras que atravesamos para alcanzar este objetivo, y a mi hermano Lauty por su aguante.

A mi novia Andrea, por creer en mí y sostenerme siempre en los momentos de debilidad.

A mi compañero de trabajo, Aldo "Pillo" Roldán, por sus consejos sobre el Trabajo Final.

A mis amigos Nico, Caro, y Beto, quienes estuvieron alentando en todo momento.

Carlos Francisco Pérez Crespo

A nuestra Directora de Trabajo Final, Dra. Rosanna Costaguta, por su paciencia y dedicación para con nosotros, siempre vamos a estar agradecidos.

A la Lic. Saritha Figueroa, por su buena onda y predisposición en cedernos su espacio para implementar nuestro trabajo final.

Carlos Francisco Pérez Crespo & María Martha Pérez Crespo
Santiago del Estero, Argentina
Septiembre de 2017

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

Muchas de las actividades que llevan a cabo grupos de estudiantes requieren que sus integrantes realicen individualmente búsquedas en la web para obtener información. A pesar de que muchos de los resultados obtenidos por cada uno de ellos se repiten, cada uno de esos resultados debe ser analizado para determinar su utilidad. Si se multiplica este proceso por la cantidad de individuos que conforman el grupo, se observa que puede producirse una considerable pérdida de tiempo y esfuerzo.

Dado el problema expuesto, se desarrolló un metabuscador basado en agentes como herramienta web de búsqueda colaborativa que indica qué integrante analizó un determinado resultado, y que además permite asignar una valoración personal a cada resultado e incluir un comentario. El metabuscador ordena los resultados obtenidos luego de disparadas las búsquedas individuales, considerando la valoración grupal de cada resultado, calculada a partir del promedio de las valoraciones individuales asignadas por cada integrante sobre dicho resultado. Además, genera un historial con los resultados de búsqueda calificados por el grupo, también ordenados por la valoración grupal.

El funcionamiento del metabuscador se validó mediante su uso por parte de grupos de estudiantes en experiencias colaborativas especialmente diseñadas dentro de asignaturas de las carreras de Licenciatura en Sistemas de Información y Programador Universitario en Informática (FCEyT – UNSE). Los resultados obtenidos demuestran que, mediante el sistema de rankeo creado, el metabuscador cumple sus propósitos de eficientizar los procesos de búsqueda en grupos y favorecer la interacción entre los integrantes de grupos de estudiantes. Las opiniones de los usuarios también fueron positivas. Mientras los estudiantes afirmaron que la aplicación les resultó útil y sencilla de usar, los profesores afirmaron que ver las calificaciones y los comentarios asignados por los otros integrantes del grupo sobre los documentos resultó un medio eficaz para que los estudiantes pudieran comunicarse e indicarse cuáles resultados eran interesantes, permitiéndoles generar así un producto final de buena calidad.

Palabras clave

Metabuscador, Agentes de Software, Búsqueda colaborativa, Grupos de estudiantes colaborativos.

INTRODUCCIÓN

En estos últimos años, muchos trabajos se realizan a distancia por lo que Internet ha dejado de ser aquel lugar donde se consultaba información o se intercambiaban mensajes para pasar a ser un espacio abierto en el que todos pueden participar. Este concepto es la esencia de lo que se denomina Web 2.0, donde se concentran las herramientas 2.0 síncronas (Chat, Video llamadas, etc.) y asíncronas (Foros de Discusión, Wikis, Blog, etc.). Estas herramientas son soluciones tecnológicas que permiten a cualquier persona pasar de ser un mero receptor de información a ser partícipe de esa información, ya sea generándola, compartiéndola o mejorándola a través de redes de colaboración (Junta de Castilla y León, 2012). Estas nuevas capacidades de participación en la web posibilitó el surgimiento de los conocidos Espacios Virtuales, donde las herramientas 2.0 son utilizadas para aprendizaje y tutoría remota, generación de equipos de trabajo, prácticas orientadas a escenarios, eventos globales multipropósito con complejas interacciones sociales, entre otros fines.

En contextos educativos, los Espacios Virtuales posibilitan el trabajo de los estudiantes organizados en grupos independizados de las variables tiempo y espacio. Usualmente para alcanzar los objetivos de enseñanza y de aprendizaje establecidos por el profesor los integrantes de los grupos deben realizar búsquedas, luego analizar los resultados para determinar su utilidad, para finalmente, consensuar sus opiniones para elaborar una producción grupal. El presente trabajo presenta un metabuscador basado en agentes para eficientizar este tipo de trabajo colaborativo. El metabuscador desarrollado cuenta con dos agentes: un Agente de Consulta (AC) y un Agente de Actualizaciones (AA).

Este documento se organiza como se describe a continuación. El capítulo I describe el planteamiento del problema, los objetivos propuestos, el alcance del presente trabajo y se describe de manera general los resultados obtenidos. El capítulo II describe el marco conceptual correspondiente a la temática del trabajo propuesto, que incluye un conjunto de conceptos y características básicas, y las herramientas, técnicas y metodologías necesarias para llevarlo a cabo. En el capítulo III se describe el documento de especificación de requerimientos necesario para el diseño del metabuscador. En el capítulo IV se presenta la solución propuesta para la problemática planteada en el capítulo I, definiendo la arquitectura del metabuscador, el modelo de base de datos y el algoritmo implementado para el rankeo de los resultados de búsqueda. El capítulo V presenta la aplicación de las metodologías de diseño para el desarrollo del metabuscador. En el capítulo VI se describen las pruebas de

caja negra y caja blanca realizadas, y en el capítulo VII la experimentación llevada a cabo mediante el uso del metabuscador por grupo de estudiantes colaborativos. Por último, se presentan algunas conclusiones vinculadas con el trabajo realizado, limitaciones detectadas y posibles líneas de trabajo futuro. El código fuente del metabuscador y el detalle de las pruebas de caja blanca de los módulos más relevantes del metabuscador están contenidos en el Anexo I. Los manuales de Instalación, de Configuración, y de Usuarios están contenidos en los Anexos II, III, y IV, respectivamente. Por último, los modelos de las encuestas confeccionadas para relevar la opinión de los estudiantes y de los docentes se incluyen en el Anexo V.

CAPITULO I - PLANTEAMIENTO DEL PROBLEMA

Los espacios virtuales dedicados al trabajo colaborativo, especialmente los dedicados a los procesos de enseñanza y de aprendizaje, están destinados a facilitar el diálogo entre los integrantes de un grupo cuyos miembros no están físicamente contiguos, pero que tienen que desarrollar de manera colaborativa alguna actividad.

Este trabajo se orienta en particular, a la realización de búsquedas colaborativas de material digital en la web por parte de estudiantes que conforman un grupo. Se sabe que cada individuo tiene sus preferencias en cuanto a buscadores, técnica de búsqueda, etc., y que cuando efectúa una búsqueda generalmente usa un buscador al que le proporciona un conjunto de palabras clave y espera en respuesta una lista de resultados relacionados con esas palabras. Cuando la búsqueda debe realizarse en el marco de un grupo de estudiantes colaborativos, usualmente el proceder antes descripto se replica por parte de cada uno de sus integrantes. En esta situación, cada individuo debe revisar sus resultados, lo que consume una considerable cantidad de tiempo y esfuerzo, para seguramente encontrar sólo unos cuantos resultados verdaderamente pertinentes. Además, dado que se trata de un grupo, los diferentes resultados obtenidos por la búsqueda efectuada por un integrante pueden repetirse en las búsquedas realizadas por otros integrantes del mismo grupo, independientemente del buscador utilizado, ya que dependerá altamente de las palabras clave ingresadas al disparar la búsqueda. Indudablemente, el esfuerzo, el tiempo y el resultado final del grupo se ven afectados cuanto mayor sea la cantidad de resultados repetidos. Sea de manera síncrona o asíncrona, una buena comunicación entre los participantes del grupo ayudaría a que esto no ocurra pero es normal que el tiempo empleado para indicar los resultados encontrados, pertinentes o no, por cada integrante al resto del grupo, aumente notoriamente cuando el número de miembros es mayor.

Por lo expuesto, la pregunta que guía esta investigación es la siguiente: *¿Cómo se puede hacer más eficiente la búsqueda de información en la web por parte de los integrantes de grupos de estudiantes colaborativos?* Para dar respuesta al interrogante planteado, se desarrolló un metabuscador basado en agentes como herramienta web de búsqueda colaborativa. El metabuscador creado cuenta con dos tipos de agentes: un Agente de Consultas (AC) y un Agente de Actualizaciones (AA), y su funcionamiento se explicará en detalle en los próximos capítulos.

I.1. OBJETIVOS GENERALES

- Propiciar búsquedas colaborativas eficientes.
- Favorecer la interacción entre los integrantes de grupos de estudiantes colaborativos.

I.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una herramienta de búsqueda colaborativa de materiales web basada en agentes que permita rankear los resultados considerando valoraciones individuales y grupales.
- Determinar las funciones principales del metabuscador.
- Implementar un prototipo de herramienta de búsqueda colaborativa.
- Diseñar experiencias de aprendizaje colaborativo soportado por computadora donde se utilice el metabuscador creado.

I.3. ALCANCE

La concreción de este proyecto implicó diseñar e implementar un metabuscador basado en agentes, como herramienta web de búsqueda colaborativa, que pueda ser utilizado por grupos de estudiantes a fin de eficientizar las búsquedas de material digital. Los agentes aplican un sistema de rankeo de resultados especialmente diseñado que considera las valoraciones individuales y grupales asignadas. Específicamente el metabuscador basado en agentes puede:

- Facilitar al usuario el ingreso datos de búsqueda.
- Facilitar al usuario el ingreso de una calificación a un resultado además de la posibilidad de agregar un comentario.
- Recuperar y procesar todos los resultados de búsqueda obtenidos de los distintos buscadores.
- Recuperar resultados que fueron valorados y/o comentados por los usuarios.
- Actualizar la base de datos con nuevos resultados valorados y/o comentados
- Computar una valoración grupal para cada resultado promediando las valoraciones individuales que cada uno haya recibido.
- Rankear resultados de búsquedas utilizando la valoración grupal.

- Mostrar al usuario los resultados de búsquedas rankeados por la valoración grupal, indicando quienes los calificaron, sus calificaciones y comentarios si los hubiese.
- Facilitar al grupo un historial con los resultados de búsqueda que fueron calificados con o sin comentarios, ordenados por la valoración grupal.

El funcionamiento del metabuscador propuesto fue validado mediante su uso por grupos de estudiantes en experiencias colaborativas especialmente diseñadas, dentro de asignaturas de las carreras de Licenciatura en Sistemas de Información y Programador Universitario en Informática (FCEyT – UNSE).

I.4. RESULTADOS OBTENIDOS

Con el presente trabajo se obtuvo un producto software, en particular un metabuscador basado en agentes, aplicable en situaciones donde grupos colaborativos de estudiantes requieren realizar búsquedas de material en la web. El metabuscador ordena los resultados obtenidos mediante las búsquedas individuales que realizan los integrantes, calculando una valoración que permite rankear los resultados de forma grupal promediando las valoraciones individuales asignadas por los integrantes del grupo. Además, genera un historial con los resultados de búsqueda calificados por el grupo, también ordenados por la valoración grupal.

CAPITULO II – MARCO CONCEPTUAL

II.1. APRENDIZAJE COLABORATIVO

II.1.1. Aprendizaje Colaborativo

En la actualidad existen varias definiciones para conceptos que hacen al aprendizaje colaborativo, tal es así que Norma Scagnoli (Scagnoli, 2005), puntualiza algunos para dar una idea de lo que trata este tipo de aprendizaje.

El aprendizaje colaborativo es la instancia de aprendizaje que se concreta mediante la participación de dos o más individuos en la búsqueda de información, o en la exploración tendiente a lograr una mejor comprensión o entendimiento compartido de un concepto, problema o situación.

La autora plantea que el aprendizaje colaborativo hace referencia al aprendizaje que resulta del trabajo en grupos formales o informales. Los participantes en una situación de aprendizaje colaborativo pueden ser partes de un grupo formal o predeterminado, como compañeros de una clase; o pueden ser miembros de grupos no formales, como los grupos de colegas, miembros de una lista de distribución de información, o investigadores.

También menciona que en situaciones de aprendizaje colaborativo cada uno de los participantes está comprometido con la búsqueda de información y su contribución al grupo no es competitiva sino que genera una interdependencia positiva[1], el logro de un resultado es más importante que las contribuciones individuales de cada uno.

El aprendizaje colaborativo se basa en el proceso de construcción del conocimiento a partir del aprendizaje que surge de la interacción con un grupo de individuos y a su vez por medio de actividades llevadas a cabo en cooperación con ellos.

La determinación de la dimensión ideal para la conformación de grupos es una tarea compleja. (Johnson, y otros, 1999) mencionan que la cantidad de integrantes de un grupo dependerá, de manera general, de los objetivos a alcanzar y de la experiencia en trabajo en grupo, además de otros aspectos que pueden surgir y deben tenerse en cuenta. También

[1]La Interdependencia Positiva es el corazón del aprendizaje colaborativo. Los estudiantes deben de creer que están ligados con otros de una forma que uno no puede tener éxito a menos que los otros miembros del equipo también tengan éxito. Los estudiantes deben de trabajar juntos para completar el trabajo(Betancourt, 2008).

presentan una serie de factores que se deben considerar para determinar la dimensión de un grupo:

- Cuanto mayor sea la cantidad de integrantes en un grupo, se debe considerar que mayor será la gama de destrezas y capacidades presentes, mayor será la cantidad de personas que procesen la información y los puntos de vistas que se obtengan. Cuantos más miembros tenga un grupo, se contará con más recursos que contribuyan al éxito del trabajo del grupo.

- En un grupo se deben coordinar las acciones y tareas de cada miembro, y a su vez cada miembro debe tener la oportunidad de expresarse. Esto permite llegar a un consenso de la información que se procese y se analice. Es decir, se debe llegar a un acuerdo para que todos los miembros cumplan con la tarea y haya buena relación de trabajo. Pero cuanto mayor sea el número de integrantes, mayor será el número de interacciones y por ende, mayor deberá ser la habilidad de cada miembro de manejar esas interacciones.

- En muchos casos, es necesario resaltar que cuando los grupos tienen una dimensión bastante grande, las interacciones personales entre los miembros suelen disminuir. Esto suele producir un grupo menos cohesionado, lo que conlleva a una baja en la responsabilidad individual para contribuir al éxito del trabajo del grupo.

- Se debe considerar el tiempo que conlleve alcanzar el producto final. Se recomienda que si el tiempo disponible para entregar un trabajo es poco, lo mejor es un grupo reducido. Esto se debe a que, un grupo menor podrá organizarse mejor en menor tiempo, operará con mayor rapidez y posibilitará una participación activa y prolongada por parte de cada miembro.

- El desempeño de cada miembro es mucho más visible en grupos pequeños, por lo que es más complicado que un miembro no cumpla con su tarea y no haga su aporte para el trabajo final. La responsabilidad es mayor y se garantiza la participación activa de todos.

- Cuando la cantidad de integrantes de un grupo es pequeña, se detecta fácilmente cuando ocurre alguna dificultad o algún integrante tiene algún inconveniente para trabajar junto a otros.

Por su parte, (García, y otros, 2008) sostienen que los grupos grandes (más de 12 o 13 personas) estarán dispersos y no lograrán establecer una comunicación y una interacción

eficaz; mientras que los grupos muy pequeños (de 2 o 3 personas) no tendrán diversidad para alcanzar soluciones.

La educación a distancia en entornos virtuales también conocida como "Educación Virtual", ha evolucionado en cuanto a herramientas, técnicas didácticas y la forma en que se da la interacción entre participantes, facilitando la exploración y búsqueda individual de información y conocimiento. Los estudiantes pueden, a través de los entornos educativos virtuales, reforzar sus habilidades en la investigación y construcción de su propio aprendizaje, como también adquirir nuevos conocimientos y competencias, si se incentiva a la participación en comunidades virtuales a través de foros de discusión, y otras aplicaciones como blogs, y wikis. Según (Anguiano Torres, y otros, 2013), a través de estos entornos los estudiantes pueden desarrollar competencias relacionadas con su habilidad para trabajar con otros, de presentar sus ideas y de respetar las de otros en un medio pluralista y de equidad social, con el fin de que desarrollen tanto competencias genéricas para la vida, como específicas para su desarrollo profesional.

II.1.2. Herramientas Colaborativas

Las Herramientas Web 2.0 o Herramientas Colaborativas han modificado los conceptos de información y de comunicación de tal forma que han dejado de ser entidades individualizadas y unilaterales, para convertirse en procesos colectivos y, en ocasiones, anónimos. Para todo ello, internet proporciona espacios donde el conocimiento es "construido" (Vaquerizo , y otros, 2009). El uso de dicha tecnología facilita el trabajo colaborativo, lo cual constituye un aspecto fundamental a la hora de mejorar la competitividad, así como la optimización de recursos en cualquier institución.

Los estudiantes, y de manera específica los estudiantes universitarios, han evolucionado en la medida que han irrumpido las tecnologías en la sociedad(Cascales Martínez , y otros, 2016). La utilización de las TIC permite a los profesores realizar actividades que impliquen colaboración en una nueva dimensión espacio-temporal que se configura, dando lugar a interacción entre individuos en función del momento y el lugar en que nos encontremos (De Benito, 1999).

En (Junta de Castilla y León, 2012) se enumeran las siguientes ventajas para la utilización de las TIC:

- ✓ Facilitar la comunicación.
- ✓ Mejorar la gestión del conocimiento, facilitando el acceso a una información de interés completa y actualizada.
- ✓ Disminuir los costes de las actividades.
- ✓ Aumentar la transparencia de las actuaciones.

Es para destacar que desde hace varios años, las instituciones diseñan políticas y proyectos académicos donde las TICs juegan un papel muy importante. Entre ellos se distingue el uso de aula virtuales y la propuesta de trabajos virtuales(Cascales Martínez , y otros, 2016).

Cuando los grupos de estudiantes desarrollan sus actividades de enseñanza y de aprendizaje valiéndose de las herramientas 2.0 para colaborar, comunicarse y coordinar esas actividades, se está en el ámbito del denominado Aprendizaje Colaborativo Soportado por Computadora (ACSC). A continuación se presentan algunas de las herramientas 2.0 que según (Scagnoli, 2005), suelen utilizarse con frecuencia en ACSC:

a) Foros de discusión o debate: Es un medio de comunicación entre miembros de una comunidad virtual de cualquier índole. Habitualmente, los foros son centros de debate, discusión, y pedido de información. La construcción de conocimiento se da por medio de la retroalimentación y respuesta a preguntas realizadas por los individuos. Cada uno de los miembros contribuye individualmente al conocimiento aportando puntos de vista o información relevante que permite llegar a conclusiones bien informadas.

b) Blogs: Es un espacio personal en la Web en donde sin necesidad de conocer diseño Web, un individuo puede escribir y publicar su propia página, un diario de reflexiones, una recolección de eventos, o compartir información, ideas, pensamientos con otros. El blog muestra las entradas en orden cronológico o temático.

c) Wikis: Una wiki se define como una colección de páginas Web conectadas entre sí, cada una de las cuales puede ser visitada y editada por cualquiera con acceso a Internet en cualquier lugar del mundo. Ejemplos de ellos son: Wikipedia, Wikibooks, WikEd y Wiki en Moodle. El aprendizaje colaborativo puede darse por la construcción de documentos sobre diversos temas entre los participantes.

d) Grupos virtuales: Son espacios en Internet cuyos miembros son personas que crean comunidades virtuales para compartir archivos, información, y para comunicarse sincrónica o asíncronamente[2]. Ejemplo de ello es Yahoo Grupos, donde los miembros comparten enlaces de sitios de interés común, archivos, textos, etc.

e) Correo electrónico: El correo electrónico, también llamado e-mail permite la comunicación en texto entre computadoras conectadas a la Red en cualquier parte del mundo.

f) Listas de distribución: Son listas de correos electrónicos que se usan para distribuir información o compartir opiniones entre los miembros. Generalmente nuclean personas con intereses similares o un tema común. Los miembros también pueden usar la misma vía para comunicarse con el resto del grupo. Habitualmente, este tipo de listas no se emplean para compartir conocimientos sino solo como medio de difusión de noticias.

g) Marcadores Sociales: Los marcadores sociales (Dixon, y otros, 2011) son herramientas de software social que permite a los usuarios enviar, clasificar, localizar y compartir las páginas web marcadas en un sitio central desde donde otros usuarios podrán acceder a ellas y "etiquetarlas". Es un proceso utilizado para organizar, ordenar, mantener y conservar enlaces a determinadas páginas web.

II.2. BÚSQUEDAS EN LA WEB

Actualmente, la Web es el mayor centro de información que existe, donde se puede encontrar una variedad inmensa de temas, desde cosas cotidianas y simples, a algunos de carácter exótico o importante.

Buscar información en la web se ha convertido en uno de los retos más importantes de internet. Gran parte de la información disponible se encuentra organizada por herramientas, que permiten agruparla y localizarla. Como dato destacable, en febrero de 1999 el número de páginas Web indexadas rondaban una cifra superior a los 800 millones de páginas (Lafuente, 2001).

[2]La comunicación sincrónica es el proceso de comunicación que tiene lugar de forma simultánea o en el mismo tiempo. La comunicación asincrónica es aquella que se realiza entre individuos que no coinciden en un mismo tiempo.(Castañeda Quintero , 2007)

Simplemente imaginarse que la única manera posible de que una persona busque información sobre un tema específico, sea navegando por la web, accediendo a un determinado sitio para luego hacer clic en un vínculo para acceder a otro sitio, en el cual deberá hacer clic en otro vínculo para acceder a un nuevo sitio, y así sucesivamente, sería una búsqueda bastante laboriosa. Como solución a este problema surgieron los buscadores, aplicaciones web sencillas que proveen servicios para consulta de información, es decir que tratan de hacer más corto el camino entre el usuario y las páginas o contenidos Web que son de su interés (Lafuente, 2001).

II.2.1. Buscadores

En términos generales es posible conceptualizar a un buscador web, también llamado motor de búsqueda, como un programa encargado de realizar las búsquedas dentro de las bases de datos de documentos web a partir de palabras clave – keywords - introducidas por el usuario (Stark, 2003)

Ahora bien, para profundizar este concepto existen otras definiciones que se enuncian a continuación.

- Un buscador es un recurso informático que permite localizar información en los servidores conectados a la red mediante el uso de palabras clave, brindando como resultado una lista ordenada más o menos amplia según la existencia de archivos o materiales almacenados en los servidores correspondientes relacionados con los criterios de exploración solicitados (Telmex, 2012)
- Los buscadores son sitios diseñados para facilitar al usuario el hallazgo de determinada información en Internet. (Maglione, y otros, n.d)
- Los motores de búsqueda utilizan palabras clave o árboles jerárquicos por temas; el resultado de la búsqueda es un listado de direcciones web en donde se mencionan temas relacionados con las palabras clave buscadas. (Rodríguez García, y otros, 2009)
- Un buscador web es un sistema de recuperación de información diseñado para la búsqueda de información en la web. (Bravo Marquez, 2010)
- Sistema que recibe una consulta del usuario compuesta de una o más palabras, realiza la consulta a la base de datos y muestra un listado ordenado de documentos

> que pueden o no cumplir con los objetivos de la consulta del usuario, de cumplir puede ser en forma parcial o total. (Lambertucci, 2001)

Considerando lo expuesto, y expresándolo de forma sintética, se puede decir que un buscador es un sitio web que proporciona al usuario un listado de direcciones web como resultado de una consulta determinada la cual se efectuó mediante un criterio de búsqueda. Solo es necesario conocer la dirección web del buscador para que a través de él navegar por la web.

Los buscadores pueden clasificarse en tres categorías:

A) Índices Temáticos, Directorios o Catálogos

Este tipo de buscadores mantienen la organización de las páginas en un directorio navegable por temas, también conocido como "Índice Temático", para que a través de él se realice la búsqueda de páginas Web. Los Índices Temáticos son sistemas de búsqueda por temas o categorías jerarquizados. Se trata de bases de datos de direcciones Web elaboradas "manualmente", es decir, hay personas que se encargan de asignar cada página web a una categoría o tema determinado. (Rodríguez García, y otros, 2009)

Existen dos desventajas para destacar: una tiene que ver con su mecanismo de indexación, pues la actualización se lleva a cabo de forma manual, y por ello no se actualiza de manera periódica; la otra tiene que ver con el tiempo que se demora para encontrar información, ya que se realiza un recorrido por el índice (organizado en un árbol Jerárquico) hasta llegar a la información específica.

Un conocido buscador que entraba dentro de esta categoría es: "Yahoo!". Éste presentaba una metodología de indexación por directorio, con una estructura jerárquica de categorías y sub-categorías que permitía a través de su página web navegar hacia la información específica. Como se muestra en la Figura 1, las categorías que ofrecía eran las siguientes: Negocios a negocios, Estados de Estados Unidos, Compras y Servicios, Internacional, Empleo y Trabajo, Negocios y Economía, Entretenimiento, Financias e Inversión, Ciencias, Artes, Recreación, Sociedad y Cultura, Ciencias Sociales, Gobierno, Educación, Noticias y Multimedia, y Referencias. Además, se puede observar, que cada categoría, ofrece un conjunto de subcategorías. Por ejemplo, Empleo y Trabajo, ofrece "Carreras" y "Trabajos". El responsable de un sitio web que deseaba añadir su sitio, de tal modo que sea alcanzado por las búsquedas de los usuarios, debía incorporar la información

necesaria en dichos directorios de forma manual. Yahoo! fue comprado en julio del 2016 por la compañía Verizon Communications, modificando su metodología de indexación, para aproximarse al implementado por Google.

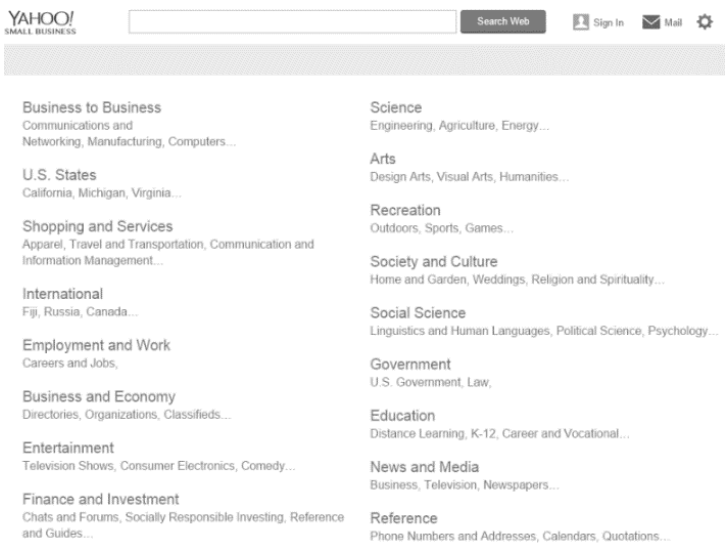


Figura 1 - Directorio del buscador Yahoo!

B) Motores de Búsqueda

Son sistemas de búsqueda por palabras clave. Son bases de datos que incorporan automáticamente páginas web mediante "robots" de búsqueda en la red. (Rodríguez García, y otros, 2009).

Este tipo de buscadores surgieron con posterioridad a los índices o directorios. En ellos, las búsquedas son llevadas a cabo por un programa, el cual implementa lo que se denomina robot (también llamado “araña”, “crawler”, “webbot” o “spider”). Un robot se encarga de visitar las páginas y tomar los datos necesarios, relacionar la dirección de la página con los datos que aparecen en ellas armando un índice y luego, actualizar la base de datos con este índice (Lafuente, 2001). A diferencia del “Índice Temático”, cuando un sitio web que se encuentra indexado en la base de datos, se actualiza, no se necesita informar los cambios realizados para actualizar el índice. Esto sucede de manera automática. Los motores de búsqueda son capaces de leer el contenido de un sitio web y encontrar aquellos datos que permitan clasificarlo. (González, n.d.).

Para realizar una consulta, utilizando un motor de búsqueda, el usuario debe ingresar una “frase de búsqueda”, o una o más “palabras clave” referidas a la información que desea buscar, y en base a ello el buscador accederá a la base de datos para realizar la búsqueda

definiendo un listado de resultados. El orden de relevancia, es decir, el orden en que se muestran los resultados de la consulta, está determinado por cada buscador. Pueden existir distintos parámetros, pero existen muchísimas estrategias, que las distintas empresas elaboran para lograr los primeros puestos.

Un ejemplo de este tipo de buscador es Google. La Figura 2 muestra, por ejemplo, el listado devuelto por el motor de búsqueda de Google. El usuario ingresa una frase de búsqueda – keywords- y Google, de acuerdo a su índice, devuelve un listado de resultados asociados a dicha frase.

Figura 2 - Listado devuelto por el motor de búsqueda Google

C) Motores de Búsqueda Híbridos

Estos son una combinación de los tipos de buscadores anteriormente descriptos. Son buscadores temáticos pero cuando se navega dentro de la página de búsqueda se puede realizar alguna consulta dentro del área temática con alguna palabra clave o comentario.

Un ejemplo de este tipo de buscador es: accoona. Una vez accedido al sitio (http://www.accoona.com/). En su barra superior de color negro se puede visualizar dos pestañas que indican la forma de búsqueda: Búsqueda por Directorio (Site Search) (Figura 3), o Búsqueda por Web (Web Search) como Motor de Búsqueda (Figura 4).

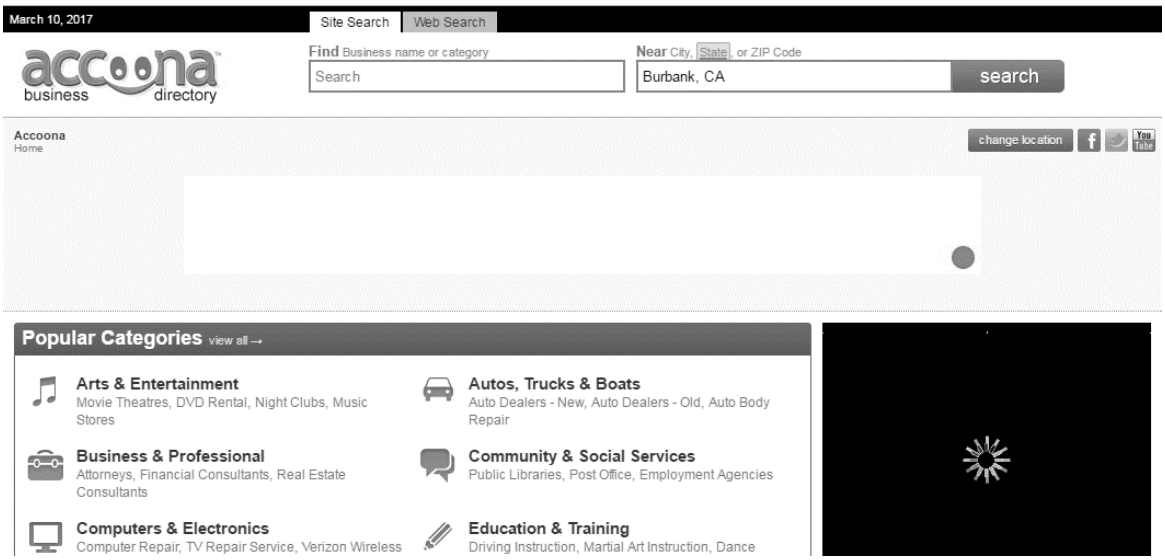


Figura 3 - Accona, motor de búsqueda híbrido – Site Search

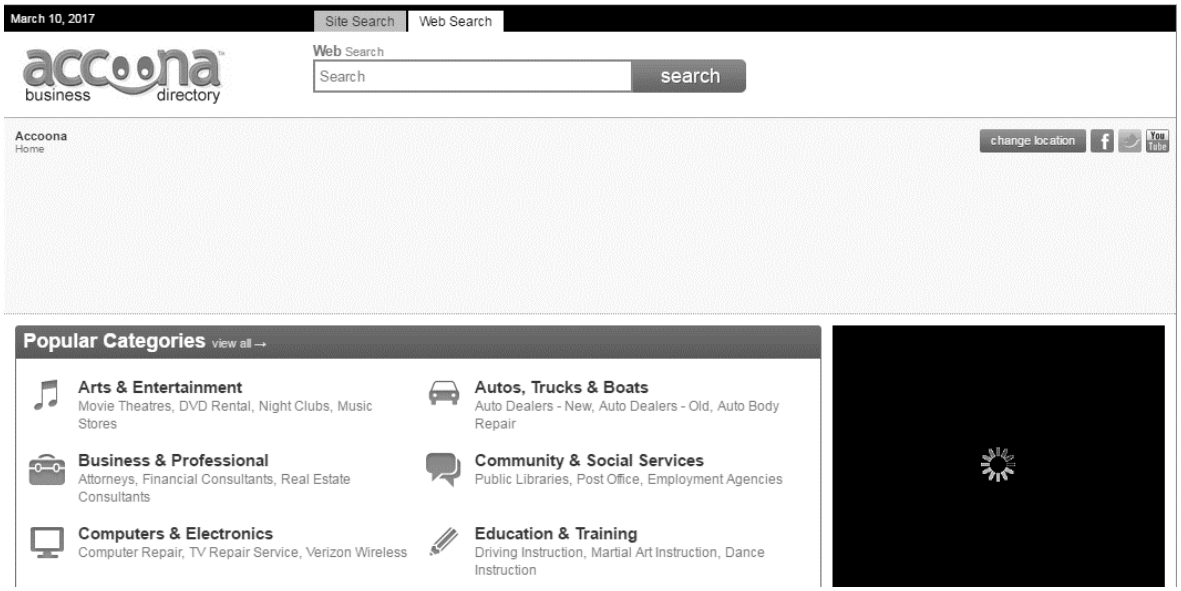


Figura 4 - Accona, motor de búsqueda híbrido – Web Search

Existe otra clasificación para los buscadores (Yu, y otros, 2002):

- Buscadores de Propósito General: Tienen la Capacidad de buscar en todas las páginas de la web. Ejemplo de estos son: Google, Yahooy Bing.

- Buscadores de Propósito Especial (o Específico): Enfocan sus búsquedas en documentos destinados a un dominio particular, como ser documentos de una organización o pertenecientes a un área temática específica. Por ejemplo, el buscador Cora (cora.whizbang.com) se enfoca en los trabajos de investigación sobre Ciencias de la Computación, mientras que el buscador Mundial de Medicina (www.mwsearch.com) es

un motor de búsqueda orientado a la información médica. Actualmente, muchos sitios de organizaciones y negocios han instalados motores de búsqueda en sus propias páginas.

II.2.2. Técnica de Web Scraping o *Escarbado de Páginas Web*

El **web scraping** se basa en la extracción de datos, de una o varias páginas web relacionadas a través de enlaces en un sitio web, para su manipulación y/o análisis posterior. Es una técnica universal adoptada por la mayoría de los motores de búsqueda.

Ritu Banerjee (Banerjee, 2014) define web scraping como una técnica utilizada para extraer información de sitios web mediante programas de software. Usualmente, estos programas simulan la navegación de un humano en la web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.

Para (Banerjee, 2014) el término web scraping está relacionado con la automatización de tareas en la web, simulando la navegación de un humano utilizando un software de computadora. Algunos de los usos del web scraping son: comparación de precios en tiendas, monitorización de datos relacionados con el clima de cierta región, detección de cambios en sitios webs, integración de datos en sitios webs, etc. Además, web scraping también está muy relacionado con la indexación de la web puesto que se indexa la información de la web utilizando un robot. Básicamente, el web scraping es el proceso de recopilar información de forma automática de la web. Una de las formas para recopilar esa información está relacionada con el Reconocimiento de Información Semántica.

El Reconocimiento de Información Semántica, según (Hernández Herrero, 2014), se basa en que las páginas que son analizadas pueden contener metadatos o información semántica, como anotaciones o comentarios. Esta “meta información” puede ser usada para extraer la información deseada. Si estas anotaciones están en las mismas páginas, como sucede con los micro formatos (por ejemplo, atributos utilizados en etiquetas html como ser “id” o “class”), estas podrían ser de utilidad cuando recopilamos información a través del DOM del documento. Según (W3C, 2004) el DOM - Document Object Model ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (API) para documentos válidos HTML y bien construidos XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula. En la especificación DOM, el término "documento" es utilizado en un sentido amplio; XML es cada vez más utilizado como un medio para

representar muchas clases diferentes de información que puede ser almacenada en sistemas diversos, y mucha de esta información se vería, en términos tradicionales, más como datos que como documentos. Sin embargo, XML presenta estos datos como documentos, y se puede utilizar DOM para manejar estos datos.

II.2.3. Metabuscadores

Con el pasar de los años la web fue creciendo hasta convertirse en lo que es hoy en día, una fuente inagotable de información, lo cual ha promovido el surgimiento de nuevos buscadores y/o el mejoramiento de los que subsisten. Aunque son cientos los motores de búsquedas ninguno puede satisfacer a todos los usuarios de la web o al menos en general, ser considerado aceptable, tal que sea lo suficientemente comprensivo en la cobertura de la web (Maity, 2015)

Cuando los usuarios realizan una búsqueda en la web para encontrar información sobre un determinado tema, se debe considerar el tiempo que conlleva, incluso cuando ya tienen una ruta más o menos específica para llevar a cabo tal búsqueda. Se debe destacar que es tedioso emplear el tiempo para realizar las búsquedas en varios buscadores con la idea de evaluar sus resultados particulares, y en base a ello determinar cuál es el más relevante para responder una demanda. Para solucionar esta problemática surgen los metabuscadores (Torres Pombert, 2013)

Un metabuscador, denominado en inglés “Meta-Search Engine”, es un sistema que proporciona un acceso unificado a los múltiples motores de búsquedas existentes (Yu, y otros, 2002). Este acceso unificado a múltiples buscadores se origina cuando habiendo recibido una consulta de un usuario, el metabuscador invoca a los diferentes motores de búsqueda subyacentes para recuperar información que responda tal consulta (Yu, y otros, 2002). Es decir, el metabuscador toma los resultados de los motores de búsqueda, elimina los duplicados y presenta un resumen de los resultados obtenidos, ordenados por relevancia y, en algunos casos, incluso indicando cuál ha sido el buscador que devolvió tal información (Miriada X, n.d.). Se debe destacar que los metabuscadores no cuentan con una base de datos propia, solamente actúan como intermediarios entre los usuarios y los buscadores. Sin embargo, un metabuscador sofisticado puede mantener información de los resultados obtenidos por los motores de búsqueda para proveer un mejor servicio (Yu, y otros, 2002).

Algunos metabuscadores ofrecen la opción de escoger los buscadores, que serán incluidos en la metabúsqueda, desde un grupo determinado de buscadores disponibles. Si se escogen aquellos conocidos por su confiabilidad, potencia y rapidez, los resultados de la búsqueda serán mejores(Torres Pombert, 2013). Ejemplos de metabuscadores son: MetaCrawler, Ixquick, Dogpile, Savvysearch, Copernic, Netbrillant 2.0, etc. En (González, n.d.) se enuncian algunas ventajas de trabajar con metabuscadores:

• Amplían el ámbito de las búsquedas que realiza un usuario, proporcionando mayor cantidad de resultados. La forma de combinar los resultados depende del metabuscador empleado.

• Hay que tener en cuenta que cada buscador utiliza su propia estrategia a la hora recoger información de una página y a la hora de ordenar los resultados de las búsquedas, esto repercute en que las páginas de mayor relevancia en un buscador no tienen por qué coincidir en los del resto, aportando puntos de vista distintos.

También se enuncian en (González, n.d.) las siguientes desventajas:

• Cada buscador dispone de su propia sintaxis de búsqueda y en el metabuscador no se puede hacer distinción entre las diferentes sintaxis de cada buscador. Por lo tanto, al buscar información muy específica es mejor emplear buscadores de los que conozcamos la sintaxis.

• No resulta muy claro qué criterios emplean los metabuscadores para la ordenación de los resultados

• Al tener que buscar en varias fuentes, las búsquedas suelen tardar más que con un buscador. Muchos de los metabuscadores permiten establecer un tiempo máximo para realizar una búsqueda.

La arquitectura de software de un metabuscador, como se muestra en la Figura 5, está compuesta por 5 componentes: interfaz de usuario, selector de base de datos, selector de documento, despachador de consulta y fusión de resultados, además de incorporar todos los motores de búsqueda que se deseen implementar(Yu, y otros, 2002). Mediante flechas dirigidas y números, se presenta la secuencia de acciones del procesamiento de una consulta. Luego de que un usuario introduce una frase de búsqueda (1), a través de una interfaz de usuario, el metabuscador debe decidir con qué motores de búsqueda trabajar (2) y cuáles son

los documentos que deberá recuperar (3), para luego disparar la consulta a los distintos motores de búsqueda (4,5), tareas de las que se encargan el Selector de Base de Datos, el Selector de Documentos y el Despachador de Consulta, respectivamente. Por último, el metabuscador recupera los resultados de cada uno de los motores de búsqueda (6), para que el componente de Fusionador de Resultados combine dichos resultados en un solo listado, el cual será presentado al usuario mediante la interfaz de usuario (7,8).

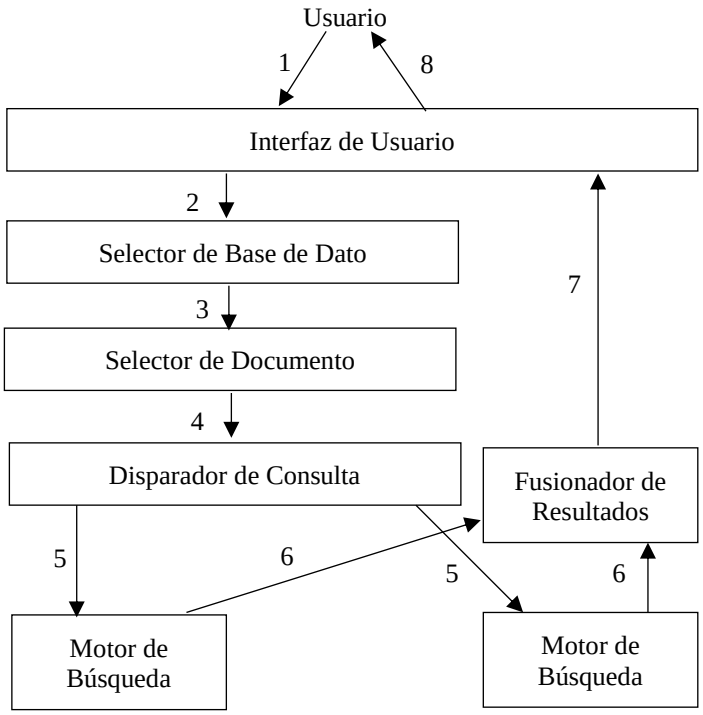


Figura 5 - Componentes de la Arquitectura de un Metabuscador

A continuación, se presenta una explicación extendida de los componentes más importantes incluidos en la Figura 5, y sus interacciones.

Selector de Base de Datos: Es necesario determinar la cantidad de motores de búsqueda que debe emplear un metabuscador. Por un lado, utilizar bases de datos locales no será de completa utilidad para satisfacer una consulta. Por otro lado, enviar la consulta de un usuario a un grupo pequeño de motores de búsqueda puede resultar una estrategia más apropiada, a diferencia de un grupo grande de motores de búsqueda. Por ejemplo, supóngase que un usuario está interesado en 10 resultados, que resultan ser los mejores adaptados a su consulta. A lo sumo estos 10 documentos estarán contenidos en un máximo de 10 bases de datos. Por ende, si consideramos que el número de bases de datos es mucho más grande que

la cantidad de resultados obtenidos, va a suceder que existirán bases de datos que no serán de utilidad a esa consulta.

Una función de similitud, calcula las similitudes de los documentos para una consulta realizada por el usuario. La similitud de un documento sobre una consulta, es un número que representa el grado de relevancia del documento para esa consulta.

Para una consulta dada, se dice que un documento “d” es potencialmente útil si satisface una de las siguientes condiciones:

1) Si se desean “m” documentos de la lista de resultados, para algún entero positivo, entonces la similitud de "d" para la consulta está entre las “m” más altas de todas las similitudes entre todos los documentos de la consulta.
2) Si todo documento con una similitud excede un umbral deseado predeterminado, entonces la similitud entre “d” es mayor que el umbral.

En otras palabras, para un “m” dado en la Condición 1, se puede determinar un umbral en la Condición 2 de manera que el número de documentos, cuyas similitudes sobrepasen el umbral, sea “m”, y viceversa. Sin embargo, en la práctica, la relación sólo puede hacerse cuando se dispone de información estadística sustancial. Por lo general, un usuario especifica el número de documentos que le gustaría ver. El sistema utiliza un umbral para determinar qué documentos deben recuperarse y sólo muestra el número de documentos deseado al usuario.

El metabuscador debe determinar si la base de datos de cada motor de búsqueda es útil o inútil (sin utilidad). Una base de datos se considera útil si contiene documentos útiles para la consulta realizada.

Realizar una consulta en motores de búsqueda cuyas bases de datos no son útiles implica considerar varias cuestiones: el desperdicio de los recursos debido al envío de la consulta a bases de datos sin utilidad y también debido a la posterior evaluación de los documentos obtenidos desde esas bases de datos inútiles; y un tráfico de red innecesario producido por la transmisión de la consulta a los motores de búsqueda sin utilidad desde el metabuscador, así como la transmisión de documentos inútiles desde esos motores de búsqueda al metabuscador.

Consecuentemente, cada consulta del usuario debe ser enviada únicamente a las bases de datos que sean potencialmente útiles. Identificar las bases de datos que son de utilidad para enviar una consulta dada es la tarea del componente de software "Selector de Base de Datos". Aquel selector de base de datos que identifica correctamente el mayor número de bases de datos potencialmente útiles como sea posible, y reduce al mínimo las bases de datos inútiles, es considerado como un buen selector de base de datos.

Selector de Documento: El Selector de Documento es el encargado de determinar qué documentos deben recuperarse de la base de datos de cada motor de búsqueda seleccionado por el selector de base de datos. Tiene por objetivo recuperar la mayor cantidad de documentos potencialmente útiles y reducir al mínimo la recuperación de documentos inútiles. Varios factores pueden afectar la selección de documentos que se recuperan de cada motor de búsqueda, como ser el número de documentos potencialmente útiles de cada motor de búsqueda y la función de similitud utilizada por el metabuscador.

La recuperación de los documentos en un motor de búsqueda se realiza en orden descendente. En el momento de realizar la selección de los documentos de cada motor de búsqueda se pueden presentar uno de los dos siguientes problemas:

- Determinar el número de documentos que se recuperarán de la base de dato. Si k documentos deben ser recuperados de una base de datos, entonces se recuperarán los k documentos con las mayores similitudes.
- Determinar un umbral local para la base de datos de tal manera que un documento de la base de datos se recupera sólo si su similitud supera el umbral.

Cualquiera sea el problema que se presente, lo que siempre se debe buscar es recuperar la totalidad o mayor número posible de documentos (resultados) potencialmente útiles de cada base de datos y reducir al mínimo la recuperación de documentos inútiles.

Los enfoques propuestos para resolver el problema de selección de documentos quedan definidos en las siguientes categorías:

- ✓ *Determinación de Usuario*: El número de documentos que se quiere recuperar de una base de datos es determinado por el usuario.

- ✓ *Asignación Ponderada*: La cantidad de documentos que se recupere dependerá de la calificación, rango o posición que tenga cada documento en la base de datos, en

comparación con la de ese resultado en otras bases de datos. Se consideran aquellos documentos que tengan una calificación alta, o una puntuación de rango superior.

- ✓ *Enfoques Basados en el Aprendizaje*: El número de documentos a recuperar de una base de datos está determinado en base a experiencias pasadas de recuperación de la base de datos.

- ✓ *Recuperación Garantizada*: Este tipo de enfoque tiene como objetivo garantizar la recuperación de todos los documentos potencialmente útiles con respecto a cualquier consulta dada. Asegura que todos los documentos potencialmente útiles a nivel mundial se recuperen incluso cuando las similitudes de los documentos globales y locales no coinciden

Despachador de Consulta: El despachador de consulta tiene como objetivo establecer una conexión con el servidor de cada motor de búsqueda seleccionado y pasar la consulta a los mismos. El Protocolo de Transferencia de Hipertexto - HTTP se utiliza para realizar la transferencia de envío de consultas y recepción de resultados. El método de solicitud HTTP (GET O POST, por ejemplo) y el formato de consulta (por ejemplo, el nombre de la caja de texto para la consulta específica) son definidos por cada motor de búsqueda.

Este componente de la arquitectura debe considerar que cada motor de búsqueda tiene sus requerimientos. La consulta enviada a un motor de búsqueda en particular puede o no ser la misma que la recibida por el motor de metabúsqueda, es decir, la consulta original será traducida a una nueva consulta antes de ser enviada a un motor de búsqueda determinado. Para realizar la traducción de la consulta solo se necesita retener todos los términos de la consulta del usuario. Pero se debe considerar el número de documentos que se recupera de un motor de búsqueda. Por ejemplo, un usuario del metabuscador indica que se deben recuperar *m* documentos. Luego, el selector de documentos decide los *k* documentos que son recuperados de un determinado motor de búsqueda. El número *k*, que por lo general suele ser diferente de *m*, se debe incluir en la consulta traducida que se enviará al motor de búsqueda de componentes.

Fusionador de Resultados: Este componente se encarga de combinar los resultados seleccionados por cada motor de búsqueda, en una sola lista ordenada.

II.3. AGENTES

Hoy en día no existe una única definición sobre agentes, eso se debe a que no hay una definición universalmente aceptada. Una de las definiciones más empleadas es la de Russell y Norvig (Russell, y otros, 2008), quienes consideran que *un agente es cualquier cosa capaz de percibir su medioambiente con la ayuda de sensores y actuar en ese medio utilizando actuadores*. El término "actuadores" hace referencia al componente del agente que concreta la acción en respuesta al estímulo percibido. Por ejemplo, un agente robot recibe pulsaciones del teclado, archivos de información y paquetes vía red a modo de entradas sensoriales y actúa sobre el medio con mensajes en el monitor, escribiendo ficheros y enviando paquetes por la red. Cuando se menciona que "percibe su medio ambiente", se hace hincapié en que el agente puede recibir entradas en cualquier instante.

Una secuencia de percepciones de un agente refleja el historial completo de lo que el agente ha recibido. Según Russel y Novig (Russell, y otros, 2008), "*un agente tomará una decisión en un momento dado dependiendo de la secuencia completa de percepciones hasta ese instante"*. Si se puede especificar qué decisión tomará un agente para cada una de las posibles secuencias de percepciones, entonces se habrá explicado más o menos todo lo que se puede decir de un agente. En términos matemáticos se puede decir que el comportamiento del agente viene dado por la función del agente que proyecta una percepción dada en una acción.

La función que describe el comportamiento de un agente se puede presentar en forma de tabla. La misma contiene todas las secuencias de percepción y sus respectivas acciones que son justamente las que el agente tomará como respuesta. Esa tabla es una caracterización externa del agente.

En la Figura 6, se puede observar gráficamente y de una forma simple el concepto de agente. El agente percibe su medio ambiente a través de sensores, realiza un procesamiento para determinar la decisión adecuada en base a su percepción, y por último, actuará en su medio ambiente a través de actuadores.

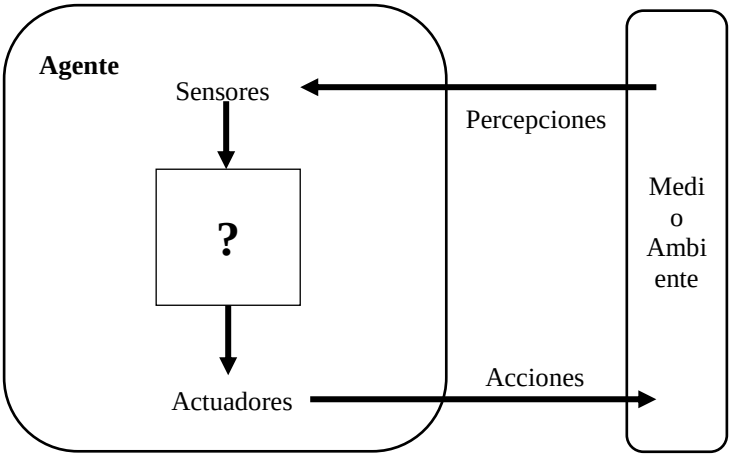


Figura 6 - Interacción de un agente con su medio ambiente a través de sensores y efectores

II.3.1. Características de un agente

Según (Franklin, y otros, 1996) es posible caracterizar un agente mediante una serie de aspectos:

- ✓ Continuidad Temporal: se considera un agente un proceso sin fin, que se ejecuta continuamente desarrollando su función.
- ✓ Autonomía: un agente es completamente autónomo, si es capaz de actuar basándose en su experiencia. El agente es capaz de adaptarse aunque el entorno cambie severamente. Un agente carece de autonomía cuando se apoya más en el conocimiento inicial que le proporciona su diseñador que en sus propias percepciones.
- ✓ Sociabilidad: este atributo permite a un agente comunicarse con otros agentes o incluso con otras entidades.
- ✓ Racionalidad: el agente siempre realiza «lo correcto» a partir de los datos que percibe del entorno.
- ✓ Reactividad: un agente actúa como resultado de cambios en su entorno. En este caso, un agente percibe el entorno y esos cambios dirigen el comportamiento del agente. ·
- ✓ Pro-actividad: un agente es pro-activo cuando es capaz de controlar sus propios objetivos a pesar de cambios en el entorno.

- ✓ Adaptabilidad: está relacionado con el aprendizaje que un agente es capaz de realizar y si puede cambiar su comportamiento basándose en ese aprendizaje.
- ✓ Movilidad: capacidad de un agente de trasladarse a través de una red telemática.
- ✓ Veracidad: asunción de que un agente no comunica información falsa a propósito.
- ✓ Benevolencia: asunción de que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos.

Determinar el grado de importancia de cada una de estas características, no es posible. Se debe destacar que las mismas, aportan una distinción a los agentes para no confundirlos con los programas que hoy en día conocemos.

II.3.2. Estructura de los agentes

Según Rusell y Novig (Russell, y otros, 2008), en la actualidad, la Inteligencia Artificial (IA) busca diseñar el *programa del agente* que implemente una función, la cual se encargue de proyectar las percepciones del agente en acciones.

$$Agente = Arquitectura + Programa$$

La arquitectura representa el aspecto físico del agente, lo que incluye a los sensores y actuadores, y es donde se ejecutará el programa. Es importante considerar que dicho programa debe ser apropiado para la arquitectura. Como ejemplos de arquitectura para un agente se pueden mencionar una computadora personal o un robot con varias computadoras, cámaras, sensores y otros componentes. La arquitectura permite que los sensores sean capaces de percibir su entorno, ejecute los programas, y luego indica a los actuadores que pongan en marcha las acciones generadas.

La diferencia entre los conceptos de *programa del agente* y *función del agente* radica en que el primero solo toma la percepción actual como entrada, porque no hay nada más disponible en el entorno, en cambio el segundo recibe la percepción histórica completa y si las acciones del agente dependen de la secuencia completa de percepciones, el agente tendría que recordar las percepciones.

Existen cuatro tipos básicos de programas para agentes que encarnan los principios que subyacen en casi todos los sistemas inteligentes: Agentes reactivos simples, Agentes reactivos basados en modelos, Agentes basados en objetivos y Agentes basados en utilidad.

a) Agentes Reactivos Simples

Los *agentes reactivos simples* son el tipo de agente más sencillo. Se encargan de seleccionar las acciones teniendo en cuenta las percepciones *actuales*, sin considerar las percepciones históricas.

Su naturaleza es del tipo estímulo – respuesta y por ello, se puede decir que el comportamiento del programa está basado en un conjunto de reglas. Estas reglas se utilizan para interpretar los estímulos y generar una respuesta si es necesario, por lo que se denominan *reglas de condición acción*. La estructura utilizada para las reglas es la sentencia de condición **Si**.... **Entonces**…. Por ejemplo:

Si *percepción* **Entonces** *acción*

El agente cuenta con interfaces para recibir señales y emitir sus respuestas, y de ese modo puede interactuar con el ambiente en dónde está ubicado. Básicamente cuenta con: una interfaz de mensajes, que se utiliza para enviar y recibir mensajes con otros agentes; sensores, los cuales perciben las variables que le interesan al agente de su ambiente; actuadores, los cuales se encargan de producir cambios en el ambiente; y una interfaz de recursos, que se utiliza para intercambiar información con estructuras externas, como por ejemplo bases de datos.

b) Agentes Reactivos Basados en Modelos

Almacenar información de las partes del mundo que no pueden ver le permite manejar la visibilidad parcial. Esto le permite al agente mantener algún tipo de *estado interno*, el cual depende de la historia percibida y refleja algunos aspectos no observables del estado actual.

Dos tipos de conocimiento en el programa del agente son necesarios para realizar la actualización de la información de estado interno. Por un lado, se necesita información acerca de cómo evoluciona el mundo independientemente del agente, y por otro lado, se necesita más información sobre cómo afectan al mundo las acciones del agente. Este conocimiento acerca de "cómo funciona el mundo", sin considerar el modo en que se

implementa, se denomina *modelo* del mundo. Un agente que utilice este modelo es un *agente basado en modelos*.

Uno de los elementos más importantes que se puede mencionar para este tipo de agentes es la base de conocimientos. En esta base, hay información predefinida y necesaria para el agente, así como también cualquier información que fue obtenida o aprendida en sus ejecuciones. Estos agentes siguen un conjunto de reglas para tomar las decisiones, similar a los agentes reactivos simples.

Otro elemento importante es el motor de inferencia. Este motor provee de facilidades al agente y la posibilidad de navegar a través de la información, contenida en la base de conocimientos, con el objetivo de deducir algo por medio de esa información. El motor utilizará reglas y la base de conocimiento para razonar sobre la información y deducir resultados de una manera organizada.

c) Agentes basados en Objetivos

Para algunos agentes no es suficiente tener conocimiento sobre el estado actual del mundo para decidir qué hacer, también necesitan algún tipo de información sobre su meta a fin de describir las situaciones que son deseables.

El programa del agente se puede combinar con información sobre los resultados de las acciones posibles (la misma información que se utilizó para actualizar el estado interno en el caso del agente reflexivo) para elegir las acciones que permitan alcanzar el objetivo.

En algunas ocasiones, la selección de acciones basadas en objetivos es directa, cuando alcanzar los objetivos es el resultado inmediato de una acción individual. En otras ocasiones, puede ser más complicado, cuando el agente tiene que considerar secuencias complejas para encontrar el camino que le permita alcanzar el objetivo. La búsqueda y planificación son sub-campos de la IA centrados en encontrar secuencias de acciones que permitan a los agentes alcanzar sus metas.

Existen dos tipos de planificadores para cualquier sistema de IA: planificadores dinámicos y planificadores estáticos. Los planificadores dinámicos toman como entrada una meta, un conjunto de operadores, y el estado del agente, y de forma dinámica generan un plan para satisfacer la meta del agente. Los planificadores estáticos toman como entrada la meta y estado del agente, y utilizan esta información para elegir el plan existente que mejor

satisfaga la meta. Aunque la manera de ejecución es totalmente diferente, la estructura básica de estos planificadores es la misma. Ambos poseen una interface de entrada – salida para recibir los mensajes de otros agentes. Los mensajes tienen la información necesaria para elegir o crear el plan de ejecución. Debido a que lo único que debe hacer el agente es generar o elegir un plan, no es necesaria ninguna otra interfaz.

d) Agentes Basados en su Utilidad

Las metas por sí solas no son realmente suficientes para generar comportamiento de gran calidad en la mayoría de los entornos. Las metas sólo proporcionan una cruda distinción binaria entre los estados de «felicidad» y «tristeza», mientras que una medida de eficiencia más general debería permitir una comparación entre estados del mundo diferentes de acuerdo al nivel exacto de felicidad que el agente alcance cuando se llegue a un estado u otro. Como el término «felicidad» no suena muy científico, la terminología tradicional utilizada en estos casos para indicar que se prefiere un estado del mundo a otro es que un estado tiene más utilidad (cualidad de ser útil) que otro para el agente.

Una función de utilidad proyecta un estado (o una secuencia de estados) en un número real, que representa un nivel de felicidad. La definición completa de una función de utilidad permite tomar decisiones racionales en dos tipos de casos en los que las metas son inadecuadas. Primero, cuando haya objetivos conflictivos, y sólo se puedan alcanzar algunos de ellos (por ejemplo, velocidad y seguridad), la función de utilidad determina el equilibrio adecuado. Segundo, cuando haya varios objetivos por los que se pueda guiar el agente, y ninguno de ellos se pueda alcanzar con certeza, la utilidad proporciona un mecanismo para ponderar la probabilidad de éxito en función de la importancia de los objetivos.

Un agente que posea una función de utilidad *explícita* puede tomar decisiones racionales, y lo puede hacer con la ayuda de un algoritmo de propósito general que no dependa de la función específica de utilidad a maximizar. De esta forma, la definición "global" de racionalidad (identificando como racionales aquellas funciones de los agentes que proporcionan el mayor rendimiento) se transforma en una restricción "local" en el diseño de agentes racionales que se puede expresar con un simple programa.

II.4. METODOLOGIAS DE DISEÑO

II.4.1. GAIA

Es una de las metodologías más conocidas y citadas en el desarrollo de sistemas multi-agentes. Se la puede considerar como una metodología de "propósito general", que no especifica una tecnología de agentes a utilizarse ni detalles de implementación. En (Lázaro Molina, 2006) se establece que esta metodología ha sido desarrollada por Franco Zambonelli, David Kinny, Nicholas Jennings y Michael Wooldridge, y que su primera versión fue en el año 2000.

La metodología permite desarrollar sistemas complejos con una estructura organizacional estática, donde el número de agentes que los componen no cambia en tiempo de ejecución. También las interacciones, habilidades y servicios ofrecidos por los agentes son estáticos. Se diferencian claramente las fases de análisis y diseño, aunque la fase de captura de requerimientos es independiente de la metodología.

GAIA es una metodología de análisis y diseño de sistemas multi-agentes, de carácter muy general, aplicable a un rango muy amplio de este tipo de sistemas. Además, permite tener un amplio conocimiento tanto a nivel organizativo (social) como a nivel de detalle de cada agente (Venturini, 2012).

Antes de entrar a desarrollar las distintas fases de GAIA, es necesario definir qué es un rol. En (Muñoz, y otros, 2010) se define un rol como una serie de tareas coherentes, y es el agente el que desempeña uno o más roles. Para GAIA, un rol se define por cuatro atributos: las responsabilidades, los permisos, las actividades y los protocolos.

Las *responsabilidades*, a su vez, pueden fraccionarse en dos categorías: las propiedades de bondad (Liveness) y las propiedades de seguridad (Safety). Ellas determinan la funcionalidad y son, tal vez, los atributos claves asociados con un rol.

Los *permisos* identifican los recursos que el rol tiene disponible para realizar sus responsabilidades. Suelen ser recursos de información. Con cada recurso habrá asociados unos derechos vinculados como por ejemplo: leer, cambiar o generar el recurso.

Las *actividades* representan acciones privadas u operaciones que el agente puede completar sin la necesidad de interactuar con otros agentes en el sistema.

Los *protocolos* definen las formas en que un agente asumiendo dicho rol podría interactuar con otros agentes.

La Figura 7 muestra cómo está estructurada la metodología GAIA. Puede observarse que está compuesta por dos fases: fase de análisis, que contiene el modelo de roles y el modelo de interacciones; y la fase de diseño, que contiene el modelo de agente, el modelo de servicio y el modelo de conocidos.

La **fase de análisis** busca dejar en claro cómo funciona el sistema y su estructura (sin detalles de implementación). Esta fase se asienta, como se dijo antes, sobre dos modelos: modelo de roles y modelo de interacciones (Muñoz, y otros, 2010); (Lázaro Molina, 2006); (Wooldridge, y otros, 2000).

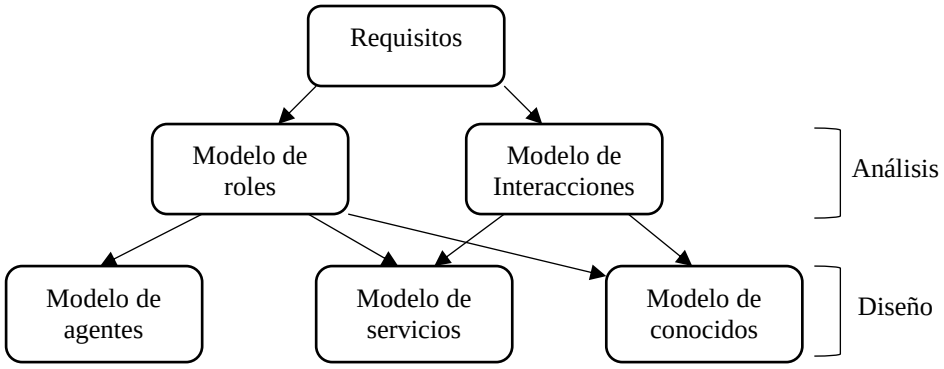


Figura 7 - Estructura de la metodología GAIA

El modelo de roles busca hacer una descripción de todas las tareas que son necesarias en el sistema. Para ello, además, debe realizarse un análisis de los recursos disponibles del sistema. Se identifican los distintos tipos de roles del sistema. Como se mencionó en la definición de rol, cada uno tiene asociado permisos y responsabilidades. Con estos conceptos, se puede especificar un esquema de rol para cada tipo de rol y conformar así, el modelo de roles. Un esquema de rol presenta una descripción del rol, sus protocolos y actividades, sus permisos y las responsabilidades de vida y seguridad.

El modelo de interacciones brinda una descripción de las comunicaciones realizadas entre los agentes dentro del modelo de roles, es decir, dichas interacciones se llevan a cabo en este último modelo. El resultado es una serie de definiciones de protocolo, una para cada tipo distinto de interacción entre los roles. En (Wooldridge, y otros, 2000) se indica que un protocolo se puede definir a partir de los siguientes atributos:

- Propósito: breve descripción textual de la naturaleza de la interacción.
- Iniciador: el/los rol/es responsable/s de iniciar la interacción.
- Receptor: el/los rol/es con los cuales el iniciador interactúa.
- Entradas: información usada por el rol iniciador los cuales desencadenan el protocolo.
- Salidas: información suministrada por/al receptor del protocolo durante el curso de la interacción.
- Proceso: breve descripción textual de cualquier proceso que el iniciador del protocolo realiza durante el curso de la interacción.

Así, la fase de análisis se puede resumir en los siguientes pasos:

- Paso 1: Identificar los roles en el sistema. Los roles en un sistema corresponderán a:

  a) Individuos, ya sea dentro de una organización o como actor independiente independientemente;
  b) Departamentos dentro de una organización;
  c) Una organización por si misma.

- Paso 2: Para cada rol, identificar y documentar los protocolos asociados. Los protocolos son los patrones de interacción que ocurren en el sistema entre los varios roles.

La **fase de diseño** permite la transformación de los modelos de la fase de análisis a un nivel de abstracción suficientemente bajo, de modo que sea posible implementar los agentes haciendo uso de las técnicas de diseño tradicionales (del mismo modo de los que hacen uso de la orientación a objetos). Esta fase implica la construcción de tres modelos: el modelo de agentes, el modelo de servicios y el modelo de conocidos(Lázaro Molina, 2006); (Muñoz, y otros, 2010) ; (Wooldridge, y otros, 2000).

El modelo de agentes permite definir los diferentes tipos de agente que habrá en el sistema, así como también el número de instancias que se tendrán de cada uno, en tiempo de ejecución. Un tipo de agente puede asumir uno o más roles (aunque lo contrario no es cierto). El modelo de agentes se construye mediante un árbol de tipos de agentes, en el cual los nodos hoja corresponden a los roles y los restantes a los tipos de agentes. Respecto a las instancias, se debe mencionar que: una instancia con una "n" significa que habrá exactamente n agentes de este tipo en el sistema en tiempo de ejecución; una notación del tipo "m…n" significa que habrá no menos que m y no más que n instancias de este tipo en un sistema en tiempo

de ejecución (m<n); una notación "*" significa que habrá cero o más instancias en tiempo de ejecución; y el "+" significa que habrá una o más instancias en tiempo de ejecución.

El modelo de servicios permite establecer los diferentes servicios que ofrece un agente al resto de los agentes del sistema. Es decir, define los servicios que cada tipo de agente va a implementar, entendiendo por todo servicio como una cierta funcionalidad. Los servicios se derivan de las actividades y protocolos, así como de sus propiedades de bondad (Livennes) y seguridad (Security) encontradas en la etapa de análisis. Cada servicio se define a partir de: las entradas, las salidas, las precondiciones y las poscondiciones. Las entradas y salidas proceden en forma directa del modelo de protocolos. Las precondiciones y las poscondiciones constituyen limitantes en los servicios y son derivadas de las propiedades de un rol.

El modelo de conocidos establece las relaciones existentes entre los agentes, determinando los servicios que un agente requiere de qué otros agentes. Además, al hacer uso de estos canales de comunicación, pueden surgir problemas de embotellamiento, y esto también puede ser identificado en este modelo. El modelo se define a partir de un grafo dirigido, donde un tipo de agente se corresponde con un nodo y los caminos de comunicación se relacionan con una arista. Así pues, un grafo **A** → **B**, muestra que hay un camino de **A** a **B**, pero no necesariamente de **B** a **A**.

Así, la fase de diseño puede resumirse en los siguientes pasos:

- Paso 1: Crear un modelo de agente:
    - Agregar roles en los tipos de agentes.
    - Documentar las instancias de cada tipo de agente usando notaciones de instancias.
- Paso 2: Desarrollar un modelo de servicio, para el examen de actividades, protocolos y propiedades Safety y Liveness de los roles.
- Paso 3: Desarrollar un modelo de conocidos del modelo de interacción y del modelo de agente.

Una vez definidos los cinco modelos indicados por las dos fases de Gaia, se da por concluida la labor de la metodología, y se procede a realizar la implementación.

II.4.2. OOHDM

OOHDM (Object Oriented Hypermedia Design Method), es una metodología desarrollada por Schwabe y Rossi (Schwabe, y otros, 1998) y está orientada a objetos. En (Aguilar Bernabé, y otros, 2013) se la describe como un proceso de desarrollo de cinco fases y (Mantilla Yánez , y otros, 2007) sostienen que su objetivo es simplificar y, a su vez, hacer más eficaz el diseño de aplicaciones hipermedia

Las metodologías tradicionales de ingeniería de software, o las existentes para sistemas de información, no contienen una abstracción capaz de facilitar la tarea de especificar aplicaciones hipermedia. Es importante destacar el crecimiento de forma acelerada tanto en tamaño, complejidad y número de aplicaciones, por lo cual una metodología de diseño sistemática es necesaria para disminuir la complejidad y admitir evolución y reusabilidad (Mantilla Yánez , y otros, 2007).

Es muy difícil de lograr el desarrollo de aplicaciones donde el usuario pueda aprovechar al máximo lo que ofrece el paradigma de la navegación de sitios web, mientras se producen una serie de transacciones sobre bases de información. La clave del éxito para estas aplicaciones hipermedia se rige en definir una estructura de navegación robusta y será una buena señal de que la aplicación ha sido bien diseñada, si el usuario entiende dónde puede ir y cómo llegar al lugar deseado.

Según se explica en (Mantilla Yánez , y otros, 2007) para poder construir la interfaz de una aplicación web se deben especificar cuáles son los objetos de la interfaz que deberían ser implementados y la manera en la cual estos objetos interactúan con el resto de la aplicación, por lo cual resulta ser una tarea compleja.

Además, para cualquier aplicación que se quiera desarrollar, existen requisitos que deben ser satisfechos en un entorno de desarrollo unificado (framework). Las aplicaciones hipermedias no quedan exentas de estos requisitos y para ello, se debe considerar que, por un lado, la navegación y el comportamiento funcional de la aplicación deberían ser integrados, y por otro lado, durante el proceso de diseño se debería poder desacoplar las decisiones de diseño relacionadas con la estructura navegacional de la aplicación, de aquellas relacionadas con el modelo del dominio.

En la Figura 8 se esquematizan las cinco etapas de OOHDM, las cuales se desarrollan en modo iterativo. Cada etapa se analiza a continuación en base a los trabajos de (Soto De

Giorgis, y otros, 2002), (Aguilar Bernabé, y otros, 2013), (Artaza Álvarez, 2010) y (Mantilla Yánez , y otros, 2007).

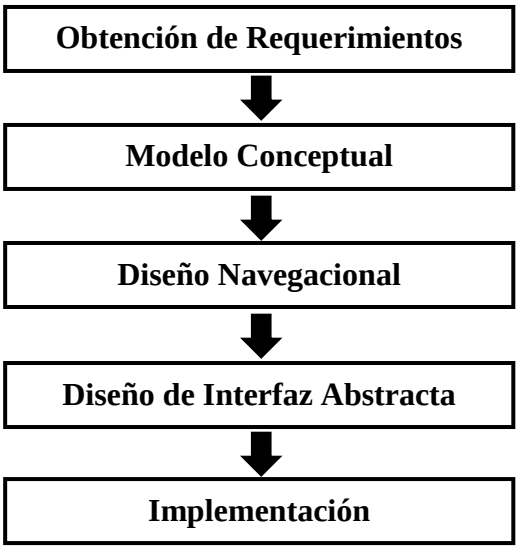


Figura 8 - Esquema de la metodología OOHDM

II.4.2.1. Obtención de Requerimientos

OOHDM propone dividir la etapa de "Obtención de Requerimientos" en cinco subetapas: Identificación de roles y tareas, Especificación de escenarios, Especificación de casos de uso, Especificación de UIDs y Validación de casos de uso y UIDs.

Identificación de roles y tareas: En esta sub etapa, lo que se busca es entender el dominio del sistema e identificar los diferentes roles que podrían cumplir cada uno de los potenciales usuarios de la aplicación. Identificar los roles, además, es de gran ayuda en cuanto a la validación de los casos de uso ya que, de acuerdo al rol que cumpla, cada usuario entregará su conformidad con respecto al caso de uso en el que participan. Una vez identificados los roles, se procede a identificar las tareas que deberá soportar la aplicación.

Especificación de escenarios: Los escenarios son descripciones narrativas de cómo la aplicación será utilizada. En esta sub etapa, se debe especificar textual o verbalmente los escenarios que describen las tareas.

Especificación de casos de uso: Un caso de uso representa la interacción entre el usuario y el sistema, agrupando las tareas representadas en los escenarios existentes. Para esta sub etapa se debe identificar la información para generar un caso de uso coherente.

Especificación de UIDs: De acuerdo a UML, los diagramas de secuencia, de colaboración y de estado son capaces de representar un caso de uso. Sin embargo, la especificación de casos de usos usando estas técnicas es un amplio trabajo y puede anticiparse inesperadamente a tomar algunas decisiones de diseño. Para evitar esto OOHDM propone la utilización de una herramienta, llamada UID, que permite representar en forma rápida y sencilla los casos de uso generados en la etapa anterior. Para poder desarrollar un UIDs desde un caso de uso, se debe identificar y organizar la secuencia de información intercambiada entre el usuario y el sistema. Identificar este intercambio es crucial, ya que es la base para la definición de los UIDs.

Validación de casos de uso y UIDs: En esta etapa, se validan los casos de uso y los UIDs, interactuando con cada usuario. Esta interacción consiste en mostrar y explicar dichos diagramas, para ver si el o los usuarios están de acuerdo.

II.4.2.2. Modelo Conceptual

En esta etapa se hace uso de las técnicas de la orientación a objetos para construir un modelo orientado a objetos, el cual represente el dominio de la aplicación. Es por ello, que se puede decir que la finalidad principal de esta fase es capturar el dominio semántico de la aplicación, en el cual se debe tener en cuenta el papel de los usuarios y las tareas que desarrollan. Las clases, relaciones y cardinalidades se definen de acuerdo a reglas que se aplican sobre los UIDs

II.4.2.3. Diseño Navegacional

Las aplicaciones hipermedia se caracterizan por lo que se conoce como “navegación”. En OOHDM, una aplicación es considerada como una vista de navegación del modelo conceptual. La misma se basa en el diseño de navegación, la cual tiene en cuenta los tipos de usuarios que interactuarán con la aplicación y el conjunto de tareas que se realizarán al utilizarla. A partir de un esquema conceptual se pueden construir diferentes modelos de navegación. Una estructura de navegación de una aplicación hipermedia es la definición de las clases de navegación que reflejan la visión del campo de aplicación. Esta estructura se define en términos de contextos de navegación, que son inducidas (de diferentes maneras, dependiendo del contexto) de las clases, tales como la navegación con Nodos y Enlaces. Durante el Diseño de Navegación se define también la manera de proceder de la

navegación, mediante la especificación de transformaciones en el espacio de navegación, es decir, el conjunto de objetos de navegación accesible en todo momento.

Este diseño, se puede expresar mediante dos esquemas: el esquema de clases navegacionales y el esquema de contextos navegacionales:

Esquema de Clases Navegacionales: A partir de los tipos predefinidos de clases, se establecen las posibles vistas del hiperdocumento, denominadas navegacionales, de las cuales pueden mencionarse los nodos, enlaces y así como también otras clases, que representan estructuras o formas alternativas de acceso a los nodos, como los índices y los recorridos guiados.

Esquema de Contexto Navegacional: Es el esquema que permite estructurar el hiperespacio de navegación en sub-espacios. Para ello debe indicar la información que será mostrada al usuario y los enlaces que estarán disponibles cuando se accede a un objeto (nodo) en un contexto determinado. Mostrar cómo será la navegación del usuario en el hiper-documento es el objetivo de este esquema.

En OOHDM, los objetos sobre los cuales navega el usuario son otro tipo de objetos que se construyen a partir de uno o más objetos conceptuales. Esto implica que el usuario navegue a través de enlaces, muchos de los cuales no se pueden derivar directamente de relaciones conceptuales.

En OOHDM hay una serie de clases especiales predefinidas, que se conocen como clases navegacionales (Aguilar Bernabé, y otros, 2013):

- **Nodos:** Los nodos se los puede definir como contenedores básicos de información para las aplicaciones hipermedia. Son vistas orientadas a objeto de las clases definidas durante el diseño conceptual. Para ellas, se usa un lenguaje predefinido e intuitivo. Esto permite que un nodo se defina mediante la combinación de atributos de clases diferentes relacionadas en el modelo de diseño conceptual. Los nodos pueden contener atributos de tipos básicos (donde se pueden encontrar tipos como imágenes o sonidos) y enlaces.

- **Enlaces:** Estos reflejan la relación de navegación que puede explorar el usuario. Como se mencionó anteriormente, un esquema conceptual puede brindar diferentes

esquemas navegacionales y los enlaces van a ser imprescindibles para poder crear esas vistas diferentes.

- **Estructuras de acceso:** Las estructuras de acceso actúan como índices o diccionarios. Se puede destacar que estas estructuras le permiten al usuario encontrar de forma rápida y eficiente la información deseada. Como ejemplo se pueden mencionar los menús, los índices o las guías de ruta. Las estructuras de acceso también se modelan como clases, compuestas por un conjunto de a objetos que son accesibles desde ella y una serie de criterios de clasificación de las mismas.

- **Contexto Navegacional**: Se debe prever los caminos que el usuario puede seguir, ya que de este modo se podrá evitar información redundante o que el usuario se pierda en la navegación. En OOHDM un contexto navegacional se puede componer por nodos, enlaces, clases de contexto y de otros contextos navegaciones. Estos son introducidos desde clases de navegación (enlaces, nodos o estructuras de acceso), pudiendo ser definidas por extensión o de forma implícita.

- **Clase de Contexto:** Es una clase que permite complementar la definición de una clase de navegación. Sirve para indicar qué información está accesible desde un enlace y desde dónde se puede llegar a él.

En la Figura 9, se hace una descripción de la nomenclatura que se utiliza para representar el Esquema de Contexto Navegacional según (Artaza Álvarez, 2010):

C	Familia de Contextos **C**
ind	Índice **ind**

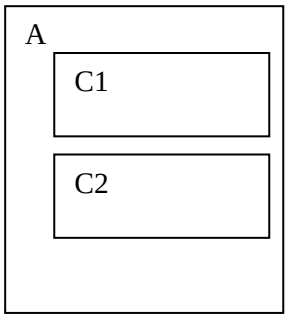
	Nodo A es recorrido en los Contextos C1 y C2
	Enlace entre contextos

Figura 9 - Simbología para la representación de contextos

II.4.2.4. Diseño de Interfaz Abstracta

En esta etapa se busca especificar las diferentes interfaces de la aplicación. Para ello se debe definir de qué manera aparecerán los objetos navegacionales en la interfaz y cuales objetos activarán la navegación. Para lograr esto se hace uso de los ADVs (Vista de Datos Abstracta). Los ADVs son modelos abstractos que especifican la organización y el comportamiento de la interfaz. Estos ADVs representan estados o interfaces y no la implementación propiamente tal. Mediante los ADVs, como se especifica(Aguilar Bernabé, y otros, 2013) lo que se busca es identificar:

- Qué objetos de interfaz va a percibir el usuario
- El camino en el cuál aparecerán los diferentes objetos de navegación
- Qué objeto de interfaz actuarán en la navegación
- La forma de sincronización de los objetos multimedia y el interfaz de transformaciones.

Saber separar claramente la interfaz abstracta y el modelo de navegación, permite la construcción de interfaces diferentes para el mismo modelo de navegación. Esto permitirá adecuar los cambios en cuanto a las necesidades y preferencias de los usuarios, o la tecnología de interfaz.

II.4.2.5. Implementación

Una vez obtenido el modelo conceptual, el modelo de navegación y el modelo de interfaz abstracta, sólo resta concretar los objetos con un lenguaje de programación, y de

este modo obtener una implementación ejecutable de la aplicación. Para ello se puede, de manera general identificar lo siguiente como puntos de esta etapa:

- Productos: Aplicación ejecutable
- Herramientas: El entorno del lenguaje de programación
- Mecanismos: Los ofrecidos por el lenguaje
- Objetivo de diseño: Obtener la aplicación ejecutable

II.5. HERRAMIENTAS DE IMPLEMENTACION

II.5.1. Servidores Web

De forma conceptual se puede decir que los servidores web son computadoras o máquina informáticas que están al servicio de otras computadoras, máquinas o personas para suministrarles todo tipo de información requerida. Los servidores web implementan diversos protocolos para ofrecer sus servicios, entre ellos se encuentra el protocolo HTTP.

"El protocolo HTTP (hypertext transfer protocol) está diseñado para transferir lo que llamamos hipertextos, páginas web páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos" (Mifsuf Talón, 2012).

II.5.1.1. Arquitectura de un Servidor Web

Mifsuf Talón, Elvira (Mifsuf Talón, 2012) menciona que la arquitectura empleada para estos servidores es la arquitectura cliente/servidor, es decir que el equipo cliente hace una solicitud o petición al equipo servidor, y éste atiende dicha solicitud.

En el equipo cliente se ejecuta una aplicación llamada "navegador o cliente web" que: por un lado, sirve de interfaz con el usuario es decir que atiende sus peticiones, muestra los resultados de las consultas y proporciona al usuario un conjunto de herramientas que facilitan su comunicación con el servidor, y por otro lado se comunica con el servidor web, es decir transmite las peticiones de los usuarios.

En el equipo servidor la principal tarea es la de atender las peticiones recibidas desde los navegadores o clientes web y hacerlo de forma eficiente y segura. Por ejemplo, el caso de los servidores web seguros que solicitan un nombre de usuario y una contraseña para

permitir el acceso sólo a usuarios registrados y por tanto, con permiso para visualizar la página o páginas.

Un aspecto más en la seguridad proporcionada por los servidores web se basa en el establecimiento de conexiones cifradas con el navegador. Este nivel de seguridad es imprescindible si se hacen transacciones comerciales mediante internet.

II.5.1.2. Servidor Web Apache

En (Mifsuf Talón, 2012)se define a un servidor HTTP Apache2 como un servidor web de software libre desarrollado por Apache Software Foundation – ASF. El producto obtenido de este proyecto es un servidor de código fuente completo, descargable y gratuito. (Mifsuf Talón, 2012)

Apache2 es robusto y con un ciclo de desarrollo muy rápido gracias a la gran cantidad de colaboradores voluntarios de los que dispone.

Básicamente es un servidor estable, eficiente, extensible y multiplataforma:

- Estable: ya que por su robustez impide caídas o cambios en el servidor inesperados.
- Flexible y eficiente: capaz de trabajar con el estándar HTTP/1.1 (RFC2616) y con la mayor parte de las extensiones web que existen en la actualidad, como son los módulos PHP, SSL, CGI, SSI, proxy, por nombrar algunos…
- Extensible: dispone de una gran cantidad de módulos que amplían su funcionalidad.
- Multiplataforma: ya que se encuentra disponible para diferentes plataformas como ser: GNU/Linux, Windows, MacOS.

En (Mifsuf Talón, 2012) se resalta que desde el año de 1996 es el servidor más utilizado en internet y es utilizado en los sistemas GNU/Linux. En concreto, en Marzo del año 2011 más del 60% de los sitios web de internet estaban utilizando Apache.

II.5.1.3. Servidor Web Apache Tomcat

En (Arenas Delgado, y otros, 2011) definen al servidor web apache Tomcat, también conocido como simplemente Tomcat o Jakarta Tomcat, como un servidor web multiplataforma que funciona como contenedor de servlets y que se desarrolla bajo el

proyecto denominado “Jackarta” perteneciente a Apache Software Foundation bajo la licencia Apache 2.0. Es una implementación de código abierto de Java Servlets, Java Server Pages, Java Lenguaje Expression y Tecnología WebSocket de Java y sus especificaciones se desarrollan bajo la Java Community Process.

II.5.1.4. Servlet

Jorge Guerra (Guerra Guerra, 2010)define a los Servlets como módulos escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java. Los Servlets son para los servidores lo que los applets para los navegadores, aunque los servlets no tienen una interfaz gráfica. Pueden ser incluidos en servidores que soporten la API de Servlet. La API no realiza suposiciones sobre el entorno que se utiliza, como tipo de servidor o plataforma, ni del protocolo a utilizar, aunque existe una API especial para HTTP.

II.5.2. Lenguajes de Programación

II.5.2.1. PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML(PHP Group, n.d.).

Lo que se destaca de este lenguaje de programación es que el código es ejecutado en el servidor, genera el código HTML y lo envía al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. Configurando los servidores web es posible procesar todos los archivos HTML con php, por lo tanto no hay manera de que los usuarios puedan saber qué hay detrás de lo que se muestra.

Es un lenguaje tan simple como complejo, puede ser utilizado desde principiantes hasta profesionales haciendo uso de todas las características avanzadas que ofrece.

II.5.2.2. JAVA

En el sitio oficial de JAVA (Oracle, n.d.) mencionan a Java como un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995, por Sun Microsystems. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos,

desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Según (Fernández, 2005), la principal característica de Java es la de ser un lenguaje compilado e interpretado. En Java, cualquiera que sea el programa debe compilarse ya que el código que se genera (bytecodes) será interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. Java es, a partir de 2012, uno de los lenguajes orientado a objetos de propósito general más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

II.5.3. PHP/Java Bridge

En (Bökemeier, y otros, n.d.) se define a PHP/Java Bridge (PJB) como una implementación de streaming (flujo de datos conocido como streams en Java), basado en XML protocolo de red, que puede utilizarse para conectar un motor de script[3] nativo, por ejemplo, PHP, Python o Scheme, con una máquina virtual de Java (JVM), en otras palabras es un protocolo de red que permite conectar en ambas direcciones scripts en PHP con clases de Java por ejemplo.

El puente (PJB) utiliza un "pasaje continuo" para invocar procedimientos / métodos de cada entorno. Aunque esto puede ser inusual, es un mecanismo rápido y fiable para conectar diferentes entornos de script.

II.5.4. JADE (Java Agent Development Framework)

Francisco Requena Moreno (Requena Moreno, 2007) menciona que según FIPA, *los agentes son entidades software que están localizados en una única plataforma.* Una plataforma de agentes FIPA[4] se define como el software que implementa un conjunto de especificaciones FIPA.

Alejandro Franco Barrios (Barrios, 2009) menciona que una plataforma de agentes es un entorno de ejecución en donde los agentes que se ejecutan tienen nombres únicos,

3Un script es un archivo que incluye un conjunto de comandos. Son ejecutados desde la primera línea hasta la última (de forma secuencial)(Velasco, y otros, 2008).

4La Foundation for Intelligent Physical Agents (FIPA) es un organismo para el desarrollo y establecimiento de estándares de software para agentes heterogéneos que interactúan, y sistemas basados en agentes(Bellver González, 2015).

pueden comunicarse con otros mediante comunicaciones punto a punto, pueden proporcionar servicios y buscar a otros agentes dependiendo de los servicios que proporcionen.

La comunicación entre agentes se lleva a cabo a través de mensajes asíncronos, es decir, el agente que envía el mensaje y el destinatario del mensaje no tienen por qué estar disponibles al mismo tiempo; el destinatario no tiene porqué existir en ese instante. Los mensajes se pueden enviar a un agente en concreto o se pueden enviar a agentes que se desconocen pero se sabe que poseen unas ciertas características. Jade proporciona mecanismos de seguridad, ya que habrá agentes a los que no se les esté permitido comunicarse con otros agentes, de manera que una aplicación puede verificar la identidad del receptor y del agente que envía el mensaje y no dejar realizar actuaciones no permitidas para un determinado agente.

Según (Marchetti, y otros, 2003) Jade es un entorno de software implementado en lenguaje Java que simplifica la implementación de sistemas multi-agente a través de un middleware que cumple con las especificaciones FIPA y por medio de un conjunto de herramientas gráficas que soportan las fases de depuración e implementación. La plataforma Jade puede ser distribuida a través de máquinas (que ni siquiera necesitan compartir el mismo sistema operativo debido a la propia característica de JAVA) y la configuración se puede controlar a través de una interfaz gráfica remota. La configuración se puede cambiar, incluso en tiempo de ejecución por parte de agentes móviles de una máquina a otra, como y cuando sea necesario. Jade está completamente implementado en lenguaje Java y la exigencia mínima del sistema es la versión 5 de Java.

Franco Barrios (Barrios, 2009) también menciona que esta plataforma incluye contenedores que no son más que una instancia del entorno de ejecución de Jade. Un contenedor puede hospedar un número indeterminado de agentes. Existe un tipo especial de contenedor denominado *"main container"*, y debe existir uno y solo uno de estos contenedores especiales por cada plataforma FIPA de agentes; el resto de contenedores de la plataforma, una vez ejecutados (en sus correspondientes entornos de ejecución Jade) deben subscribirse al principal, por lo que el responsable de ejecutarlos también es responsable de indicar en donde se localiza el contenedor principal.

Según (Marchetti, y otros, 2003) Jade crea y maneja una cola de mensajes ACL entrantes; los agentes pueden acceder a su cola mediante una combinación de varios modos: blocking, polling, timeout y pattern matching. Implementa completamente el modelo de comunicación de FIPA distinguiendo a sus componentes integrados: protocolos de interacción, ACL, lenguaje de contenido, esquemas de codificación, ontologías y finalmente protocolos de transporte. El mecanismo de transporte, en particular, se adapta a cada situación, seleccionando el mejor protocolo disponible entre RMI de Java (Java Remote Method Invocation – Invocación de Método de Manera Remota), notificación de eventos e IIOP (Internet Inter-ORB Protocol – Protocolo de Internet Inter-ORB, donde ORB: Object Request Broker). Otros protocolos pueden ser fácilmente agregados: SMTP (Simple Mail Transfer Protocol - Protocolo simple de transferencia de correo electrónico), HTTP (Hypertext Transfer Protocol - Protocolo de Transferencia de Hipertexto) y WAP (Wireless Application Protocol -Protocolo de Aplicaciones Inalámbricas).

El lenguaje de comunicación de agente (Agent Communication Language - ACL) define la estructura que debe tener cada uno de los mensajes que son enviados por los agentes a los demás agentes, permitiendo así el intercambio y entendimiento de mensajes entre estos.

Dado que JADE se implementa bajo las especificaciones establecidas por FIPA, es necesario indicar que cuenta con tres agentes especiales como se señala en (Peñaranda Cebrián, 2016):

- Agente ACC (Agent Comunnication Chanel): Agente que trabaja sobre un canal de comunicación que permite el dialogo entre los agentes del sistema y de la plataforma. Controla el intercambio de mensajes.

- Agente DF (Directory Facilator): Agente que informa sobre qué agentes están disponibles en el AMS y cuáles son los servicios que proporcionan, también conocido como “Páginas Amarillas”.

- Agente AMS (Agent Management System): Tiene el control sobre la creación y destrucción de los agentes, e incluso puede detener la plataforma. Garantiza que cada agente en la plataforma sea identificado por un único nombre AID (Agente ID – Identificación de Agente). Proporciona el ciclo de vida y mantiene el directorio de los identificadores de agentes. Cada agente se registra con el AMS para obtener un AID válido.

Todo agente definido en esta plataforma debe seguir un ciclo de la vida que es propuesto también por FIPA. Este ciclo de vida cuenta con seis estados que un agente puede atravesar:

- Iniciado: El agente ha sido creado, pero aún no ha sido registrado.
- Activo: Ha sido registrado y puede comunicarse con otros agentes.
- Suspendido: Su tarea a realizar ha sido suspendida.
- Esperando: Espera el suceso de un evento.
- Eliminado: El AMS ha eliminado al agente.
- Transito: El agente está migrando a una nueva ubicación.

En la Figura 10 se pueden observar todos los estados definidos en el ciclo de vida de un agente y la forma que los puede transitar a cada uno de ellos.

Para emplear los agentes de software y que ellos brinden sus servicios es necesario una plataforma para su ejecución. JADE (Java Agent DEvelopment Framework) constituye una plataforma y un conjunto de herramientas que facilitan el desarrollo de los agentes que se ejecutarán en ella. Existen otras tecnologías de este tipo, sin embargo se ha considerado a JADE por su trayectoria, comunidad creciente y por el hecho de estar liberada bajo una licencia de código abierto (LGPL[5]).

[5]LGPL: (*Lesser GNU Public License*) en su version 2.1. Licencia que permite el uso de la tecnología con cualquier fin (sea comercial o no); pero exige que cualquier trabajo derivado de esta, debe ser publicado con la misma licencia. Esta asegura que cualquiera pueda utilizar el producto y, además, que todas las correcciones y mejoras serán retribuidas a la comunidad para que otros puedan aprovecharlas(San Martín, 2012).

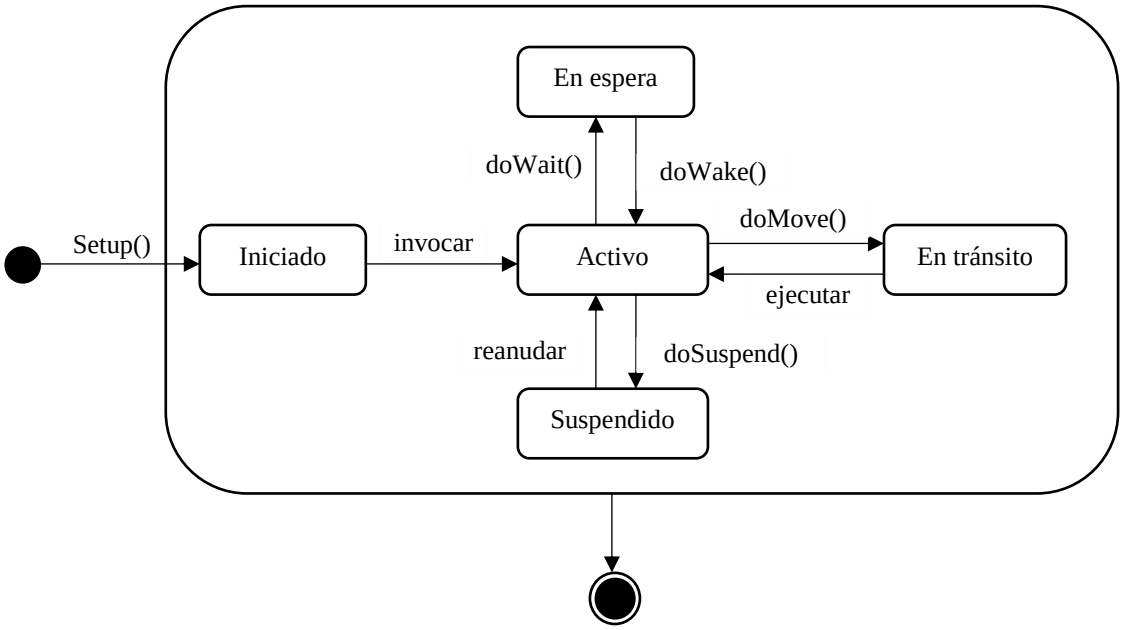


Figura 10 - Visión gráfica de los estados de un agente en JADE

II.5.5. MySQL

En (Criollo Guatapi, y otros, 2007) definen a MySQL como un sistema de gestión de bases de datos relacional, creada por la empresa sueca MySQL AB, la cual tiene el copyright del código fuente del servidor SQL, así como también de la marca.

MySQL es un software de código abierto, licenciado bajo la GPL de la GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL. El lenguaje de programación que utiliza MySQL es Structured Query Language (SQL) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

II.5.6. phpMyAdmin

Fabricio Carlos (Ibañez, 2014) hace mención de phpMyAdmin como una herramienta de software libre escrito en PHP encargada de manejar la administración de MySQL a través de la Web. phpMyAdmin es compatible con una amplia gama de operaciones de MySQL y MariaDB. Las operaciones que se realizan con más frecuencia (operaciones de gestión de bases de datos, tablas, columnas, relaciones, índices, usuarios,

permisos, etc.) se pueden realizar a través de la interfaz de usuario, que además tiene la capacidad de ejecutar directamente cualquier sentencia SQL.

CAPITULO III – ESPECIFICACIÓN DE REQUERIMIENTOS

III.1. INTRODUCCIÓN

El presente documento contiene las Especificaciones de Requisitos de Software del prototipo de aplicación web, el metabuscador colaborativo basado en agentes al que llamaremos "June". Parte de la estructura del presente documento está basada en las directrices referidas en el estándar IEEE 830-1998 Recommended Practice for Software Requirements Specifications.

Esta documentación es el punto de partida para analizar, diseñar y construir la aplicación "June". El producto final pretendido es un prototipo del metabuscador colaborativo basado en agentes de software.

III.1.1. Propósito

El propósito del presente documento es definir de manera clara y precisa tanto las funcionalidades como las posibles restricciones del prototipo a construir. A los efectos del Trabajo Final, esta documentación va dirigida a los desarrolladores mismos de la aplicación web.

III.1.2. Ámbito

La documentación es de utilidad para las etapas de diseño, codificación e implantación del prototipo metabuscador "June".

El metabuscador "June" es una aplicación web que permite a los usuarios de un grupo realizar búsquedas, acceder a los resultados obtenidos para luego, si es de su interés, poder calificarlos y comentarlos debidamente. Además, permite a los usuarios del grupo visualizar un historial con los resultados de búsqueda que fueron calificados y/o comentados de otros usuarios, ordenado por la valoración grupal de cada resultado. A su vez cada usuario del grupo podrá visualizar en los resultados las calificaciones (con o sin comentarios) de cada usuario que calificó y/o comentó.

Además permite acceder a información de ayuda o un instructivo para la utilización de dicha aplicación. También dará la posibilidad de contactarse con los desarrolladores mediante un formulario de contacto.

III.1.3. Referencias

La referencia utilizada para lograr el presente documento es el Estándar IEEE 830-1998 Recommended Practice for Software Requirements Specifications.

III.2. DESCRIPCIÓN GENERAL

III.2.1. Perspectiva del Producto

El metabuscador "June" será una aplicación web, producto diseñado para trabajar de forma descentralizada. Además será independiente por lo tanto no interactuará con otros sistemas o aplicaciones.

III.2.2. Funciones del Producto

La funcionalidad del metabuscador "June" se describe por las secciones a la que tendrán accesos los usuarios.

- Sección de Búsqueda (Sección Principal): Sección en donde un usuario realiza las búsquedas de material. Cuando se lleve a cabo una búsqueda el metabuscador mostrará un listado de resultados, dándole la posibilidad de calificarlos (y si le interesa de comentarlos). Además, los resultados del listado mostraran quienes fueron los usuarios que lo calificaron, sus calificaciones y comentarios si es que lo hicieron.

- Sección de Ayuda: Sección a la que un usuario podrá acceder para obtener información de cómo usar el metabuscador.

- Sección de Contacto: Sección a la que un usuario podrá acceder para contactarse con los desarrolladores mediante un formulario de contacto.

- Sección de Historial: Sección en la que un usuario podrá acceder a un historial de resultados de búsquedas calificados y/o comentados por el grupo.

III.2.3. Características de los Usuarios

Cada usuario es un estudiante que interactuará continuamente con el metabuscador. A los fines del presente trabajo final, los estudiantes serán de nivel universitario y no necesariamente deberá tener una capacitación respecto a la aplicación, pero si deberán tener una mínima noción de cómo utilizar los buscadores más conocidos, como ser google, yahoo, bing, y ask.

III.2.4. Restricciones

- *Políticas Reguladoras*

La aplicación se desarrollará utilizando software de licencia abierta por lo que no se pagará por el uso de:

- ✓ Sistema Operativo CentOS 6.8
- ✓ Servidor Web Apache 2.
- ✓ Servidor Web Apache Tomcat 8.0 & Jade 4.4.0.
- ✓ Sistema de Gestión de Base de Datos – MySQL.
- ✓ Lenguaje de programación: PHP, JAVA.
- ✓ Librerías: PHP / JAVA Bridge, PHPMailer v5.0, PHP Simple HTML DOM Parser.

La utilización de los distintos softwares y librerías anteriormente nombrados está sujeta a sus propias políticas de licenciamiento.

- *Limitaciones de Hardware*

Las limitaciones consideradas para el metabuscador “June” están dadas por:

- ✓ Una PC convencional o una máquina virtual con al menos 2 GB de Ram, con un procesador Dual Core como mínimo.
- ✓ Sistema Operativo CentOS 6.8
- ✓ Instalación del Servidor Web Apache, Servidor Web Apache Tomcat, Jade MySQL, PHPMailer v5.0 y de PHP / JAVA Bridge, además de ser correctamente configurados.

- *Interface con otras Aplicaciones*

El metabuscador debe implementar la API (Interfaz de Programación de Aplicación) PHP/JAVA Bridge, que permitirá la comunicación interna con Jade.

- *Requisitos del Lenguaje*

La aplicación debe estar en español.

- *Protocolos Señalados*

Se utilizarán protocolos de comunicación TCP/IP, HTTP, SSL.

- *Consideraciones de Seguridad*

A los fines del prototipo a desarrollar en el presente trabajo final, cada usuario accederá al metabuscador desde una dirección web distinta dentro del mismo dominio, para autenticarse. Todos los nombres de usuario y contraseñas están generados por los desarrolladores del metabuscador.

III.2.5. Atención y Dependencias

La red interna con la que convivirá el metabuscador deberá de estar configurada para el manejo de protocolos TCP/IP, HTTP, principalmente todo lo relacionado en cuanto a desempeño y seguridad.

Para evitar cualquier inconveniente de compatibilidad debido a sus características de innovación y al uso con las tecnologías web, el metabuscador deberá ejecutarse en las últimas versiones de los siguientes navegadores: Mozilla Firefox y Google Chrome.

III.2.6. Prorratear los Requisitos

A los fines del presente trabajo final, los siguientes aspectos no serán tenidos en cuenta en el producto obtenido:

a) No incluirá la opción de hacer búsqueda por imagen o por video.
b) No incluirá opciones sobre la cantidad de resultados a mostrar.
c) No incluirá filtros para hacer búsquedas por fecha o por idioma.

III.3. REQUISITOS ESPECIFICOS

Los Requisitos Específicos van a describir el comportamiento de la aplicación. La elaboración de este apartado, en lo que respecta a los Requisitos Funcionales, adopta actividades orientadas a la Metodología OOHDM.

III.3.1. Requisitos Funcionales - RF

- ***R1: La aplicación permite al usuario realizar una búsqueda***
  - R1.1 - El usuario debe proporcionar palabra o frase de búsqueda, la cual se replica en varios buscadores para obtener variedad de resultados.
  - R1.2 - El usuario debe seleccionar todos o al menos un buscador con los que operará la aplicación.
  - R1.3 - Los resultados de búsqueda se presentan en un listado de resultados.
  - R1.4 - El usuario puede acceder a más de un listado de resultados para una misma palabra/frase de búsqueda.
- ***R2: La aplicación ordena los resultados de búsqueda de acuerdo a una calificación grupal***
  - R2.1 - La posición de cada resultado de búsqueda quedará determinada a partir de un promedio grupal. Este promedio se obtiene de la calificación individual de cada usuario.
- ***R3: El usuario puede conocer qué usuario/s calificó/calificaron un resultado***
  - R3.1 - En cada resultado de búsqueda se identifica los usuarios que lo calificaron/comentaron junto con sus calificaciones/comentarios.
- ***R4: El usuario puede calificar y comentar cada resultado de búsqueda***
  - R4.1 - El usuario puede calificar cada resultado de búsqueda.
  - R4.2 - El usuario puede cambiar su calificación en cualquier momento.
  - R4.3 - El usuario puede quitar su calificación en cualquier momento.
  - R4.4 - El usuario puede comentar cada resultado de búsqueda.
  - R4.5 - El usuario puede cambiar su comentario en cualquier momento.
  - R4.6 - El usuario puede quitar su comentario en cualquier momento.

- ***R5: La aplicación proporciona una explicación de su funcionamiento***

- ***R6: El usuario puede comunicarse con los desarrolladores del software***

  - R6.1 - El usuario puede enviar un mensaje a los responsables de la aplicación.

- ***R7: El usuario puede acceder a un historial de resultados de búsquedas calificados y/o comentados por el grupo.***

- ***R8: El usuario puede salir de la aplicación.***

III.3.2. RF - Identificación De Roles Y Tareas

Existen 3 roles en la aplicación:

- Estudiante: Es el estudiante el que interactúa con la aplicación, por lo tanto deberá ingresar una URL en la barra de dirección de un navegador para acceder a ella. El estudiante puede seleccionar los buscadores con los que operará la aplicación (la selección se realiza sobre un conjunto pequeño que involucra los buscadores más conocidos). Puede realizar una búsqueda a partir de una palabra o frase, calificar y comentar un resultado de búsqueda como también modificar o quitar esa calificación y/o comentario en cualquier momento, puede visualizar quién califico/comentó un resultado de búsqueda, obtener una explicación del funcionamiento (modo de uso) del sistema, puede contactarse con los desarrolladores del sistema, y por último, puede acceder a un historial de resultados de búsquedas calificados y/o comentados por el grupo.

  Tareas:

  - ✓ *RealizarBúsqueda*: Selecciona todos o al menos un buscador. Ingresa una palabra o frase de búsqueda en un campo de texto de la aplicación e inicia el proceso para generar el primer listado de resultados o los siguientes listados para esa palabra o frase.
  - ✓ *Calificar/ComentarResultado:* Califica uno o varios resultados del listado de resultados. Al calificar un resultado puede o no comentarlo
  - ✓ *Modificar/Quitar Calificación de Resultado:* Modifica o quita su calificación de uno o varios resultados del listado de resultados.

- ✓ *Modificar/Quitar Comentario de Resultado:* Modifica o quita su comentario de uno o varios resultados del listado de resultados.
- ✓ *Ayuda*: Consulta una explicación del funcionamiento de la aplicación.
- ✓ *Contactar Desarrolladores*: Se comunica con los desarrolladores de la aplicación.
- ✓ *Generar Historial*: Permite generar un historial ordenado con los resultados de búsquedas calificados y/o comentados por el grupo.
- ✓ *Salir*: Sale de la aplicación.

- Agente de Consulta: Es el que se encarga de recibir la lista de resultados de búsqueda o un pedido de historial de resultados de búsquedas calificados y/o comentados. Luego devuelve a la aplicación, según corresponda, un listado ordenado o un historial ordenado con los resultados de búsqueda del grupo, indicando en ambos casos quiénes calificaron, sus calificaciones y/o sus comentarios sobre cada resultado.

  Tareas:

  - ✓ *Recibir Lista de Resultados de Búsqueda*.
  - ✓ *Enviar Lista Ordenada con los Resultados Calificados y/o Comentados.*
  - ✓ *Recibir pedido de Historial de Resultados de Búsquedas.*
  - ✓ *Enviar Historial Ordenado con los Resultados Calificados y/o Comentados.*

- Agente de Actualización de BD: Es el que se encarga de recibir los parámetros para asociarlos a un resultado: identificación del estudiante, URL, calificación y comentario. El Agente de Actualización, envía una señal que indica que se han asociado los parámetros al resultado.

  Tareas:

  - ✓ *Recibir Parámetros.*
  - ✓ *Enviar Señal de Actualización*

III.3.3. RF - Especificación De Los Escenarios

- *Realizar Búsqueda:* Un estudiante desea realizar una búsqueda con la aplicación, para ello debe seleccionar todos o al menos un buscador con los que operará la aplicación, luego ingresa una palabra o una frase en un campo de texto de la aplicación e inicia el proceso de búsqueda. También podrá realizar sucesivas búsquedas para esa misma

palabra o frase. La aplicación devolverá los resultados de búsqueda con la forma de un listado confeccionado y ordenado. En cada resultado se visualizará también quién o quiénes son los estudiantes que lo calificaron, sus calificaciones correspondientes junto con sus comentarios, si es que lo hicieron.

- *Calificar y/o Comentar un Resultado:* Un estudiante podrá calificar un resultado que considere de interés de un listado de resultados de búsqueda. Luego, de haber calificado un resultado, el estudiante podrá comentarlo, es decir: si el estudiante no calificó un resultado, no podrá comentarlo. Una vez calificado y/o comentado, la aplicación volverá a mostrar en pantalla el mismo listado de resultados pero con el resultado calificado y comentado.

- *Modificar/Quitar Calificación de Resultado:* Un estudiante podrá modificar o quitar la calificación que realizó él mismo sobre un resultado. El valor de la calificación va a incidir en si se actualiza la calificación del resultado o directamente se la quita.

- *Modificar/Quitar Comentario de Resultado:* Un estudiante podrá modificar o quitar la calificación que realizó él mismo sobre un resultado.

- *Contactar Desarrolladores:* Un estudiante podrá escribir un mensaje a los desarrolladores con la finalidad de sugerir mejoras o manifestar algún inconveniente con la aplicación.

- *Solicitar Ayuda:* Un estudiante podrá visualizar una sucesión de explicaciones sobre el funcionamiento de la aplicación con la finalidad de que la misma sea bien entendida por los estudiantes.

- *Generar Historial:* Un estudiante podrá acceder a un historial ordenado de resultados de búsquedas calificados con o sin comentarios por todos los integrantes del grupo.

- *Salir:* Un estudiante podrá salir de la aplicación cuando lo desee.

III.3.4. RF -Especificación De Casos De Uso

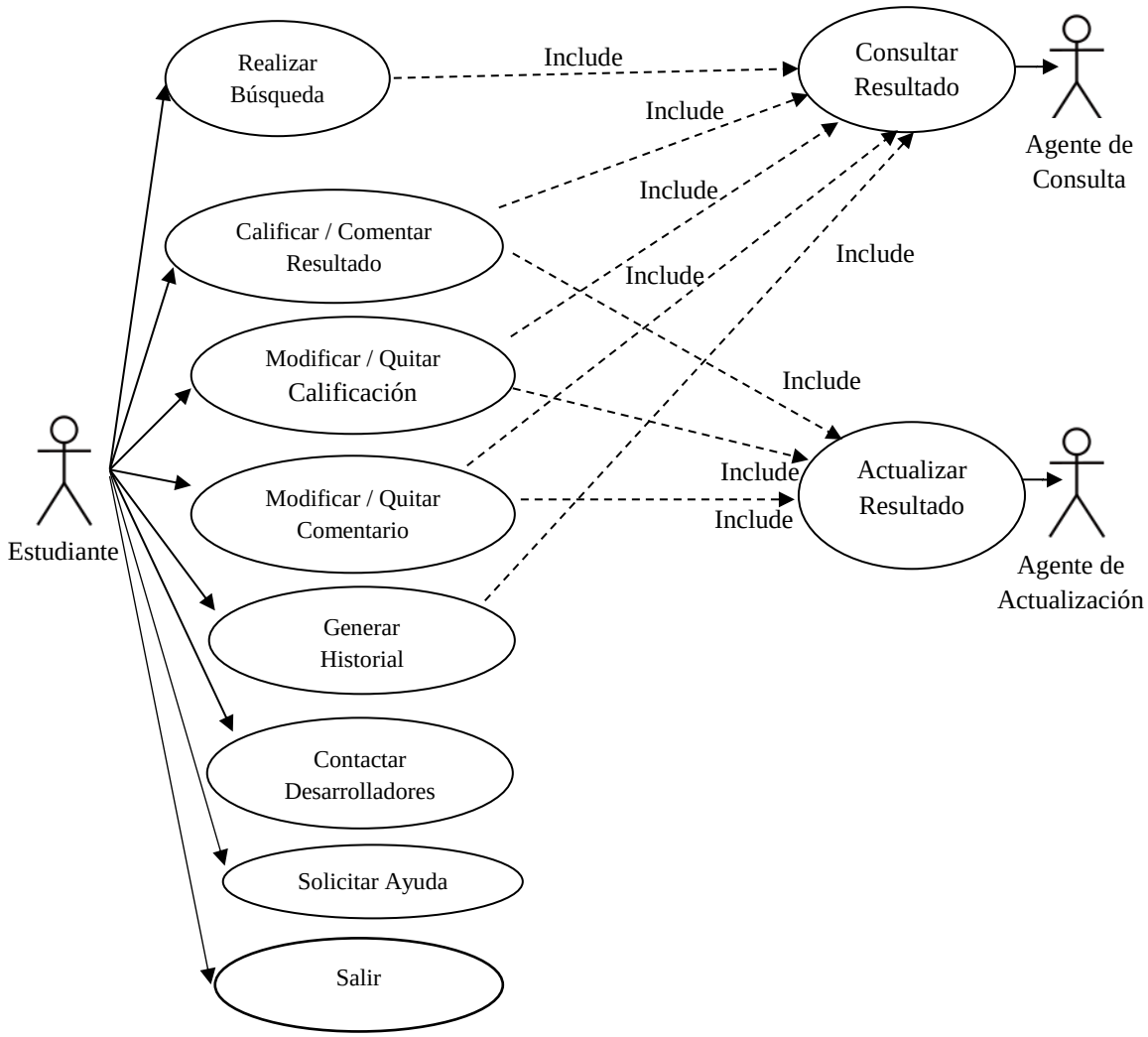


Figura 11 - Especificación de caso de uso

CASO DE USO	*REALIZAR BUSQUEDA*
ACTORES	Estudiante (Actor Primario) Agente de Consulta (Actor Secundario)
REFERENCIAS	R1.1, R1.2, R1.3, R1.4, R2.1, R3.1
PRECONDICIONES	El usuario debe tener asignada una dirección web propia, asociada al dominio del metabuscador, para ingresar a la aplicación.
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante accede a la aplicación mediante el ingreso de su URL	

	2. Solicita el ingreso de la palabra o frase de búsqueda, y la selección de los buscadores con los que puede operar
3. El Estudiante selecciona al menos un buscador e ingresa la palabra o frase de búsqueda	
	4. Opera con los buscadores seleccionados para obtener los resultados de cada uno de ellos
	5. Genera una lista de resultados de búsqueda inicial eliminando los resultados duplicados y ordenándola priorizando (al comienzo de la lista) aquellos que más se repiten entre los obtenidos de los buscadores
	6. Envía la lista de resultados al Agente de Consulta
	7. El Agente de Consulta proporciona a la aplicación una lista con los resultados consultados
	8. Muestra los resultados de la lista ordenados e indicando en cada uno de ellos quiénes lo calificaron, sus calificaciones y/o sus comentarios
FLUJO ALTERNATIVO	---
POSCONDICIONES	El estudiante visualiza el listado de resultados de búsqueda.
COMENTARIOS	El Estudiante puede realizar una búsqueda con una palabra o frase de búsqueda como también puede proceder a una siguiente búsqueda para la misma palabra o frase.

Tabla 1 - Caso de uso “Realizar Búsqueda"

CASO DE USO	*CALIFICAR Y/O COMENTAR RESULTADO*
ACTORES	Estudiante (Actor Primario) Agente de Actualización (Actor Secundario) Agente de Consulta (Actor Secundario)
REFERENCIAS	R2.1, R3.1, R4.1, R4.4
PRECONDICIONES	La aplicación debe mostrar previamente una lista de resultados de búsqueda.
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante indica que desea calificar un resultado	
	2. Solicita la calificación para el resultado
3. El Estudiante ingresa una calificación	

	4. Solicita al estudiante agregar un comentario
5. El estudiante ingresa un comentario	
	6. Verifica que el Estudiante ingresó al menos la calificación
	7. Envía los datos ingresados por el estudiante, junto con otros parámetros asociados (identificación del estudiante, URL, calificación y/o comentario) del resultado al Agente de Actualización
	8. El Agente de Actualización ingresa una señal (dato) de actualización a la aplicación
	9. Envía la lista de resultados al Agente de Consulta
	10. El Agente de Consulta proporciona a la aplicación una lista con los resultados consultados
	11. Muestra los resultados de la lista indicando en cada uno de ellos quienes lo calificaron, sus calificaciones y/o sus comentarios
FLUJO ALTERNATIVO	---
POSCONDICIONES	---
COMENTARIOS	Se requiere al menos el ingreso de la calificación.

Tabla 2 - Caso de Uso "Calificar y/o comentar resultado"

CASO DE USO	MODIFICAR/QUITAR CALIFICACIÓN DE RESULTADO
ACTORES	Estudiante (Actor Primario) Agente de Actualización (Actor Secundario) Agente de Consulta (Actor Secundario)
REFERENCIAS	R2.1, R3.1, R4.2, R4.3
PRECONDICIONES	La aplicación debe mostrar previamente una lista de resultados de búsqueda. Además, el estudiante debe haber calificado previamente el resultado al que se le va a modificar o quitar su calificación.
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante solicita cambiar su calificación de un resultado	
	2. Devuelve al estudiante la calificación del resultado solicitado
3. El Estudiante ingresa una nueva calificación para el resultado	

	4. Envía los datos ingresados por el estudiante, junto con otros parámetros asociados (identificación del estudiante, URL, calificación y/o comentario) al resultado al Agente de Actualización
	5. El Agente de Actualización ingresa una señal (dato) de actualización a la aplicación
	6. Envía la lista de resultados al Agente de Consulta
	7. El Agente de Consulta proporciona a la aplicación una lista con los resultados consultados
	8. Muestra los resultados de la lista indicando en cada uno de ellos quienes lo calificaron, sus calificaciones y/o sus comentarios
FLUJO ALTERNATIVO	---
POSCONDICIONES	---
COMENTARIOS	El valor de la calificación va a incidir en si se actualiza la calificación del resultado o directamente se la quita.

Tabla 3 - Caso de Uso “Modificar/Quitar calificación resultado”

CASO DE USO	MODIFICAR/QUITAR COMENTARIO DE RESULTADO
ACTORES	Estudiante (Actor Primario) Agente de Actualización (Actor Secundario) Agente de Consulta (Actor Secundario)
REFERENCIAS	R2.1, R3.1, R4.5, R4.6
PRECONDICIONES	La aplicación debe mostrar previamente una lista de resultados de búsqueda. Además, el estudiante debe al menos haber calificado previamente el resultado al que se le va a modificar o quitar su comentario.
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante solicita cambiar su comentario de un resultado	
	2. Devuelve al estudiante el comentario del resultado solicitado
3. El Estudiante modifica su comentario	
	4. Envía los datos ingresados por el estudiante, junto con otros parámetros asociados (identificación del estudiante, URL, calificación y/o comentario) al resultado al Agente de Actualización

	5. El Agente de Actualización ingresa una señal (dato) de actualización a la aplicación
	6. Envía la lista de resultados al Agente de Consulta
	7. El Agente de Consulta proporciona a la aplicación una lista con los resultados consultados
	8. Muestra los resultados de la lista indicando en cada uno de ellos quienes lo calificaron, sus calificaciones y/o sus comentarios
FLUJO ALTERNATIVO	---
POSCONDICIONES	---
COMENTARIOS	El contenido del comentario (si está vacío o no) va a incidir en si se actualiza el comentario del resultado o directamente se lo quita.

Tabla 4 - Caso de Uso “Modificar/Quitar comentario resultado”

CASO DE USO	*CONTACTAR DESARROLLADORES*
ACTORES	Estudiante (Actor Primario)
REFERENCIAS	R6
PRECONDICIONES	---
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante solicita contactarse con los desarrolladores	
	2. Muestra un formulario de contacto solicitando que ingrese su nombre, apellido, correo electrónico, asunto y comentario.
3. El Estudiante ingresa los datos solicitados	
	4. Verifica que todos los datos hayan sido ingresados
	5. Muestra una pantalla de despedida indicando que se han enviado todos los datos a los desarrolladores
6. El Estudiante sale de la pantalla	
FLUJO ALTERNATIVO	
4. Verifica que todos los datos hayan sido ingresados 4.a. La aplicación informa que hay datos que no han sido ingresados 4.b. La aplicación solicita que ingrese todos los datos. 4.c. Vuelve al paso 3	

POSCONDICIONES	---
COMENTARIOS	---

Tabla 5 - Caso de Uso “Contactar Desarrolladores”

CASO DE USO	*SOLICITAR AYUDA*
ACTORES	Estudiante (Actor Primario)
REFERENCIAS	R5
PRECONDICIONES	---
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante solicita hacer Consulta (Modo de Uso)	
	2. Muestra textos con imágenes que indican cómo utilizar la aplicación
3. El Estudiante sale de la pantalla	
FLUJO ALTERNATIVO	---
POSCONDICIONES	---
COMENTARIOS	---

Tabla 6 - Caso de Uso “Solicitar Ayuda”

CASO DE USO	*GENERAR HISTORIAL*
ACTORES	Estudiante (Actor Primario) Agente de Consulta (Actor Secundario)
REFERENCIAS	R7
PRECONDICIONES	Los estudiantes del grupo deben haber calificado al menos un resultado.
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante solicita un historial.	
	2. Genera un historial ordenado con los resultados de búsquedas calificados y/o comentados por el grupo.
	3. Muestra los resultados del historial, indicando en cada uno de ellos quiénes lo calificaron, sus calificaciones y/o sus comentarios

FLUJO ALTERNATIVO	---
POSCONDICIONES	El estudiante visualiza el historial de resultados de búsqueda de un grupo.
COMENTARIOS	---

Tabla 7 - Caso de uso "Generar Historial"

CASO DE USO	*SALIR*
ACTORES	Estudiante (Actor Primario)
REFERENCIAS	R8
PRECONDICIONES	El estudiante debe haber accedido a la aplicación.
FLUJO PRINCIPAL	
Actores	**Aplicación**
1. El Estudiante solicita salir de la aplicación.	
	2. Muestra mensaje de despedida.
3. El Estudiante sale de la pantalla	
FLUJO ALTERNATIVO	---
POSCONDICIONES	---
COMENTARIOS	---

Tabla 8 - Caso de Uso "Salir"

III.3.5. RF -Especificación De Diagramas De Interacción De Usuarios

UID correspondiente al caso de uso de "Realizar Búsqueda"

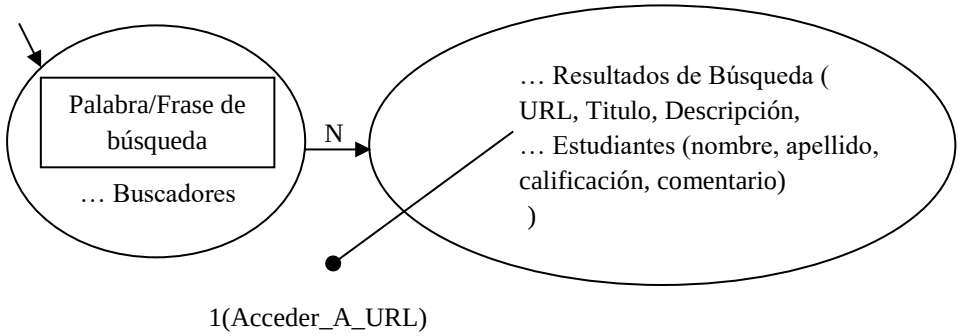


Figura 12 - UID: "Realizar Búsqueda"

UID correspondiente al caso de uso de "Calificar y/o Comentar Resultado"

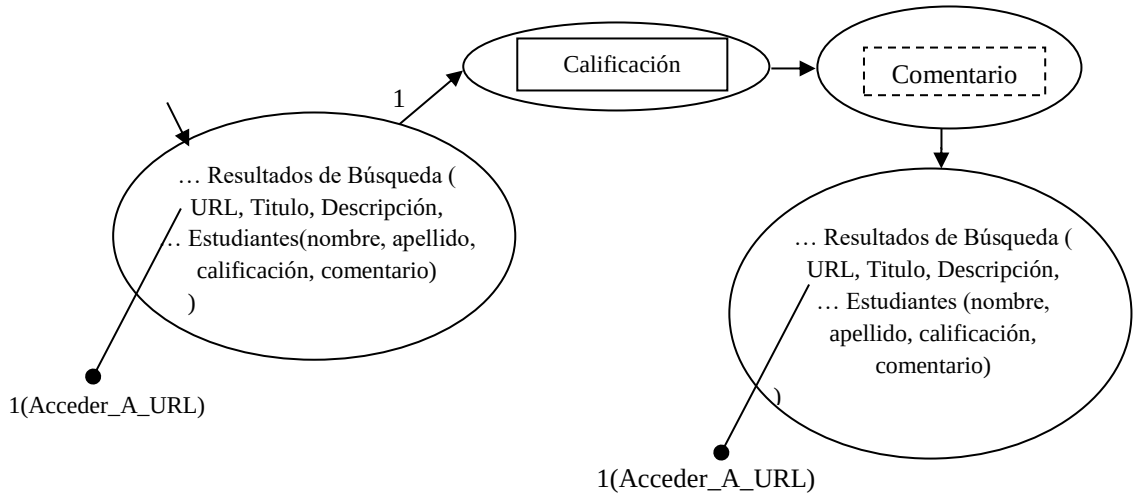


Figura 13 - UID: "Calificar y/o Comentar Resultado"

UID correspondiente al caso de uso de "Modificar/Quitar Calificación de Resultado"

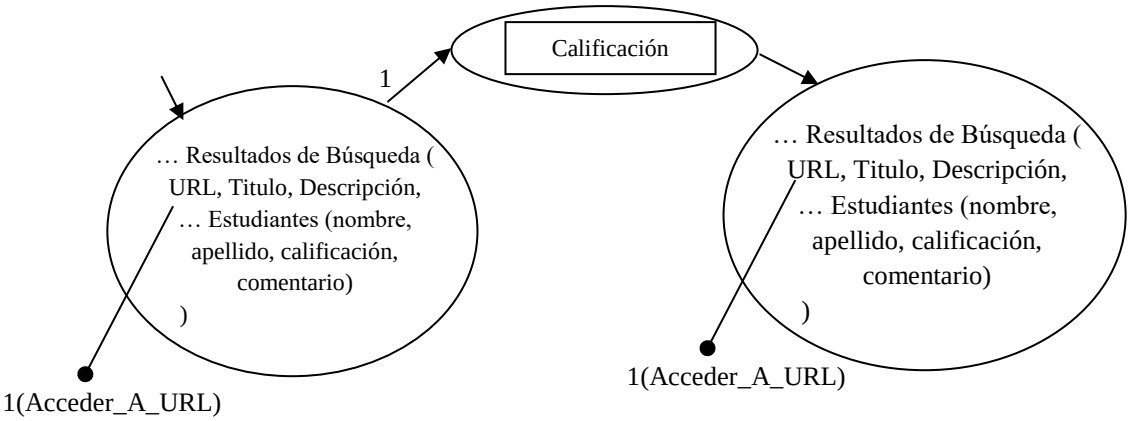


Figura 14 - UID: "Modificar/Quitar Calificación de Resultado"

UID correspondiente al caso de uso de "Modificar/Quitar Comentario de Resultado"

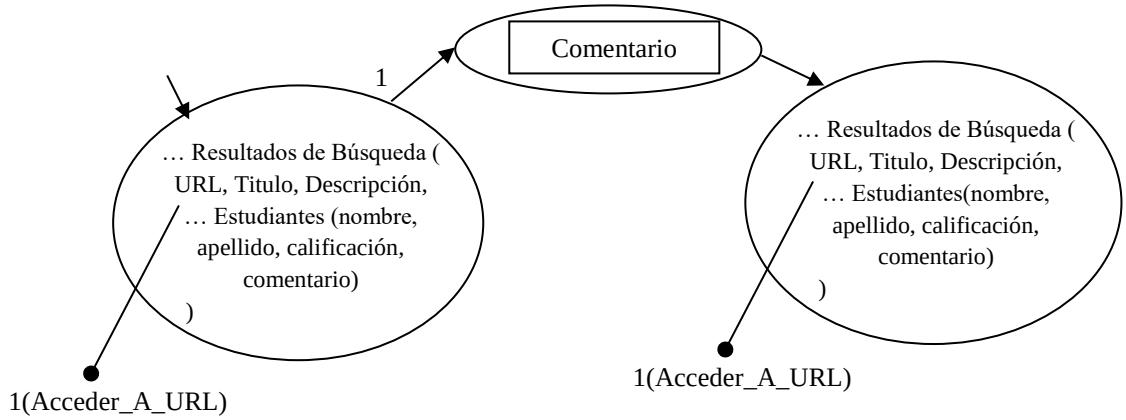


Figura 15 - UID: "Modificar/Quitar Comentario de Resultado"

UID correspondiente al caso de uso de "Contactar Desarrolladores"

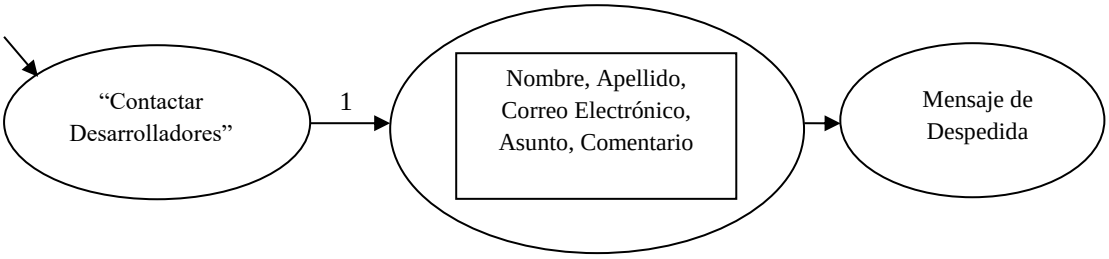


Figura 16 - UID: "Contactar Desarrolladores"

UID correspondiente al caso de uso de "Ayuda"

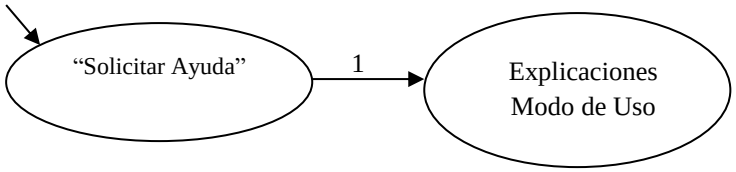


Figura 17 - UID: "Ayuda"

UID correspondiente al caso de uso de "Generar Historial"

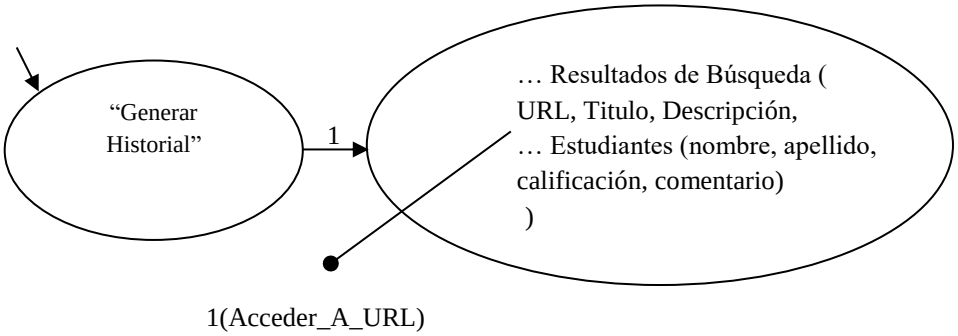


Figura 18 - UID: "Generar Historial"

UID correspondiente al caso de uso de "Salir"

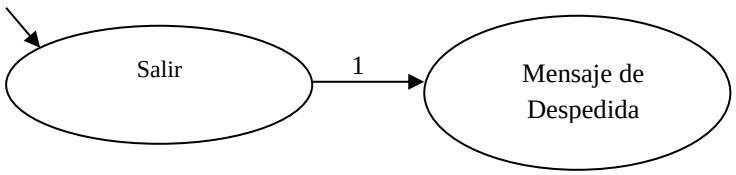


Figura 19 - UID: "Salir"

III.3.6. Requisitos No Funcionales

- *Seguridad*: Todo nombre de usuario y contraseña será generado y entregado por los desarrolladores. El único acceso al metabuscador será mediante una dirección web (dentro del mismo dominio) que identifique al usuario.

- *Disponibilidad*: El metabuscador está pensado para que esté disponible 24(horas)/7(días) siempre que el servidor que lo albergue esté funcionando y no detenido.

- *Mantenibilidad*: La mantenibilidad del metabuscador debe ser realizada por los desarrolladores del mismo.

- *Portabilidad*: El metabuscador debe funcionar correctamente con el Sistema Operativo Windows 7, 8, 10 y con los navegadores Chrome y Firefox.

CAPITULO IV – SOLUCIÓN PROPUESTA

Como se ha mencionado anteriormente, los integrantes de un grupo de estudiantes colaborativos generalmente realizan actividades que requieren búsquedas en la web para obtener información. A pesar de que muchos de los resultados obtenidos por cada uno de ellos se repiten, cada uno de esos resultados debe ser analizado por cada estudiante para determinar su utilidad. Si se multiplica este proceso por la cantidad de individuos que conforman el grupo, es evidente que se produce una considerable pérdida de tiempo y esfuerzo al analizar material que ya fue considerado por otro integrante.

La problemática planteada en el párrafo previo dio origen al metabuscador colaborativo basado en agentes de software que le da solución. La Figura 20 muestra un esquema de funcionamiento del metabuscador propuesto. A continuación se describen con mayor detalle sus componentes.

Como puede verse en la Figura 20, el usuario ingresa las palabras clave en el metabuscador (1), quien se encarga de recuperar los resultados de búsquedas de los motores seleccionados para este trabajo (2,3) (bing, yahoo, google, ask). A partir de las palabras clave ingresadas, se realiza un filtrado para que se eliminen los resultados duplicados. El listado obtenido (listado de URLs) es enviado al Agente de Consulta (AC) (4) que se encargará de consultar la Base de Datos (BD) para recuperar aquellos resultados que ya fueron valorados y comentados. Por cada coincidencia de url entre el listado y el contenido de la BD (5), el AC recupera la identificación del usuario, su valoración y comentario (6). A continuación, el AC devuelve al metabuscador el listado de urls resultantes ordenadas de acuerdo con una valoración grupal, obtenida al promediar las valoraciones individuales de dichos resultados (7). Este listado rankeado es mostrado al usuario (8), quien a partir de él puede acceder a cada resultado, analizarlo y realizar una valoración y comentario del mismo (9). Toda valoración consiste en la asociación de un valor numérico entre 1 y 10 a un resultado (url contenida en el listado). El metabuscador envía el resultado valorado y comentado al Agente de Actualización (AA) (10), quien se encarga de actualizar la BD (11): si la url no existe, la agrega y coloca la valoración, comentario e identificación del usuario que las formula; y si existe, verifica el usuario que realizó la nueva valoración, la agrega junto con el comentario y la identificación correspondiente a dicho usuario. El AA, después de actualizar la BD, envía una señal al metabuscador (12) para que refresque el listado actual de resultados con

el que están trabajando los usuarios a fin de mostrar las actualizaciones efectuadas (se repite 4, 5 ,6 ,7 y 8).

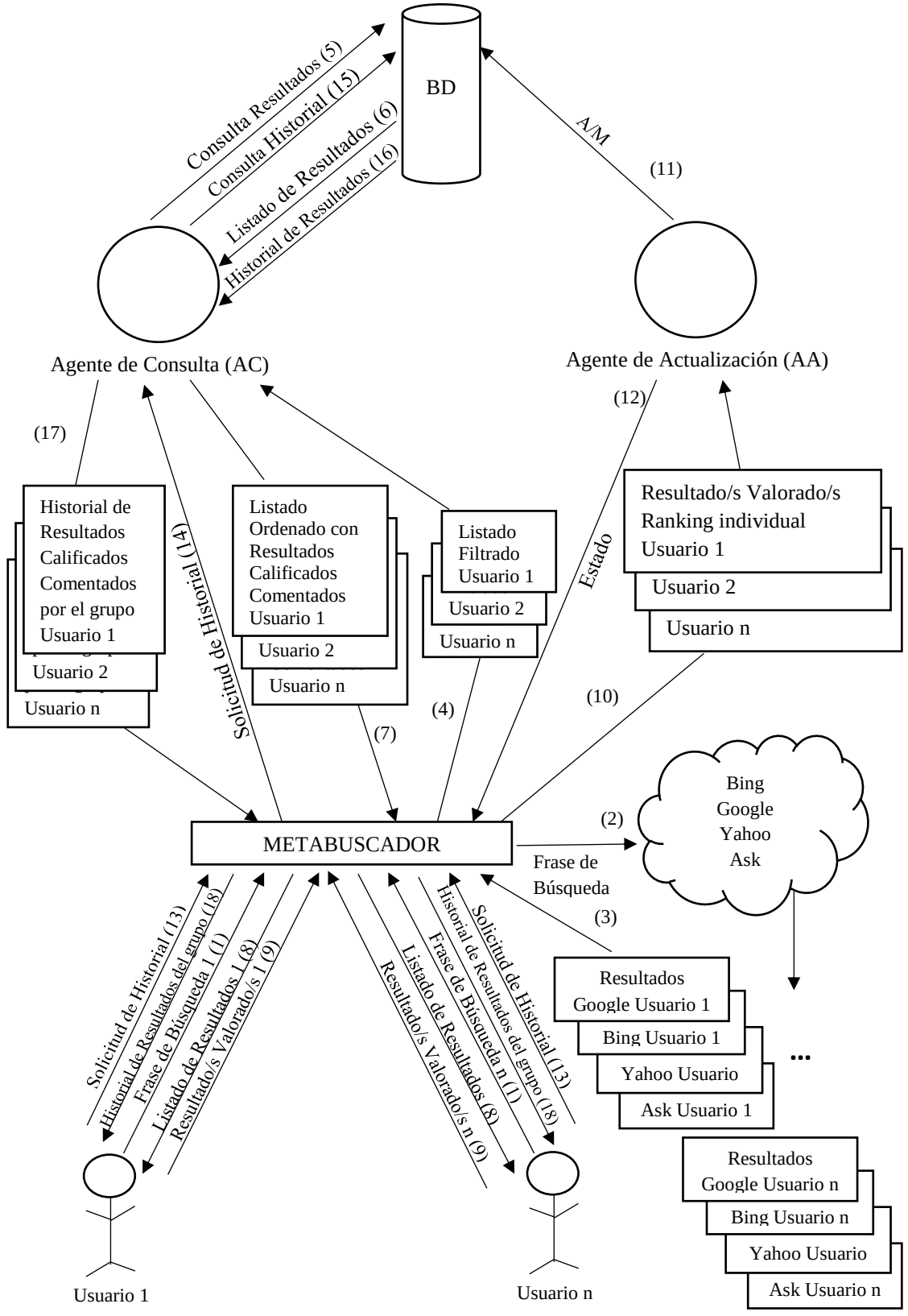


Figura 20 - Esquema de funcionamiento del metabuscador basado en agentes

En el caso del ingreso inicial, cuando la BD se encuentra vacía, el AC devuelve al metabuscador el mismo listado filtrado que le envió éste, sin ordenar y sin valoraciones y/o comentarios. Éste es el listado de resultados que muestra al usuario. En las ejecuciones posteriores el procedimiento se adecúa a la explicación formulada en el párrafo anterior.

El usuario, además, puede solicitar un historial con todos los resultados de búsqueda calificados por el grupo (13). El metabuscador envía el pedido del historial al AC (enviando la identificación del grupo) (14); éste se encargará de consultar la BD (15) para recuperar aquellos resultados que ya fueron valorados y comentados por el grupo (16).

El AC recupera el título, url y descripción de cada resultado a partir de la identificación del grupo, además de recuperar los usuarios que lo calificaron y/o comentaron con sus respectivas calificaciones y comentarios. A continuación, el AC devuelve al metabuscador el historial de urls resultantes ordenadas de acuerdo con una valoración grupal (17), y éste es mostrado al usuario (18).

III.1. ARQUITECTURA DEL METABUSCADOR BASADO EN AGENTES DE SOFTWARE

Para desarrollar el prototipo que responde al esquema de funcionamiento descripto en la sección anterior, debieron realizarse modificaciones a la arquitectura clásica de un metabuscador. En la Figura 21 se presenta la arquitectura resultante para el prototipo del metabuscador basado en agentes de software propuesto.

Las modificaciones introducidas involucraron modificar componentes y quitar otros, como por ejemplo quitar el “Selector de Base de Datos”, ya que el usuario va a realizar esa tarea escogiendo los motores de búsqueda con los que desea trabajar. La tarea del “Selector de Documentos” debió ser considerada en la del “Fusionador de Resultados”.

El componente “Interfaz de Usuario” debe en una primera medida identificar al usuario que utilizará el metabuscador y también al grupo de trabajo al que éste pertenece. Luego realiza tres tareas: una de ellas es la de recibir la frase de búsqueda y los buscadores seleccionados por el usuario para entregárselos al componente “Procesamiento de Datos”, otra es recibir los datos vinculados con una calificación realizada por el usuario sobre un resultado (es decir, la url del resultado, calificación del usuario, y comentario del usuario si lo hiciese) para entregárselos también al componente “Procesamiento de Datos”, y la tercera,

consiste en recibir un pedido de historial de resultados de búsquedas que fueron calificados y/o comentados por los integrantes del grupo, dicho pedido contiene la identificación del grupo para entregárselo al componente "Procesamiento de Datos". Por último, el componente "Interfaz de Usuario" espera recibir del componente "Procesamiento de Datos" el listado de resultados con las calificaciones y/o comentarios asociados a ellos. Luego, debe preparar el listado de manera que el usuario pueda visualizarlo y comprenderlo rápidamente.

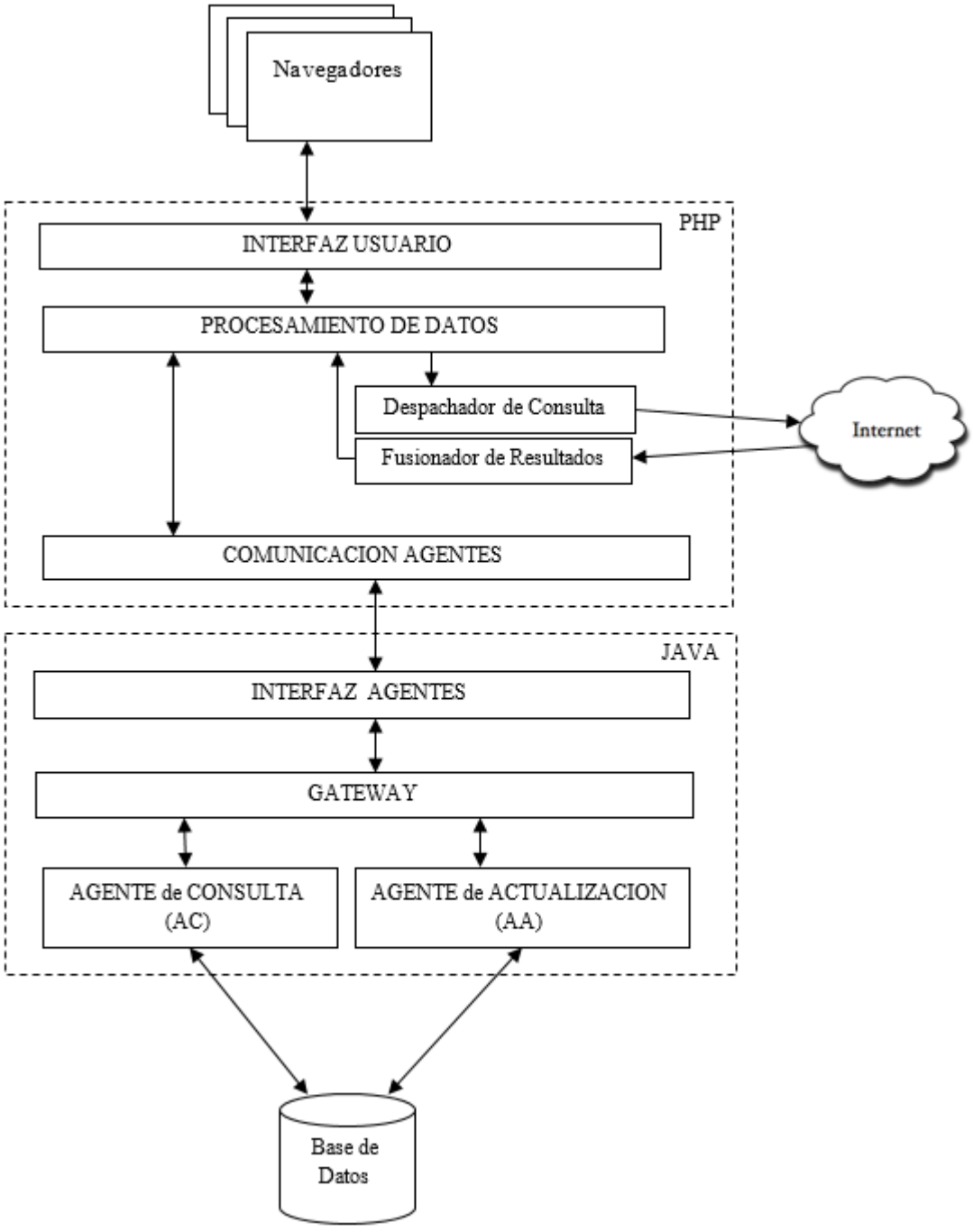


Figura 21 - Arquitectura del Metabuscador basado en Agentes de Software

El nuevo componente “Procesamiento de Datos” tiene la obligación de diferenciar los datos que son destinados al AC (listado de resultados o historial de resultados calificados) de los datos que son destinados al AA (paquete de actualización). En otras palabras, este componente es el encargado de dirigir el procesamiento del metabuscador tomando una decisión por sobre dos caminos.

Uno de los caminos indica que habrá comunicación con el AC. Por un lado, se identificará que el usuario ha realizado una búsqueda. Este camino se inicia con la vinculación al componente “Despachador de Consulta”. Y por otro lado, se identificará que el usuario ha realizado un pedido de historial con todos los resultados de búsquedas calificados y/o comentados. Este camino se inicia con la vinculación al componente “Comunicación Agente”.

El otro camino indica que habrá comunicación primero con el AA y luego con el AC, es decir que el usuario ha realizado una calificación (con o sin comentario) a un resultado y muy posiblemente el orden de los resultados del listado cambie a raíz de la nueva calificación por lo que debe mostrarse nuevamente dicho listado. Este camino se inicia con la vinculación al componente “Comunicación Agente”.

El componente “Despachador de Consulta” tiene la tarea de establecer una conexión con el servidor de cada motor de búsqueda seleccionado y pasar la consulta a los mismos, utilizando la “Técnica de Web Scraping” descripta en el Capítulo II. Cada motor de búsqueda tiene sus propios requisitos sobre el método de solicitud HTTP (por ejemplo, el método GET o el método POST) y el formato de la consulta (por ejemplo, el nombre de la caja de texto para la consulta específica). El despachador de consulta debe considerar los requerimientos de cada motor de búsqueda correctamente.

El componente “Fusionador de Resultados” se encarga de obtener los resultados devueltos por cada buscador utilizando la “Técnica de Web Scraping” (descripta en el capítulo anterior), procesar y fusionar los resultados combinándolos en una sola lista.

Para llevar a cabo la tarea de procesar los resultados, se ha abordado un enfoque sencillo llamado “Recuperación Garantizada” del componente “Selector de Documentos”, que indica la recuperación de todos los resultados útiles o de importancia de cada buscador para cualquier consulta dada. En otras palabras, los listados de resultados devueltos por cada motor de búsqueda no necesariamente van a estar ordenados de la misma manera entre sí, y

además la posición de cada resultado dentro de un listado devuelto por cualquiera de ellos, indica su importancia o utilidad para la frase de búsqueda por la que fue obtenido, entonces es por este motivo que se ha considerado la recuperación de todos los resultados devueltos por cada uno de los motores de búsqueda.

Por último, la tarea de fusionar tiene por objetivo generar una lista con todos los resultados devueltos por los motores de búsqueda seleccionados, que no incluya aquellos que se repiten y además que esté ordenada descendentemente por la cantidad de veces que se repiten entre los distintos motores de búsqueda.

El nuevo componente "Comunicación Agentes" y el nuevo componente "Interfaz Agentes" se complementan como si fueran un único componente ya que tienen la importante tarea de proveer el medio de comunicación con los agentes. Llevan a cabo dos tareas:

1) Recibir del componente "Procesamiento de Datos" el listado de resultados, el paquete de actualización o el pedido de historial, convertir su estructura de datos desarrollada en PHP a una prácticamente idéntica en JAVA agregando un "rótulo" que indique el nombre del agente de destino. Luego se enviará al componente "Gateway" el listado de resultados, el paquete de actualización o el pedido de historial convertido.

2) Recibir del componente "Gateway" el listado de resultados con las calificaciones y/o comentarios, el paquete de actualización con un estado que indicará si se ha llevado a cabo la actualización o el historial con los resultados de búsquedas calificados y/o comentados, reconvertir a la estructura original para ser finalmente devueltos al componente de "Procesamiento de Datos".

El nuevo componente "Gateway" es un tipo de agente de software especial codificado en JAVA que se ejecuta sobre la plataforma JADE, que provee un puente el cual conecta código no JADE con agentes ejecutados sobre dicha plataforma. Para él se han definido las siguientes tareas:

1) Recibir el listado de resultados, el paquete de actualización o un pedido de historial del componente "Interfaz Agente", determinar si el componente de destino es el AC o el AA, crear un mensaje apropiado y luego enviárselo.

2) Recibir un mensaje que contiene el listado de resultados con las calificaciones y/o comentarios, el paquete de actualización con el estado indicando si la actualización fue

realizada o el historial de resultados de búsquedas calificados y/o comentados, del componente correspondiente, para luego entregárselo al componente "Interfaz Agente".

Los componentes de destino pueden ser:

- El componente "Agente Consulta", codificado en JAVA y que se ejecuta sobre la plataforma JADE, el cual desarrolla las siguientes tareas:

  1) Recibir el mensaje del componente "Gateway".
  2) Identificar si el contenido del mensaje corresponde a un listado de resultados, conectarse con la BD, completar el listado de resultados con las calificaciones y/o comentarios a partir de la consulta con la BD; y devolver el listado completo y ordenado al componente "Gateway".
  3) Identificar si el contenido del mensaje corresponde a un pedido de historial, conectarse con la BD, recuperar todos los resultados calificados y/o comentados por el grupo, y devolver un listado ordenado al componente "Gateway".

- El componente "Agente Actualización", codificado en JAVA y que se ejecuta sobre la plataforma JADE, el cual desarrolla las siguientes tareas:

  1) Recibir el paquete de actualización del componente "Gateway".
  2) Conectarse con la BD.
  3) Actualizar BD con los datos incluidos en el paquete.
  4) Indicar en estado del paquete de actualización si la actualización se ha llevado a cabo correctamente.
  5) Devolver el paquete al componente "Gateway".

El prototipo del presente trabajo se implementó considerando la arquitectura básica de un metabuscador debido a que existe amplia documentación al respecto. Se ha optado, además, por la implementación de agentes de software debido a que existe documentación y experiencias sobre ellos aplicados tanto al Aprendizaje Colaborativo Soportado por Computadora (ACSC), como a la recuperación de información.

Los agentes de software que se han desarrollado para el presente trabajo son de tipo reactivos, ya que la solución propuesta a la problemática planteada no requiere de agentes complejos, sino más bien de agentes relativamente simples que interactúen con otros agentes

de manera sencilla, que respondan a un estímulo lo más rápido posible, ya que se trata de un metabuscador cuya característica (como la de cualquier buscador y metabuscador) entre otras es la velocidad.

Además, se implementaron en un servidor, el servicio de Servidor Web Apache con una configuración multi-hilos para la ejecución del código PHP y también el servicio de Servidor Web Apache Tomcat para la ejecución del código en JAVA. La elección de ellos se debe a que son gratuitos, de código abierto, y que además son estables y eficientes.

La elección de PHP se debe a que: es gratuito, es Open Source, es uno de los lenguajes para web más popular soportado por una comunidad enorme de desarrolladores, el rendimiento es eficiente ya que con un servidor modesto puede atender millones de peticiones, posee librerías incluidas para trabajar con un conjunto muy amplio de funciones, está disponible para la mayoría de los sistemas operativos, está orientado a objeto desde la versión PHP5 y tiene soporte para conectarse con una gran variedad de base de datos. Además se hace uso de la técnica web scraping con el objetivo de obtener los resultados de los buscadores debido a que no se ha encontrado en la web el código libre de un metabuscador que se adapte a nuestra propuesta.

La elección de Java se debe principalmente a que es gratuito, es Open Source, es rápido, seguro, y fiable, es multiplataforma, es orientado a objetos, tiene un número grande de librerías, además de ser uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

La elección de PHP/Bridge como vinculación entre PHP y Java, en el desarrollo del prototipo del presente trabajo, se debe a que es más rápido que cualquier otro conocido, es hasta 50 veces más rápido que RPC vía SOAP, requiere menos recursos en el lado del servidor web. Es más rápido y más fiable que la comunicación directa a través de la interfaz nativa de Java, y no requiere componentes adicionales para invocar procedimientos de Java desde PHP o procedimientos de PHP desde Java.

La elección de MySQL se debe a que es una herramienta muy utilizada en aplicaciones web además de estar muy ligada a PHP, y también porque es un software de código abierto licenciado bajo la GPL de la GNU.

La elección de phpmyadmin como herramienta de administración de MySQL se debe simplemente a que existe disponible documentación sobre ella y es de software libre.

III.2. MODELO ENTIDAD-RELACION

La arquitectura planteada debe implementar el modelo entidad-relación para su base de datos, como se muestra en la Figura 22.

De manera general, se puede decir que un estudiante para buscar información puede formar parte de distintos grupos de trabajo, en distintas materias a lo largo de su carrera. Un estudiante tiene los siguientes atributos: legajo, apellido y nombre. Cuando realiza búsquedas, a través de la aplicación puede calificar y agregar un comentario, si lo desea, a los resultados obtenidos.

Un resultado tiene los siguientes atributos: idResultado, título, url y descripción. Un resultado puede ser calificado por los integrantes de un grupo y tener asociada una calificación general, obtenida de ellos. Esto se observa en la tabla "resultadoporgrupo" de la Figura 22, donde se definen los siguientes atributos: idGrupo, idResultado y calificacionGeneral. Al definir un idGrupo, los grupos pueden calificar un mismo resultado, y de ese modo la calificación general calculada es independiente para cada grupo.

En la tabla "calificacionxestudiate" de la Figura 22, puede observarse que también se incluye un idGrupo. Esto se debe a que un integrante puede pertenecer a más de un grupo y calificar en cada uno de ellos el mismo resultado, con distinta calificación.

Además, en la Tabla 9 se definen los atributos de las distintas tablas utilizadas en el Modelo de Entidad relación, indicando el tipo de variable y una breve descripción.

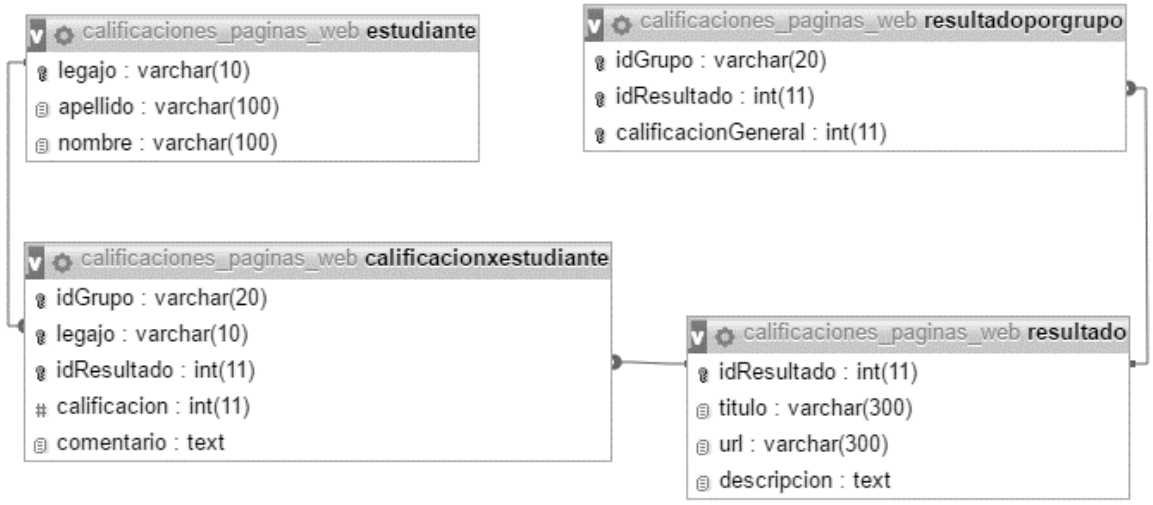


Figura 22 - Modelo de entidad-relación

Base de Datos: *calificaciones_paginas_web*		
Tabla: *estudiante*		
Atributo	**Tipo**	**Descripción**
legajo	varchar(10)	Legajo del Estudiante
apellido	varchar(100)	Apellido del Estudiante
nombre	varchar(100)	Nombre del Estudiante
Tabla: *calificacionxestudiante*		
Atributo	**Tipo**	**Descripción**
idGrupo	varchar(20)	Identificación del Grupo al que pertenece el estudiante
legajo	varchar(10)	Legajo del Estudiante que calificó un resultado
idResultado	int(11)	Identificación de un Resultado calificado por un estudiante
calificacion	int(11)	Calificación de estudiante sobre un resultado
comentario	text	Comentario de estudiante sobre un resultado

Tabla: *resultado*		
Atributo	**Tipo**	**Descripción**
idResultado	int(11)	Identificación de un resultado
url	varchar(300)	Dirección Web (URL) del resultado
titulo	varchar(300)	Título del resultado
descripcion	text	Descripción del resultado
Tabla: *resultadoporgrupo*		
Atributo	**Tipo**	**Descripción**
idGrupo	varchar(20)	Identificación del Grupo al que pertenece el estudiante
idResultado	int(11)	Identificación de un resultado
calificacionGeneral	int(11)	Calificación promedio de un resultado

Tabla 9 - Definición de las variables del Modelo Entidad-Relación

III.3. ALGORITMO DE RANKEO

El componente "Agente de Consulta" recibe, del componente "Gateway", un listado de resultados, lo procesa y lo ordena para luego devolvérselo nuevamente. Para ordenar el listado, dicho componente utiliza un algoritmo de rankeo que tiene como tarea ordenar descendentemente el listado de resultados a partir de una calificación general asociada con cada uno de ellos, calculada por el componente "Agente de Actualización", y que resulta de promediar las calificaciones individuales de cada estudiante.

El algoritmo de rankeo desarrollado en el presente trabajo utiliza un método[6] de JAVA que implementa un algoritmo de ordenamiento híbrido llamado TimSort, para ordenar el listado. En la Figura 23 puede observarse la invocación del método Collections.sort al que se le brindan dos parámetros: el listado desordenado de resultados con sus calificaciones

[6] El método Sort de la librería de JAVA java.util.Collections.sort(), implementa un algoritmo de ordenamiento modificado de merge denominado TimSort. Es actualmente utilizado por JAVA para ordenamiento de listas de objetos ya que garantiza rendimiento y estabilidad en el proceso de ordenamiento (Oracle - API Specification).

generales y un objeto de tipo Comparator que incluye la función compare a la que se le suministra dos parámetros de tipo Resultado.

```
public void ordenarListado(){
    Collections.sort(
        listado, new Comparator<Resultado>(){
            public int compare(Resultado o1, Resultado o2){
                return new Integer(o2.getCalificacionGeneral()).compareTo(new Integer(o1.getCalificacionGeneral()));
            }
    });
}
```

Figura 23 - Código del Colection.sort

El algoritmo trabaja de la forma que se explica a continuación. En primera instancia se necesita de una lista con N elementos y de un valor mínimo (min) calculado internamente por el algoritmo a partir de la cantidad de elementos de la lista (N), siguiendo una regla general igual a N/min <= 2^n, es decir que el resultado de N/min debe ser o estar próximo a una potencia de dos.

Si la lista tiene menos de 64 elementos, entonces el método utiliza un algoritmo simple de ordenamiento por inserción llamado “Insertion Sort” para ordenar la lista. Si tiene más de 64 elementos entonces procede de la manera que se explica a continuación.

El algoritmo trabaja recorriendo la lista (desde el primer elemento hasta el último) siguiendo aquellos elementos consecutivos ascendentes, o descendentes, agregándolos a sublistas. Cada sublista debe tener como mínimo “min” elementos, en el caso de encontrar un conjunto de elementos cuya cantidad de elementos sea menor a “min”, el algoritmo sigue con los próximos elementos aunque no respeten el ordenamiento hasta que la cantidad de elementos de la sublista que se está generando sea igual a “min”. La única situación en la que una sublista puede tener menos de “mín” elementos es cuando el algoritmo ha llegado al último elemento de la lista y no tiene más elementos para agregar a una sublista.

Para el caso de un ordenamiento descendente, una vez obtenidas las sublistas, el algoritmo debe ordenarlas descendentemente a cada una siguiendo las siguientes condiciones:

A) Para cualquier sublista ordenada ascendentemente, el algoritmo aplica una función de reversa que se encargará de invertir el orden de cada una.
B) Para cualquier sublista no ordenada, el algoritmo aplica el método de ordenamiento por inserción (Insertion Sort).

C) Si todas las sublistas están ordenadas descendentemente, el algoritmo aplica el método de ordenamiento por mezcla (Merge Sort) para obtener la lista completa ordenada descendentemente.

A modo de ejemplo, como se muestra en la Figura 24, se define una lista (listado) de tamaño N=20 cuyos elementos son los resultados que serán representados por sus respectivas calificaciones generales. El valor mínimo será min=4 para llevar a cabo el ejemplo.

Para este ejemplo, se hace caso omiso a Si N<64 entonces el algoritmo aplica "Insertion Sort".

El algoritmo recorre la lista de principio a fin para generar sublistas. Estas sublistas mínimamente van a tener "min" elementos, y pueden estar ordenadas de forma ascendente, descendente, o sin ordenar.

Figura 24- Ejemplo de lista de elementos

Las sublistas se van a formar a partir de los elementos ascendentes de la lista, descendentes, o sin ordenar en el caso de que el tamaño de la sublista que se está generando sea menor que "min=4", como se muestra en la Figura 25.

Figura 25 - Construcción de Sublistas

Como el objetivo del algoritmo de rankeo es ordenar la lista descendentemente, entonces las sublistas son ordenadas de esa misma forma. Si se observa detenidamente, la sublista 5 posee elementos ordenados de forma ascendente, la sublista 4 posee 5 elementos

ordenados de forma descendente, y por último las sublistas 1, 2 y 3 poseen elementos sin ordenar.

Como se muestra en la Figura 26, en el caso de la sublista 5 el algoritmo utiliza una función de reversa para ordenarlas de forma descendente. En el caso de la sublista 4, el algoritmo no efectúa cambios. Por último, en el caso de las sublistas 1,2 y 3, al no estar ordenados sus elementos aplica el método de "Insertion Sort".

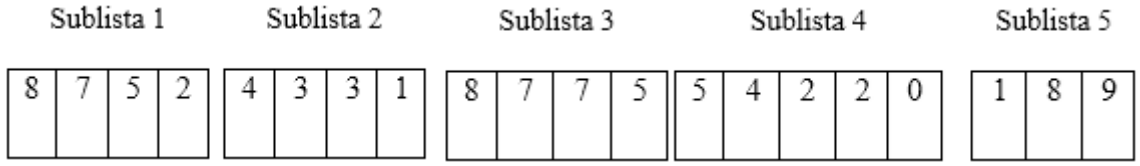


Figura 26 - Sublistas ordenadas

Una vez ordenadas descendentemente todas las sublistas, el algoritmo procede con la reconstrucción de la lista utilizando el método de ordenamiento por mezcla "Merge Sort" como se muestra en la Figura 27. Comienza por la sublista 1 y 2, luego con la resultante y la sublista 3, luego con la resultante y la sublista 4, y por último la resultante con la sublista 5.

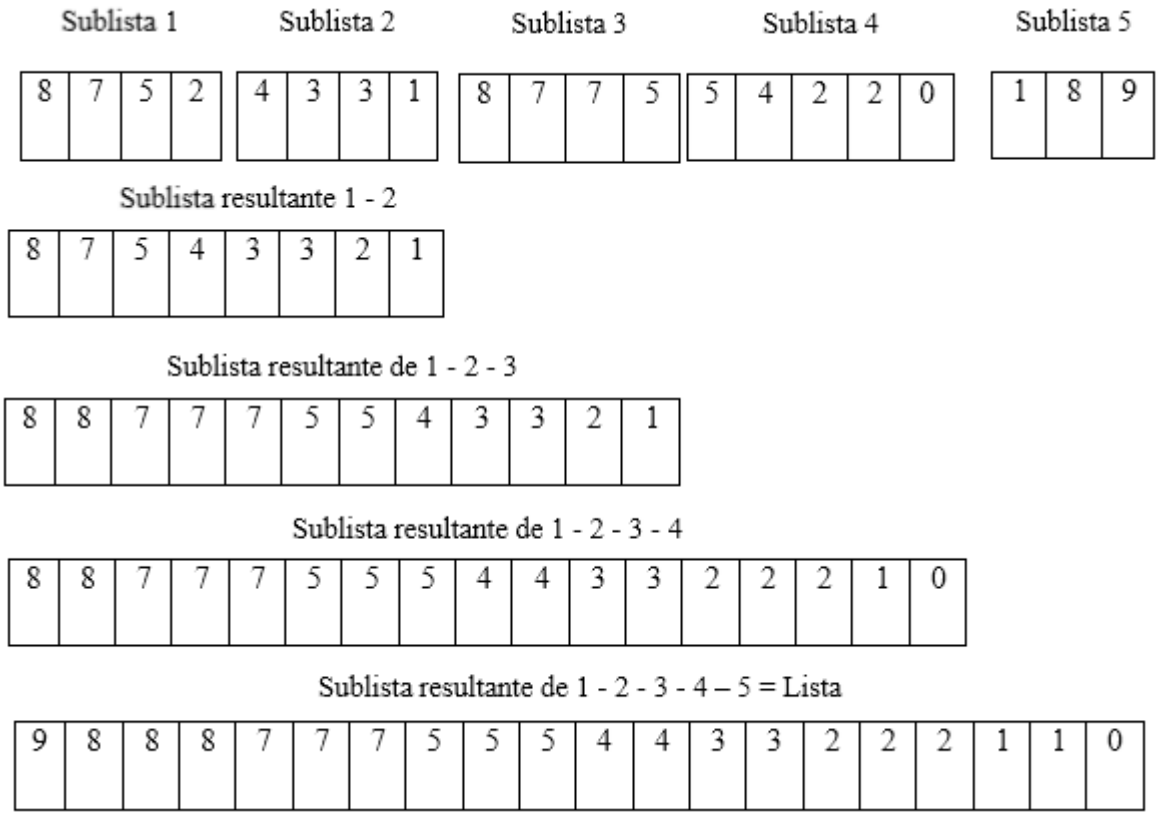


Figura 27 - Unificación ordenada de sublistas

Finalmente, se obtiene la lista ordenada de forma descendente.

CAPITULO V – DISEÑO

Para llevar a cabo el diseño del prototipo de metabuscador basado en agentes de software se emplearon dos metodologías de diseño: Metodología GAIA y Metodología OOHDM.

La construcción del prototipo del metabuscador colaborativo basado en agentes, que responde a una aplicación web, se ha llevado a cabo sobre OOHDM. La primera etapa de esta metodología trata la Obtención de Requerimientos la cual está definida en el Capítulo III. En ella se presentan los requisitos básicos que debe cumplir la aplicación y sus actores participantes, con sus roles y tareas. Dentro de estos actores se encuentran los agentes de software. Por lo tanto, fue necesario considerar previamente, la construcción de ellos para implementarlos como actores, y de ese modo completar la etapa de requerimientos. Para la construcción de estos agentes se implementó la metodología GAIA

Se ha considerado la utilización de la metodología GAIA debido a que existe variada documentación sobre ella, además de que es aplicable al número pequeño de agentes que se desarrollan en el presente trabajo. Siendo que es una metodología centrada solo en el análisis y diseño, se adecua y es suficiente para los agentes simples que se implementan. La estructura de la organización del sistema de agentes para este trabajo es estática, es decir que la relaciones entre agentes no cambian en tiempo de ejecución, y por otro lado, tanto las habilidades como los servicios que poseen y bridan estos agentes, son estáticos, es decir que no cambian en tiempo de ejecución.

También se ha considerado la utilización de la metodología OOHDM ya que es una de las metodologías con mayor aceptación en cuanto al desarrollo de aplicaciones multimedia. Además, proporciona varias ideas que son asumidas en otros desarrollos, para los cuales se han obtenido buenos resultados. OOHDM hace una separación clara entre lo conceptual, lo navegacional y lo visual. Además, hace uso de la orientación a objetos y de un diagrama de clases para representar el aspecto de la navegación a través de las clases navegacionales.

IV.1. METODOLOGIA GAIA

Al emplear esta metodología se obtienen cinco modelos principales como resultado de las siguientes fases: Modelos de Roles y Modelo de Interacción de la fase de análisis, y Modelo de Agentes, Modelo de Servicio y Modelo de Conocidos de la fase de Diseño.

IV.1.1. Fase de Análisis

Los roles del AC y del AA han sido delimitados a partir de la "Identificación de Roles y Tareas de Actores" definidos en la Especificación de Requerimientos (Sección III.3.1. del Capítulo III). El rol del Agente Gateway ha sido delimitado a partir de la "Arquitectura del Metabuscador" (Sección III.1. del Capítulo IV).

A continuación en Tablas 10, 11 y 12 se describen los tres roles, indicando en cada caso una "D*escripción"*, en donde se explica brevemente una serie de tareas que lo identifican; los "*Protocolos y Actividades*" que definen las formas en que el agente (asumiendo dicho rol) interactúa con otro agente, y representan las acciones privadas que el agente realiza sin necesidad de interactuar con otro agente, respectivamente; los "P*ermisos*" que identifican los recursos que el rol tiene disponible para realizar sus responsabilidades; y por último las "R*esponsabilidades*" que determinan la funcionalidad del rol.

Modelo de Roles

Rol: Consulta	
Descripción	Trabaja con la Base de Datos. Cuando un usuario realiza una consulta, éste recupera los datos asociados de aquellos resultados que fueron anteriormente calificados y/o comentados. Cuando el usuario solicita un historial, éste recupera los resultados que fueron anteriormente calificados y/o comentados por el grupo del usuario y los datos asociados a cada uno de ellos.
Protocolos y actividades	GeneraNuevoListado, GeneraHistorial,OrdenaListado, DevuelveListadoOrdenado, DevuelveHistorialOrdenado
Permisos	Lee base de datos de calificaciones de resultados, lee la estructura de datos conocida.
Responsabilidades	Liveness: Consulta: (RecibeConsulta|GeneraHistorial) $^{\omega}$ RecibeConsulta: GeneraNuevoListado. OrdenaListado.DevuelveListadoOrdenado GeneraHistorial: GeneraHistorial. OrdenaListado. DevuelveHistorialOrdenado Safety: Una conexión completa con la base de dato de calificaciones de resultados es establecida.

Tabla 10 - Modelo de Roles "Rol Consulta"

Rol: Actualización	
Descripción	Trabaja con la Base de Datos. Cuando un usuario realiza una calificación y/o un nuevo comentario, éste actualiza la Base de Datos de calificación de resultados con los nuevos datos enviados por el usuario.
Protocolos y actividades	ActualizaBD, InformaNuevaActualización
Permisos	Lee y actualiza la Base de Datos de calificaciones de resultados, lee la estructura de datos conocida.
Responsabilidades	Liveness: Actualizacion: (RecibeActualizacion) $^{\omega}$ RecibeActualizacion:ActualizaBD. InformaNuevaActualización Safety: Una conexión completa con la base de dato de calificaciones de resultados es establecida.

Tabla 11 - Modelo de Roles "Rol Actualización"

Rol: Gateway	
Descripción	Determina para qué agente es el mensaje y el contenido del mismo, y luego lo envía al agente correspondiente.
Protocolos y actividades	EmpaquetaContenido, EnviaPaquete, RecibePaquete, DesempaquetaContenido
Permisos	Crea y lee las estructuras de datos conocida.
Responsabilidades	Liveness: GateWay: (EnvioMensaje||RecepciónMensaje) $^{\omega}$ EnvioMensaje: EmpaquetaContenido. EnviaPaquete RecepcionMensaje: RecibePaquete .DesempaquetaContenido Safety: True

Tabla 12 - Modelo de Roles "Rol Gateway"

Una vez definido el Modelo de Roles, Gaia propone el Modelo de Interacción, el cual representa la descripción de las comunicaciones entre agentes, definiendo qué acción retorna de una solicitud e indicando qué rol la inicia y cuáles responden. La Tabla 13 que se muestra a continuación, contiene la información necesaria para dicho modelo, en la que se indica el

"*Propósito*", que es una descripción textual de la interacción; el "*Iniciador*", que indica el/los rol/es responsable/s de iniciar la interacción; el "*Receptor*", que indica el/los rol/es con los cuales el iniciador interactúa; la "*Acción de retorno*", que define qué acción retorna de una solicitud; y por último el "*Proceso*", que es una descripción textual del proceso que realiza el agente que inicia el protocolo.

Modelo de Interacción

Protocolo	EnviaPaquete	InformaNueva Actualización	DevuleveListado Ordenado	DevuelveHistorial Ordenado
Iniciador	GateWay	Actualizacion	Consulta	Consulta
Receptor	Actualizacion o Consulta	GateWay	GateWay	GateWay
Acción de respuesta	RecibePaquete	-	-	-
Proceso	Crea y envía un mensaje de acuerdo al agente destinatario.	Informa al GateWay que la actualización de la base de datos se ha realizado	Envía un listado ordenado de los resultados que ya fueron previamente calificados y/o comentados, almacenados en la base de datos de calificaciones de resultados.	Envía un historial ordenado con todos los resultados que fueron calificados y/o comentados por todo el grupo, almacenados en la base de datos de calificaciones de resultados.

Tabla 13 - Modelo de Interacción

IV.1.2. Fase de Diseño

Esta fase implicó la creación del modelo de agente que muestra la Figura 28. Allí puede observarse los diferentes tipos de agentes que fueron usados en el desarrollo del Metabuscador así como el número de instancias de cada agente que funcionarán en la aplicación. En el modelo se puede ver que los tres agentes tienen una única instancia en el Metabuscador, en el que la flecha indica el rol desempeñado por cada uno de ellos.

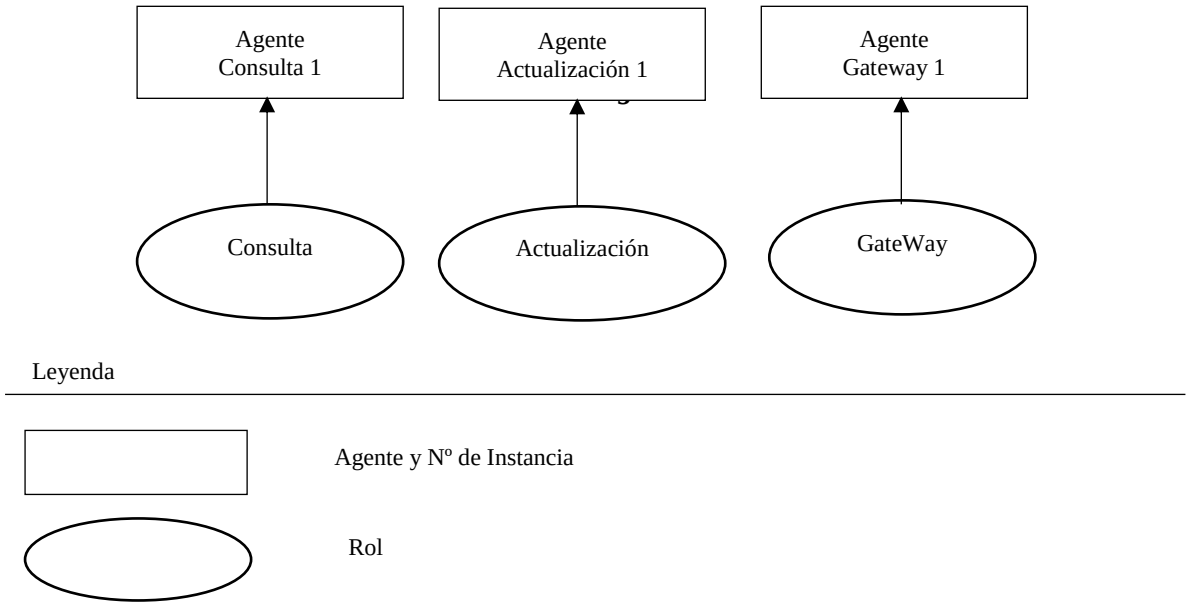


Figura 28 - Modelo de Agente

También se creó el modelo de servicio documentado en Tabla 14. En este modelo fue necesario incluir las "*Entradas*" y las "*Salidas*", que serán derivadas en una forma obvia del modelo de protocolos; y las "*Pre-Condiciones" y* "*Post-Condiciones*", que representan restricciones en los servicios.

Modelo de Servicio

Servicio	*Consulta*	*Actualización*	*GateWay*
Entradas	Listado de resultados. Pedido de historial.	Resultado con su calificación, con o sin comentario	Nombre de agente. Listado de resultados.
			Nombre de agente. Pedido de historial.
			Nombre de agente. Resultado con su calificación, con o sin comentario
Salidas	Listado de resultados ordenado con las correspondientes calificaciones y/o los comentarios. Historial de todos los resultados calificados y/o ordenados con sus respectivas calificaciones y/o los comentarios	Mensaje que indica que la actualización se ha realizado.	Listado de resultados ordenado con las correspondientes calificaciones y/o los comentarios.
			Historial de todos los resultados calificados y/o ordenados con sus respectivas calificaciones y/o los comentarios.

			Mensaje que indica que la actualización se ha realizado.
Pre-Condición	---	---	El usuario debe estar registrado en el sistema.
Post-Condición	---	---	El usuario puede calificar un resultado y/o comentarlo. Además puede de modificar o eliminar dicha calificación.

Tabla 14 - Modelo de Servicio

Por último, el tercer modelo es el de conocidos (Figura 29). En este último modelo puede observarse los vínculos de comunicación que existen entre los tipos de agentes. Ellos no definen qué mensajes son enviados o cuándo los mensajes son enviados, ellos simplemente indican que existe un camino de comunicación. En la Figura 29, pueden identificarse los distintos tipos de agentes, y sus correspondientes arcos.

Modelo de Conocidos

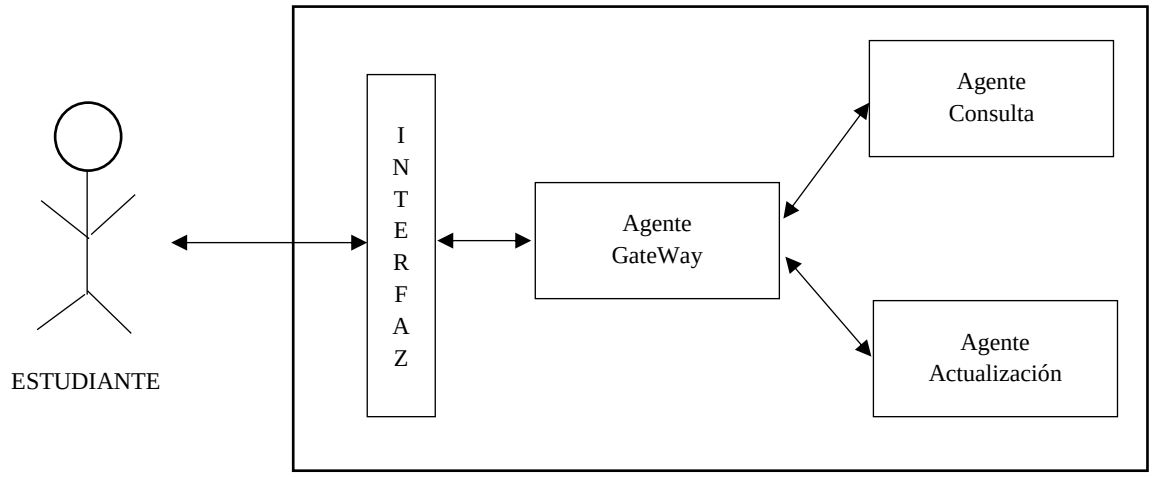


Figura 29 - Modelo de Conocidos

Aclaraciones

- Interfaz: Es una interfaz implementada en lenguaje de programación PHP, que corresponde a un componente del metabuscador, que permite la relación con los agentes.

Una vez obtenido los cinco modelos sólo queda llevar los agentes a un lenguaje concreto de programación. Esta actividad se documentará en el Capítulo VI Codificación y Prueba.

IV.2. METODOLOGIA OOHDM

Dentro de la metodología OOHDM se definen tres etapas para el diseño de una aplicación: Diseño Conceptual, Diseño Navegacional, y Diseño de Interfaz Abstracta. Estas etapas se realizan como un proceso iterativo e incremental.

IV.2.1. Diseño Conceptual

En esta fase se construye el modelo de dominio de la aplicación usando los conocimientos del modelado orientado a objetos, con notación similar a UML. El producto de esta fase es el Esquema Conceptual también llamado Modelo Conceptual, que muestra las clases y las relaciones del dominio.

Descripción de las clases representativas:

- *metabuscador:* Es la clase que en sí viene a representar la aplicación hipermedia.
- *ayuda*: Clase que representa el modo de utilización de la hipermedia
- *contacto*: Clase que representa la posibilidad de contactarse con los responsables de la aplicación hipermedia.
- *listado*: Clase que representa las búsquedas de frases o palabra de búsquedas que serán posibles de realizar en la aplicación hipermedia.
- *historial*: Clase que representa el historial de resultados de búsquedas calificados y/o comentados.
- *resultado*: Clase que representa al subconjunto de resultados de búsqueda.
- *estudiante:* Clase que representa la calificación y/o comentario de cada resultado.

En la Figura 30 se muestra el Esquema Conceptual de la aplicación. Se hace uso de la abstracción de UML para indicar de manera más clara las relaciones entre cada una de las clases del domino.

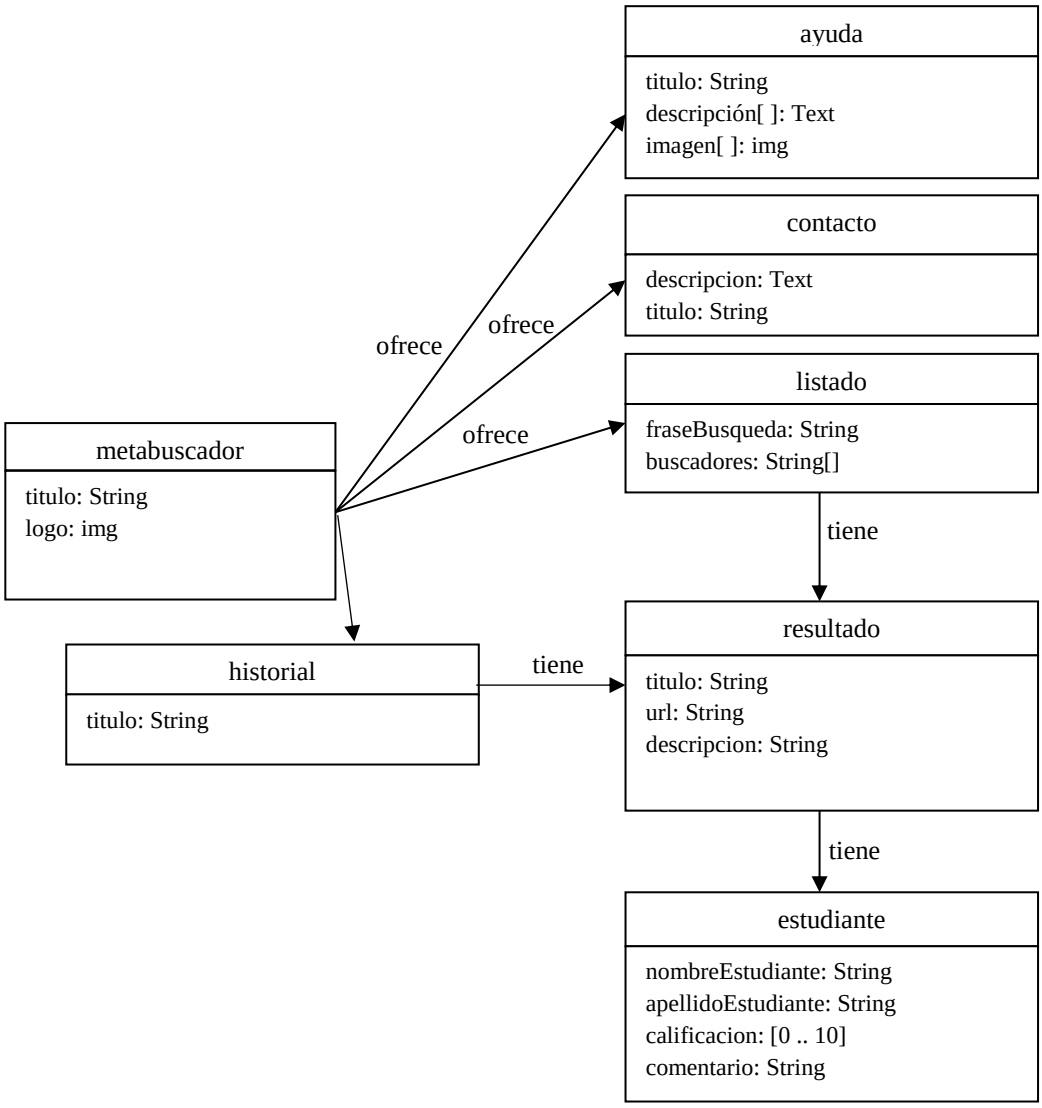


Figura 30 - Esquema Conceptual

IV.2.2. Diseño Navegacional

En esta etapa de la metodología OOHDM, una aplicación debe abordarse como una "Vista de Navegación" sobre el Modelo Conceptual de la etapa anterior.

Una vez obtenido el Modelo Conceptual es necesario enfocarse en los objetos y en las relaciones, es por ello que aparece el concepto de Esquema de Clases de Navegación donde se determinan los objetos de navegación que un usuario visualizará. También aparece el concepto de Esquema de Contexto de Navegación vinculado con las vistas o contextos creados a partir de las diferentes necesidades de la aplicación, por ejemplo una vista podría ser mostrar los resultados a partir de una búsqueda.

Al finalizar esta etapa se obtiene el Esquema de Navegación construido sobre los nodos y enlaces obtenidos en los esquemas de clases y de contexto de navegación.

En la Figura 31 se presenta un Esquema de Clases de Navegación, en el que se puede ver mayor detalle de la estructura de navegación de la aplicación. Se muestran los enlaces de navegación, los nodos que unen estos enlaces y además los atributos de los nodos, que brindan una idea de la información que se presentará en la interfaz del usuario. La información sigue presentándose de una manera abstracta, sin determinar cómo será la interfaz con respecto a su diseño, ni qué herramientas se utilizarán. Se debe destacar que esta abstracción permite que el impacto producido por algún cambio en el diseño sea mínimo o no afecte a la estructura de navegación de la aplicación. Si se debe realizar un ajuste, este cambio se podrá hacer con facilidad aislando la parte que se verá afectada de las demás que no se ven afectadas. Los enlaces determinan las relaciones para lograr la navegación. Orientan sobre el origen y el destino de la navegación, así como la cardinalidad, la cual da más información de la relación entre las clases de nodo que une.

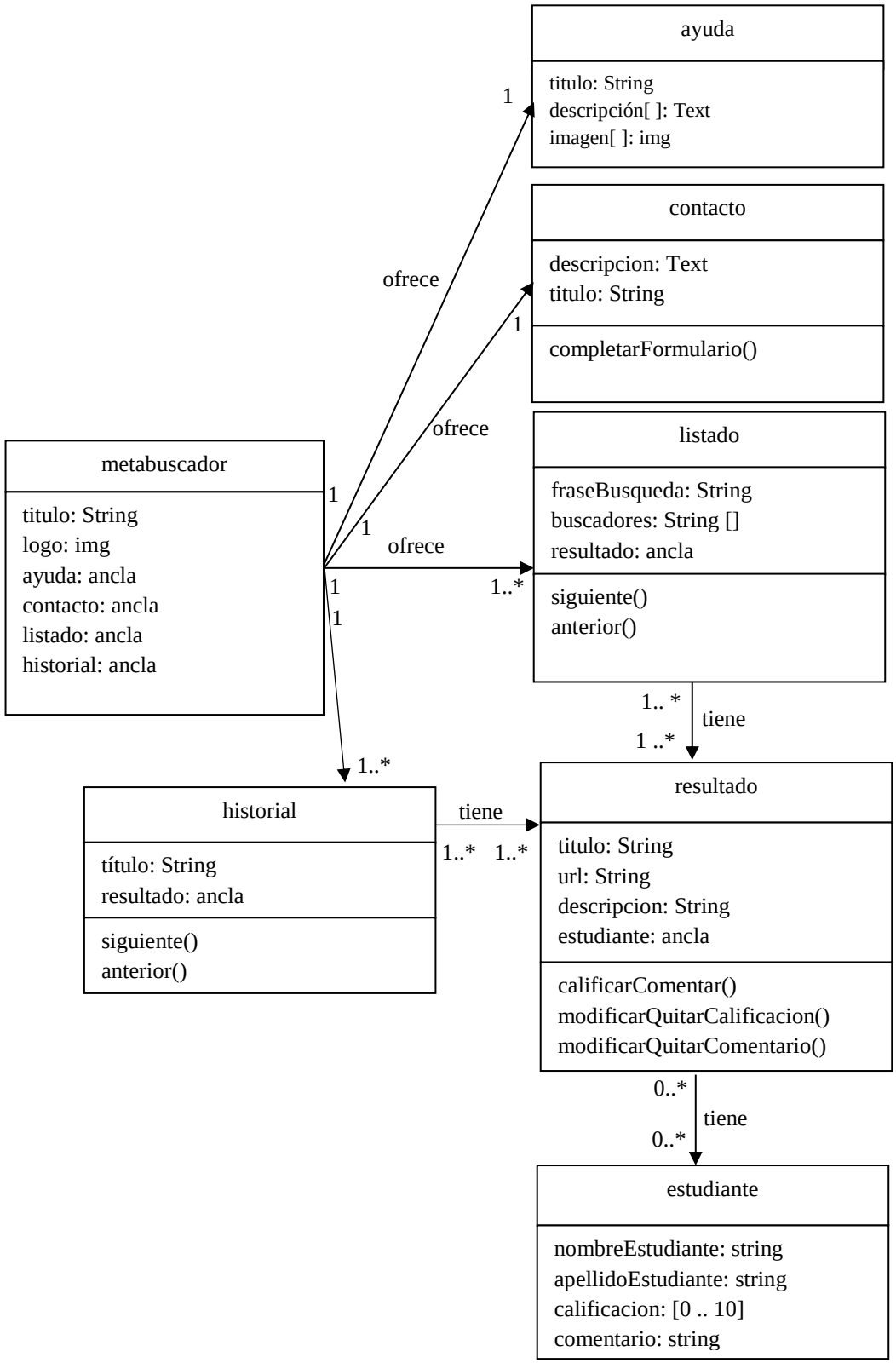


Figura 31 - Esquema de Clases de Navegación

IV.2.3. Esquema de Contexto de Navegación

En la Figura 32, se muestra el Esquema de Contexto de Navegación para el usuario estudiante. En el diagrama se puede observar la representación de la información según la nomenclatura detallada en la Sección II.4.2 OOHDM. Cada contexto recorre un nodo del Esquema de Navegación. Así por ejemplo el contexto porAyuda recorre el nodo Ayuda y muestra información respecto al modo de uso del Metabuscador. El índice porHistorial, del nodo Historial, nos indica que el historial estará representado por una estructura de acceso a todos los resultados calificados por un grupo de estudiantes. De manera similar el índice porBúsqueda, indica que tendremos una estructura de acceso desde la interfaz a un listado de resultados de búsqueda del nodo ListadoXBúsqueda.

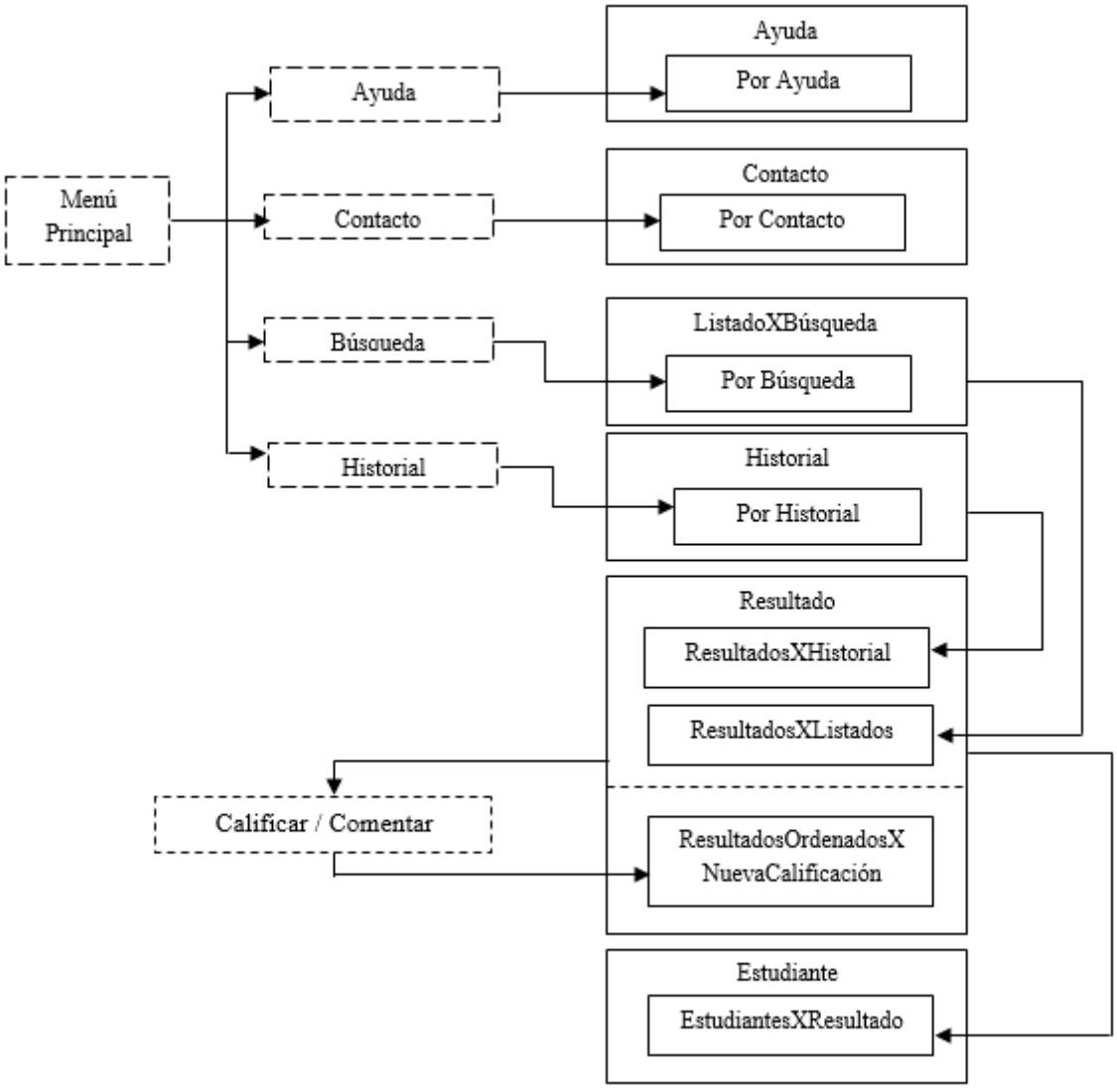


Figura 32 - Esquema de Contexto de Navegación

Aclaración: Contacto, ayuda y Calificar/Comentar son de tipo “visita guiada”. Búsqueda e Historial son del tipo “índice”.

IV.2.4. Diseño de Interfaz Abstracta

En este diseño se especifica qué objetos de interfaz podrá percibir el usuario. Este diseño es útil para plantear interfaces a niveles abstractos independiente del entorno de desarrollo.

Los ADV se determinan por la estructura de nodos, es decir debería definirse un ADV para cada nodo y para sus atributos que tengan representatividad en el diseño (como anclas). Siguiendo con esta definición en la Figura 33 (tanto parte A como parte B) se muestra el Esquema de ADVs para los nodos más representativos de la aplicación.

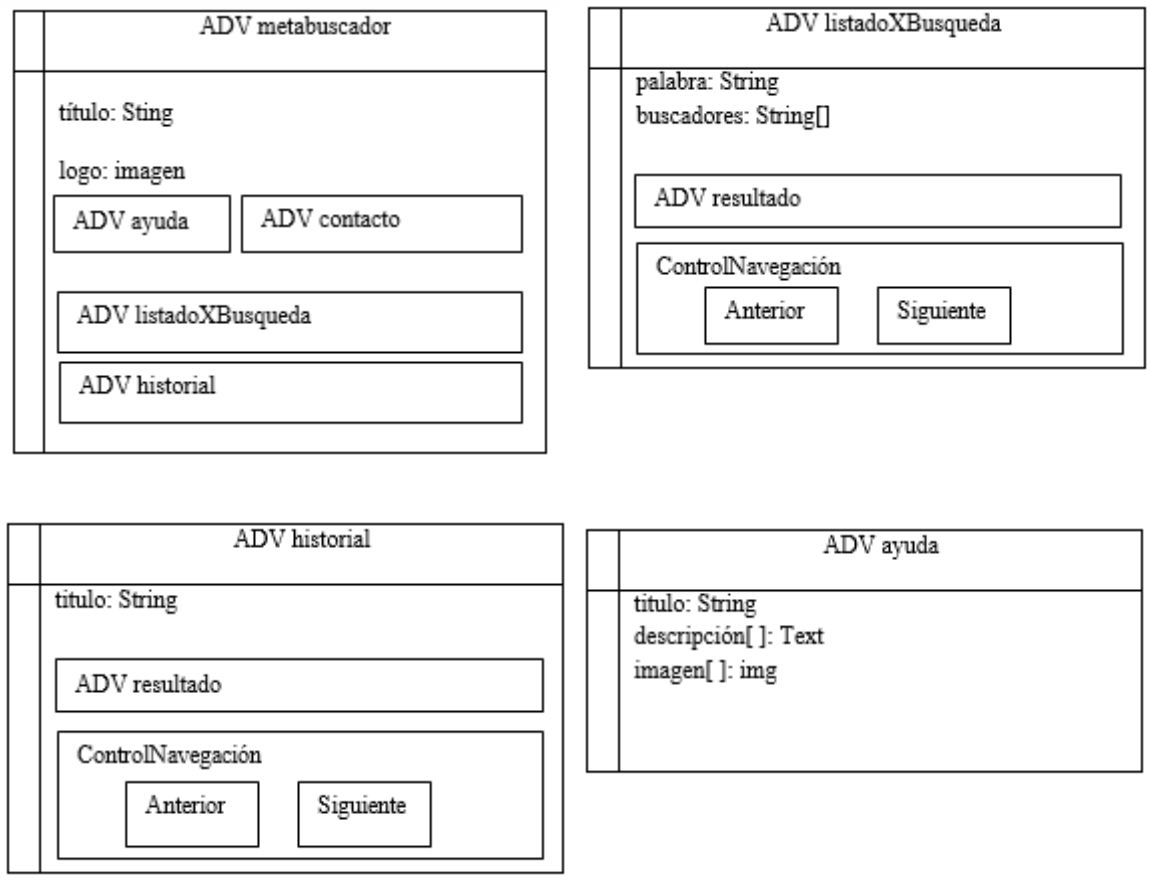


Figura 33 - Diseño de Interfaz Abstracta - parte A

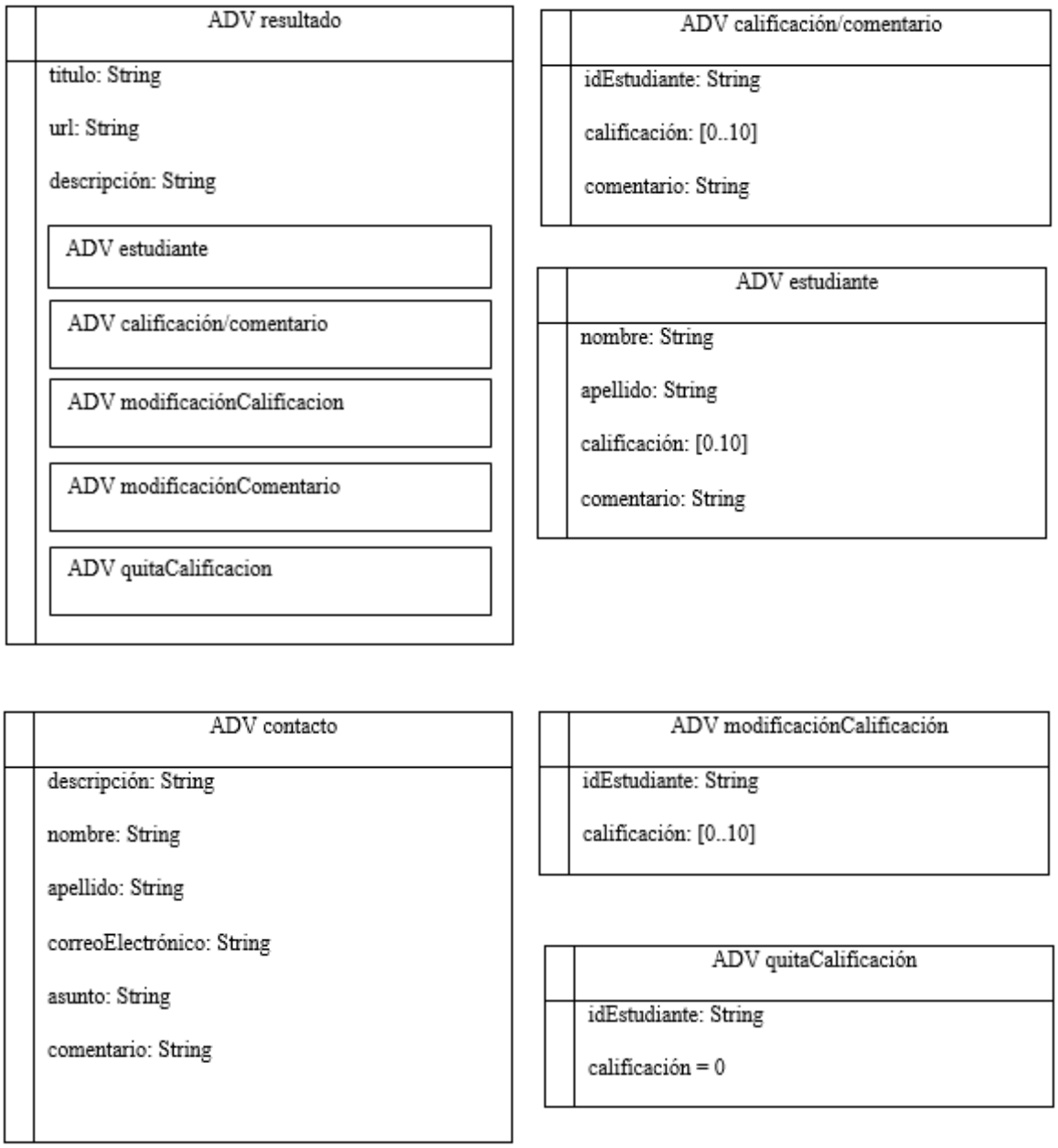


Figura 34 - Diseño de Interfaz Abstracta – parte B

Como puede observarse en el ADV Historial y ADV listadoXBúsqueda de la Figura 33, parte A, se define un ADV ControlNavegación que provee los controles de navegación dentro del contexto navegacional: resultados por listado.

Se explican a continuación los ADV más representativos del metabuscador.

El ADV Metabuscador muestra la estructura general del metabuscador. Grafica los objetos de interfaz que aparecerán en la vista principal de la aplicación. La correspondencia entre el ADV Metabuscador con el Nodo metabuscador puede verse en la Figura 35.

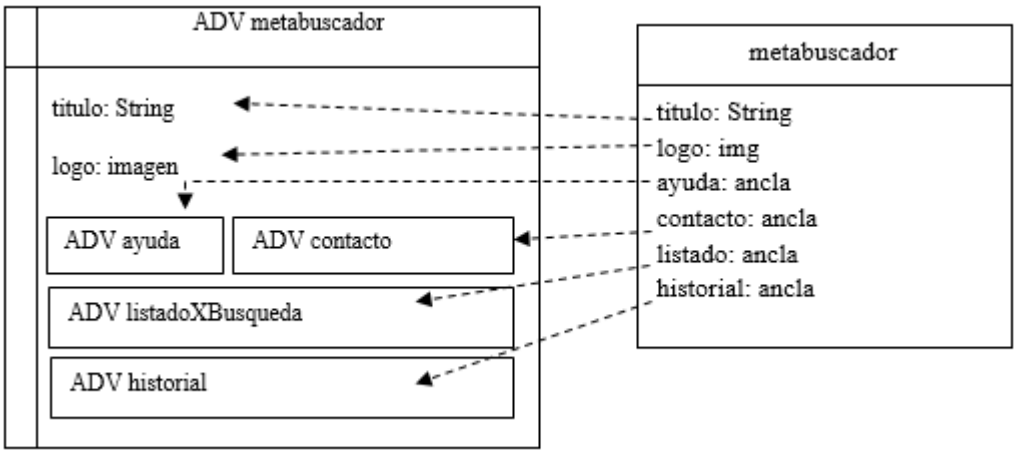


Figura 35 - Correspondencia ADV metabuscador vs Nodo metabuscador

El ADV ListadoXBúsqueda muestra el objeto de interfaz resultado que se presentará en la vista. En la Figura 36, se puede observar la correspondencia del ADV ListadoXBúsqueda con el nodo listado.

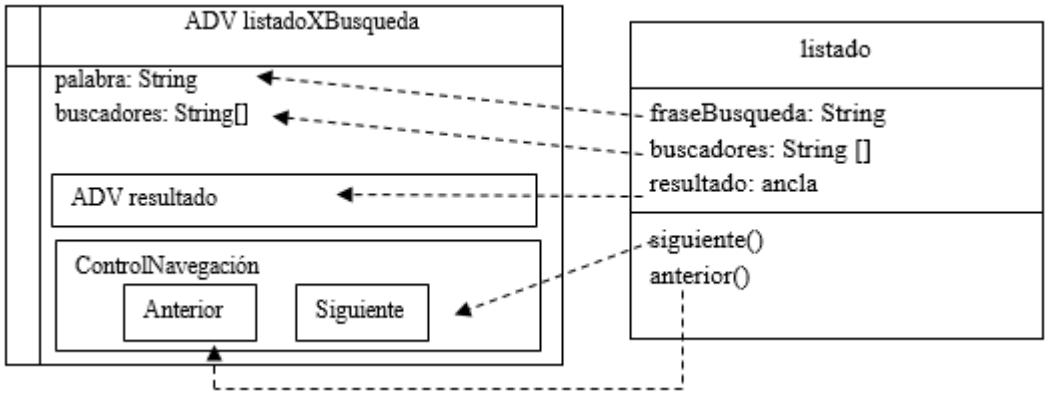


Figura 36 - Correspondencia ADV metabuscador vs Nodo metabuscador

Una vez obtenido el esquema conceptual, el esquema de navegación y el esquema de interfaz abstracta, sólo queda llevar los objetos a un lenguaje concreto de programación. Esta actividad se llevará a cabo en el capítulo de Codificación.

CAPITULO VI – CODIFICACIÓN & PRUEBA

Para llevar a cabo la codificación del Metabuscador y sus componentes se tuvieron en cuenta varios aspectos propios de la programación. A continuación se explica con más detalle la Arquitectura del Metabuscador detallada en el Capítulo IV.

Básicamente, la Arquitectura del Metabuscador se divide en dos partes diferenciadas por el lenguaje de programación por el que fueron concebidas. Esos lenguajes de programación son: PHP y JAVA, por lo tanto se ha considerado que todo lo desarrollado en PHP se ejecute en un servicio de Servidor Apache 2 y lo desarrollado en JAVA se ejecute en un servicio de Servidor Apache Tomcat 8. Ambos servicios se implementan dentro de un mismo equipo físico.

La **Parte Superior de la Arquitectura** (Parte PHP) está compuesta por los componentes: "*Interfaz Usuario*", "*Procesamiento de Datos*", "*Despachador de Consulta*", "*Fusionador de Resultados*" y "*Comunicación Agentes*", como se muestra en la Figura 37.

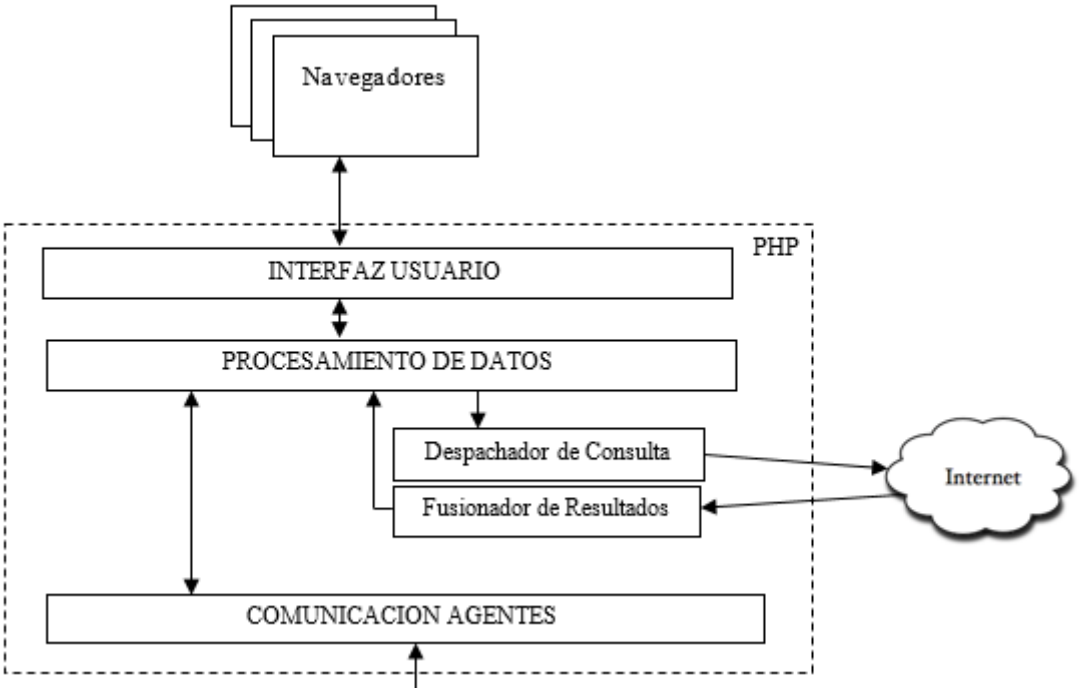


Figura 37 - Parte superior de la Arquitectura del Metabuscador - Parte PHP

En el componente "***Interfaz Usuario***" se ha utilizado una combinación entre el Lenguaje de Etiquetas HTML y el Lenguaje de Programación PHP, para generar una interfaz gráfica intuitiva y así facilitar la interacción del usuario con el metabuscador.

Mediante el uso de formularios implementados en HTML, se capturan los datos que serán entregados al componente "*Procesamiento de Datos*, para que luego a partir de ellos, el mismo componente devuelva los resultados que se mostrarán al usuario a través de la interfaz.

La Figura 38 muestra un formulario para realizar una búsqueda compuesto por: un campo de texto para ingresar la frase de búsqueda, por los checkbox para tildar los buscadores a utilizar, el botón "Buscar" que será el disparador de la búsqueda y por último un botón "Ver Resultados Calificados" con el que el usuario podrá acceder al "Historial de Resultados Calificados".

Figura 38 - Interfaz Gráfica – Formulario de Búsqueda

La Figura 39, a modo de ejemplo, muestra parte del listado resultados devueltos por el metabuscador para la frase de búsqueda "Agentes Inteligentes", como también los motores de búsqueda utilizados: GOOGLE y ASK. Además, de todos los resultados desplegados, se observan aquellos resultados calificados con una valoración del 1 al 10 representada por estrellas y el/los usuarios que los calificaron.

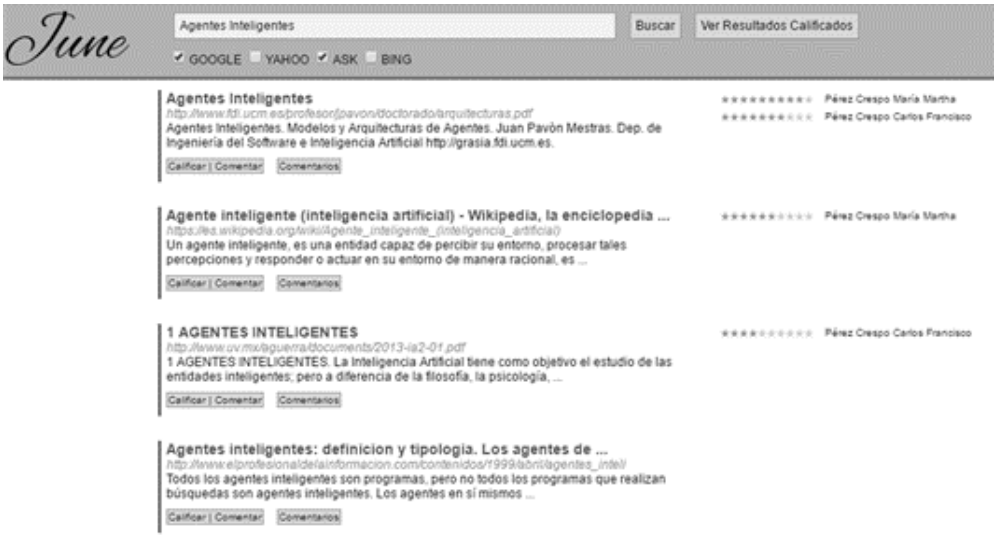


Figura 39 - Interfaz Gráfica – Resultados de Búsqueda

Por último, la interfaz gráfica también incluye una sección de ayuda compuesta de imágenes y texto para guiar al usuario en el uso de la aplicación web, y un formulario de contacto para realizar algún tipo de consulta sobre el metabuscador. El formulario de contacto está compuesto por los campos de texto: nombre, apellido, correo electrónico, asunto y comentario, acompañado de un botón "Enviar" para enviar el comentario y un botón "Cerrar" para salir del formulario. Los datos de los campos de texto son enviados a una cuenta de correo asociada al metabuscador. La asociación se establece mediante un archivo de configuración llamado "enviarcorreo.php", ubicado en el directorio donde está instalado el metabuscador, modificando los valores de siguientes parámetros en dicho archivo: dirección ssl de correo ($mail->Host = 'ssl://smtp.gmail.com';), puerto ($mail->Port = 465;), autenticación smtp ($mail->SMTPAuth = true;), nombre de usuario (($mail->Username = 'junemetabuscador@gmail.com';) y contraseña ($mail->Password = 'junemeta';). Los valores para cada parámetro son provistos por el servicio de correo electrónico.

El componente "***Procesamiento de Datos***" recibe los datos del componente "*Interfaz Usuario*", a través del método $_POST, para procesarlos y así obtener una estructura de datos con los resultados de búsqueda que será entregada al mencionado componente. Los datos recibidos pueden ser por ejemplo la frase de búsqueda, los buscadores seleccionados, la identificación del usuario y su grupo, el título de un resultado, la url del resultado, el comentario, entre otros, y la estructura de datos entregada es un objeto que incluye los resultados, el índice de los resultados, y la cantidad de páginas con resultados. El método $_POST es utilizado para recuperar los datos de los formularios implementados en HTML.

También recibe por parámetro dos objetos: el primero de ellos es el objeto $comunicacionAgentes (vinculación con el componente "*Comunicación Agentes*") a través del cual se van a realizar todas las comunicaciones y operaciones con los agentes de software; el segundo de ellos es el objeto $metabuscador (vinculación con los componentes "*Despachador de Consulta*" y "*Fusionador de Resultados*") a través del cual se va a realizar las operaciones necesarias para obtener los resultados a partir de una frase de búsqueda.

A partir de los datos recibidos, el componente decide como procesarlos para obtener una estructura de datos. La decisión en cuestión ha sido desarrollada como una función cuya parte del código se puede observar en la Figura 40.

En ella se muestra una primera condición “if(strlen($_POST['senialDeUsuario']) > 2)” que indica si el usuario a calificado un resultado, ha modificado su calificación y/o comentario, o simplemente ha eliminado su calificación. De ser así se procede con el “Procesamiento de Actualización” en el que interviene el AA a través de la función “$comunicacionAgentes->agenteActualizacion($paquetePHP)”. En el código se puede observar un arreglo $paquetePHP cuyos elementos serán el título, la url, y la descripción de un resultado, además del comentario, calificación ($_POST[‘rb’]), la identificación y el grupo del usuario. Dicho arreglo se prepara y se envía al AA con la función $comunicaciónAgentes->agenteActualizacion ($paquetePHP). Por último, se espera la señal AA que indica la finalización del proceso de actualización.

Con las dos siguientes condiciones: “if(isset($_POST['btnCalificados']))” y “if(isset($_POST['btnBuscar']))”, se verifica si el usuario quiere obtener los resultados que han sido calificados por el grupo como “Historial de Resultados Calificados” o si el usuario quiere realizar una búsqueda para obtener un “Listado de Resultados”, respectivamente. Y por último con la condición: if($_POST['subPaginas'] == '' || (strlen($_POST['btnIndice']) <= 2 && $_SESSION['resultadosCalificados'] == 0) || strlen($_POST['btnCalificados']) >= 2), y todas las condiciones internas a ella, se definen todos los caminos que puede tomar el metabuscador, y de ellos los más relevantes son:

- Realizar una búsqueda a través del “Despachador de Consulta” y de la “Fusión de Resultados” para obtener los resultados con la función “$metabuscador->despachadorYfusion(str_replace("","+",$_POST['keywords']),intval($_POST ['btnIndice']))”, para luego pasarlos al AC dos a través de la función “$comunicacionAgentes->agenteConsulta($this->getResultadosCrudos(), $_SESSION['grupoUsuario'])”.

- Obtener el “Historial de Resultados Calificados” pasasando al AC un arreglo vacío través de la función “$comunicacionAgentes->agenteConsulta($this-> getResultadosCrudos(), $_SESSION['grupoUsuario'])”.

- Al momento de que un usuario califique un resultado con o sin comentario, o lo modifique o lo elimine, y luego del “Procesamiento de Actualización”, el componente recupera los resultados obtenidos de la consulta con la función “$this->setResultadosCrudos($_SESSION['resultadosCrudos'])” y se lo pasa al AC a través

de la función "$comunicacionAgentes->agenteConsulta($this->getResultados Crudos(),$_SESSION['grupoUsuario'])" para q devuelva nuevamente los resultados pero con la actualización indicada por el usuario.

Independientemente, de si se realiza una consulta para obtener los resultados de búsqueda o si desea obtener el "Historial de Resultados Calificados", existe una función "$this->obtenerListados($comunicacionAgentes->getListadoAgenteConsulta(),$pag, $pagSig)" que se encarga de tomar el listado obtenido y procesarlo para generar listados más pequeños de a lo sumo 10 resultados, para generar un índice como referencia que permita elegir un listado de entre los generados, y por último para determinar la cantidad de listados generados.

```
function busquedaResultados($comunicacionAgentes,$metabuscador){

    $estado = "";

    if(strlen($_POST['senialDeUsuario']) > 2 ){

        $paquetePHP = array(utf8_encode($_POST['tituloResultado']), utf8_encode($_POST['urlResultado'])
            ,utf8_encode($_POST['descripcionResultado']),$_SESSION['idUsuario'],$_POST['rb'],
            utf8_encode($_POST['comentario']),$_SESSION['grupoUsuario']);
        $comunicacionAgentes->agenteActualizacion($paquetePHP);
        $estado = $comunicacionAgentes->getEstadoAgenteActualizacion();
        unset($paquetePHP);
    }

    if(isset($_POST['btnCalificados']) ) $_SESSION['resultadosCalificados']= 1;
    if(isset($_POST['btnBuscar']))  $_SESSION['resultadosCalificados']= 0;

    if($_POST['subPaginas'] == '' || (strlen($_POST['btnIndice']) <= 2 && $_SESSION['
        resultadosCalificados'] == 0) || strlen($_POST['btnCalificados']) >= 2 ){

        $pag = intval($_POST['btnIndice']);

        if($_SESSION['resultadosCalificados'] == 1){
            $this->setResultadosCrudos($this->resultadoVacio());
            $pagSig = $pag;
        }else{
            $metabuscador->despachadorYfusion(str_replace(" ","+",$_POST['keywords']),intval($_POST['
                btnIndice']));
            $this->setResultadosCrudos($metabuscador->getResultadoMetabuscador());
            $pagSig = $pag + 1;
        }

        $_SESSION['resultadosCrudos'] = $this->getResultadosCrudos();
        $comunicacionAgentes->agenteConsulta($this->getResultadosCrudos(),$_SESSION['grupoUsuario']);

        $this->obtenerListados($comunicacionAgentes->getListadoAgenteConsulta(),$pag,$pagSig);
        $this->setSubpagina(0);
```

Figura 40 - Parte del código del componente "Procesamiento de Datos"

Para el componente "***Despachador de Consulta***" se ha utilizado una técnica llamada web scraping que permite obtener los resultados de cada motor de búsqueda, y a su vez se debió considerar el tiempo que le puede tomar al metabuscador procesar la estructura de datos, llamada DOM devuelta por cada motor de búsqueda al hacer la consulta aplicando

web scraping. Por ello, se trabajó la consulta realizada por el metabuscador a cada motor de búsqueda, de forma independiente, generando un hilo de ejecución para cada motor de búsqueda, es decir generando la simultaneidad de una consulta para cada motor de búsqueda.

En la Figura 41 se muestra parte de código fuente que involucra al Componente "*Despachador de Consulta*", y en ella se puede ver como se genera un objeto por cada motor de búsqueda para una consulta determinada ($consulta), y a su vez se van guardando en un arreglo de objetos llamado $resultadoBuscador[].

Cuando se invoca a la función despachadorConsulta junto con el parámetro $resultadoBuscador, cada objeto de $resultadoBuscador realiza la consulta, recibe la estructura de datos del motor de búsqueda correspondiente y la procesa aplicando la técnica del "Web Scraping" para generar una nueva estructura de datos que contenga los resultados de búsqueda.

```
function despachadorYfusion($consulta,$pagina){

    $direccion_simple_html_dom = "".$this->getDirectorios(2)."/simple_html_dom.php";
    $resultadoBuscador = array();
    for($i=0; $i < $this->CantBuscadoresSeleccionados(); $i++){

        $direccion_buscador = "".$this->getDirectorios(2)."/".$this->getBuscadoresSeleccionados($i)."/mod_init.php";

        $resultadoBuscador[] = new buscador($direccion_simple_html_dom,
                                            $direccion_buscador,
                                            $consulta,
                                            $this->getBuscadoresSeleccionados($i),
                                            $pagina);
    }

    despachadorConsulta($resultadoBuscador);
    esperaFinalizacionConsulta($resultadoBuscador);
    destruirHilosConsulta($resultadoBuscador);
```

Figura 41 - Parte del Código Fuente del Componente Despachador

En la Figura 42 se puede visualizar la función google que utiliza la técnica de web scraping para el buscador google.

```
function google($frase,$buscador,$inicioPagina){

    $numInicial = (($inicioPagina-1)*100)/10;
    $numFinal = $numInicial;
    $google_html = new simple_html_dom();
    $google_html->load_file("https://www.google.com.ar/search?q=".$frase."&biw=1100&bih=767&start=".$numInicial."&sa=N");

    $google_items = $google_html->find('div[id=search]');
    $google_resultados = array();
    $google_ij = 0;

    foreach($google_items as $items_body){
        foreach($items_body->find('div[class=g]') as $items_g){
            foreach($items_g->find('h3[class=r]') as $items_r){

                $titulo=strip_tags($items_r->children(0));
                $url = getUrl($items_r->children(0)->href);

                if($url != "NoGoogleUrl" ){

                    foreach($items_g->find('span[class]') as $items_st){
                        $descripcion = utf8_encode(strip_tags($items_st));

                    }

                    $google_resultados[$google_ij] = array($buscador,
                                                           utf8_encode($titulo),
                                                           $url,
                                                           $descripcion);
                    $google_ij  = $google_ij + 1;
                }

            }
        }
    }

return $google_resultados;

}
```

Figura 42 - Código de la Técnica WebScraping para Google

Una vez que el metabuscador ya tiene las nuevas estructuras de datos derivadas de los objetos de motores de búsqueda, las combina para generar otra nueva estructura de datos quitando aquellos resultados de búsqueda que se repiten entre los obtenidos de los motores de búsqueda. Luego, los resultados restantes son ordenados priorizando aquellos que se hayan repetido más veces. Por último, la estructura de datos con los resultados es regresado al componente "*Procesamiento de Datos*". En la Figura 43, se muestran las funciones principales del componente "Fusión de Resultados": combinarResultadosSinFiltrar, para combinar las estructuras de datos derivadas de los motores de búsqueda, y filtrarOrdenarResultados, para quitar resultados que se repiten y ordenarlos.

```
function fusionResultados($cantResultados,$resultadoBuscador){
    $ii=0;
    for($i=0; $i<$cantResultados; $i++){
        combinarResultadosSinFiltrar($resultadosTOTALES,$resultadoBuscador[$i]->getResultado(),$ii);
    }
    $resultadosTOTALESfiltradosOrdenados = filtrarOrdenarResultados($resultadosTOTALES);
    return $resultadosTOTALESfiltradosOrdenados;
}
```

Figura 43 - Funciones principales del componente "Fusión de Resultados"

La **Parte Inferior de la Arquitectura** (Parte JAVA) está compuesta por los componentes: “Interfaz Agentes”, “Gateway”, “Agente Consulta”, “Agente Actualización”, como se muestra en la Figura 44.

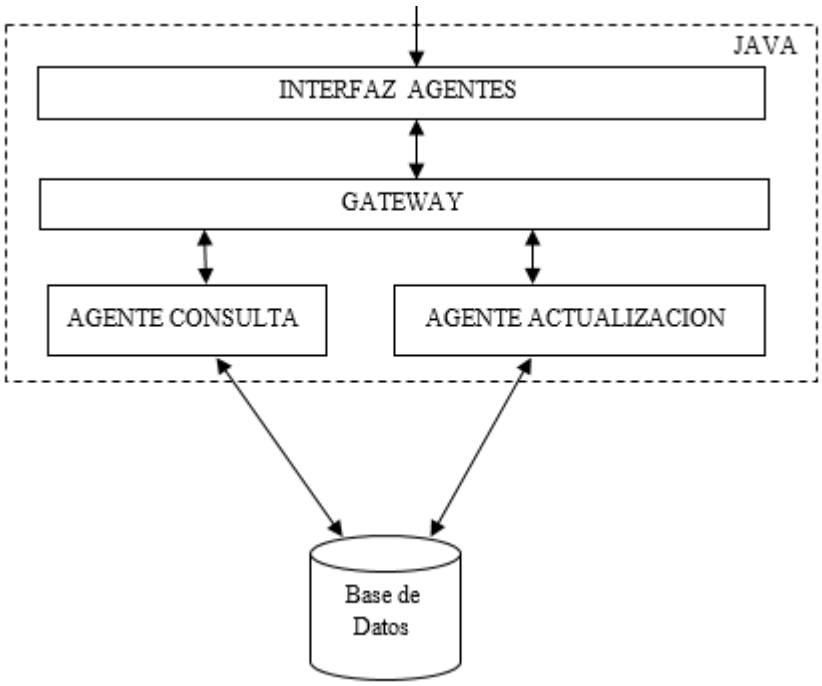


Figura 44 - Parte Superior de la Arquitectura del Metabuscador – Parte JAVA

La tarea de programar los agentes en el lenguaje de programación JAVA y vincularlos con el lenguaje de programación PHP para que funcionen como un todo fue un desafío importante.

En la Figura 45 puede verse parte de la arquitectura del metabuscador que involucra al componente “*Comunicación Agentes*” desarrollado en el lenguaje de programación PHP y al componente “*Interfaz Agentes*” desarrollado en el lenguaje de programación JAVA.

Figura 45 - Parte de Agentes de la Arquitectura del Prototipo de Metabuscador

El componente “***Comunicación Agentes***” de la Parte PHP es quién, a través de una implementación de streaming de java llamada “PHP/Java Bridge”, permite convertir las estructuras de datos, de PHP a JAVA y viceversa. Dicha implementación “PHP/Java Bridge”

facilita el uso de funciones y/o métodos entre ambos lenguajes, a tal punto que cuando se pone en funcionamiento el metabuscador, es posible invocar desde la Parte PHP funciones y/o métodos de JAVA que se ejecutan en Tomcat. En la Figura 46, puede verse como se da la conversión de estructuras de datos entre lenguajes así también la invocación desde PHP a funciones de JAVA, por ejemplo cuando se invoca a la función cargarDatosParaAgenteActualizacion.

```
function agenteActualizacion($paquetePHP){

    $this->setEstadoAgenteActualizacion("");

    $paqueteActualizacionJAVA = new JList();

    for($i=0; $i<7; $i++){
        $paqueteActualizacionJAVA->add(java_values($paquetePHP[$i]));
    }

    $this->interfaz->cargarDatosParaAgenteActualizacion($paqueteActualizacionJAVA);
    $this->interfaz->enlaceGateway("AgenteActualizacion");

    $estadoAgente = java_values($this->interfaz->obtenerEstadoDeActualizacion());
    $this->setEstadoAgenteActualizacion($estadoAgente);

}
```

Figura 46 - Código Fuente en PHP de Función de Agente de Actualización

El componente "***Interfaz Agentes***" de la Parte JAVA es quién tiene todas aquellas funciones que serán invocadas desde la Parte PHP por el componente "*Comunicación Agentes.* En la Figura 47 puede verse la función cargarDatosParaAgenteActualizacion que recibe por parámetro (de la Parte PHP) un paquete de tipo List de JAVA, a partir del cual se generará un nuevo paquete que será el contenido del mensaje destinado al AA.

```
public void cargarDatosParaAgenteActualizacion(List paqueteAct){
        this.paquete = new PaqueteActualizacion();
        this.paquete.setNombreAgente("AgenteActualizacion");
        paquete.setTitulo((String)paqueteAct.get(0));
        paquete.setUrl((String)paqueteAct.get(1));
        paquete.setDescripcion((String)paqueteAct.get(2));
        paquete.setLegajo((String)paqueteAct.get(3));
        paquete.setCalificacion((String)paqueteAct.get(4));
        paquete.setComentario((String)paqueteAct.get(5));
        paquete.setGrupo((String)paqueteAct.get(6));

}
```

Figura 47 - Código Fuente en JAVA de una Función del Componente "Interfaz Agentes"

Entre las funciones que incluye el componente "*Interfaz Agentes*", se encuentra aquella que permite invocar al componente "*Agente Gateway*". En la Figura 48, la función

enlaceGateway recibe por parámetro un String (agente), y a partir de él determina el objeto con el que será invocado el agente Gateway. Dicho objeto representa el contenido del mensaje que enviará el agente Gateway al agente correspondiente, es decir que si el agente Gateway va a comunicarse con el AC entonces el objeto (listado) debe ser de tipo "Listado", de lo contrario si va a comunicarse con el AA entonces el objeto (paquete) debe ser de tipo "PaqueteActualizacion". Se han desarrollado las clases de JAVA, Listado y PaqueteActualización para dar una estructura al contenido del mensaje tanto del agente de consulta como del agente de actualización, respectivamente.

```
public void enlaceGateway(String agente){

    if(agente.equals("AgenteConsulta")){
        try{ JadeGateway.execute(listado);
        } catch(Exception e) { e.printStackTrace(); }
    }
    if(agente.equals("AgenteActualizacion")){
        try{ JadeGateway.execute(paquete);
        } catch(Exception e) { e.printStackTrace(); }
    }
}
```

Figura 48 - Función enlaceGateway del Componente "Interfaz Agentes"

El componente "***Agente Gateway***" es el que se comunicará con el componente "*Agente Consulta*" o con el "*Agente Actualización*", dependiendo del tipo de objeto que reciba por parámetro. Como se puede ver en la Figura 49, la función processCommand del componente "*Agente Gateway*" recibe por parámetro un objeto que representa el contenido del mensaje (obj), luego determina si dicho objeto es un mensaje para AC o para AA, y por último envía el mensaje al componente correspondiente.

Para determinar si el mensaje es para AC o AA, la función compara el objeto recibido por parámetro y pregunta si dicho objeto es una instancia de PaqueteActualización, que de ser así la función procede a armar el mensaje para luego enviarlo al componente "*Agente Actualización*". Caso contrario, la función procede a armar el mensaje para luego enviarlo al componente "*Agente Consulta*".

```
protected void processCommand(java.lang.Object obj) {


    if (obj instanceof PaqueteActualizacion){

        paquete = (PaqueteActualizacion)obj;
        bandera = 1;
        try{
            //Enviando mensaje al Agente de Actualizacion
            ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
            msg.addReceiver(new AID(paquete.getNombreAgente(), AID.ISLOCALNAME) );
            msg.setContentObject(paquete);
            send(msg);
        }catch(IOException e){ }
    }else{

        listado = (Listado)obj;
        bandera = 2;

        try{
            //Enviando mensaje al Agente de Actualizacion
            ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
            msg.addReceiver(new AID(listado.obtenerAgente(), AID.ISLOCALNAME) );
            msg.setContentObject(listado);
            send(msg);
        }catch(IOException e11){ }
    }
}
```

Figura 49 - Función processCommand del Componente “Interfaz Agentes”

El componente “***Agente de Consulta***” se encarga de recibir el mensaje enviado por el componente “*Agente Gateway*” y determinar si corresponde a un pedido para generar un “Listado de Resultados”, o si corresponde a la generación de un “Historial de Resultados Calificados”. Esto puede verse reflejado en la Figura 50. Para determinar a qué tipo de pedido corresponde el paquete, el AC verifica, de manera de control que el listado no esté vacío, y el campo “titulo” del listado sea distinto de “null”. En caso de que el “titulo” del primer resultado no sea “null”, se define que el pedido corresponde a “Generar una consulta” y deja la variable “estado” en “0”, para que la estructura de control “switch” colocada a continuación interprete esa orden. En caso de que el “titulo” si sea igual a “null”, el Gateaway interpreta que es un pedido de actualización y coloca la variable “estado” en “1”.

```
protected void setup()
{

    addBehaviour(new CyclicBehaviour(this)
    {
        public void action()
        {
            ACLMessage msg = receive();
            int estado = 0;
            if (msg!=null) {

                try {
                    Listado listado = (Listado)msg.getContentObject();
                    int cantidadDeResultados = listado.cantidadResultados();

                    //Determina si el pedido es de "Generar Listado" o "Historial de Grupo"
                    if(cantidadDeResultados == 1 && listado.obtenerResultado(0).getTitulo().equals("null")) estado  = 1;

                    switch(estado){
                        case 0: listado = comportamientoUno(listado,cantidadDeResultados); break;
                        case 1: listado = comportamientoDos(listado,cantidadDeResultados); break;

                    }

                    ACLMessage reply = msg.createReply();
                    reply.setPerformative( ACLMessage.INFORM );
                    reply.setContentObject(listado);
                    send(reply);

                } catch(Exception e) { e.printStackTrace(); }

            } else block();

        }
    });


}
```

Figura 50 - Proceso principal del componente "Agente Consulta"

Si el AC interpreta que el pedido es para generar un “Listado de Resultados” (estado=0), entonces redirige el listado de resultados al comportamientouno() del agente. Como puede verse en la Figura 51, el agente primero establece la conexión con la BD (final conection bd = new conection();), recupera los resultados calificados por el grupo y luego procede a realizar la comparación de las url de los resultados recuperados de la BD con las url del listado de resultados (lo que corresponde a todo el bloque de la primer estructura de control While()). Por cada coincidencia verdadera (dada en if(urlBD.equals(urlListado))), el AC recupera los datos de los estudiantes (legajo, apellido, nombre, calificación y comentario) de un grupo que calificaron ese resultado y los asocia a él.

```
public Listado comportamientoUno(Listado listadoSinProcesar, int cantidadDeResultados){
    Listado listado = listadoSinProcesar;
    System.out.println("Comportamiento 1");

    try{

        //Conexión con la BD//
        final conection bd = new conection();

        String urlBD = "";
        String grupoListado = new String(listado.getGrupo());

        //Corroboro todos los resultados asociados a mi grupo
        System.out.println("Defino el conjunto de resultados asociados a mi grupo");
        String Query = "SELECT * FROM resultadoporgrupo INNER JOIN resultado ON (resultadoporgrupo.idResultado = resultado.idResultado)" +
        " WHERE resultadoporgrupo.idGrupo='"+grupoListado+"'";
        Statement st = conection.getConexion().createStatement();
        java.sql.ResultSet resultSet;
        resultSet = st.executeQuery(Query);

        //GENERANUEVOLISTADO
        while(resultSet.next()){
            urlBD = resultSet.getString("url");
            String urlListado = new String("");
            String calificacionGeneralListado = new String("");

            for(int i = 0; i<cantidadDeResultados; i++){
                urlListado = listado.obtenerResultado(i).getUrl();
                calificacionGeneralListado = resultSet.getString("calificacionGeneral");

                if(urlBD.equals(urlListado)){
                    System.out.println("Como la url se encuentra en la BD (en mi grupo), entonces recupero los datos de estudiantes asociados a dicha url");
                    String Query2 = "SELECT * FROM calificacionxestudiante "
                            + "INNER JOIN estudiante ON (calificacionxestudiante.legajo=estudiante.legajo) "
                            + "INNER JOIN resultado ON (calificacionxestudiante.idResultado=resultado.idResultado)"
                            + "INNER JOIN resultadoporgrupo ON (resultadoporgrupo.idResultado=calificacionxestudiante.idResultado AND "
                            + "calificacionxestudiante.idGrupo=resultadoporgrupo.idGrupo)"
                            + "WHERE resultado.url= '"+urlListado+"' AND calificacionxestudiante.idGrupo='"+grupoListado+"'";
                    Statement st2 = conection.getConexion().createStatement();
                    java.sql.ResultSet resultSet2 = st2.executeQuery(Query2);

                    listado.obtenerResultado(i).agregarCalificacionGeneral(calificacionGeneralListado);
                    listado.obtenerResultado(i).iniciarEstudiantes();//SI o SI VA ESTE METODO NUEVO

                    while(resultSet2.next()){

                        listado.obtenerResultado(i).agregarEstudiante(resultSet2.getString("legajo"),
                                                                      resultSet2.getString("apellido"),
                                                                      resultSet2.getString("nombre"),
                                                                      resultSet2.getString("calificacion"),
                                                                      resultSet2.getString("comentario"));
                    }
                }

            }
        }

        //ORDENALISTADO
        listado.ordenarListado();

        //CIERRA CONEXION BD
        conection.Close();

    } catch (SQLException ex){ System.out.println(ex);}

    //DEVUELVE LISTADO ORDENADO
    return listado;
}
```

Figura 51 - Comportamiento Uno - generación de Listado de Resultados

Una vez que el AC recorrió el listado y comparó todos los resultados, se encarga de ordenar el listado modificado. La función ordenarListado(), como se observa en la siguiente Figura invoca el método Collections.sort al que se le pasan dos parámetros: el listado desordenado de resultados con sus calificaciones generales y un objeto del tipo Comparator, que incluye la función Comparator(), a la cual se le pasa dos parámetros de tipo Resultado. El funcionamiento de dicho método fue definido con mayor detalle en el Capítulo IV “Algoritmo de Rankeo”.

```
public void ordenarListado(){
    Collections.sort(
        listado, new Comparator<Resultado>(){
            public int compare(Resultado o1, Resultado o2){
                return new Integer(o2.getCalificacionGeneral()).compareTo(new Integer(o1.getCalificacionGeneral()));
            }
    });
}
```

Figura 52 - Método ordenarListado()

En caso de que el pedido fuera de generar “Historial de Resultados Calificados” (estado=1), el AC envía el listado al comportamientodos(). Como puede verse en la Figura 53, primero establece la conexión con la BD (final conection bd = new conection();) y luego recupera todos los resultados calificados por un grupo, asociando a cada resultado los datos de los estudiantes que lo calificaron (correspondiente a todo el bloque de la primer estructura de control while()). Una vez obtenido este listado, lo ordena, convocando a la misma función de ordenarListado(), descripta en el comportamientouno().

```
public Listado comportamientoDos(Listado listadoSinProcesar, int cantidadDeResultados){
    Listado listado = listadoSinProcesar;
    System.out.println("Comportamiento 2");

    try{

        //CONEXION BD
        final conection bd = new conection();

        String urlBD = "";
        String grupoListado = new String(listado.getGrupo());

        listado.vaciarListadoDeResultados();

        //Corroboro todos los resultados asociados a mi grupo
        System.out.println("Defino el conjunto de resultados asociados a mi grupo");
        String Query = "SELECT * FROM resultadoporgrupo INNER JOIN resultado ON (resultadoporgrupo.idResultado = resultado.idResultado) "
        + "WHERE resultadoporgrupo.idGrupo='"+grupoListado+"'";
        Statement st = conection.getConexion().createStatement();
        java.sql.ResultSet resultSet;
        resultSet = st.executeQuery(Query);

        //GENERAHISTORIAL
        while(resultSet.next()){

            Resultado resultado = new Resultado();
            urlBD = resultSet.getString("url");

            resultado.setTitulo(resultSet.getString("titulo"));
            resultado.agregarUrl(resultSet.getString("url"));
            resultado.setDescripcion(resultSet.getString("descripcion"));
            resultado.agregarCalificacionGeneral(resultSet.getString("calificacionGeneral"));

            System.out.println("Como la url se encuentra en la BD (en mi grupo), entonces recupero los datos de estudiantes asociados a dicha url");
            String Query2 = "SELECT * FROM calificacionxestudiante "
                    + "INNER JOIN estudiante ON (calificacionxestudiante.legajo=estudiante.legajo) "
                    + "INNER JOIN resultado ON (calificacionxestudiante.idResultado=resultado.idResultado)"
                    + "INNER JOIN resultadoporgrupo ON (resultadoporgrupo.idResultado=calificacionxestudiante.idResultado "
                    + "AND calificacionxestudiante.idGrupo=resultadoporgrupo.idGrupo)"
                    + "WHERE resultado.url= '"+urlBD+"' AND calificacionxestudiante.idGrupo='"+grupoListado+"'";

            Statement st2 = conection.getConexion().createStatement();
            java.sql.ResultSet resultSet2 = st2.executeQuery(Query2);

            while(resultSet2.next()){

                resultado.agregarEstudiante(resultSet2.getString("legajo"),
                                            resultSet2.getString("apellido"),
                                            resultSet2.getString("nombre"),
                                            resultSet2.getString("calificacion"),
                                            resultSet2.getString("comentario"));
            }

            listado.agregarResultado(resultado);


        }

        //ORDENALISTADO
        listado.ordenarListado();

        //CIERRA CONEXION BD
        conection.Close();

    } catch (SQLException ex){ System.out.println(ex); }

    //RETORNA HISTORIAL ORDENADO
    return listado;
}
```

Figura 53 - Comportamiento Dos - generación de Historial de Resultados Calificados

El componente “***Agente Actualización***” recibe el mensaje del componente “*Agente Gateway*” y procede como se muestra en la Figura 54. El agente primero realiza la conexión

con la BD (final conection bd = new conection();) y luego busca recuperar los datos asociados, en la BD, del nuevo resultado calificado.

```
protected void setup()
{
    addBehaviour(new CyclicBehaviour(this)
    {
        public void action()
        {
            ACLMessage msg = receive();
            boolean flag = false;

            if (msg!=null) {

                try {
                    try {

                        final conection bd = new conection();

                        PaqueteActualizacion listado = (PaqueteActualizacion)msg.getContentObject();
                        String Query = "SELECT * FROM resultado INNER JOIN resultadoporgrupo ON (resultado.idResultado=resultadoporgrupo.idResultado) "
                                + "WHERE resultado.url='"+listado.getUrl()+"' AND resultadoporgrupo.idGrupo='"+listado.getGrupo()+"'";
                        Statement st = conection.getConexion().createStatement();
                        java.sql.ResultSet resultSet;
                        resultSet = st.executeQuery(Query);
                        resultSet.next();

                        if (resultSet.getRow()!=0){

                            System.out.println("La url fue anteriormente calificada/comentada por este grupo");
                            Query = "SELECT * FROM calificacionxestudiante INNER JOIN resultado ON calificacionxestudiante.idResultado=resultado.idResultado"
                                    + " WHERE calificacionxestudiante.legajo='"+listado.getLegajo()+"' AND calificacionxestudiante.idGrupo='"
                                    + listado.getGrupo() + "' AND resultado.url='"+listado.getUrl()+"'";
                            st = conection.getConexion().createStatement();
                            resultSet = st.executeQuery(Query);
                            resultSet.next();
                            if(resultSet.getRow()!=0){
                                System.out.println("La url fue anteriormente calificada/comentada por este estudiante en este grupo");

                                if(!listado.getCalificacion().equals("0")){

                                    System.out.println("La calificacion es distinto de 0, entonces debo modificar");
                                    if(listado.getComentario().equals(resultSet.getString("comentario"))){
                                        System.out.println("Como el comentario enviado es igual al anterior, entonces modifico la calificacion");
                                        modificaCalificacion(listado.getCalificacion(), resultSet.getInt("idResultado"), listado.getLegajo(),listado.getGrupo());
                                    }else{

                                        if(listado.getCalificacion().equals(resultSet.getInt("calificacion")+"")){
                                            System.out.println("Como calificacion enviada es igual al anterior, entonces modifico el comentario");
                                            modificaComentario(resultSet.getInt("idResultado"),listado.getComentario(),listado.getLegajo(),listado.getGrupo());
                                        }else{
                                            System.out.println("Como tanto la calificacion como el comentario enviados son distintos, entonces modifico ambos");
                                            modificarCalyCom(resultSet.getInt("idResultado"),listado.getCalificacion(),listado.getLegajo(),listado.getComentario(),
                                            listado.getGrupo());
                                        }
                                    }
                                }else{
                                    System.out.println("La calificacion es igual a 0, entonces tengo q eliminar");
                                    int idResultado = obtenerIdResultado(listado.getUrl());

                                    Query = "SELECT * FROM calificacionxestudiante"
                                    + " WHERE idGrupo='"+listado.getGrupo()+"' AND idResultado='"+idResultado+"'";
                                    st = conection.getConexion().createStatement();
                                    resultSet = st.executeQuery(Query);
                                    resultSet.last();
                                    int cantidadComentarios = resultSet.getRow();

                                    Query = "SELECT idGrupo FROM resultadoporgrupo WHERE idResultado='"+idResultado+"'";
                                    st = conection.getConexion().createStatement();
                                    resultSet = st.executeQuery(Query);
                                    resultSet.last();
                                    int cantidadGrupos = resultSet.getRow();
                                    st.close();

                                    System.out.println(cantidadComentarios);
                                    eliminarAporte(listado.getLegajo(), listado.getGrupo(), listado.getUrl(),cantidadComentarios, idResultado, cantidadGrupos);
                                }
                            }else{
                                System.out.println("La url no fue anteriormente calificada/comentada por este estudiante en este grupo");
                                nuevoComYCal(listado.getUrl(), listado.getCalificacion(), listado.getComentario(), listado.getLegajo(),listado.getGrupo());
                            }
                        }else{
                            System.out.println("La url no fue anteriormente calificada/comentada por este grupo");
                            nuevoUrl(listado);
                        }

                        conection.Close();

                        ACLMessage reply = msg.createReply();
                        reply.setPerformative( ACLMessage.INFORM );
                        listado.setEstado("ACTUALIZADO");
                        reply.setContentObject(listado);
                        send(reply);

                    } catch (SQLException ex) { System.out.println(ex); }

                } catch(Exception e) { e.printStackTrace(); }
            }
            else block();
        }
    });
}
```

Figura 54 - Proceso principal del componente "Agente Actualización"

Para agregar una nueva calificación y/o comentario a un resultado, para modificar los mismos o para eliminarlos, se deberán realizar distintos controles ya que pueden darse distintos casos. En primera instancia, se verifica si el resultado fue calificado anteriormente, para lo cual se controla que la cantidad de resultados devueltos de la BD sean distinto de cero (resultSet.getRow()!=0). En caso de que no se haya calificado anteriormente, se agrega el nuevo resultado a la BD mediante el método nuevoUrl(…).

Si el resultado fue calificado anteriormente, se debe verificar si fue realizado anteriormente por ese estudiante. Para ello recupera los datos del estudiante de la BD y controla que la cantidad de resultados devueltos de la BD sean distinto de cero (resultSet.getRow()!=0). Si la condición es falsa, el AA llama al método de nuevoComyCal(…), el cual se encarga de agregar el nuevo comentario y calificación de este estudiante al resultado (el cual ya fue calificado y comentado por otro estudiante del mismo grupo).

Si el resultado ya fue calificado y/o comentado anteriormente por el mismo estudiante, se debe verificar si lo que desea es modificar sus ingresos, o eliminarlos. Para ello el AA controla el nuevo ingreso del estudiante mediante la estructura de control if(listado.getCalificacion()!=0). Si el estudiante ingresa una calificación como “0”, el AA interpreta que quiere eliminar el resultado. Para ello, llama al método eliminarAporte(…), el cual eliminará el aporte del estudiante, y a su vez controlará y eliminará el resultado, siendo el caso de que éste no tenga ningún otro aporte por otro estudiante.

Si la estructura de control if(listado.getCalificacion()!=0) da como resultado un valor verdadero, el AA interpreta que quiere modificar la calificación, el comentario, o ambos. Para ello, mediante una serie de estructuras “If”, controla cuál es el parámetro que difiere al de la BD, y luego llama a los métodos modificarCalificacion(…), modificarComentario(…) o modificarCalyCom(…), según corresponda.

Todos los controles realizados, tienen en cuenta el grupo al que pertenece el estudiante. De modo que si un estudiante pertenece a distintos grupos y califica, agregando o no un comentario, al mismo resultado, dicho aporte será particular e independiente para cada grupo.

Por último, para poner en funcionamiento los agentes y poder interactuar con ellos, se ha utilizado una plataforma llamada JADE. En la Figura 45 puede verse a la plataforma

con dos contenedores, en el principal están ubicados los agentes AC y AA, y en el otro está ubicado el Agente Gateway.

Al ponerse en marcha el prototipo de metabuscador, es necesario iniciar la plataforma para poner en funcionamiento los agentes, de lo contrario no se tendrá respuesta por parte de ellos. Es importante remarcar nuevamente que el componente "Interfaz Agente" es el que va a invocar al Agente Gateway como mediador con los agentes.

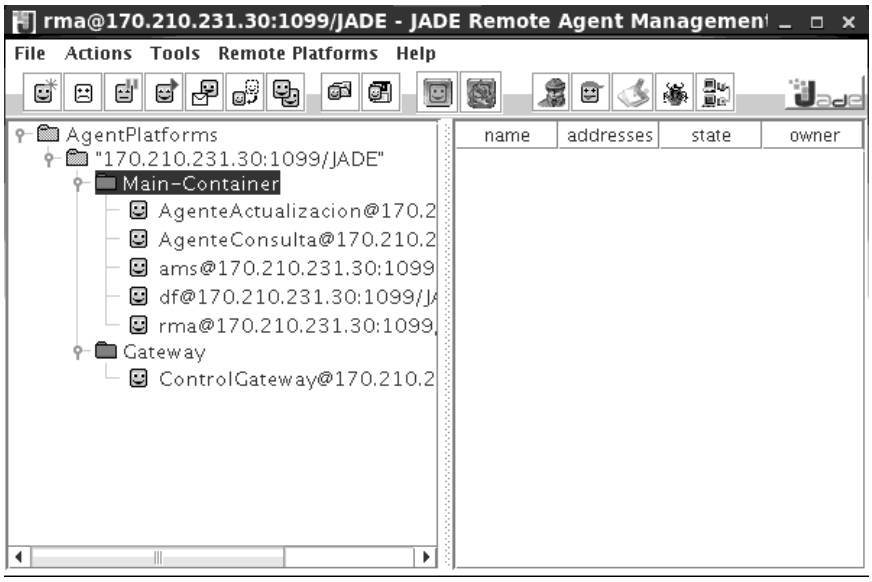


Figura 55 - Plataforma Jade

VI.I. PRUEBAS DE CAJA BLANCA & CAJA NEGRA

Para probar el Prototipo de Metabuscador Colaborativo Basado en Agentes para Grupos de Estudiantes, durante su proceso de desarrollo y una vez finalizado, se realizaron diferentes casos de prueba para encontrar errores.

- *Pruebas de caja negra*: Estas pruebas fueron llevadas a cabo para demostrar que cada función del prototipo de metabuscador está operativa, considerando que la entrada y salida de la función sean correctas.
- *Pruebas de caja blanca*: Estas pruebas fueron llevadas a cabo para determinar que cada operación del prototipo de metabuscador se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada. Se han comprobado los caminos lógicos del programa.

Existen varios tipos de prueba dentro de las Pruebas de Caja Negra. Para este trabajo se hizo uso de la *Prueba de Partición Equivalente* que divide el dominio de entrada de un

programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada.

En el desarrollo de la prueba de partición equivalente para el presente trabajo final, se identifican las condiciones, clases válidas y clases inválidas que se muestran en la Tabla 15. Las condiciones consideradas fueron “Grupo”, que representa el identificación del grupo del estudiante; “Legajo”, es la identificación del estudiante; “Buscadores”, que representa la cantidad de buscadores seleccionados; “Frase de Búsqueda”, que define la frase de búsqueda que realiza el estudiante; “Comentario”, es el comentario que ingresa un estudiante para resultado; “Nombre”, “Apellido”, “Correo”, “Asunto” y “Comentario Contacto” son los datos de nombre, apellido, correo electrónico, asunto y comentario respecto a una consulta a los responsables de la aplicación, que son ingresados por un estudiante. Cada una de las clases de equivalencia está identificada por un número único y representan un conjunto de estados válidos o inválidos para las condiciones de entrada.

Condición	Clases Válidas	Clases Inválidas
Grupo	1) Cadena alfanumérica mayor que 0	2) Cadena nula
Legajo	3) Cadena de 10 caracteres, donde los caracteres del 1 al 5 son numéricos, el carácter 6 = “_” y los caracteres de 7 a 10 son numéricos.	4) Cadena de caracteres del 1 al 5 no es numérica, 5) Cadena de caracteres del 7 al 10 no es numérica. 6) El carácter 6 es distinto de “_” 7) Cadena de caracteres menor que 10 8) Cadena de caracteres mayor que 10
Buscadores	9) Número de buscadores mayor o igual a 1 y menor o igual que 4	10) Número de buscadores igual a 0
Frase de Búsqueda	11) Cadena alfanumérica mayor que 0	12) Cadena nula
Comentario	13) Cadena alfanumérica menor o igual que 300 caracteres	14) Cadena alfanumérica mayor a 300 caracteres.
Nombre	15) Cadena alfanumérica mayor que 0.	16) Cadena nula
Apellido	17) Cadena alfanumérica mayor que 0.	18) Cadena de más de 30 caracteres.
Correo	19) Cadena alfanumérica mayor que 0.	20) Cadena nula.
Asunto	21) Cadena alfanumérica mayor que 0.	22) Cadena nula.
Comentario Contacto	23) Cadena alfanumérica mayor que 0.	24) Cadena nula.

Tabla 15 - Caja Negra - Condiciones, Clases Válidas y Clases Inválidas

En Tabla 16 se muestran los casos de prueba considerados para cada una de las condiciones establecidas en la Tabla 15. Por ejemplo, el caso de prueba para el grupo “LSI-IA-2” y el estudiante con legajo “300030_2008” (correspondiente a la primera fila de la Tabla 16) responden a las clases de equivalencia válidas 1 y 3, donde el “Grupo” es una cadena alfanumérica mayor que 0 y el “Legajo” es una cadena alfanumérica mayor que 0, permitiendo un ingreso exitoso al metabuscador. En el caso de que no se ingrese un “Grupo”, como puede observarse en el caso de prueba de la fila 2 de la Tabla 16, se presenta una clase de equivalencia válida, correspondiente al “Legajo” y una clase de equivalencia inválida, correspondiente al “Grupo”, produciendo un ingreso fallido al metabuscador.

Otro ejemplo puede verse en la fila 5 de la Tabla 16, donde la “Frase de Búsqueda” y la cantidad de “Buscadores” son “null” y “0” respectivamente, respondiendo a Clases de Equivalencias Inválidas para ambos casos e impidiendo realizar la búsqueda.

Casos de Prueba

Caso de Prueba	Clases Válidas	Clases Inválidas	Salida
LSI-IA-2,30030_2008	1,3	----	Ingreso exitoso al metabuscador
null, 30030_2008	3	2	Ingreso fallido al metabuscador
LSI-IA-1,3005_2a09	1	5,7	Ingreso fallido al metabuscador
LSI-SIM,300a5/20091	1	4,6,8	Ingreso fallido al metabuscador
LSI-IA-2,”Buscadores”	9,11	--	Búsqueda exitosa
0, null	--	10,12	Búsqueda fallida
“Adrián, Suarez”, “adrians@gmail.com”, “Resultados”, “No muestra mis calificaciones…”	15,17,19,21,23	---	Contacto Exitoso
null, null, null, null, null	---	16,18,20,22,24	Contacto Fallido

Tabla 16 - Caja Negra - Caso de Prueba

La técnica elegida para realizar las Pruebas de Caja Blanca es el de *Prueba del Camino Básico.* Lo que se busca con este tipo de prueba es obtener una medida de la complejidad de un diseño procedimental, para luego utilizar esta medida como guía para la definición de la serie de caminos básicos de ejecución. Con ello, luego, se puede realizar el diseño de casos de prueba que garanticen que cada camino se ejecuta al menos una vez.

Los pasos de esta técnica pueden ser resumidos en los siguientes:

- Obtener el grafo de flujo, a partir del diseño o del código del módulo
- Obtener la complejidad ciclomática del grafo de flujo
- Definir el conjunto básico de caminos independientes
- Determinar los casos de prueba que permitan la ejecución de cada uno de los caminos anteriores
- Ejecutar cada caso de prueba y comprobar que los resultados son los esperados

A continuación, se realiza la prueba de caja blanca sobre una de las funciones principales del metabuscador. El Anexo I contiene la documentación correspondiente a las pruebas de las restantes funciones principales del metabuscador.

Función Principal – Despachador de Consulta y Fusión de Resultados

En la porción A de la Figura 56 muestra el código correspondiente al método "Despachador de Consulta y Fusión de Resultados". Cada parte de código separada con los indicadores 1, 2 y 3, permiten graficar el "Grafo de Control de Flujo" correspondiente, como se muestra en la porción D.

```
function despachadorYfusion ($consulta,$pagina){

   $direccion_simple_html_dom = "../".$this->getDirectorios(1)."/".
                                $this->getDirectorios(2).
                                "/simple_html_dom.php";
   $resultadoBuscador = array();                                          1

   for($i=0; $i < $this->CantBuscadoresSeleccionados(); $i++){

        $direccion_buscador = "../".$this->getDirectorios(1)."/".
                              $this->getDirectorios(2)."/".
                              $this->getBuscadoresSeleccionados($i).
                              "/mod_init.php";
        $resultadoBuscador[] = new buscador($direccion_simple_html_dom,   2
                                            $direccion_buscador,
                                            $consulta,
                                            $this->getBuscadoresSeleccionados($i),
                                            $pagina);
   }

   despachadorConsulta($resultadoBuscador);
   esperaFinalizacionConsulta($resultadoBuscador);
   destruirHilosConsulta($resultadoBuscador);

   $resultadosFusionados = fusionResultados($this->CantBuscadoresSeleccionados(),   3
                                            $resultadoBuscador);

   $this->setResultadoMetabuscador($resultadosFusionados);
   unset($resultadoBuscador);
   unset($resultadosFusionados);
}
```

A

Complejidad Ciclomática V(G)

B

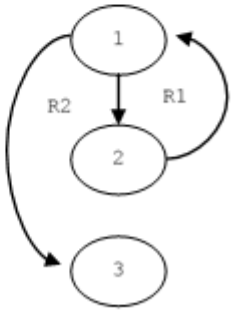
Aristas: 3
Nodos: 3
Regiones = 2
Nodos Predicado: 1
V(G): 3 (Aristas) - 3 (Nodos) + 2 = 2
V(G): 1 (Nodos Predicado) + 1 = 2

Caminos independientes

C

a) 1, 2, 1, 3
b) 1, 3

Grafo de Control de Flujo



D

Figura 56 - Prueba de Caja Blanca para el método Despachador de Consulta y Fusión de Resultados

A partir del grafo obtenido, se puede definir la “Complejidad Ciclomática”, como se muestra en la porción B. Dicha complejidad, simbolizada con V(G), se puede calcular de distintas maneras: contando la cantidad de regiones; restando la cantidad de nodos a la cantidad de aristas y sumando el valor “2”; o sumándole a la cantidad de nodos predicados el valor “1”. Todos estos valores deben ser iguales. Es importante calcular el V(G) porque este indica la cantidad de caminos independientes que existen y se deben controlar. Estos caminos independientes están definidos en la porción C de la Figura 56.

CAPITULO VII - EXPERIMENTACIÓN

El funcionamiento de June, nuestro metabuscador basado en agentes para grupos de estudiantes colaborativos, fue validado mediante su uso por parte de alumnos de las carreras de Licenciatura en Sistemas de Información (LSI) y Programador Universitario en Informática (PUI), pertenecientes a la Universidad Nacional de Santiago del Estero. Específicamente la experimentación se llevó a cabo con un total de 23 (veintitrés) estudiantes en el marco de las asignaturas Base de Datos del PUI, Criptografía, e Inteligencia Artificial de LSI. Para la asignatura Base de Datos se definió 1 grupo de 3 integrantes. Los estudiantes de Criptografía se distribuyeron en 3 grupos, 2 de 3 integrantes, y 1 de 4 integrantes. La asignatura Inteligencia Artificial contó con 4 grupos, 2 grupos de 3 integrantes y 2 grupos de 2 integrantes.

Para realizar esta experimentación los docentes responsables de las asignaturas mencionadas crearon los grupos colaborativos, diseñaron una actividad de aprendizaje en la que los estudiantes debieran utilizar el metabuscador, y la asignaron a los diferentes grupos para que la realizaran en el plazo de una semana. Los grupos trabajaron sin intervención alguna por parte del docente, y finalizaron exitosamente la actividad encomendada respetando las pautas de trabajo y los plazos de tiempo establecidos para ello.

Todas las intervenciones de los estudiantes utilizando el metabuscador fueron almacenadas para su posterior análisis. Además, finalizadas las experiencias de aprendizaje, en cada una de las asignaturas se recabaron las opiniones acerca de la utilización de June por parte de los estudiantes y de los docentes involucrados. A tal efecto se diseñaron dos cuestionarios, uno constituido por 30 preguntas destinado a los estudiantes, y otro de 7 preguntas, dirigido a los profesores. Cada una de las preguntas contenidas en ambos cuestionarios, incluyeron respuestas prefijadas dentro de las cuales el encuestado marcó aquella que se ajustó a su opinión. Los modelos de dichos cuestionarios están incluidos en el Anexo V.

El cuestionario dirigido a los estudiantes se organiza en dos partes. La primera permite por un lado, recabar datos personales del estudiante (materia en la cual usó el metabuscador y buscadores que utiliza con más frecuencia) y por otro lado, recopilar opiniones sobre June en relación con el diseño realizado para su interfaz (la organización de contenidos, presentación de resultados, ubicaciones tanto de las calificaciones como de los comentarios, forma de acceder a los comentarios, forma de acceder a la opción de calificar

y comentar, utilización de estrellas para calificar, etc.). La segunda parte permite recolectar opiniones sobre el funcionamiento y uso de June (la facilidad de su uso, la cantidad y calidad de los resultados de búsqueda, el impacto de las calificaciones y de los comentarios efectuados por otros usuarios en los procesos de búsqueda disparados, etc.). Se incluyó al final la opción de sugerir mejoras a realizar por parte de los desarrolladores.

El cuestionario destinado a los docentes permite recabar opiniones relacionadas con la facilidad de uso y utilidad del metabuscador para los estudiantes que trabajen en grupo, el impacto en la productividad y motivación de los alumnos, y la calidad y cantidad de material obtenido por los estudiantes mediante el uso de June. En el caso de los docentes también se incluyó al final la opción de sugerir mejoras a realizar por parte de los desarrolladores.

Análisis de resultados

Finalizada la experimentación se llevó a cabo el procesamiento y análisis de los 116 (ciento dieciséis) registros almacenados en la base de datos de June respecto al uso que los estudiantes hicieron del metabuscador, es decir, la información registrada acerca de los procesos de búsqueda disparados, los comentarios y las calificaciones efectuadas. Recordemos que el metabuscador June facilita a los estudiantes la posibilidad de calificar un resultado para indicar su utilidad, pudiendo agregar o no un comentario que justifique esa calificación, como así también les permite observar junto a los resultados de búsqueda (ordenados descendentemente por las calificaciones asociadas a ellos) tanto las calificaciones como los comentarios de todos los integrantes del grupo, permitiéndoles calificar y comentar aquellos resultados que ya fueron calificados por otros integrantes. La posibilidad de visualizar a los resultados de búsqueda ordenados por su calificación, se debe al algoritmo de rankeo, quien es el encargado de posicionar resultados con mayor utilidad por encima de aquellos con menor utilidad.

Para realizar el procesamiento de los datos se decidió evaluar las siguientes variables: cantidad de resultados útiles e inútiles, cantidad de calificaciones efectuadas, cantidad de resultados comentados.

Respecto a la primera variable evaluada, cantidad de resultados útiles e inútiles, cabe aclarar primero que significado se dio a estos adjetivos. Un resultado de búsqueda es útil cuando la información que posee el sitio, accedido a través del enlace de dicho resultado, se ajusta a la necesidad o preferencia o es del agrado del estudiante que realiza la búsqueda.

Caso contrario, es un resultado de búsqueda inútil o no útil, cuando la información que posee el sitio no se ajusta a la necesidad o preferencia o no es del agrado del estudiante que realiza la búsqueda.

Considerando que la escala con la que los estudiantes podían calificar los resultados de búsqueda, va del 1 al 10 inclusive (mediante el uso de estrellas), se ha decidido tomar por *Resultado Inútil* a aquel resultado cuya calificación sea *Menor o Igual* a cuatro (calificación <= 4), y por *Resultado Útil* a aquel cuya calificación sea *Mayor o Igual* a cinco (calificación >= 5), como se muestra en la Tabla 17.

Resultado	**Calificación**
Útil	[5,10] Intervalo entre 5 y 10 inclusive
Inútil	[1,4] Intervalo entre 1 y 4 inclusive

Tabla 17 - Resultado Útil/Inútil

El criterio que se ha utilizado para determinar si un resultado de búsqueda es inútil recae en el análisis de los comentarios realizados por los estudiantes y sus calificaciones. Se ha observado que para calificaciones que oscilan entre 1 y 4 inclusive, los comentarios están relacionados con: escasa o ninguna información que brinda el sitio web, poca claridad en la información que brinda el sitio web, impedimentos para acceder a la información en el sitio web (por ejemplo: registrarse en el sitio creando una cuenta paga). Por el contrario, respecto a los resultados de búsqueda útiles, se ha observado que para calificaciones que oscilan entre 5 y 10 inclusive, los comentarios están relacionados con: la suficiente, buena, importante o excelente información que brinda el sitio web sin importar su formato (videos, gráficos, esquemas o textos).

En la Tabla 18 se muestra el porcentaje de resultados útiles e inútiles sobre el total de 116 (ciento dieciséis) resultados calificados sin diferenciar los grupos de trabajos.

Resultados Útiles	88	76%
Resultados Inútiles	28	24%
Total	116	100%

Tabla 18 - Total de Resultados Calificados

En la Tabla 19 se indican los porcentajes de resultados de búsqueda útiles e inútiles pero discriminando subtotales considerando los 8 (ocho) grupos que participaron en la experimentación.

Cátedra – Grupo	*% Resultados Útiles*	*% Resultados Inútiles*
Base de Datos - Grupo 1	81,82%	18,18%
Criptografía - Grupo 1	100%	0%
Criptografía - Grupo 2	85,19%	14,81%
Criptografía - Grupo 3	65,22%	34,78%
Inteligencia Artificial - Grupo 1	66,67%	33,33%
Inteligencia Artificial - Grupo 2	88,89%	11,11%
Inteligencia Artificial - Grupo 3	60%	40%
Inteligencia Artificial - Grupo 4	54,55%	45,45%

Tabla 19 - Resultados Calificados Útiles / Inútiles por Grupo

Respecto a la segunda variable evaluada, cantidad de resultados calificados, se computaron los totales por grupo, es decir, la cantidad de calificaciones que ha recibido cada resultado con su correspondiente porcentaje sobre el total de resultados (Tabla 20). Cabe aclarar que, puesto que un estudiante puede ingresar una sola calificación (o modificarla), entonces la cantidad máxima de calificaciones que puede tener un resultado de búsqueda está determinada por la cantidad de integrantes del grupo, es decir, sólo es posible una calificación por estudiante.

La información suministrada por la Tabla 20 se interpreta de la siguiente forma. Véase, por ejemplo, la fila del Grupo 4 de Inteligencia Artificial. Del total de resultados calificados por este grupo, la primera columna indica que el 91% de los resultados han sido calificados por un único estudiante, la segunda columna que el 9% fue calificado por dos estudiantes, mientras que el 0% de la tercera columna indica que no hay un resultado que haya sido calificado por 3 o más integrantes del grupo.

Cátedra – Grupo	*1 Calificación*	*2 Calificaciones*	*3 Calificaciones o más*
Base de Datos - Grupo 1	64%	27%	9%
Criptografía - Grupo 1	69%	31%	0%
Criptografía - Grupo 2	93%	7%	0%
Criptografía - Grupo 3	83%	17%	0%
Inteligencia Artificial - Grupo 1	100%	0%	0%
Inteligencia Artificial - Grupo 2	100%	0%	0%
Inteligencia Artificial - Grupo 3	100%	0%	0%
Inteligencia Artificial - Grupo 4	91%	9%	0%

Tabla 20 - Porcentajes de Resultados Calificados por Integrantes por Grupo

La Tabla 21 muestra el Total de Resultados Inútiles Calificados por Integrantes por Grupo. La información suministrada por la Tabla 21 se interpreta de la siguiente forma. Por ejemplo, el contenido de la fila del Grupo 3 de Criptografía indica que del total de resultados inútiles calificados por este grupo, 7 de esos resultados han sido calificados solo por un estudiante, y sólo 1 fue calificado por dos estudiantes. No existieron casos en que tres estudiantes de un grupo hubieran calificado un mismo resultado inútil.

Cátedra - Grupo	*1 Calificación*	*2 Calificaciones*
Base de Datos - Grupo 1	2	0
Criptografía - Grupo 1	0	0
Criptografía - Grupo 2	4	0
Criptografía - Grupo 3	7	1
Inteligencia Artificial - Grupo 1	4	0
Inteligencia Artificial - Grupo 2	1	0
Inteligencia Artificial - Grupo 3	4	0
Inteligencia Artificial - Grupo 4	4	1

Tabla 21 - Total de Resultados Inútiles Calificados por Integrantes por Grupo

Considerando el total de resultados calificados por los estudiantes sin discriminar grupos, se obtuvieron los porcentajes de cantidad de calificaciones por resultado que muestran la Tabla 22 y la Figura 57. Allí puede observarse que es notoria la diferencia que

existe en lo que respecta a aquellos resultados con 1 (una) calificación respecto a aquellos resultados con 2 (dos) o más calificaciones, cuyos porcentajes son muy bajos.

Cantidad de Calificaciones	*Porcentaje de Resultados Calificados*
3	1%
2	12%
1	87%

Tabla 22 - Porcentaje de Resultados Calificados sin discriminar Grupos

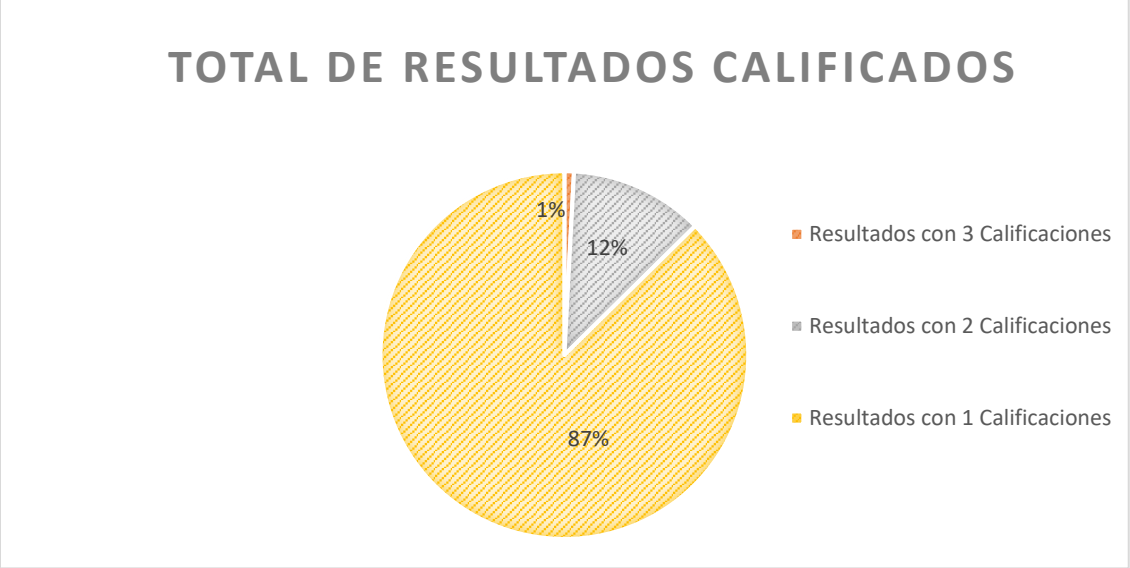


Figura 57 - Gráfico del porcentaje del Total de Resultados Calificados

Respecto a la tercera variable evaluada, cantidad de resultados comentados, se computaron los totales por grupo, es decir, la cantidad de comentarios que ha recibido cada resultado con su correspondiente porcentaje sobre el total de resultados comentados (Tabla 23). Cabe recordar que un estudiante, al calificar un resultado, opcionalmente puede agregar un comentario, lo que llevó a que se registraran 81 resultados comentados (sobre los 116 resultados calificados).

La información suministrada por la Tabla 23 se interpreta de la siguiente forma. El contenido de la fila correspondiente al Grupo 1 de Criptografía indica que del total de resultados calificados por este grupo, el 53.84% también han sido comentados por un estudiante, y que el 7.69 % fue comentado por dos estudiantes. No registrándose resultados comentados por tres o más estudiantes.

Cátedra – Grupo	*Sin Comentario*	*1 Comentario*	*2 Comentarios*
Base de Datos - Grupo 1	45.46%	54.54%	0%
Criptografía - Grupo 1	38.47%	53.84%	7.69%
Criptografía - Grupo 2	48.15%	48.15%	3.70%
Criptografía - Grupo 3	0%	86.96%	13.04%
Inteligencia Artificial - Grupo 1	58.34%	41.66%	0%
Inteligencia Artificial - Grupo 2	100%	0%	0%
Inteligencia Artificial - Grupo 3	20%	80%	0%
Inteligencia Artificial - Grupo 4	0%	90.91%	9.09%

Tabla 23 - Porcentajes de Resultados Comentados por Integrantes por Grupo

Procesamiento de los cuestionarios a usuarios

Como ya se dijo antes, finalizada la experiencia utilizando el metabuscador June, se solicitó a los estudiantes y a los docentes responder de manera individual y voluntaria un cuestionario (incluido en Anexo V). El procesamiento de tales cuestionarios implicó analizar las respuestas brindadas por 19 estudiantes sobre un total de 23 estudiantes participantes en la experiencia, lo cual representa aproximadamente el 83% de la población total afectada, y también de las respuestas brindadas por 3 docentes que representan el 100 % de los profesores participantes. Los resultados obtenidos son los siguientes:

A) Respecto al cuestionario a estudiantes:

A continuación se describen los resultados obtenidos respecto a las opiniones sobre las pantallas del metabuscador.

- El 100 % de los encuestados utiliza Google como Buscador de preferencia.
- El 100% de los estudiantes coincide en que la organización del contenido es correcta.
- El 100% opina que la presentación de los resultados, priorizando aquellos con mayor calificación, es agradable.
- La mayoría de los participantes (94.7%) coincide en que la calificación de sus compañeros mediante estrellas (de 1 a 10 estrellas) es de su agrado, mientras que el 5.3% (1 estudiante sobre 19) no comparte esta opinión.

- El 100% de los encuestados opina que disponer de la calificación de sus compañeros a la derecha de los resultados, la forma de acceder a los comentarios de cada resultado y su ubicación, son de su agrado.
- El 94.7% coincide en que la información proporcionada en los comentarios de cada resultado es suficiente, mientras que el 5.3% (1 estudiante sobre 19) no comparte esta opinión.
- El 84.2% coincide en que la forma de acceder a la opción de calificar y/o comentar un resultado es agradable, mientras que el restante el 15.8% (3 estudiantes sobre 19) no comparte esta opinión.
- La mayoría (94.7%) opina que utilizar estrellas para calificar y la ubicación de las mismas es agradable, mientras que el 5.3% (1 estudiante sobre 19) no comparte esta opinión.
- El 100% de los encuestados coincide en que la ubicación del cuadro de texto para ingresar un comentario es agradable.

A continuación se describen los resultados obtenidos respecto a las opiniones sobre la facilidad de uso del metabuscador.

- El 52.6% de los estudiantes coincide en que "Tal Vez" utilizaría el metabuscador con frecuencia, el 26.3% está "De acuerdo", el 15.8% está en "Desacuerdo" con ello, mientras que 5.3% restante está "Completamente de acuerdo" en que lo utilizaría.
- La mayoría de los encuestados (68.4%) se encuentra en "Desacuerdo" respecto a afirmar que el metabuscador es innecesariamente complejo. El 15.8% está en "Completo Desacuerdo", el 10.5% de ellos opina que "Tal vez" es innecesariamente complejo, mientras que el restante 5.3% afirma que lo es.
- El 52.6% opina que está en "Completo de acuerdo" respecto a que el metabuscador es fácil de usar, el 31.6% está "De acuerdo", el 10.5% está en "Completo desacuerdo", mientras que el 5.3% opina que "Tal vez" es fácil de usar.
- El 57.9% de los estudiantes está en "Desacuerdo" con que necesita ayuda de otra persona para utilizar el metabuscador, el 31.6% está en "Completo Desacuerdo" respecto a esta ayuda, mientras que el 10.5% cree que "Tal vez" necesite algo de ayuda.
- El 52.6% está "De acuerdo" en que están bien integradas las funciones del metabuscador, el 21.1% está "Totalmente de acuerdo" y el 26.3% opina que "Tal vez" lo están.

- El 57.9% de los encuestados afirman que se encuentran en “Desacuerdo” con la afirmación de que el metabuscador presenta inconsistencias, el 31.6% opina que “Tal vez” se presentan inconsistencias, el 5.3% está en “Completo desacuerdo”, y el restante 5.3% opinan que están “De acuerdo”.
- El 36.8% de los estudiantes está “Totalmente de acuerdo” en afirmar en que la mayoría de los estudiantes aprenderían rápidamente a usar el metabuscador, el 36.8% está “De acuerdo”, el 10.5% opina que “Tal vez”, el 10.5% está en “Completo Desacuerdo” sobre que sería rápido de aprender, mientras que el 5.3% restante está en “Desacuerdo”.
- El 63.2% de los encuestados se encuentra “De acuerdo” en que se sintió confiado en cuanto al manejo del metabuscador, el 15.8% opinó que “Tal vez” se sintió confiado, el 10.5% “Completamente de acuerdo”, el 5.3% restante se encontró en “Completo Desacuerdo”, y el 5.3% restante se encontró en “Desacuerdo”.
- El 73.7% de los estudiantes opina que está en “Desacuerdo” respecto a que es necesario aprender muchas cosas antes de manejar el metabuscador, el 15.8% está “Completamente en desacuerdo”, mientras que el 5.3% está “Completamente de acuerdo”, y el 5.3% “De acuerdo”.
- El 100% de los estudiantes afirma que el funcionamiento del metabuscador se ajusta a su finalidad (realizar búsquedas e indicar aquellos resultados calificados, con o sin comentarios, entre los compañeros de grupo, mostrando nombre, apellido, calificación y comentario de quienes calificaron).
- La mayoría de los encuestados (78.9%) opina que utilizando el metabuscador es más rápido comunicar a sus compañeros de grupo, los resultados que son importantes o no, en lugar de utilizar WhatsApp, Facebook, Mensaje de Texto, una red social, entre otros, mientras que el 21.1% discrepa de esta afirmación.
- El 94.7% respondió que el metabuscador incentiva al estudiante a buscar otros resultados (que no hayan sido calificados), mientras que el restante 5.3% no comparte esta opinión.
- El 78.9% de los estudiantes opina que el metabuscador incentiva a que sus compañeros de grupo no ingresen a un resultado que ha sido calificado (con o sin cometario), evitándole a ellos tener que buscar y analizar la información dentro del sitio vinculado a ese resultado. El 21.1% restante, está en desacuerdo.
- El 94.7% coincide en que información brindada en la sección de ayuda es suficiente, mientras que el restante 5.3% no comparte esta opinión.

- El 100% se encuentra de acuerdo en afirmar que los campos de la sección contacto son suficientes.
- El 94.7% de los encuestados opina que los resultados brindados por June para desarrollar su tarea le fueron de utilidad, mientras que el restante 5.3% no comparte esta opinión.

B) Respecto al cuestionario a docentes:

- El 100% de los docentes encuestados opinan que el metabuscador June es fácil de usar y es útil para los trabajos en grupo.
- Todos los docentes (100%) están de acuerdo en afirmar que si los estudiantes observan las calificaciones, con o sin comentarios de sus compañeros, esto impacta en su motivación y/o productividad.
- El 100% de los docentes coincide en que la posibilidad de calificar e incluir un comentario impacta positivamente en la motivación y/o productividad de los estudiantes.
- El total de los docentes (100%) considera que los documentos devueltos por el metabuscador contribuyeron a mejorar la calidad del trabajo en los grupos.
- El 100% de los docentes afirma que el uso del metabuscador permitió a los estudiantes disponer de fuentes de información en cantidad apropiada.
- La mayoría de los docentes (66.7%) cree que no es necesario considerar ningún otro aspecto en el metabuscador, mientras que el 33.3% considera oportuno agregar otros motores de búsqueda como una mejora adicional.

Análisis y discusión de resultados

Una vez realizada la experiencia con los 23 estudiantes de las distintas asignaturas distribuidos en 8 grupos distintos, fue posible procesar la información almacenada por June y la recopilada mediante cuestionarios, tal como se mostró en la sección anterior. Sin embargo, un análisis detallado sobre los resultados obtenidos se puede afirmar que el uso del metabuscador June contribuye a disminuir el tiempo de comunicación entre los integrantes de un grupo, y también contribuye a disminuir el tiempo de lectura de un mismo documento sin contenido significativo para los integrantes de un grupo. Esto queda demostrado en los párrafos que se incluyen a continuación.

En la Tabla 19 se pueden observar los porcentajes de resultados útiles e inútiles encontrados en cada uno de los grupos. Es para destacar que si bien las calificaciones y comentarios que hace un estudiante sobre los resultados le sirven a él mismo durante del

proceso de búsqueda de información, la importancia de calificar los resultados radica en comunicar, a los demás integrantes del grupo, que tan útil o no puede ser el resultado. De esa manera un estudiante puede ver la calificación y el comentario de aquellos integrantes del grupo que hayan calificado y comentado un resultado, y a partir de ahí decidir si accede o no al sitio web, teniendo en cuenta que al accederlo deberá analizar, si lo desea, la información que brinda dicho sitio para determinar si se ajusta o no, a la información buscada.

Sujeto a lo anterior, como se observa en la Tabla 23, los porcentajes de resultados que han sido comentados por los grupos oscilan entre el 41.66% al 100%. Es decir, que si bien los estudiantes podían comentar opcionalmente, ellos han hecho uso de esta posibilidad para argumentar e indicar si eran o no de su agrado esos resultados.

Independientemente de la importancia de que un resultado sea útil o no, como puede observarse en la Tabla 20, en todos los grupos se detectó un porcentaje de resultados de búsqueda con dos calificaciones, oscilando entre el 0% y el 31% inclusive, y es necesario mencionar que solo dos resultados de esos fueron resultados inútiles, como se muestra en la Tabla 21. No existieron casos donde haya resultados inútiles con más de 2 calificaciones. Además, se observó un porcentaje elevado de resultados de búsqueda con una sola calificación, oscilando entre el 64% hasta el 100%. En la Tabla 22, se puede observar los porcentajes relacionados con el Total de Resultados Calificados (sin considerar los grupos), y es en ella donde se destaca la diferencia porcentual entre aquellos resultados con una calificación con aquellos de dos calificaciones. El porcentaje más pequeño corresponde a los resultados con tres calificaciones. Considerando lo expuesto, se puede afirmar de todo lo anterior mencionado, que la cantidad de resultados inútiles con dos calificaciones es realmente baja, dando un indicio de que es significativamente menor el análisis a resultados inútiles realizado por un grupo, es decir que es baja la cantidad de análisis repetidos de resultados inútiles por grupo. Por otro lado, se destaca un porcentaje bajo de resultados calificados dos veces, consideramos esto como un indicio de que si el estudiante observa un resultado que ya fue calificado tiende a analizar otros resultados que no hayan sido calificados.

Teniendo en cuenta la opinión de los estudiantes, aproximadamente el 95% de los mismos considera amigable al metabuscador. El 100% considera agradable como están presentados los resultados, priorizando aquellos con mayor calificación. El 84% de los

estudiantes consideran que el metabuscador es fácil de usar y el 89% no necesita de ayuda de otra persona para utilizarlo. Además, el 74%, considera que se aprende rápidamente la utilización del metabuscador, y el 74% de ellos se sintió confiado al manejarlo. El 90% de los encuestados no encuentra complejo el metabuscador. El 79% de los estudiantes considera que con el metabuscador es más rápido para comunicar los resultados importantes a los compañeros del grupo de trabajo (en comparación a si se utilizan otros medios como ser WhatsApp, Redes Sociales, etc.). El 87% de los encuestados considera que observar la calificación y/o comentario de sus compañeros lo incentiva a buscar otros resultados. De manera general, el 95% de los estudiantes considera que la sección ayuda y sección contacto contienen la información necesaria, y que los resultados de June fueron de utilidad para realizar su actividad académica. Por otra parte, resultaron de interés algunos comentarios respecto a la interfaz gráfica: un estudiante manifestó que en lugar de utilizar estrellas para calificar, prefieren una notación numérica; tres estudiantes consideran que es mejor realizar la calificación e ingreso de un comentario mediante ventanas emergentes, ya que el espacio utilizado para esto es relativamente grande; y uno de ellos declaró que sería útil incorporar "Google Académico" como mejora para el metabuscador.

Para aseverar los resultados de la experiencia, también se tuvieron en cuenta las opiniones de los docentes. El 100% de ellos considera que June es fácil de usar y útil para trabajos en grupo que incluyan tareas de búsqueda en la web, además consideran que el hecho de que un estudiante califique y comente, así como también que pueda observar las calificaciones y comentarios realizados por sus compañeros, impacta en su motivación y/o productividad. Todos ellos (100%) están de acuerdo en afirmar que los documentos devueltos por el metabuscador contribuyeron a mejorar la calidad de trabajo en los grupos y que permite a los estudiantes disponer de fuentes de información en cantidad apropiada. El 66.7% de los docentes opina que no es necesario considerar alguna mejora en el metabuscador, mientras que el 33.3% manifestó que podría ser útil incorporar algún otro motor de búsqueda utilizado por los estudiantes.

Lo manifestado en los párrafos previos respecto a las opiniones de estudiantes y docentes nos permite afirmar que el metabuscador June es una herramienta amigable, fácil de usar y eficiente, para grupos que deban realizar búsquedas de información en la web.

También es necesario destacar que, por un lado, cuando un estudiante encuentra varios resultados que son de su interés, los medios que puede utilizar para comunicárselos a

sus compañeros de grupo suelen ser redes sociales, mensajes de texto, llamada telefónica a través de teléfonos fijos o móviles, foros o chats que comparta con su grupo, y si a esto se le agregan los métodos empleados para tomar esos resultados, como ser a través de una foto a la pantalla de la PC para captar las direcciones web, o escribirlas en un archivo de texto, o escribirlas directamente en el chat, red social, foro, o mensaje de texto, esto genera tiempo de comunicación que debe sumarse al tiempo empleado para el proceso y análisis de búsqueda de información en la web. Por otro lado, el hecho de que los estudiantes integrantes de un grupo estén buscando individualmente y asincrónicamente información en la web para realizar una misma actividad, limitados en tiempo por el plazo para realizar la actividad establecido por el profesor, muy posiblemente lleve a que los resultados que consideren de interés sean en gran parte los mismos entre los integrantes. Esto da una idea de esfuerzo y tiempo desperdiciado si se tiene en cuenta que cada uno de ellos tiene que ingresar al sitio para analizar si tiene sentido la información del sitio.

Mediante la utilización del metabuscador June con su algoritmo de rankeo, partiendo de la posibilidad que tiene un grupo de estudiantes de calificar y comentar los resultados, como también de visualizar los resultados de búsqueda ordenados por su utilidad (resultados con mayor utilidad por encima de aquellos con menor utilidad), se ha favorecido a la comunicación entre los estudiantes respecto a indicar cuales son los resultados de búsqueda que pueden o no aportar información de interés, lo que indirectamente contribuye a disminuir el tiempo de comunicación que debería sumarse al tiempo empleado para el proceso y análisis de búsqueda de información en la web, como también contribuye a disminuir el tiempo y el esfuerzo invertido en análisis innecesarios sobre resultados que no son útiles. Son los estudiantes quienes a lo largo de la experiencia tuvieron siempre la posibilidad de contribuir al proceso de búsqueda de información a través de sus calificaciones y opiniones sobre los resultados de búsqueda, tal es el caso que un estudiante puede decidir analizar un resultado útil que ya posea una o más calificaciones, pues puede no estar de acuerdo con la calificación y/o comentario de sus compañeros ya que pueden ocurrir discrepancias entre sus calificaciones y/o comentarios, ya sea porque no confía en la calificación o interpretación de su compañero o en la fuente del resultado (su dirección web).

CONCLUSIONES

Muchas de las actividades que llevan a cabo grupos de estudiantes requieren que sus integrantes realicen individualmente búsquedas en la web para obtener información. A pesar de que muchos de los resultados obtenidos por cada uno de ellos se repiten, cada uno de esos resultados debe ser analizado para determinar su utilidad. Si se multiplica este proceso por la cantidad de individuos que conforman el grupo, se observa que puede producirse una considerable pérdida de tiempo y esfuerzo.

Dado el problema expuesto, se desarrolló un metabuscador basado en agentes como herramienta web de búsqueda colaborativa que indica qué integrante analizó un determinado resultado, y que además permite asignar una valoración personal a cada resultado e incluir un comentario. El metabuscador ordena y muestra los resultados obtenidos luego de disparadas las búsquedas individuales, considerando la valoración grupal de cada resultado, calculada a partir del promedio de las valoraciones individuales asignadas por cada integrante sobre dicho resultado. Además, genera un historial con los resultados de búsqueda calificados por el grupo, también ordenados por la valoración grupal. El metabuscador creado cuenta con dos agentes de software: un Agente de Consulta y un Agente de Actualización. La tarea de ordenar los resultados de acuerdo con las calificaciones la realiza el Agente de Consulta a partir de las calificaciones generales, calculadas por el Agente de Actualización el cual promedia las calificaciones individuales. El buen desempeño del metabuscador propuesto en el presente trabajo fue comprobado a través de experimentación con estudiantes universitarios. También se recabaron y analizaron las opiniones de los estudiantes y docentes involucrados en las experiencias. Los resultados obtenidos permiten afirmar que se alcanzaron los objetivos planteados para la realización de este trabajo: propiciar búsquedas colaborativas eficientes, y favorecer la interacción entre los integrantes de grupos de estudiantes colaborativos.

A continuación se describen las principales contribuciones del trabajo, se enuncian las limitaciones encontradas, las publicaciones generadas, y se finaliza indicando algunas líneas futuras de trabajo.

Contribuciones

La realización del presente trabajo permitió efectuar las contribuciones enunciadas a continuación:

- Se definió un agente de software (Agente de Consulta) capaz de recuperar resultados que fueron valorados y/o comentados por los usuarios, y rankearlos utilizando un algoritmo de rankeo.
- Se definió un agente de software (Agente de Actualización) capaz de actualizar la base de datos con nuevos resultados valorados y/o comentados, y de computar una valoración grupal para cada resultado, promediando las valoraciones individuales que cada uno haya recibido.
- Se definió un algoritmo de rankeo, que implementa un método de ordenamiento propio Java (Método TimSort), para rankear los resultados de búsqueda utilizando la valoración grupal.
- Se desarrolló un metabuscador con una arquitectura básica modificada para incluir los agentes de software, capaz de facilitar al usuario el ingreso de datos de búsqueda como también el ingreso de calificaciones a los resultados permitiéndole agregar un comentario; también es capaz de recuperar y procesar todos los resultados de los distintos buscadores, de mostrar al usuario los resultados de búsquedas rankeados por la valoración grupal, indicando quienes los calificaron, sus calificaciones y comentarios si los hubiese; y por ultimo también es capaz de facilitar al grupo un historial con los resultados de búsqueda que fueron calificados y/o comentados, ordenados por la valoración grupal.
- Se aplicó la Técnica de Web Scraping (Técnica de Escarbado de Páginas Web) para la consulta a los distintos motores de búsqueda y recuperación de los resultados de búsqueda devueltos por ellos.

Limitaciones

En la experiencia realizada con los estudiantes, el metabuscador desarrollado se mostró estable, con buen rendimiento, sin interrupciones ni impedimentos al momento de ser utilizado. De esta manera, los estudiantes pudieron llevar a cabo sus actividades sin inconveniente alguno durante la dinámica de trabajo. Sin embargo, al tratarse de un prototipo no se han considerados algunos filtros que suelen ser de mucha utilidad al momento de hacer más precisa una búsqueda, por ejemplo filtros que permitan hacer búsqueda solo de imágenes, por fecha, por región. etc.

Líneas Futuras

A partir de las limitaciones anteriormente nombradas, en un futuro próximo podría estudiarse el desempeño del metabuscador incorporando a su interfaz de usuario filtros para búsqueda por imágenes, o por fecha, o por región. Esto llevaría a que las búsquedas sean más precisas a criterio de quien lo usa dándole la posibilidad de elección. También puede considerarse la incorporación de otros motores de búsqueda, como "Google Académico", y la utilización de ventanas emergentes para realizar la calificación e ingresar un comentario por cada resultado. Por otro lado, se podría incorporar una interfaz de usuario de administración que facilite la creación y gestión de grupos como también de los enlaces de acceso asignados a los integrantes de cada uno de ellos.

En un futuro un poco más lejano podrían incluirse perfiles de usuario, dando la posibilidad, a quien utilice el metabuscador de tener su propia configuración, además de guardar permanentemente los resultados para futuras revisiones.

Publicaciones

La realización de este trabajo permitió realizar dos envíos a congresos. El artículo "Metabuscador basado en agentes para grupos de estudiantes colaborativos" fue presentado en el Workshop de Investigadores en Ciencias de la Computación (WICC 2016), realizado en la ciudad de Paraná, Entre Ríos (Argentina), los días 14 y 15 de abril del 2016. El artículo titulado "JUNE: Metabuscador basado en agentes para grupos de estudiantes colaborativos" fue aprobado como full paper en INTERACTION 2017, a realizarse en Cancún, (México) en el mes de septiembre.

REFERENCIAS

Aguilar Bernabé, Roberto Jesús y Arroyo Flores, Elder Anderson . 2013. Diseño e Implementación de una plataforma web, aplicada a un aula digital empleando la metodología OOHDM. [En línea] 2013. [Citado el: 5 de Septiembre de 2016.] http://www.inf.unitru.edu.pe/revista/8.pdf.

Anguiano Torres, Susana, López Arreguín, María Josefina y Martínez Negrete, Eduardo. 2013. El trabajo colaborativo en la educación virtual: estrategias aplicadas en la Universidad Virtual del Estado de Guanajuato. [En línea] Diciembre de 2013. [Citado el: 3 de Octubre de 2016.] http://bdistancia.ecoesad.org.mx/?articulo=el-trabajo-colaborativo-en-la-educacion-virtual-estrategias-aplicadas-en-la-universidad-virtual-del-estado-de-guanajuato. 2007-4751.

Arenas Delgado, Gloria , y otros. 2011. Apache Software Foundation. [En línea] 6 de Enero de 2011. [Citado el: 20 de Junio de 2016.] https://apachefoundation.wikispaces.com/Apache+Tomcat.

Artaza Álvarez, Haydeé Maribel. 2010. Estudio de una metodología para el desarrollo de aplicaciones hipermedia educativas accesibles. [En línea] 2010. [Citado el: 15 de Septiembre de 2016.] http://e-archivo.uc3m.es/handle/10016/9589.

Banerjee, Ritu. 2014. Website Scraping. [En línea] Abril de 2014. [Citado el: 5 de Julio de 2016.] http://www.happiestminds.com/whitepapers/website-scraping.pdf.

Barrios, Alejandro Franco. 2009. Manual de Jade para Principiantes. [En línea] 2009. [Citado el: 23 de Agosto de 2016.] http://biblioteca.unitecnologica.edu.co/notas/tesis/0053723.pdf.

Bellver González, Miguel. 2015. Coordinación y comunicación entre agentes físicos basados en la plataforma SPADE. [En línea] 2015 de Septiembre de 2015. [Citado el: 15 de Diciembre de 2016.] https://riunet.upv.es/bitstream/handle/10251/56410/Memoria.pdf?sequence=1.

Betancourt, Julián. 2008. *Atmósferas creativas 2: Rompiendo candados mentales, segunda edición.* México, D.F. : El Manual Moderno S.A. de C.V., 2008. 978-970-729-320-5.

Bökemeier, Jost y Koerber , Jon. n.d.. PHP / JAVA Bridge. [En línea] n.d. [Citado el: 30 de Julio de 2016.] http://php-java-bridge.sourceforge.net/pjb/.

Bravo Marquez, Felipe José. 2010. Diseño e implementación de un metabuscador de párrafos para la recuperación de documentos similares en la web. [En línea] Octubre de 2010. [Citado el: 1 de Septiembre de 2016.] http://www.cs.waikato.ac.nz/~fjb11/memoria.pdf.

Cascales Martínez , Antonia , Martínez Segura, María José y Gomariz Vicente , María Ángeles . 2016. Competencia tecnológica y trabajo colaborativo en las prácticas curriculares del Grado en Pedagogía en la Universidad de Murcia. [En línea] 2016. [Citado el: 27 de Agosto de 2016.] http://reined.webs.uvigo.es/ojs/index.php/reined/article/view/1135/371#page=36.

Castañeda Quintero , Linda . 2007. Herramientas sincrónicas y cuasi-sincrónicas para la comunicación educativa. *En PRENDES ESPINOSA, M. P. Herramientas Telemáticas Para La Enseñanza Universitaria En El Marco Del Espacio Europeo De Educación Superior.* [En línea] 2007. [Citado el: 5 de Diciembre de 2016.] http://ocw.um.es/gat/contenidos/mpazherramientas/documentos/videoymsn.pdf. ISBN: 978-84-611-7947-3.

Criollo Guatapi, Mayra Carmita y Serrano Fuel, Jose Guillermo. 2007. Diseño de un Portal Web Dinámico con Acceso a Datos para la agencia de viajes "Absolut Joy Expeditions Travel Agency". [En línea] Marzo de 2007. [Citado el: 24 de 09 de 2016.] http://bibdigital.epn.edu.ec/bitstream/15000/1306/1/CD-0618.pdf.

De Benito, Barbara. 1999. Taller: Redes y trabajo colaborativo entre profesores. [En línea] 1999. [Citado el: 03 de Julio de 2016.] http://tecnologiaedu.us.es/nweb/htm/pdf/gte43.pdf.

Dixon, Angela y College Swansea, Gower . 2011. Marcadores Sociales. [En línea] 2011. [Citado el: 2 de Octubre de 2016.] http://www.svea-project.eu/www.svea-project.eu/fileadmin/_svea/downloads/Social_Bookmarking_03.pdf.

Fernández, Oscar Belmonte. 2005. Introducción al lenguaje de programación Java: Una guía básica. [En línea] 6 de Junio de 2005. [Citado el: 1 de Septiembre de 2016.] http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf.

Franklin, Stan y Graesser, Art . 1996. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. [En línea] 1996. [Citado el: 7 de Diciembre de 2016.] http://robotics.cs.tamu.edu/dshell/cs631/papers/franklingraesser96agents.pdf.

García, Lourdes y Fernandez, Salmones. 2008. Satisfacción y Eficacia: Un estudio con grupos de trabajo en un entorno productivo. [En línea] 2008. [Citado el: 1 de Julio de 2016.] http://e-spacio.uned.es/fez/eserv/tesisuned:Psicologia-Lgarcia/Documento.pdf.

González, Carlos Mario . n.d.. Motores de búsqueda o Buscadores. [En línea] n.d. http://aprendeenlinea.udea.edu.co/lms/moodle/pluginfile.php/38229/mod_resource/content/0/Modulo2/Buscadores.pdf.

Guerra Guerra, Jorge. 2010. Sistemas Distribuidos. [En línea] Febrero de 2010. [Citado el: 4 de Septiembre de 2016.] http://jguerra91.wixsite.com/sistdist20141.

Hernández Herrero, Cristina. 2014. Aplicación de Técnicas de Web Scraping al BOCyL. [En línea] 2014. [Citado el: 15 de Mayo de 2016.] https://uvadoc.uva.es/handle/10324/5794.

Ibañez, Fabricio Carlos. 2014. Desarrollo de un Portal Web con Tecnología Apache/Mysql. [En línea] 24 de Julio de 2014. [Citado el: 1 de Septiembre de 2016.] https://riunet.upv.es/bitstream/handle/10251/46575/Ibanez Raad, Fabricio Carlos.pdf?sequence=1.

Johnson, David W., Johnson, Roger T. y Holubec, Edythe J. . 1999. *El aprendizaje cooperativo.* Buenos Aires : Paidós SAICF, 1999.

Junta de Castilla y León. 2012. Herramientas para el trabajo colaborativo. [En línea] 2012. [Citado el: 15 de Julio de 2016.] https://universoabierto.com/2016/01/04/herramientas-para-el-trabajo-colaborativo/.

Lafuente, Gustavo Hernán. 2001. Motores de Búsqueda en Internet. [En línea] 10 de Noviembre de 2001. [Citado el: 3 de Octubre de 2016.] http://www.unlu.edu.ar/~tyr/tyr/TYR-motor/lafuente-motor.pdf.

Lambertucci, David . 2001. Motores de búsqueda en internet. [En línea] 10 de Noviembre de 2001. [Citado el: 2 de Septiembre de 2016.] http://www.unlu.edu.ar/~tyr/tyr/TYR-motor/lambertucci-motor.pdf.

Lázaro Molina, Jorge. 2006. Apuntes Metodológicos de Desarrollo orientado a agentes: aplicación a una agencia de viajes. [En línea] Julio de 2006. [Citado el: 2 de Septiembre de 2016.] http://oa.upm.es/948/.

Maglione, Carla y Varlotta , Nicolás . n.d. Investigación, gestión y búsqueda de información en Internet. [En línea] n.d. [Citado el: 5 de Septiembre de 2016.] http://bibliotecadigital.educ.ar/uploads/contents/investigacion0.pdf.

Maity, Sheuli. 2015. Meta Search Engine using Multi-Objective Partial Rank Aggregation: Application in Ranking WebPages. [En línea] 9 de Julio de 2015. [Citado el: 7 de Junio de 2016.] http://ijirset.com/upload/2015/ncetas15/16_P_ID_21.pdf.

Mantilla Yánez , Daniel Augusto y Santos Castillo, Ana Cristina. 2007. Desarrollo de un portal web para el ingreso y consultas de notas para el colegio nacional mixto “María Angélica Carrilo de Mata Martínez”. [En línea] Marzo de 2007. [Citado el: 8 de Junio de 2016.] http://bibdigital.epn.edu.ec/handle/15000/1359.

Marchetti, Tulio José y García, Alejandro Javier . 2003. Plataformas para Desarrollo de Sistemas Multiagente. Un Análisis Comparativo. [En línea] Mayo de 2003. [Citado el: 3 de Junio de 2016.] http://sedici.unlp.edu.ar/handle/10915/21445.

Mifsuf Talón, Elvira. 2012. Apache. [En línea] 2012. [Citado el: 3 de Agosto de 2016.] https://sede.educacion.gob.es/publiventa/apache/ensenanza-informatica/15824.

Miriada X. n.d.. Las mejores herramientas. [En línea] n.d. [Citado el: 6 de Junio de 2016.] https://miriadax.net/documents/21772177/21772203/Las+mejores+herramientas.pdf.

Muñoz, Norman , y otros. 2010. Uso de la metodología GAIA para modelar el comprotamiento de personajes en un juego de estrategia en tiempo real. [En línea] Junio de 2010. [Citado el: 4 de Septiembre de 2016.] http://www.scielo.org.co/pdf/rfiua/n53/n53a20.pdf.

Oracle - API Specification. Java™ Platform, Standard Edition 6 - API Specification. [En línea] Oracle. [Citado el: 15 de Febrero de 2017.] https://docs.oracle.com/javase/6/docs/api/java/util/Collections.html#sort(java.util.List).

Oracle. n.d.. Java. [En línea] n.d. [Citado el: 29 de Agosto de 2016.] https://www.java.com/es/download/faq/whatis_java.xml.

Peñaranda Cebrián, Cristian. 2016. Un sistema multiagente para mejorar la localización de un robot NAO. [En línea] 2016. [Citado el: 1 de Octubre de 2016.] https://riunet.upv.es/bitstream/handle/10251/68705/Pe%C3%B1aranda%20-%20Un%20sistema%20multiagente%20para%20mejorar%20la%20localizaci%C3%B3n%20de%20un%20robot%20NAO.pdf?sequence=3.

PHP Group. n.d.. PHP. [En línea] n.d. [Citado el: 4 de Septiembre de 2016.] http://php.net/manual/es/intro-whatis.php.

Requena Moreno, Francisco J. 2007. Caracterización de Juegos Basados en Agentes Móviles. Diseño e Implementación de un caso práctico y su incorporación en el programa docente de una asignatura de redes de computadores. [En línea] Junio de 2007. [Citado el: 15 de Julio de 2016.] https://ddd.uab.cat/pub/trerecpro/2007/hdl_2072_8936/PFCRequenaMoreno.pdf.

Rodríguez García, Raquel y Zayas de Diego , María Luisa. 2009. Buscadores Web y Redes Semánticas. [En línea] Agosto de 2009. [Citado el: 15 de Julio de 2016.] http://www.it.uc3m.es/~jvillena/irc/practicas/09-10/08mem.

Russell, Stuart y Norvig, Peter . 2008. *Inteligencia Artificial: Un Enfoque Moderno.* Madrid, España : PEARSON. Prentice Hall, 2008. 978-84-205-4003-0.

San Martín, Matías I. 2012. Un servicio de movilidad de agentes de software para JADE basado en OSGi. [En línea] 2012. [Citado el: 4 de Noviembre de 2016.]

Scagnoli, Norma I. 2005. Estrategias para Motivas el Aprendizaje Colaborativo en Cursos a Distancia. [En línea] 2005. [Citado el: 27 de Mayo de 2016.] https://www.ideals.illinois.edu/bitstream/handle/2142/10681/aprendizaje-colaborativo-scagnoli.pdf?sequence=4.

Schwabe, Daniel y Rossi, Gustavo . 1998. Developing Hypermedia Applications using OOHDM. [En línea] Agosto de 1998. [Citado el: 30 de Junio de 2016.] www.inf.puc-rio.br/~schwabe/papers/ExOOHDM.pdf.gz.

Soto De Giorgis, Ricardo , Palma Muñoz, Wenceslao y Roncagliolo De La Horra., Silvana . 2002. Propuesta de un modelo navegacional para el desarrollo de aplicaciones basadas en OOHDM. [En línea] Noviembre de 2002. [Citado el: 15 de Septiembre de 2016.] https://tallerinf281.wikispaces.com/file/view/Aplicacion-OOHDM.pdf.

Stark, Natalia S. . 2003. Motores de búsqueda en internet. [En línea] 28 de Agosto de 2003. [Citado el: 5 de Diciembre de 2016.] http://www.unlu.edu.ar/~tyr/tyr/TYR-motor/stark-motor.pdf.

Telmex. 2012. buscadores o motores búsqueda. [En línea] 14 de Febrero de 2012. [Citado el: 6 de Junio de 2016.] http://www.telmexeducacion.com/proyectos/DocsDobleclic/14-Doble%20clic-Buscadores%20o%20motores%20de%20busqueda.pdf.

Torres Pombert, Ania . 2013. El uso de los buscadores en Internet. [En línea] Junio de 2013. [Citado el: 5 de Mayo de 2016.] http://eprints.rclis.org/5089/1/uso.pdf..

Vaquerizo , Belén , Renedo, Eduardo y Valero, Miguel . 2009. Aprendizaje colaborativo en grupo: Herramientas Web 2.0. [En línea] Julio de 2009. [Citado el: 5 de Mayo de 2016.] http://upcommons.upc.edu/bitstream/handle/2099/7855/p186.pdf?sequence=6&isAllowed=y.

Velasco, Sergio y Martín, Alicia. 2008. Creación de Script en Linux. [En línea] 26 de Mayo de 2008. [Citado el: 2016 de Diciembre de 13.] http://pendientedemigracion.ucm.es/info/aulasun/archivos/SCRIPTS.pdf.

Venturini, Verónica M. 2012. istema multi-agente basado en contexto, localización y reputación para dominios de inteligencia artificial. [En línea] Octubre de 2012. [Citado el: 15 de Mayo de 2016.] http://e-archivo.uc3m.es/handle/10016/16341.

W3C. 2004. ¿Qué es el Modelo de Objetos del Documento? [En línea] 7 de Abril de 2004. [Citado el: 2 de Octubre de 2016.] https://www.w3.org/2005/03/DOM3Core-es/introduccion.html.

Wooldridge, Michael , Jennigs, Nicholas R. y Kinny, David. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. [En línea] 2000. [Citado el: 29 de Mayo de 2016.] http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/jaamas2000b.pdf.

Yu, Clement, Meng, Weiyi y Liu, King-Lup. 2002. Building Efficient and Effective Metasearch Engines. [En línea] Marzo de 2002. [Citado el: 1 de Junio de 2016.] https://www.ischool.utexas.edu/~i385df04/readings/Meng(2002)-mewtasearch_engines.pdf.

ANEXO I

CODIGO Y PRUEBAS DE CAJA BLANCA

- JUNE -

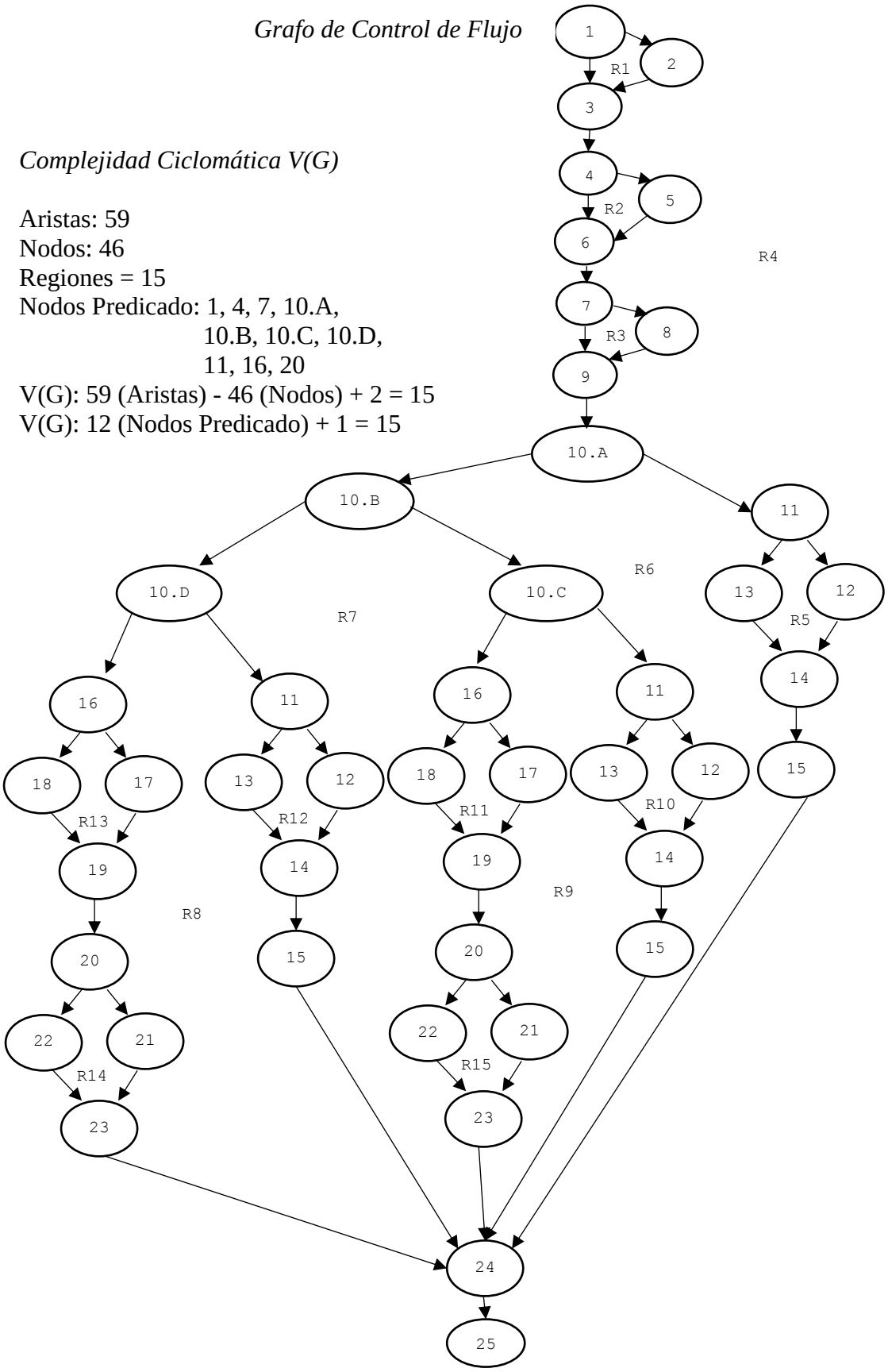
A continuación, se detallan las pruebas de caja blanca realizadas sobre las funciones principales del metabuscador.

Función Principal del Procesamiento de Datos

```
function busquedaResultados($comunicacionAgentes,$metabuscador){

    $estado = "";
1
    if(strlen($_POST['senialDeUsuario']) > 2 ){

    $paquetePHP = array(utf8_encode($_POST['tituloResultado']),
                        utf8_encode($_POST['urlResultado']),
                        utf8_encode($_POST['descripcionResultado']),
                        $_SESSION['idUsuario'],$_POST['rb'],
  2                     $_POST['comentario'],
                        $_SESSION['grupoUsuario']);
    $comunicacionAgentes->agenteActualizacion($paquetePHP);
    $estado = $comunicacionAgentes->getEstadoAgenteActualizacion();
    unset($paquetePHP);
3   }

4   if(isset($_POST['btnCalificados']) ){

  5 $_SESSION['resultadosCalificados']= 1;
6   }

7   if(isset($_POST['btnBuscar'])){

  8 $_SESSION['resultadosCalificados']= 0;

9   }

10.A
10.B
10.C
10.D
    if($_POST['subPaginas'] == '' || (strlen($_POST['btnIndice']) <= 2 &&
    $_SESSION['resultadosCalificados'] == 0) || strlen($_POST['btnCalificados']) >= 2 ){

    $pag = intval($_POST['btnIndice']);
11
    if($_SESSION['resultadosCalificados'] == 1){
        $this->setResultadosCrudos($this->resultadoVacio());
  12    $pagSig = $pag;
        $_SESSION['resultadosCalificados']= 1;

    }else{
        $metabuscador->despachadorYfusion(
               str_replace(" ","+",$_POST['keywords']),
               intval($_POST['btnIndice']));
  13    $this->setResultadosCrudos($metabuscador->getResultadoMetabuscador());
        $pagSig = $pag + 1;
        $_SESSION['resultadosCalificados']= 0;
14  }

    $_SESSION['resultadosCrudos'] = $this->getResultadosCrudos();
    $comunicacionAgentes->agenteConsulta($this->getResultadosCrudos(),
                                         $_SESSION['grupoUsuario']);
15
    $this->obtenerListados($comunicacionAgentes->getListadoAgenteConsulta(),
                           $pag,$pagSig);
    $this->setSubpagina(0);

}else{
    $opc = explode(' - ', $_POST['btnIndice']);
    $this->setSubpagina(ord($opc[1])-65);
16  $pag = intval($opc[0]);

    if($_SESSION['resultadosCalificados'] == 1){
```

```
            $pagSig = $pag;
        }else{
            $pagSig = $pag + 1;
        }
    if($estado == "ACTUALIZADO"){
            $this->setResultadosCrudos($_SESSION['resultadosCrudos']);
            $comunicacionAgentes->agenteConsulta($this->getResultadosCrudos(),
                                                 $_SESSION['grupoUsuario']);
            $this->obtenerListados($comunicacionAgentes->getListadoAgenteConsulta(),
            $pag,$pagSig);

    }else{
            $this->listados = $_SESSION['listados'];
            $this->listados->setIndice($_SESSION['indice']);
            $this->listados->setCantidadSubpaginas($_SESSION['cantidadSubpaginas']);
        }
}
        $this->listados->setPaginaSeleccionada($this->getSubpagina());
}
```

Grafo de Control de Flujo



Complejidad Ciclomática V(G)

Aristas: 59
Nodos: 46
Regiones = 15
Nodos Predicado: 1, 4, 7, 10.A,
10.B, 10.C, 10.D,
11, 16, 20
V(G): 59 (Aristas) - 46 (Nodos) + 2 = 15
V(G): 12 (Nodos Predicado) + 1 = 15

Caminos independientes

- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 11, 13, 14, 15, 24, 259
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24,25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 104.A, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 11, 13, 14, 15, 24, 259
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24,25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25

- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 5, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 11, 13, 14, 15, 24, 259
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 5, 6, 7, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24,25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 2, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 11, 13, 14, 15, 24, 259
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 11, 13, 14, 15, 24,25

- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 11, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 15, 17, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10, 12, 13, 14, 16, 17, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 12, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 9, 10.A, 10.B, 10.C, 10.D, 11, 13, 14, 15, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 6, 7, 8, 9, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 17, 19, 20, 22, 23, 24, 25
- 1, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 21, 23, 24, 25
- 1, 3, 4, 6, 7, 3, 10.A, 10.B, 10.C, 10.D, 16, 12, 19, 20, 22, 23, 24, 25

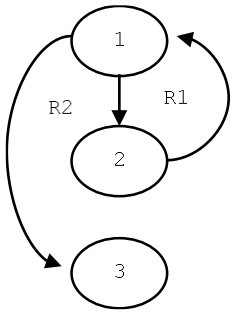
Función Principal – Despachador de Consulta y Fusión de Resultados

```
functiondespachadorYfusion ($consulta,$pagina){

   $direccion_simple_html_dom = "../".$this->getDirectorios(1)."/".
                                $this->getDirectorios(2).
                                "/simple_html_dom.php";
   $resultadoBuscador = array();                                          1

   for($i=0; $i < $this->CantBuscadoresSeleccionados(); $i++){

        $direccion_buscador = "../".$this->getDirectorios(1)."/".
                                $this->getDirectorios(2)."/".
                                $this->getBuscadoresSeleccionados($i).
                                "/mod_init.php";
        $resultadoBuscador[] = new buscador($direccion_simple_html_dom,    2
                                            $direccion_buscador,
                                            $consulta,
                                            $this->getBuscadoresSeleccionados($i),
                                            $pagina);
   }

   despachadorConsulta($resultadoBuscador);
   esperaFinalizacionConsulta($resultadoBuscador);
   destruirHilosConsulta($resultadoBuscador);

   $resultadosFusionados = fusionResultados($this->CantBuscadoresSeleccionados(),   3
                                            $resultadoBuscador);

   $this->setResultadoMetabuscador($resultadosFusionados);
   unset($resultadoBuscador);
   unset($resultadosFusionados);
}
```

Complejidad Ciclomática V(G)

Aristas: 3
Nodos: 3
Regiones = 2
Nodos Predicado: 1
V(G): 3 (Aristas) - 3 (Nodos) + 2 = 2
V(G): 1 (Nodos Predicado) + 1 = 2

Grafo de Control de Flujo



Caminos independientes

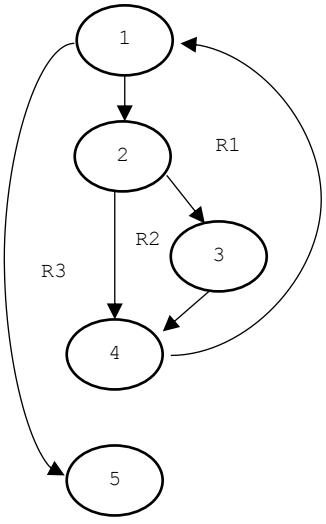
a) 1, 2, 1, 3
b) 1, 3

```
function filtrarOrdenarResultados($resultadosTotalesSinFiltrar){

   $filtrados = array();
   $filtradoOrdenado = array();
   $j = 0;                                                              ] 1
   for($i = 0; $i < count($resultadosTotalesSinFiltrar);$i++){          ]

        if($resultadosTotalesSinFiltrar[$i][1] != ""){    ] 2

            $filtrados[$j] = agregarFiltrado($resultadosTotalesSinFiltrar[$i]);
            eliminarLineasRepetidas($resultadosTotalesSinFiltrar,$filtrados, $j);   ] 3
            $j = $j + 1;

        } ] 4

   }

   $filtradoOrdenado = ordenarFiltrados($filtrados);                    ]
   unset($filtrados);                                                   ] 5
   return $filtradoOrdenado;                                            ]
}
```

Complejidad CiclomáticaV(G)

Aristas: 6
Nodos: 5
Regiones = 3
Nodos Predicado: 1, 2
V(G): 6 (Aristas) - 5 (Nodos) + 2 = 3
V(G): 2 (Nodos Predicado) + 1 = 3

Grafo de Control de Flujo

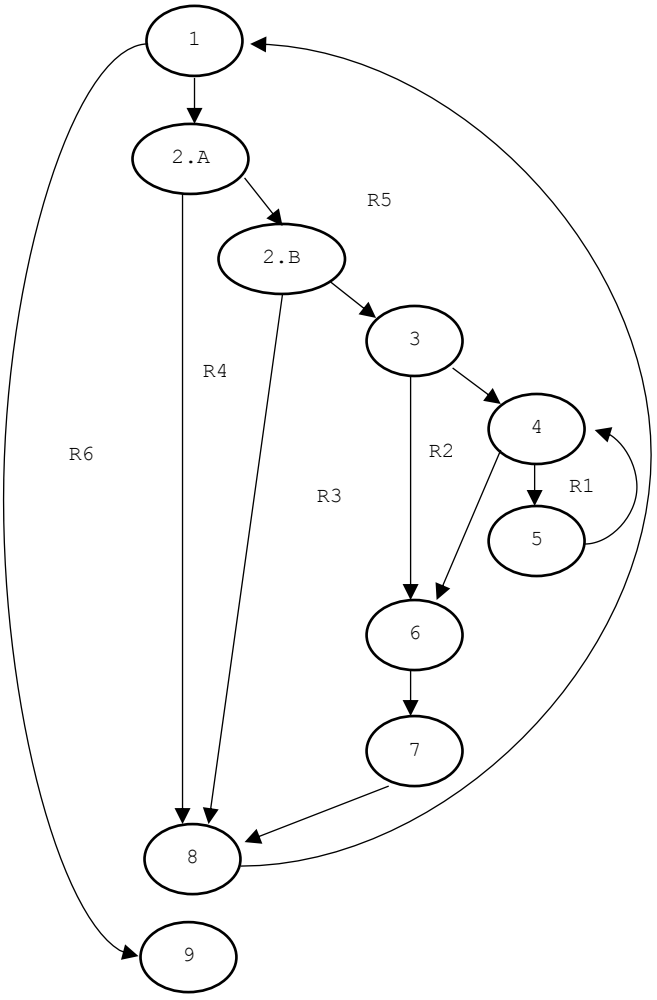


Caminos independientes

a) 1, 5, 3, 4, 1, 5
b) 1, 5
c) 1, 2, 4, 1, 5

```
function eliminarLineasRepetidas(&$resPuros,&$filtrado,$lineaf){

   for($i = 0; $i < count($resPuros);$i++){ ] 1

      if(($resPuros[$i][2] == $filtrado[$lineaf][2]) && ($resPuros[$i][0] != "")){ ] 2.A
                                                                                    2.B
       if($resPuros[$i][4] < $filtrado[$lineaf][4]){ ] 3

              for($j = 0; $j < 5;$j++){ ] 4

                     $filtrado[$lineaf][$j] = $resPuros[$i][$j]; ] 5
              }
       } ] 6


       $resPuros[$i][0] = "";                                          ]
       $resPuros[$i][1] = "";                                          ]
       $resPuros[$i][2] = "";                                          ] 7
       $resPuros[$i][3] = "";                                          ]
       $filtrado[$lineaf][5] = $filtrado[$lineaf][5] + 1;              ]

      } ] 8

   } ] 9

}
```

Grafo de Control de Flujo



Complejidad CiclomáticaV(G)

Aristas: 14
Nodos: 10
Regiones = 6
Nodos Predicado: 1, 2.A, 2.B, 3, 4
V(G): 14 (Aristas) - 10 (Nodos) + 2 = 6
V(G): 5 (Nodos Predicado) + 1 = 6

Caminos independientes

a) 1, 2.A, 2.B, 3, 4, 5, 4, 6, 7, 8, 1, 9
b) 1, 2.A, 2.B, 3, 4, 6, 7, 8, 1, 9
c) 1, 2.A, 2.B, 3, 6, 7, 8, 1, 9
d) 1, 2.A, 2.B, 8, 1, 9
e) 1, 2.A, 8, 1, 9
f) 1, 9

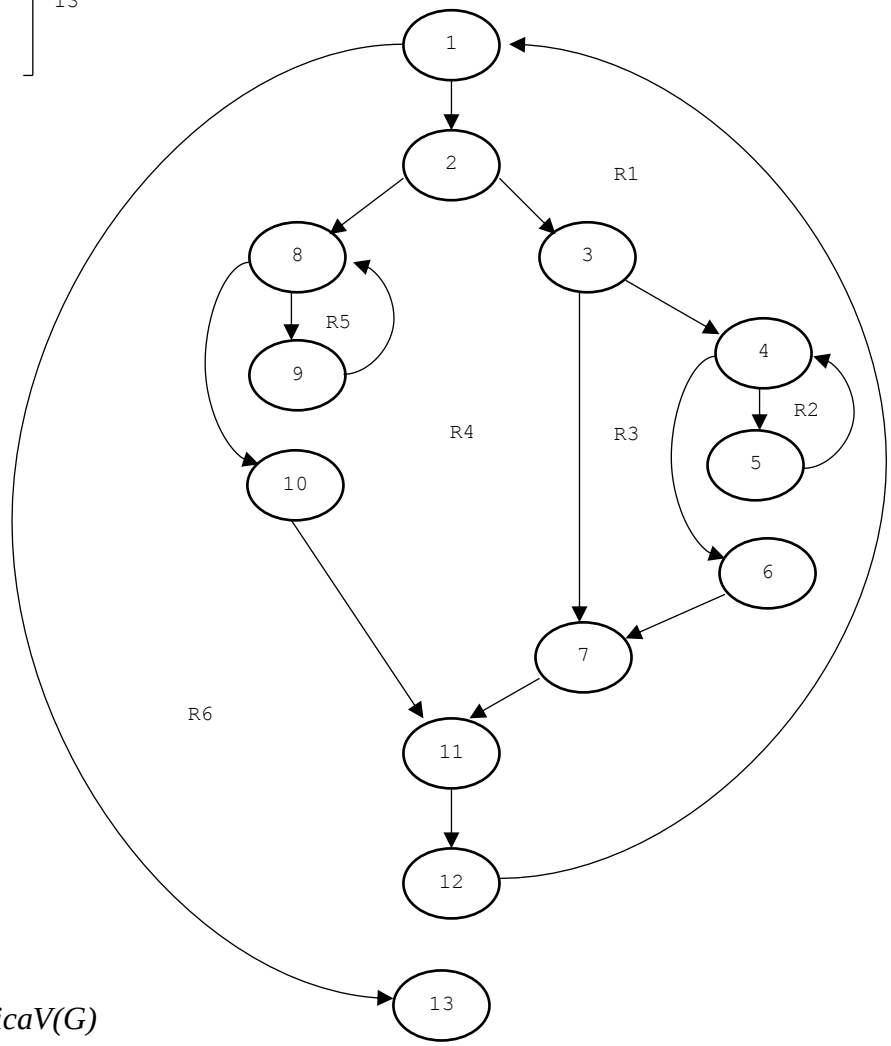
```
function ordenarFiltrados($resultados){

$resultadosAux = ordenarDESC($resultados,5);
$ordenado = array();
$ordenadoAux = array();
$ordenadoFinal = array();
$i_total = 0;
$i_ord = 0;
$i = 0;
$tamanio = count($resultadosAux);

while($i < $tamanio){

   $ordenado[$i_ord] = agregarFiltrado($resultadosAux[$i]);
   $i_ord = $i_ord + 1;

   if(($i + 1) != $tamanio){

        if($resultadosAux[$i][5] != $resultadosAux[$i+1][5] ){

                $ordenadoAux = ordenarASC($ordenado,4);

                for($k = 0; $k < count($ordenadoAux); $k++){

                        $ordenadoFinal[$i_total][0] = $ordenadoAux[$k][1];
                        $ordenadoFinal[$i_total][1] = $ordenadoAux[$k][2];
                        $ordenadoFinal[$i_total][2] = $ordenadoAux[$k][3];
                        $ordenadoFinal[$i_total][3] = null;
                        $i_total = $i_total + 1;
                }
                $i_ord = 0;
                unset($ordenado);
                $ordenado = array();
        }

   }else{
        $ordenadoAux = ordenarASC($ordenado,4);

                for($k = 0; $k < count($ordenadoAux); $k++){
                        $ordenadoFinal[$i_total][0] = $ordenadoAux[$k][1];
                        $ordenadoFinal[$i_total][1] = $ordenadoAux[$k][2];
                        $ordenadoFinal[$i_total][2] = $ordenadoAux[$k][3];
                        $ordenadoFinal[$i_total][3] = null;
                        $i_total = $i_total + 1;
                }
                $i_ord = 0;
                unset($ordenado);
                $ordenado = array();
   }
   $i = $i + 1;
}
```

1 2 3 4 5 6 7 8 9 10 11 12

```
unset($resultadosAux);
unset($ordenado );
unset($ordenadoAux);
return $ordenadoFinal;
}
```

13

Grafo de Control de Flujo



Complejidad CilomáticaV(G)

Aristas: 17
Nodos: 13
Regiones:6
Nodos Predicado: 1, 2, 3, 4, 8
V(G): 17 (Aristas) - 13 (Nodos) + 2 = 6
V(G): 5 (Nodos Predicado) + 1 = 6

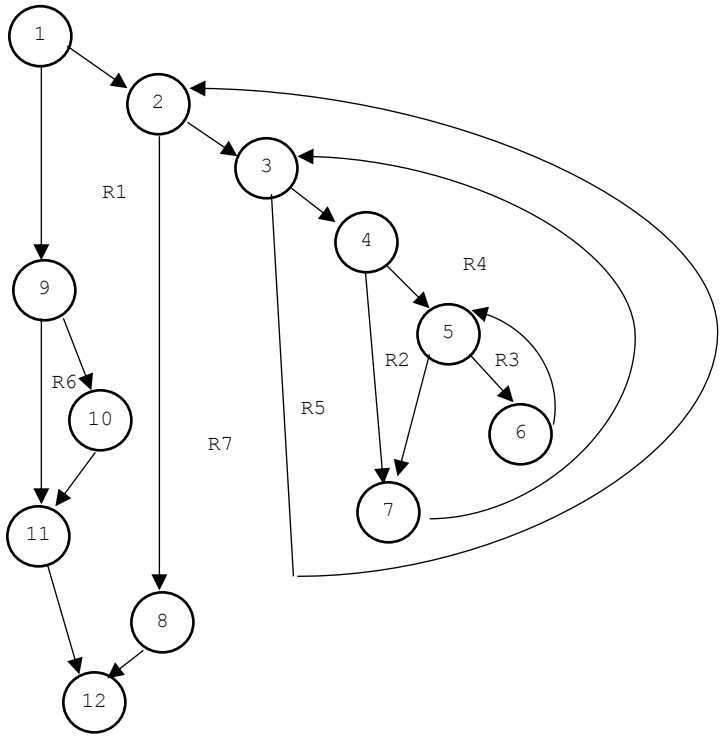
Caminos independientes

a) 1, 2, 3, 4, 5, 4, 6, 7, 11, 12, 1, 13
b) 1, 2, 3, 4, 6, 7, 11, 12, 1, 13
c) 1, 2, 3, 7, 11, 12, 1, 13
d) 1, 2, 8, 9, 10, 11, 12, 1, 13
e) 1, 2, 8, 10, 11, 12, 1, 13
f) 1, 13

Función de Agente de Consulta

```
public Listado comportamientoUno(Listado listadoSinProcesar, int cantidadDeResultados){
1   Listado listado = listadoSinProcesar;
    System.out.println("Comportamiento 1");
    try{

2       final conection bd = new conection();
        String urlBD = "";
        String grupoListado = new String(listado.getGrupo());
        System.out.println("Defino el conjunto de resultados asociados a mi grupo");
        String Query = "SELECT * FROM resultadoporgrupo INNER JOIN resultado ON
        (resultadoporgrupo.idResultado = resultado.idResultado) WHERE
        resultadoporgrupo.idGrupo='"+grupoListado+"'";
        Statement st = conection.getConexion().createStatement();
        java.sql.ResultSet resultSet;
        resultSet = st.executeQuery(Query);
        while(resultSet.next()){

3           urlBD = resultSet.getString("url");
            String urlListado = new String("");
            String calificacionGeneralListado = new String("");
            for(int i = 0; i<cantidadDeResultados; i++){

4               urlListado = listado.obtenerResultado(i).getUrl();
                calificacionGeneralListado = resultSet.getString("calificacionGeneral");
                if(urlBD.equals(urlListado)){

5                   System.out.println("Como la url se encuentra en la BD (en mi grupo),
                    entonces recupero los datos de estudiantes asociados a dicha url");
                    String Query2 = "SELECT * FROM calificacionxestudiante "
                    + "INNER JOIN estudiante ON
                    (calificacionxestudiante.legajo=estudiante.legajo) "
                    + "INNER JOIN resultado ON
                    (calificacionxestudiante.idResultado=resultado.idResultado)"
                    + "INNER JOIN resultadoporgrupo ON
                    (resultadoporgrupo.idResultado=calificacionxestudiante.idResultado AND
                    calificacionxestudiante.idGrupo=resultadoporgrupo.idGrupo)"
                    + "WHERE resultado.url= '"+urlListado+"' AND
                    calificacionxestudiante.idGrupo='"+grupoListado+"'";
                    Statement st2 = conection.getConexion().createStatement();
                    java.sql.ResultSet resultSet2 = st2.executeQuery(Query2);
                    listado.obtenerResultado(i).agregarCalificacionGeneral(calificacionGen
                    eralListado);
                    listado.obtenerResultado(i).iniciarEstudiantes();
                    while(resultSet2.next()){

6                       listado.obtenerResultado(i).agregarEstudiante(resultSet2.getString
                        ("legajo"),
                        resultSet2.getString("apellido"),
                        resultSet2.getString("nombre"),
                        resultSet2.getString("calificacion"),
                        resultSet2.getString("comentario"));
                    }
7               }
            }
        }

8       listado.ordenarListado();
        conection.Close()

9   } catch (SQLException ex){
10      System.out.println(ex);
11  }

12  return listado;}
```

Grafo de Flujo de Control



Complejidad Ciclomática V(G)

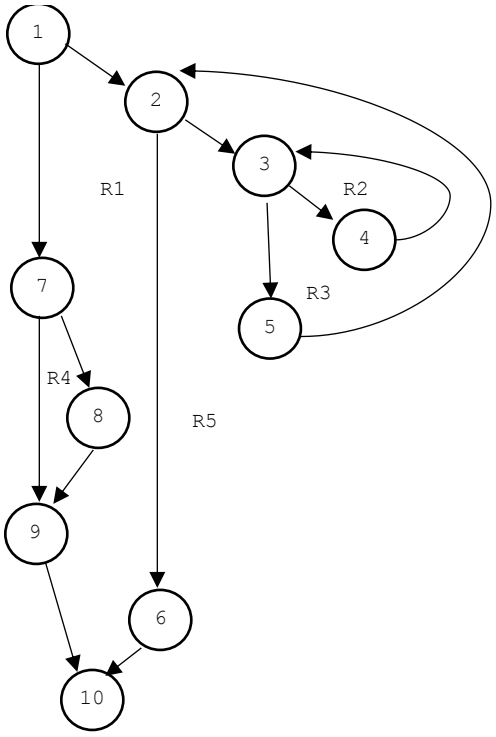
Aristas: 17
Nodos: 12
Regiones: 7
Nodos Predicado: 1,2,3,4,5,9
V(G): 17 (Aristas) - 12 (Nodos) + 2 = 7
V(G): 6 (Nodos Predicado) + 1 = 7

Caminos independientes

a) 1,2,3,4,5,6,5,7,3,2,8,12
b) 1,2,3,4,5,7,3,2,8,12
c) 1,2,3,4,7,3,2,8,12
d) 1,2,3,2,8,12
e) 1,2,8,12
f) 1,9,10,11,12
g) 1,9,11,12

```
public Listado comportamientoDos(Listado listadoSinProcesar, int cantidadDeResultados){
    1  Listado listado = listadoSinProcesar;
       System.out.println("Comportamiento 2");

       try{

          2  final conection bd = new conection();
             String urlBD = "";
             String grupoListado = new String(listado.getGrupo());
             listado.vaciarListadoDeResultados();
             System.out.println("Defino el conjunto de resultados asociados a mi grupo");
             String Query = "SELECT * FROM resultadoporgrupo INNER JOIN resultado ON
             (resultadoporgrupo.idResultado = resultado.idResultado) WHERE
             resultadoporgrupo.idGrupo='"+grupoListado+"'";
             Statement st = conection.getConexion().createStatement();
             java.sql.ResultSet resultSet;
             resultSet = st.executeQuery(Query);
             while(resultSet.next()){

                3  Resultado resultado = new Resultado();
                   urlBD = resultSet.getString("url");
                   resultado.setTitulo(resultSet.getString("titulo"));
                   resultado.agregarUrl(resultSet.getString("url"));
                   resultado.setDescripcion(resultSet.getString("descripcion"));
                   resultado.agregarCalificacionGeneral(resultSet.getString("calificacionGenera
                   l"));
                   System.out.println("Como la url se encuentra en la BD (en mi grupo),
                   entonces recupero los datos de estudiantes asociados a dicha url");
                   String Query2 = "SELECT * FROM calificacionxestudiante INNER JOIN estudiante
                   ON (calificacionxestudiante.legajo=estudiante.legajo) INNER JOIN resultado
                   ON (calificacionxestudiante.idResultado=resultado.idResultado)INNER JOIN
                   resultadoporgrupo ON
                   (resultadoporgrupo.idResultado=calificacionxestudiante.idResultado AND
                   calificacionxestudiante.idGrupo=resultadoporgrupo.idGrupo)WHERE
                   resultado.url= '"+urlBD+"' AND
                   calificacionxestudiante.idGrupo='"+grupoListado+"'";
                   Statement st2 = conection.getConexion().createStatement();
                   java.sql.ResultSet resultSet2 = st2.executeQuery(Query2);
                   while(resultSet2.next()){

                      4  resultado.agregarEstudiante(resultSet2.getString("legajo"),
                         resultSet2.getString("apellido"),
                         resultSet2.getString("nombre"),
                         resultSet2.getString("calificacion"),
                         resultSet2.getString("comentario"));
                   }

                5  listado.agregarResultado(resultado);
             }

          6  listado.ordenarListado();
             conection.Close();

       7  } catch (SQLException ex){
             8  System.out.println(ex);
       9  }

      10 return listado;    }
```

Grafo de Flujo de Control



Complejidad Ciclomática V(G)

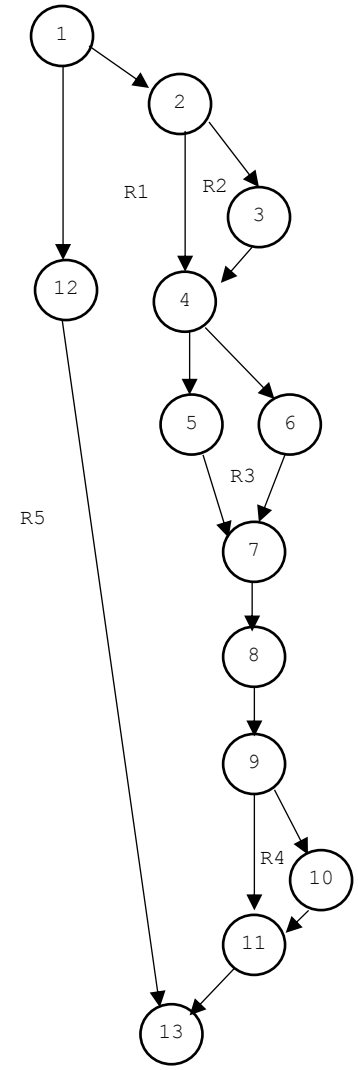
Aristas: 13
Nodos: 10
Regiones: 5
Nodos Predicado: 1,2,3,7
V(G): 13 (Aristas) - 10 (Nodos) + 2 = 5
V(G): 4 (Nodos Predicado) + 1 = 5

Caminos independientes

a) 1,2,3,4,3,5,2,6,10
b) 1,2,3,5,2,6,10
c) 1,2,6,10
d) 1,7,8,9,10
e) 1,7,9,10

```
public void action() {
    ACLMessage msg = receive();
 1  int estado = 0;
    if (msg!=null) {

        try {

           Listado listado = (Listado)msg.getContentObject();
         2 int cantidadDeResultados = listado.cantidadResultados();
           if(cantidadDeResultados == 1 &&
           listado.obtenerResultado(0).getTitulo().equals("null"))

             3     estado  = 1;
           4 switch(estado){

               5       case 0: listado = comportamientoUno(listado,cantidadDeResultados);
                       break;

               6       case 1: listado = comportamientoDos(listado,cantidadDeResultados);
                       break;

             7 }
```

```
            ACLMessage reply = msg.createReply();
            reply.setPerformative( ACLMessage.INFORM );
          8 reply.setContentObject(listado);
            send(reply);
      9   } catch(Exception e) {
           10 e.printStackTrace();
   11      }
     } else{

   12  block();

     }
}       13
```



Complejidad Ciclomática V(G)

Aristas: 16
Nodos: 13
Regiones: 5
Nodos Predicado: 1,2,3,7
V(G): 16 (Aristas) - 13 (Nodos) + 2 = 5
V(G): 4 (Nodos Predicado) + 1 = 5

Caminos independientes

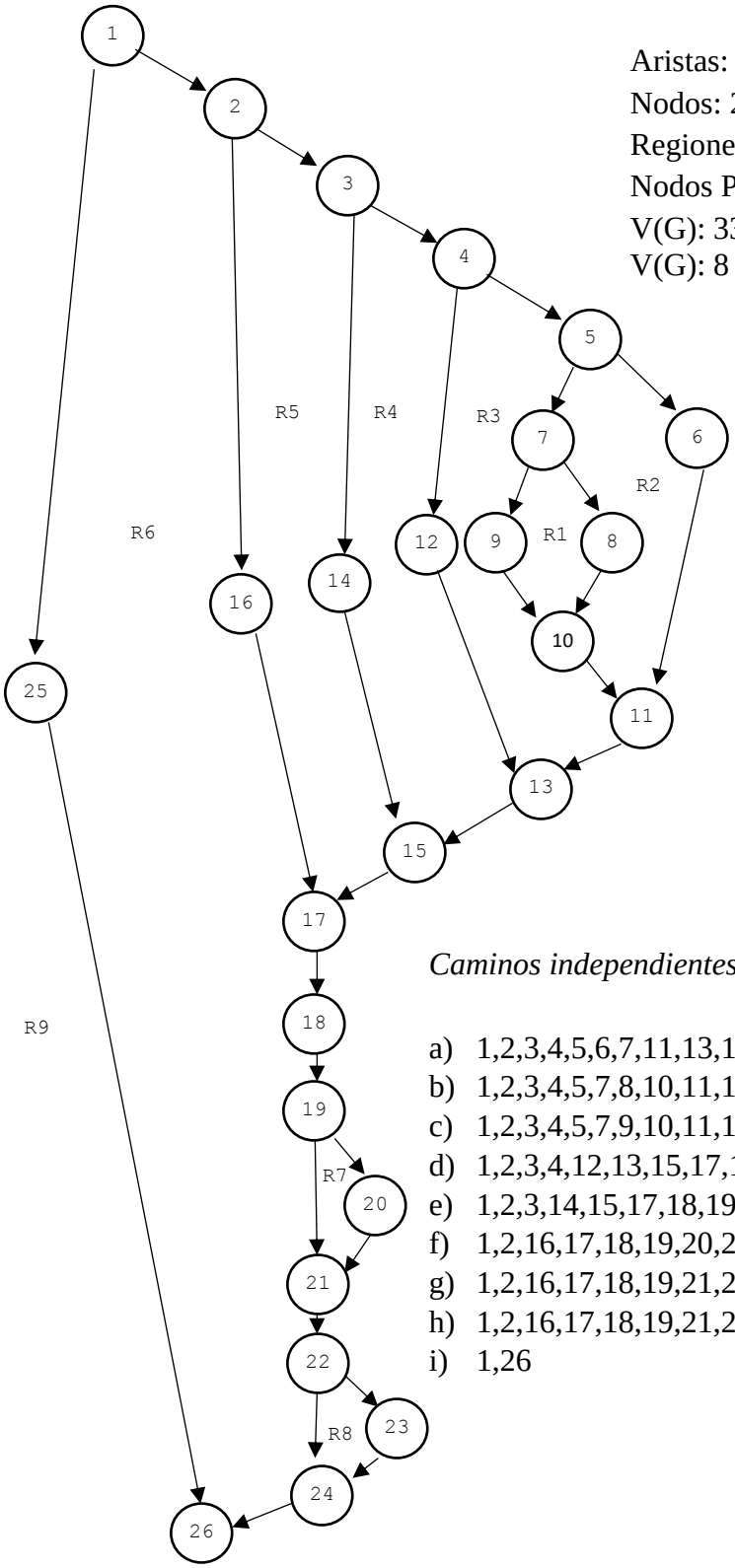
a) 1,2,3,4,6,7,8,9,10,11,13
b) 1,2,3,4,5,7,8,9,10,11,13
c) 1,2,4,5,7,9,11,13
d) 1,2,4,6,7,8,9,11,13
e) 1,12,13

Función del Agente de Actualización

```
public void action(){
   ACLMessage msg = receive();
 1 boolean flag = false;
   if (msg!=null) {

      try {

         try {

            final conection bd = new conection();
            PaqueteActualizacion listado = (PaqueteActualizacion)msg.getContentObject();
            String Query = "SELECT * FROM resultado INNER JOIN resultadoporgrupo ON
            (resultado.idResultado=resultadoporgrupo.idResultado) WHERE
          2 resultado.url='"+listado.getUrl()+"' AND
            resultadoporgrupo.idGrupo='"+listado.getGrupo()+"'";
            Statement st = conection.getConexion().createStatement();
            java.sql.ResultSet resultSet;
            resultSet = st.executeQuery(Query);
            resultSet.next();
            if (resultSet.getRow()!=0){

               System.out.println("La url fue anteriormente calificada/comentada por
               este grupo");
               Query = "SELECT * FROM calificacionxestudiante INNER JOIN resultado ON
               calificacionxestudiante.idResultado=resultado.idResultadoWHERE
             3 calificacionxestudiante.legajo='"+listado.getLegajo()+"' AND
               calificacionxestudiante.idGrupo='"+listado.getGrupo()+"' AND
               resultado.url='"+listado.getUrl()+"'";
               st = conection.getConexion().createStatement();
               resultSet = st.executeQuery(Query);
               resultSet.next();
               if(resultSet.getRow()!=0){

                  System.out.println("La url fue anteriormente calificada/comentada por
                4 este estudiante en este grupo");
                  if(!listado.getCalificacion().equals("0")){

                     System.out.println("La calificacion es distinto de 0, entonces
                   5 debo modificar");
                     if(listado.getComentario().equals(resultSet.getString("comentario"
                     ))){

                        System.out.println("Como el comentario enviado es igual al
                      6 anterior, entonces modifico la calificacion");
                        modificaCalificacion(listado.getCalificacion(),
                        resultSet.getInt("idResultado"),
                        listado.getLegajo(),Integer.parseInt(listado.getGrupo()));
                     }else{

                        if(listado.getCalificacion().equals(resultSet.getInt("califica
                      7 cion")+"")){

                           System.out.println("Como calificacion enviada es igual al
                           anterior, entonces modifico el comentario");
                         8 modificaComentario(resultSet.getInt("idResultado"),listad
                           o.getComentario(),listado.getLegajo(),Integer.parseInt(li
                           stado.getGrupo()));
                        }else{
                           System.out.println("Como tanto la calificacion como el
                           comentario enviados son distintos, entonces modifico
                           ambos");
                         9 modificarCalyCom(resultSet.getInt("idResultado"),listado.
                           getCalificacion(),listado.getLegajo(),listado.getComentar
                           io(),Integer.parseInt(listado.getGrupo()));
```

```
                    }
                }

                }else{
                    System.out.println("La calificacion es igual a 0, entonces tengo
                    q eliminar");
                    int idResultado = obtenerIdResultado(listado.getUrl());
                    Query = "SELECT * FROM calificacionxestudianteWHERE
                    idGrupo='"+listado.getGrupo()+"' AND
                    idResultado='"+idResultado+"'";
                    st = conection.getConexion().createStatement();
                    resultSet = st.executeQuery(Query);
                    resultSet.last();
                    int cantidadComentarios = resultSet.getRow();;
                    Query = "SELECT idGrupo FROM resultadoporgrupo WHERE
                    idResultado='"+idResultado+"'";
                    st = conection.getConexion().createStatement();
                    resultSet = st.executeQuery(Query);
                    resultSet.last();
                    int cantidadGrupos = resultSet.getRow();
                    st.close();
                    System.out.println(cantidadComentarios);
                    eliminarAporte(listado.getLegajo(),
                    Integer.parseInt(listado.getGrupo()),
                    listado.getUrl(),cantidadComentarios, idResultado,
                    cantidadGrupos);
                }
                }else{
                    System.out.println("La url no fue anteriormente
                    calificada/comentada por este estudiante en este grupo");
                    nuevoComYCal(listado.getUrl(), listado.getCalificacion(),
                    listdo.getComentario(),
                    listado.getLegajo(),Integer.parseInt(listado.getGrupo()));
                }

            }else{
                System.out.println("La url no fue anteriormente calificada/comentada por
                este grupo");
                nuevoUrl(listado);
            }

            conection.Close();
            ACLMessage reply = msg.createReply();
            reply.setPerformative( ACLMessage.INFORM );
            listado.setEstado("ACTUALIZADO");
            reply.setContentObject(listado);
            send(reply);

        } catch (SQLException ex) {
            System.out.println(ex);
        }

    } catch(Exception e) {
        e.printStackTrace();
    }

    }

    else block();
}
```

Grafo de Flujo de Control



Complejidad Ciclomática V(G)

Aristas: 33
Nodos: 26
Regiones: 9
Nodos Predicado: 1,2,3,4,5,7,19,20
V(G): 33 (Aristas) - 26 (Nodos) + 2 = 9
V(G): 8 (Nodos Predicado) + 1 = 9

Caminos independientes

a) 1,2,3,4,5,6,7,11,13,15,17,18,19,20,21,22,23,24,26
b) 1,2,3,4,5,7,8,10,11,13,15,17,18,19,20,21,,22,23,24,26
c) 1,2,3,4,5,7,9,10,11,13,15,17,18,19,20,21,,22,23,24,26
d) 1,2,3,4,12,13,15,17,18,19,20,21,22,23,24,26
e) 1,2,3,14,15,17,18,19,20,21,22,23,24,26
f) 1,2,16,17,18,19,20,21,22,23,24,26
g) 1,2,16,17,18,19,21,22,23,24,26
h) 1,2,16,17,18,19,21,22,24,26
i) 1,26

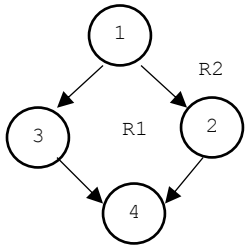
Función de Agente GateWay

```
protected void processCommand(java.lang.Object obj) {

 1 [if (obj instanceof PaqueteActualizacion){

    2 [paquete = (PaqueteActualizacion)obj;
       bandera = 1;
       ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
       msg.addReceiver(new AID(paquete.getNombreAgente(), AID.ISLOCALNAME) );
       msg.setContentObject(paquete);
       send(msg);
}
else{
       [listado = (Listado)obj;
       bandera = 2;
    3  ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
       msg.addReceiver(new AID(listado.obtenerAgente(), AID.ISLOCALNAME) );
       msg.setContentObject(listado);
       send(msg);
        }
          4
}
```

Complejidad Ciclomática V(G)

Aristas: 4
Nodos: 4
Regiones: 1
Nodos Predicado: 1
V(G): 4 (Aristas) - 4 (Nodos) + 2 = 2
V(G): 1 (Nodos Predicado) + 1 = 1

Grafo de Flujo de Control



Caminos independientes

a) 1,2,4
b) 1,3,4

```
public void action(){
  0 [ACLMessage msg = receive();

 1[if ((msg!=null)&&(paquete!=null)||(msg!=null)&&(listado!=null)){
            1.A           1.B           1.C           1.D
      try {
    2 [     if(bandera == 1){
            [    PaqueteActualizacion paqueteAux =
            [    (PaqueteActualizacion)msg.getContentObject();
         3  [    paquete.setEstado(paqueteAux.getEstado());
            [    releaseCommand(paquete);
            [    paquete = null;
    4 [}
    5 [     if(bandera == 2){
            [    Listado lista = (Listado)msg.getContentObject();
         6  [    listado.asociarListado(lista.obtenerListado());
            [    releaseCommand(listado);
            [    listado = null;
```

```
        7 [}
        8 [       bandera = 0;
    9 [ }catch (Exception e2) {

            10 [  e2.printStackTrace();
     11 [ }

    [}else {
12  [     block();
13  [}
  }
```

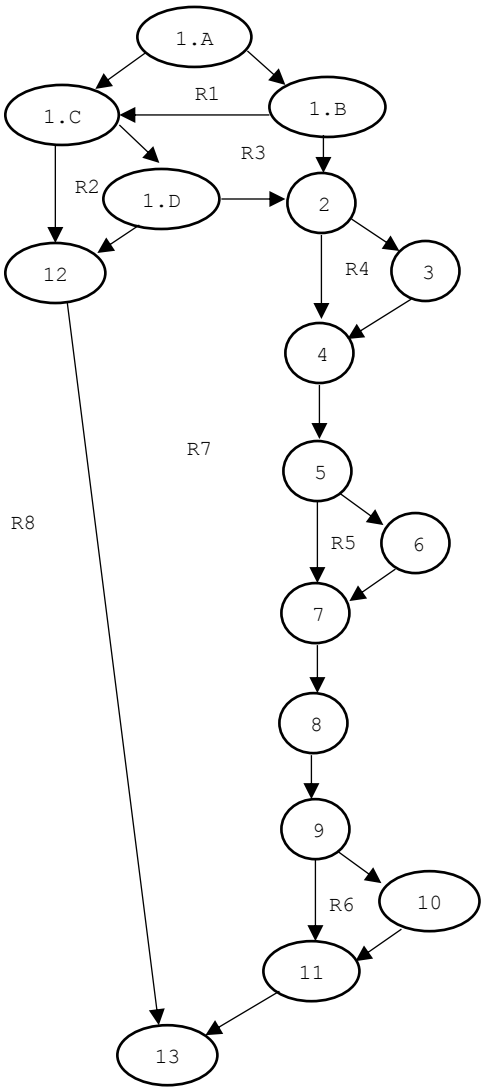
Complejidad Ciclomática V(G)

Aristas: 22
Nodos: 16
Regiones: 8
Nodos Predicado: 1.A, 1.B, 1.C, 1.D, 2, 5, 9
V(G): 22 (Aristas) - 16 (Nodos) + 2 = 8
V(G): 7 (Nodos Predicado) + 1 = 8

Caminos independientes

a) 1.A, 1.B, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13
b) 1.A, 1.B, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13
c) 1.A, 1.B, 2, 4, 5, 6, 7, 8, 9, 11, 13
d) 1.A, 1.B, 2, 4, 5, 7, 8, 9, 11, 13
e) 1.A, 1.B, 1.C, 1.D, 12, 13
f) 1.A, 1.B, 1.C, 1.D, 2, 4 ,5 ,7 ,8 ,9 ,11, 13
g) 1.A, 1.C, 12, 13
h) 1.A, 1.C, 1.D, 12, 13

Grafo de Flujo de Control



ANEXO II

MANUAL DE INSTALACION

- JUNE -

INDICE

INDICE DE FIGURAS

I. INTRODUCCIÓN

El siguiente documento corresponde al manual de instalación de June, el metabuscador colaborativo basado en agentes para grupos de estudiantes. El objetivo de este manual es simplificar en la mayor medida posible el proceso de instalación del metabuscador para garantizar el buen funcionamiento del mismo.

Este documento ofrece una breve explicación de los pasos a seguir para la de instalación y dos alternativas para instalar a June.

II. METABUSCADOR JUNE

June nace como propuesta al trabajo final presentado por quienes redactaron este mismo documento. Es un metabuscador colaborativo basado en agentes para apoyar las tareas de grupos de estudiantes. De manera general, el metabuscador permite realizar búsquedas de material en la web, revisando, calificando y/o comentando los resultados mostrados en el listado de resultados de búsqueda devuelto. Dicho listado está rankeado de acuerdo con la calificación general de cada resultado, la cual se obtiene como promedio de las calificaciones individuales de los integrantes del grupo. También puede consultar la ayuda sobre cómo se maneja el metabuscador o contactar con los desarrolladores ante cualquier consulta. Por último, puede consultar el historial de resultados de búsquedas del grupo.

III. MANUAL DE INSTALACION

Debido a la complejidad que puede resultar la instalación del prototipo de "metabuscador colaborativo basado en agentes para grupos de estudiantes" llamado "June", se han dispuesto dos alternativas para ponerlo en funcionamiento.

> *IMPORTANTE*: La instalación de cualquiera de las dos alternativas se lleva a cabo dentro de una red LAN, por lo tanto, todos los estudiantes deben estar conectados dentro de la misma red para que puedan acceder a June. Si se requiere acceder a June desde fuera de la red LAN entonces es necesario configurar la red de la organización en donde actualmente está instalando. Esa configuración no está incluida en la presente documentación.

La primera alternativa, y las más sencilla, se da a partir de la importación de una máquina virtual en Oracle VM VirtualBox (versión 5.1.4 r110228). Dicha máquina virtual

tiene toda la configuración y servicios necesarios para poner en funcionamiento el prototipo. Solo debe configurarse manualmente la red de la máquina virtual para que pertenezca a la misma red local en la que se desea poner en funcionamiento a JUNE.

La segunda alternativa, y la más compleja, consiste en instalar el prototipo desde "cero" en un servidor (hosting) pago o propio. Para ello debe tenerse en cuenta una serie de requisitos fundamentales y necesarios para que el prototipo funcione correctamente.

A continuación se detallan las alternativas.

Accesos utilizados en el Manual de Instalación

Motivo	Usuario	Contraseña
Para iniciar Sesión con usuario June	juneTF	04091983
Para utilizar el usuario root	root	root@root
Para acceder phpMyAdmin	root	root@june
Para acceder a Base de Datos sin phpMyAdmin	root	root@june
Para acceder a Tomcat 8	admin	manager

Formulario de Contacto:

Cualquier sea la alternativa de instalación seleccionada, para que la sección de "Contacto" del metabuscador funcione, es necesario asociarlo con una cuenta de correo mediante un archivo de configuración llamado "enviarcorreo.php" ubicado en su directorio raíz. La asociación se establece a través de los siguientes parámetros: dirección ssl de correo, puerto, autenticación smtp, nombre de usuario y contraseña, como se muestra a continuación.

```
$mail->Host = 'ssl://smtp.gmail.com';
$mail->Port = 465;
$mail->SMTPAuth = true;
$mail->Username = 'junemetabuscador@gmail.com';
$mail->Password = 'junemeta';
```

IV. ALTERNATIVA Nº1 – IMPORTAR MAQUINA VIRTUAL

IV.1. INSTALACIÓN DEL ORACLE VIRTUALBOX 5.1

Se recomienda instalar la versión 5.1.14 del VirtualBox, debido a que sobre ella se trabajó la máquina virtual del prototipo. La recomendación se hace solo por garantizar la compatibilidad de la máquina virtual con el VirtualBox.

Para instalar el VirtualBox se debe ejecutar el archivo "VirtualBox-5.1.14-112924-Win.exe" ubicado en la siguiente ruta: D:\DVD_JUNE\virtualbox\, es decir:

D:\DVD_JUNE\virtualbox\VirtualBox-5.1.14-112924-Win.exe

IMPORTANTE: Dependiendo del equipo donde se desea instalar el programa, la letra que identifica la unidad de DVD puede variar, pudiendo ser la misma la letra D, o E, o F, etc…

Al ejecutar el archivo "VirtualBox-5.1.14-112924-Win.exe" aparecerá la siguiente ventana como se muestra en la Figura 58.

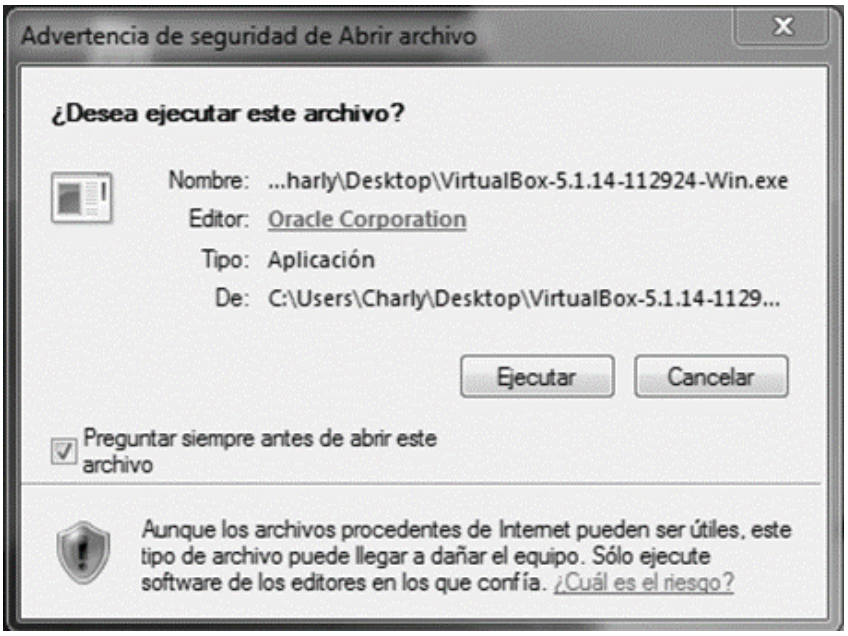


Figura 58 – Instalación VirtualBox - Advertencia de Seguridad de Abrir archivo de instalación

Al hacer click con el botón izquierdo del mouse (**de ahora en más se mencionará "hacer click" o "doble click" o "clickear" en referencia al botón izquierdo del mouse,**

salvo que se indique expresamente el botón derecho) sobre el botón de "Ejecutar" aparecerá la ventana que se muestra en la Figura 59.

Figura 59 - Instalación VirtualBox - Bienvenida de Instalación del VirtualBox

Luego de hacer click en el botón "Next", aparecerá una nueva ventana que permite seleccionar las características de instalación del VirtualBox como se muestra en la Figura 60. No debe cambiarse ninguna configuración de la pantalla, se deja todo como está para luego hacer click sobre en el botón "Next"

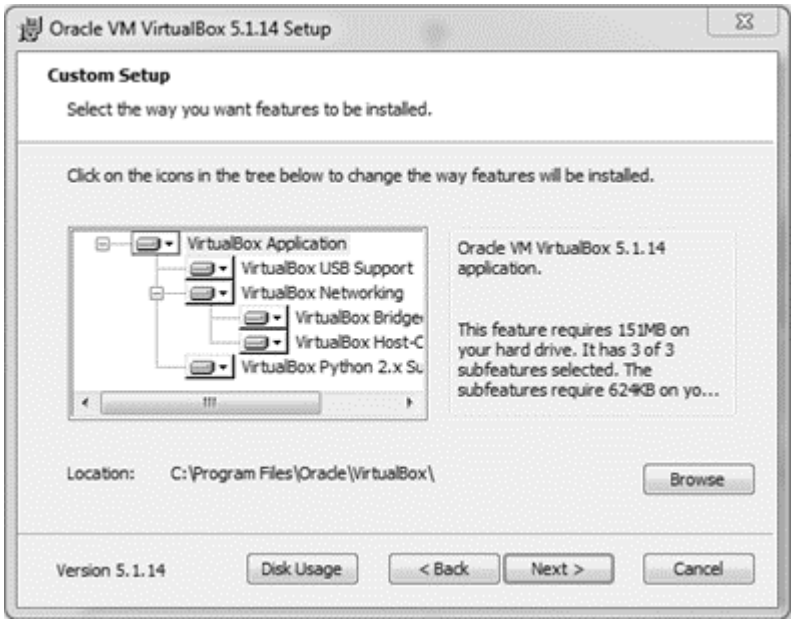


Figura 60 - Instalación VirtualBox - Instalación de las características del VirtualBox

Luego aparecerá una nueva ventana con otras características de instalación. Seleccionando todas las opciones como muestra en la Figura 61, se debe hacer click en el botón “Next”.

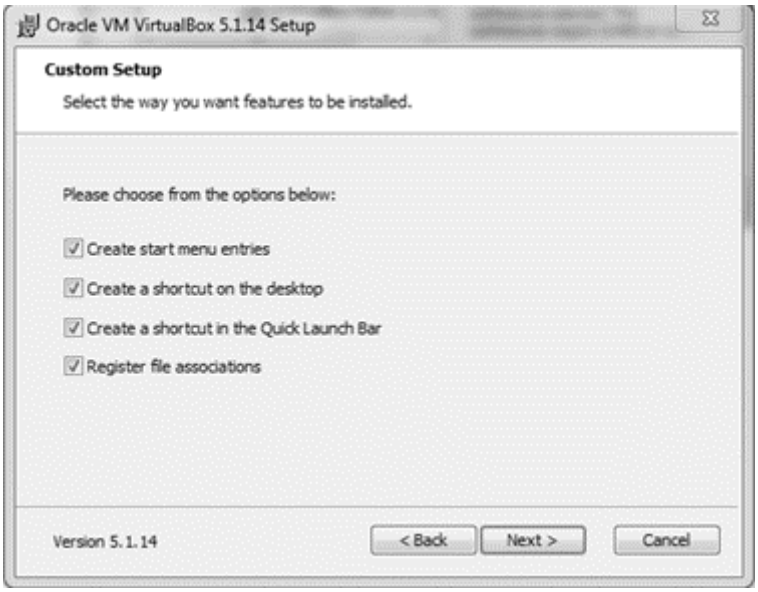


Figura 61 – Instalación VirtualBox - Instalación de otras características del VirtualBox

Seguidamente se mostrará una ventana de advertencia (Figura 62), indicando que reiniciará todas las interfaces de red (placas de red) del equipo físico donde se está instalando el VirtualBox por lo que se perderá momentáneamente por unos segundos la conexión a internet.

Figura 62 – Instalación VirtualBox - Advertencia sobre las Interfaces de Red

Al hacer click en el botón “Yes” aparecerá otra ventana, como se muestra en la Figura 63, indicando que el instalador del VirtualBox tiene todas las configuraciones necesarias para comenzar con la instalación.

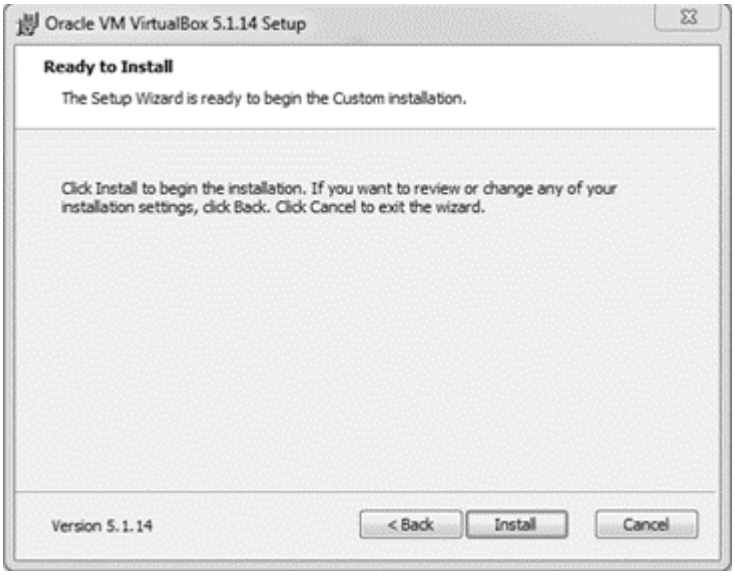


Figura 63 – Instalación VirtualBox - Iniciar la Instalación

Comenzará el proceso de instalación de Oracle VM VirtualBox 5.1.14. Durante la instalación aparecerá una ventana, como se muestra en la Figura 64, en la que se deberá tildar la opción de: Siempre confiar en el software de “Oracle Corporation”. Luego hacer click en el botón “Instalar” para que continúe la instalación.

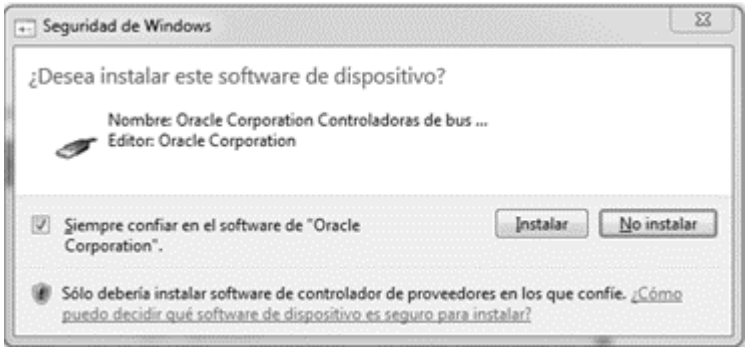


Figura 64 – Instalación VirtualBox - Seguridad de Windows

Cuando finalice la instalación aparecerá una última ventana como se muestra en la Figura 65 indicando que ha finalizado correctamente la instalación. Por último, se debe clickear una vez el botón “Finish”.

Figura 65 – Instalación VirtualBox - Fin de Instalación de VirtualBox

IV.2. IMPORTAR MÁQUINA VIRTUAL DEL PROTOTIPO

Para importar una máquina virtual en el Oracle VirtualBox 5.1.14, primero se debe abrir/ejecutar el VirtualBox. Una vez iniciado el programa se mostrará una ventana como la de la Figura 66.

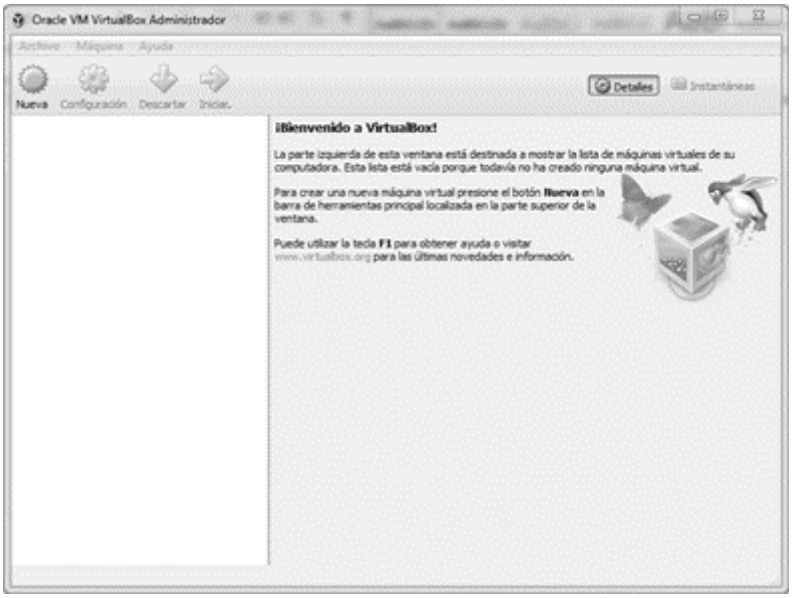


Figura 66 – VirtualBox Administrador – Administrador del VirtualBox

Ahora sí, para importar una máquina virtual se debe clickear en “Archivo”, luego se desplegará un menú en el que se deberá clickear en “Importar Servicio Virtualizado” como se muestra en la Figura 67.

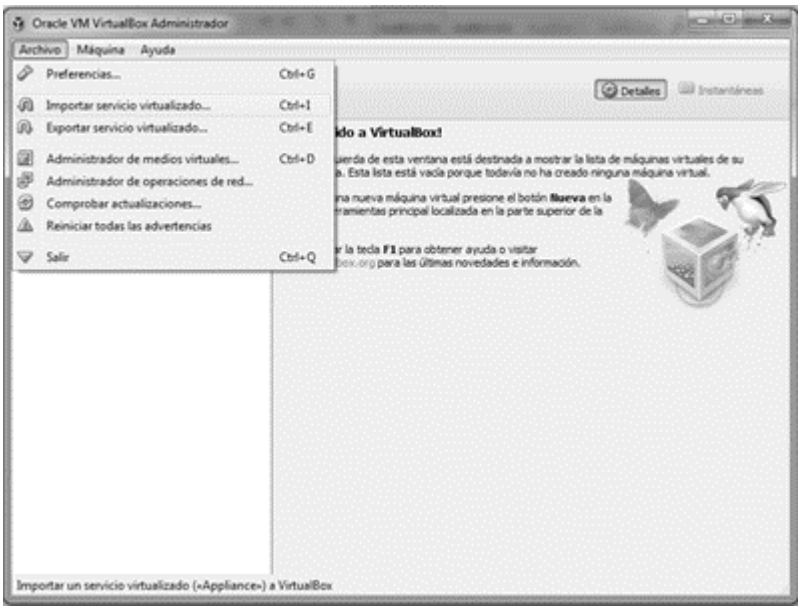


Figura 67 – VirtualBox Administrador – Menú Desplegable del Item “Archivo”

Luego de clickear en “Importar Servicio Virtualizado” aparecerá una ventana como se muestra en la Figura 68 en la que se deberá ingresar la ubicación del archivo “june.ova” (archivo de Máquina Virtual), para ello se debe clickear en el ícono con forma de carpetita amarilla con una flecha verde apuntando hacia arriba.

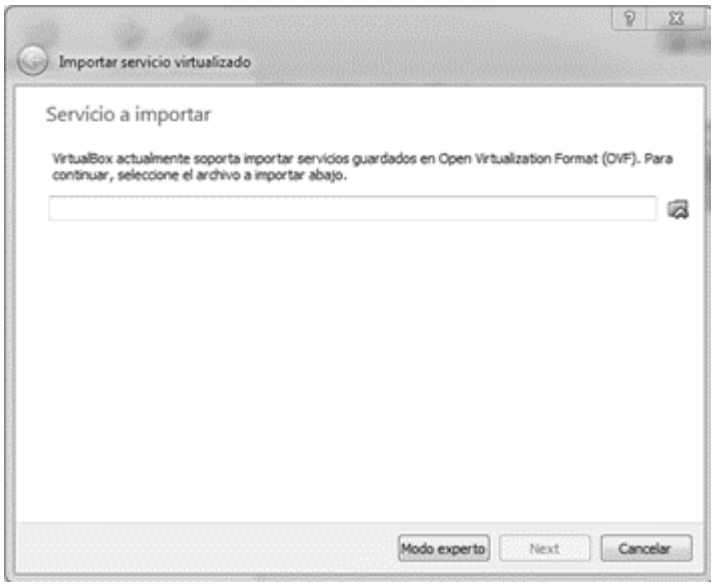


Figura 68 – VirtualBox Administrador – Servicio a Importar Sin Completar

Luego de clickear en el ícono anteriormente nombrado aparecerá una ventana, como se muestra en la Figura 69, en la que se debe buscar y seleccionar el archivo “june.ova”.

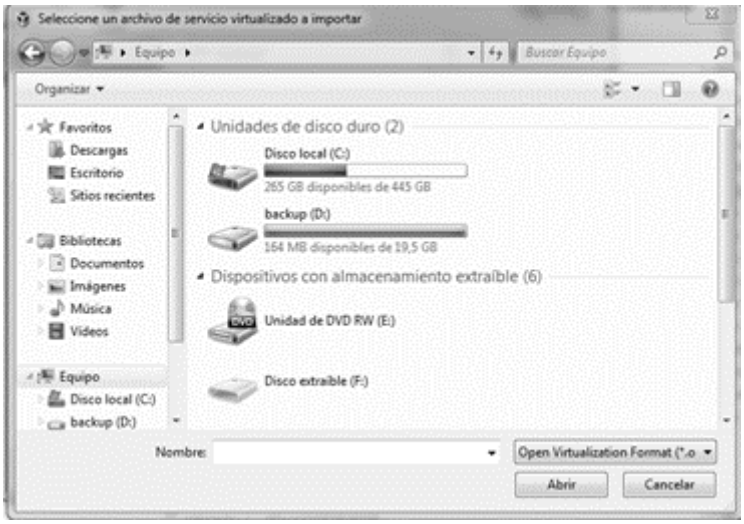


Figura 69 – VirtualBox Administrador – Selección de Archivo de Servicio Virtualizado a Importar

La ruta debe ser “D:\DVD_JUNE\june\june.ova”, como se muestra en la Figura 70. Luego se debe clickear en “Next”.

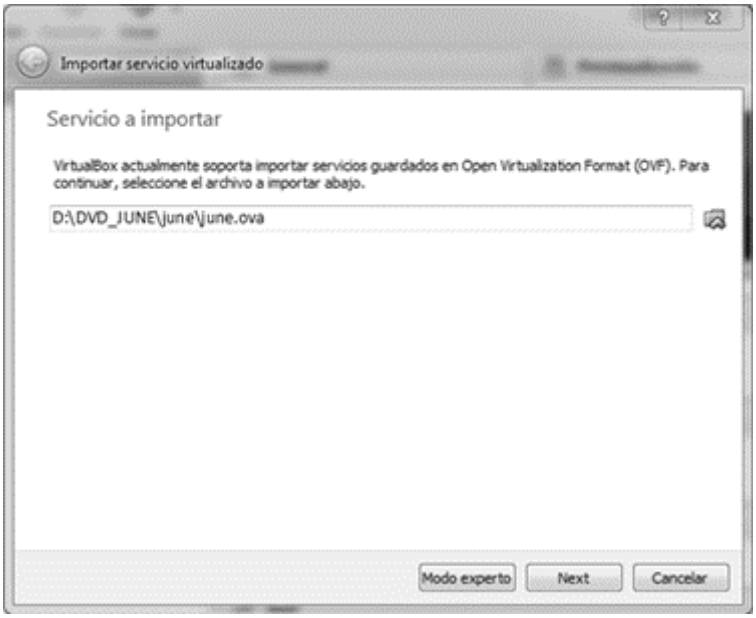


Figura 70 – VirtualBox Administrador – Servicio a Importar Completado

A continuación aparecerá una ventana como se muestra en la Figura 71 con todas las preferencias de la máquina virtual por importar. Se debe seleccionar la opción “Reinicializar la dirección MAC de todas las tarjetas de red” y luego clickear en “Importar”.

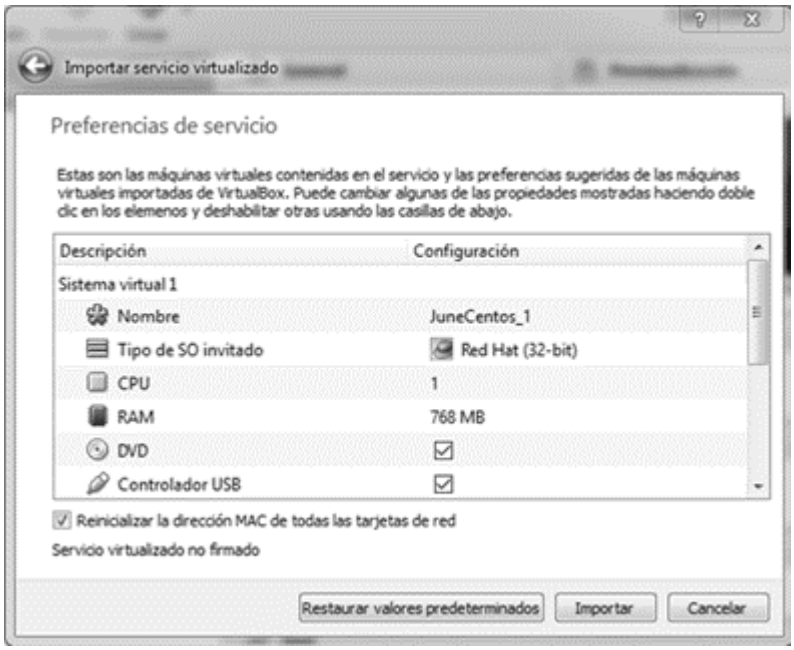


Figura 71 – VirtualBox Administrador – Preferencias de Servicio

Una vez finalizada la importación aparecerá en el Administrador de VirtualBox la máquina virtual llamada “June” del Prototipo de Metabuscador Basado en Agentes para Grupos de Estudiantes Colaborativos, como se muestra en la Figura 72.

Figura 72 – VirtualBox Administrador – Administrador de VirtualBox con Máquina Virtual del Prototipo

IV.3. INICIAR MÁQUINA VIRTUAL

Para poner en funcionamiento la Máquina Virtual “June” solo se debe seleccionar “June” en el panel izquierdo, como se muestra en la Figura 72, y clickear en el botón “Iniciar” (con forma de flecha verde apuntando hacia la derecha) sobre el menú superior. Una vez clickeado en “Iniciar” la máquina virtual comenzará a cargar como se muestra en la Figura 73.

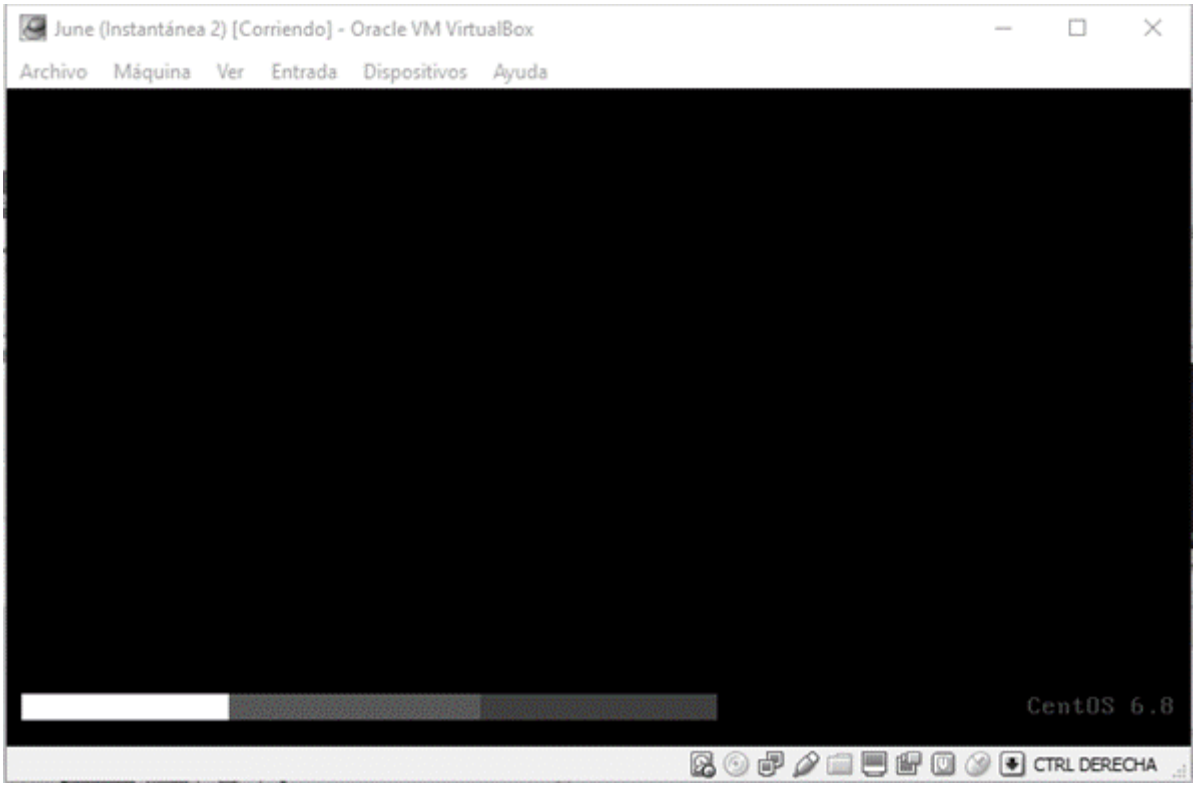


Figura 73 - Máquina Virtual – Iniciando Linux CentOS

Una vez iniciada la máquina virtual como se muestra en la Figura 74, se podrá hacer uso del Sistema Operativo (S.O.) Linux CentOS.

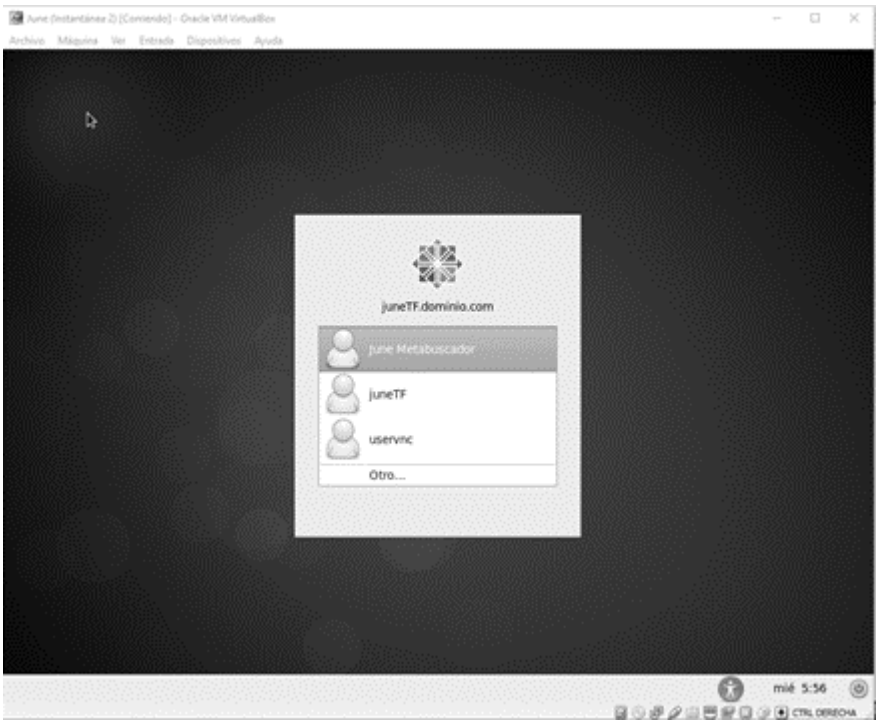


Figura 74 - Máquina Virtual - Linux CentOS Iniciado

Para continuar con la instalación es necesario ingresar al "Escritorio" del usuario asignado para la utilización del Prototipo de Metabuscador "June", seleccionando el usuario e ingresando su contraseña, como se muestra en la Figura 75, cuyo caso el usuario es: "**juneTF**" y la contraseña: "**04091983**" (sin comillas dobles)

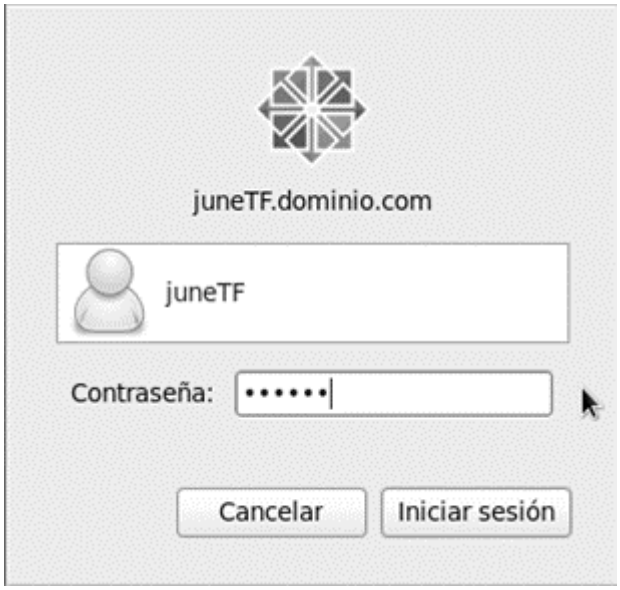


Figura 75 - CentOS - Ingreso de Usuario y Contraseña

Luego de presionar "Iniciar Sesión", se mostrará en pantalla el escritorio del usuario como se muestra en la Figura 76.

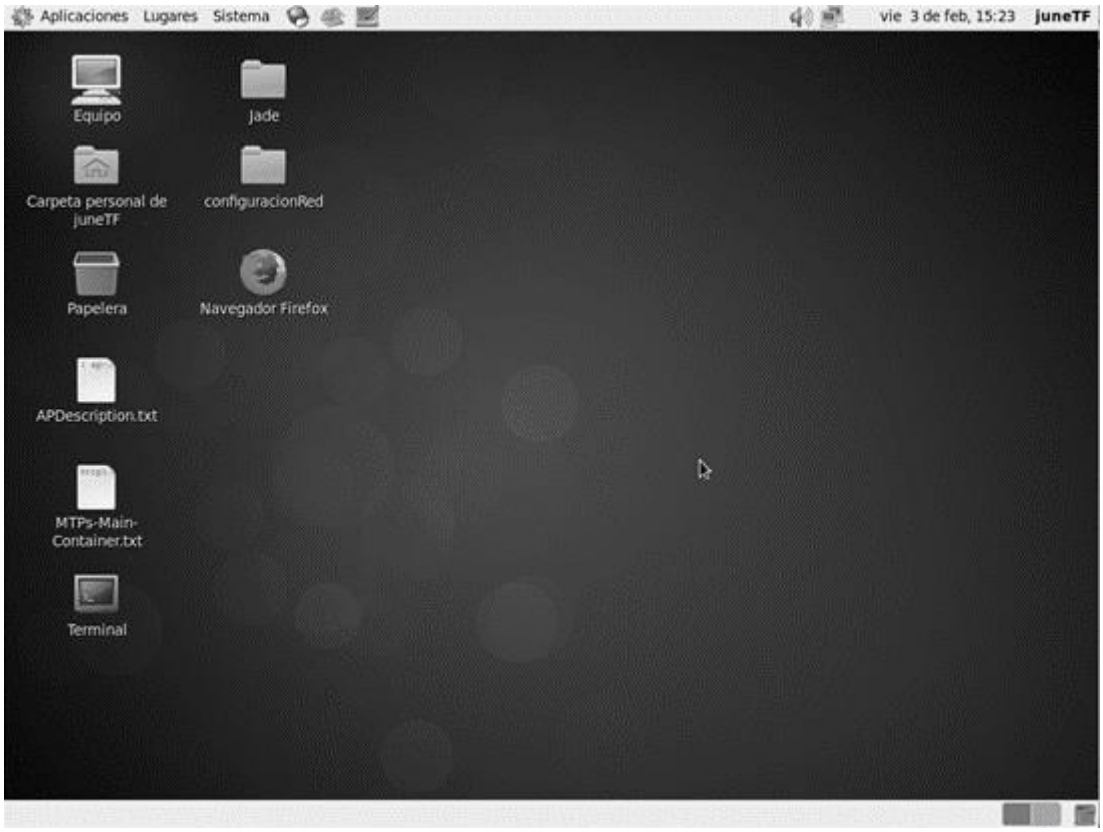


Figura 76 - CentOS - Escritorio de juneTF

IV.4. CONFIGURACIÓN FINAL

Antes de poner en funcionamiento el Prototipo de Metabuscador "June" es necesario realizar un último paso de instalación sobre el S.O. Linux CentOS, es necesario configurar la conexión de red del CentOS.

Es **IMPORTANTE** que se lleve a cabo este paso de configuración final. A partir de esta configuración, el S.O Linux CentOS quedará preparado para ser usado.

Se recuerda que el Linux CentOS debe tener una configuración de red acorde a la configuración de red en donde se lo va a utilizar, es decir, "*debe estar dentro de la misma red en donde se lo va a utilizar*".

Generalmente en organizaciones pequeñas o en los hogares, las configuraciones de red tienen la siguiente forma por ejemplo:

RED: 192.168.1.0
Mascara de Subred: 255.255.255.0
Puerta de Enlace: 192.168.1.1

DNS Principal: 192.168.1.1

Cabe aclarar que la dirección IP 192.168.1.1, por lo general, es la dirección IP del Router, Modem Router o Cable Modem; y también que las direcciones IP de las PC (también conocidas como Host) de las organizaciones pequeñas o los hogares tienen la siguiente forma, por ejemplo:

PC1: 192.168.1.2
PC2: 192.168.1.4
PC3: 192.168.1.7

En la máquina virtual, el S.O. Linux CentOS ya viene con una configuración de red:

Dirección IP del Host: **192.168.1.123**

Mascara de Subred: **255.255.255.0**
Puerta de Enlace: **192.168.1.1**
DNS Principal: **192.168.1.1**

La configuración de red debe llevarse a cabo **SI O SI** como se indica a continuación.

IV.4.1. Configuración de Red

Para configurar la red en el S.O. Linux CentOS se debe acceder a la carpeta "configuracionRed" (Figura 77), dispuesta en el escritorio del usuario juneTF, haciendo doble click en el ícono de la carpeta.

Figura 77 – Configuración de Red del CentOS – Ícono de Carpeta de Configuración de Red

Al acceder a la carpeta, como se muestra en la Figura 78, se debe ejecutar el archivo llamado "iniciarConfiguracionRed.sh" haciendo doble click sobre él.

Figura 78 - Configuración de Red del CentOS – Archivos de Carpeta de Configuración de Red

Una vez que se hizo doble click sobre el archivo "iniciarConfiguracionRed.sh" aparecerá una ventana, como se muestra en la Figura 79, solicitando que elija que es lo que se quiere ejecutar. Se debe clickear una sola vez sobre el botón "**Ejecutar en un terminal**".

Figura 79 - Configuración de Red del CentOS – Selección de Ejecución

Al hacer click en el botón "**Ejecutar en un terminal**" aparecerá una ventana, como se muestra en la Figura 80, indicando un ejemplo como opción de configuración de red para S.O. Linux CentOS, y esperando que confirme si desea aceptar (ACEPTAR) la configuración mostrada o si desea cambiarla (MODIFICAR). Siempre que ingrese el número se debe presionar seguidamente la tecla ENTER.

Figura 80 - Configuración de Red del CentOS – Opciones de Configuración

En el **caso de aceptar la configuración** se debe presionar la tecla número "0" (cero) y presionar la tecla "Enter". Luego de hacerlo, se solicitará que se presione nuevamente la tecla "Enter" para reiniciar el S.O. Linux CentOS como se muestra en la Figura 81.

Figura 81 - Configuración de Red del CentOS – Reinicio del S.O. Linux CentOS

Una vez presionada la tecla "Enter", el S.O. procederá a reiniciarse. Una vez reiniciado, estará configurado y listo para poner en funcionamiento el Prototipo de Metabuscador "June".

En el **caso de cambiar la configuración** se debe presionar la tecla número "1" (uno) y presionar la tecla "Enter". Luego de hacerlo, se solicitará una serie de datos para ingresar.

Primero, solicitará la Dirección IP del Host (dirección IP para la Máquina Virtual). Se debe ingresar la dirección IP, y luego seguidamente presionar la tecla "Enter". A modo de ejemplo, se ingresó la dirección IP 192.168.1.123, como se muestra en la Figura 82.

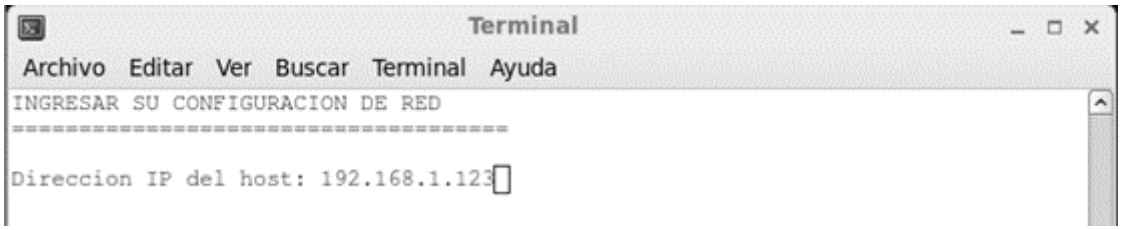


Figura 82 - Configuración de Red del CentOS – Ingreso de Dirección IP

Segundo, solicitará la Dirección de la Máscara de Subred. Se debe ingresar la dirección de la máscara, y luego seguidamente presionar la tecla “Enter”. A modo de ejemplo, se ingresó la dirección IP 255.255.255.0, como se muestra en la Figura 83.

Figura 83 - Configuración de Red del CentOS – Ingreso de Mascara de Subred

Tercero, solicitará la Dirección de la Puerta de Enlace (Gateway). Se debe ingresar la dirección de la Puerta de Enlace, y luego seguidamente presionar la tecla “Enter”. A modo de ejemplo, se ingresó la dirección IP 192.168.1.1, como se muestra en la Figura 84.

Figura 84 - Configuración de Red del CentOS – Ingreso de Puerta de Enlace

Cuarto, solicitará la Dirección del DNS Principal (DNS1). Se debe ingresar la dirección del DNS, y luego seguidamente presionar la tecla “Enter”. A modo de ejemplo, se ingresó la dirección IP 192.168.1.1, como se muestra en la Figura 85.

Figura 85 - Configuración de Red del CentOS – Ingreso de DNS Principal

Por último, se solicitará que confirme si desea aceptar (ACEPTAR) la nueva configuración mostrada o si desea cambiarla nuevamente (MODIFICAR). Siempre que ingrese el número se debe presionar seguidamente la tecla ENTER, como se muestra en la Figura 86.

Figura 86 - Configuración de Red del CentOS – Confirmación de Modificación de la Configuración de Red

Para aceptar se debe presionar la tecla número “0” (cero) y seguidamente presionar la tecla “Enter”. Luego de hacerlo, se solicitará que se presione nuevamente la tecla “Enter” para reiniciar el S.O. Linux CentOS como se muestra en la Figura 81.

Una vez presionada la tecla “Enter”, el S.O. procederá a reiniciarse. Una vez reiniciado, estará configurado y listo para poner en funcionamiento el Prototipo de Metabuscador “June”.

V. ALTERNATIVA Nº2 - INSTALACION DESDE CERO

Para llevar a cabo la instalación del Prototipo de *"June"*, el servidor (hosting) que lo aloje deberá cumplir con todos los requisitos expresados a continuación:

El servidor debe tener instalado los siguientes servicios:

- Apache Version 2.2.15, además de tener activado el Módulo "Worker" para la ejecución de múltiples hilos, también debe tener instalado el módulo HTTP_Response2 para ejecución de la librería (API) del motor de búsqueda Bing.

- PHP Version 5.6.23 Zend Engine v2.6.0, además de tener activado el empleo de clases y objetos, e instalado los paquetes de pthreads y php-zts para tratar múltiples hilos.

- Mysql 5.5.47

- phpMyAdmin - 2.11.11.3

- Webmin 1.8, en conjunto con sendmail smtp y habilitado por el firewall del sistema operativo.

- Apache Tomcat/8.5.5

Cabe destacar que la aplicación se desarrolló e implementó sobre el Sistema Operativo Linux CentOS 6.8, el cual ofrecía esos servicios, por lo tanto los mismos se consideran necesarios como mínimo para el correcto funcionamiento. De no cumplirse con la instalación de alguno de los servicios o de no respetar las versiones de cada uno de ellos, no se garantiza el correcto funcionamiento de la aplicación.

Aclaración: El Firewall (en caso de estar activado) debe tener los siguientes puertos abiertos (habilitados):

- ✓ Para Tomcat los puertos TCP: 8080, 8008, 8009, 8443
- ✓ Para MySql el puerto TCP: 3306
- ✓ Para Apache el puerto TCP: 80

Aclaración: Como el S.O. Linux CentOS de la máquina virtual ya viene con una configuración de red (descripta a continuación), se explicará la **instalación de cero** como si la máquina virtual fuese un Hosting de Servidor Linux Pago o Propietario que cumple con los requisitos anteriormente nombrados.

Dirección IP del Host:	**192.168.1.123**
Mascara de Subred:	**255.255.255.0**
Puerta de Enlace:	**192.168.1.1**
DNS Principal:	**192.168.1.1**

Esta alternativa de instalación se divide en 3 etapas:

1. Importación de Base de Datos del Prototipo
2. Instalación de la parte PHP del Prototipo
3. Instalación de la parte JAVA del Prototipo

V.1. IMPORTACIÓN DE BASE DE DATOS DEL PROTOTIPO

A modo de prototipo la aplicación emplea la siguiente configuración para su base de datos:

- ✓ Usuario: *june*
- ✓ Contraseña: *june*
- ✓ Nombre de base de datos: *calificaciones_paginas_web*
- ✓ Puerto: *3306*

Ante cualquier inconveniente dirigirse a la carpeta "D:\DVD_JUNE\june\june_java\june\WEB-INF\src" y abrir el archivo "*conection*" con cualquier editor de texto. En él se encuentra la configuración de conexión con la base de datos de la aplicación.

Para comenzar con la instalación, en primer lugar, se debe acceder a la aplicación phpMyAdmin ingresando la siguiente dirección en la barra de direcciones del navegador:

http://192.168.1.123/phpmyadmin

Nota: Tener en cuenta que la dirección IP 192.168.1.123 es la correspondiente a la que viene en la máquina virtual. Si se ha cambiado dicha dirección por una nueva, entonces esta nueva IP es la que debe ingresarse en la barra de direcciones del navegador. Por ejemplo:

http://XXX.XXX.XXX.XXX/phpmyadmin

Donde XXX.XXX.XXX.XXX es la nueva dirección IP de la máquina virtual.

Luego de ingresada la dirección de phpMyAdmin, solicitará un usuario y contraseña como se muestra en la Figura 87. Estos datos deben ser los mismos que fueron asignados al motor de base de datos MySql.
Por ejemplo:

- Usuario: *root*
- Contraseña: *root@june*

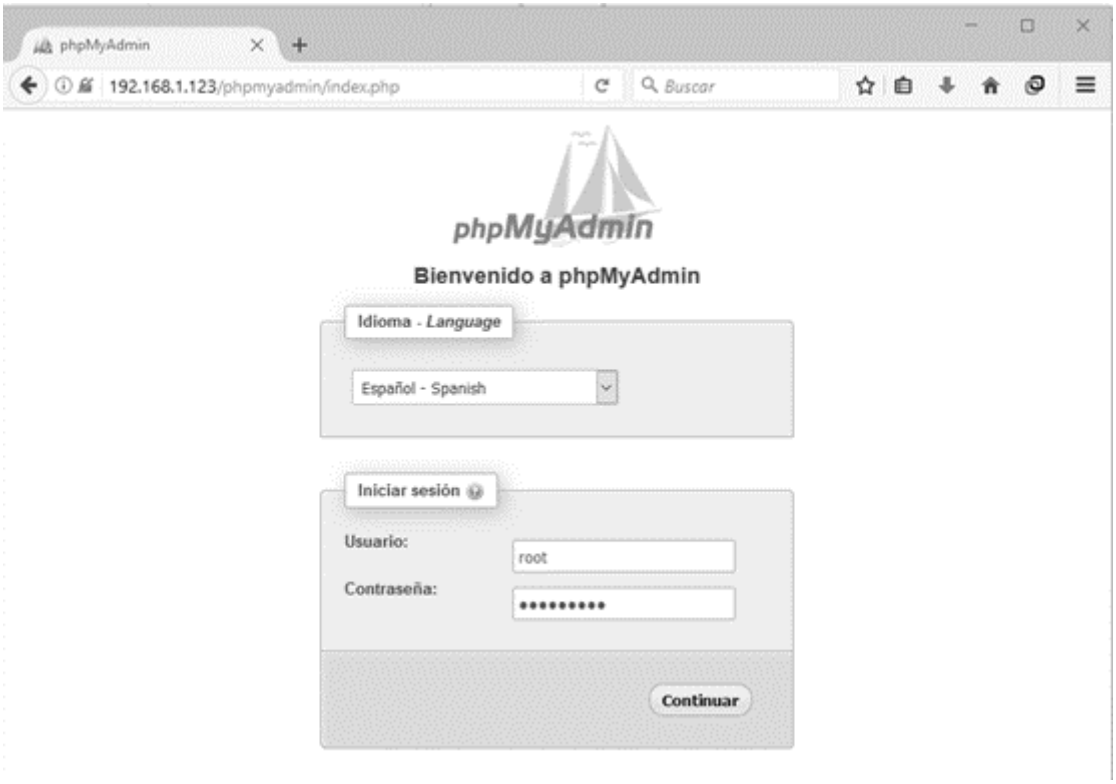


Figura 87 – phpMyAdmin - Pantalla de Solicitud de Nombre de Usuario y Contraseña

Una vez que se ingresa a la aplicación phpMyAdmin, se debe crear una base de datos con el siguiente nombre: "***calificaciones_paginas_web****",* como se observa en la Figura 88.

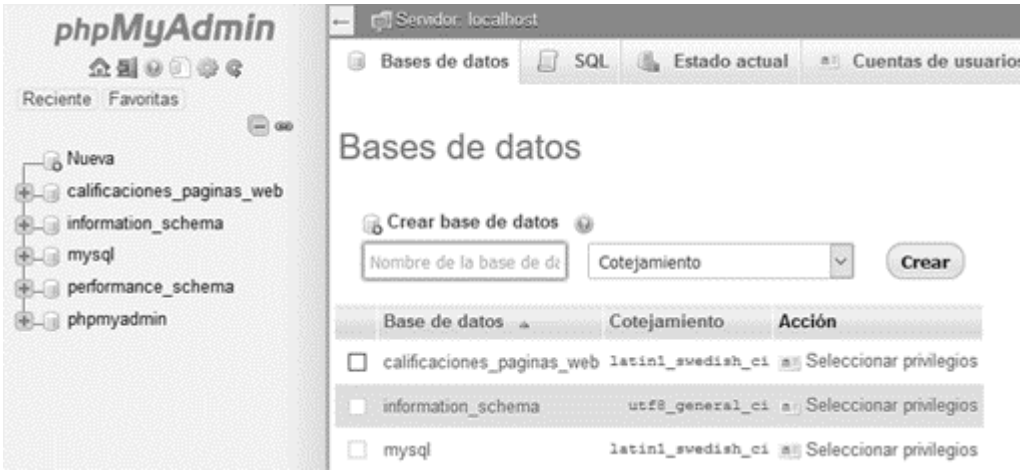


Figura 88 – phpMyAdmin - Bases de datos del Servidor

Luego de creada, se debe importar la base de datos del prototipo de metabuscador. Para ello, se seleccionará la opción de "Importar" que aparece en la Figura 89.

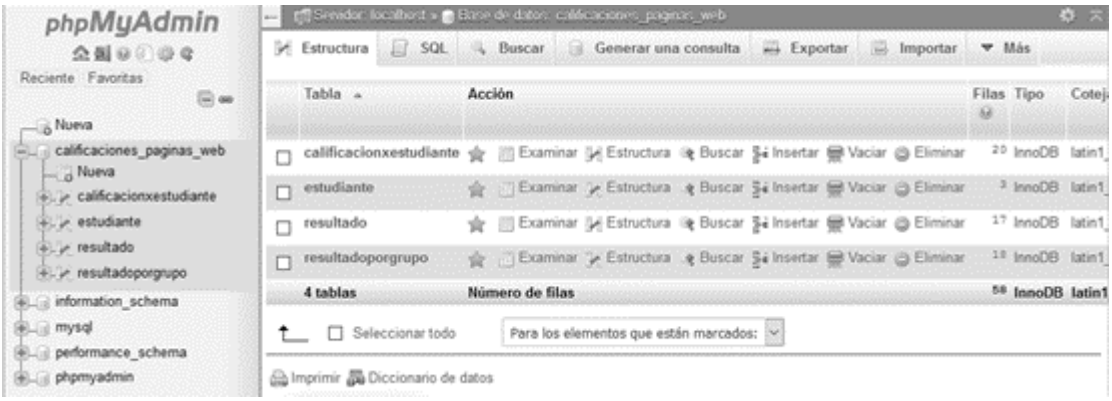


Figura 89 – phpMyAdmin - Base de Datos "calificaciones_paginas_web"

Una vez mostradas las opciones para importar, se debe seleccionar la opción: "Buscar en su ordenador", presionar el botón "Examinar", dirigirse a la carpeta "*june*" dentro de "D:\DVD_JUNE" y seleccionar el archivo sql "*calificaciones_paginas_web.sql*" y presionar el botón aceptar. De vuelta en la pantalla de importación presionar el botón "continuar". De esta manera se importará la base de datos de la aplicación.

V.2. INSTALACIÓN DE LA PARTE PHP DEL PROTOTIPO

Para instalar la parte PHP del Prototipo de Metabuscador “June” solo se deben hacer dos pasos.

El **primero** de ellos es tener en cuenta la carpeta utilizada por el S.O. para publicar los sitios, como ser “/home/juneTF/public_html/” como se muestra en la Figura 90.

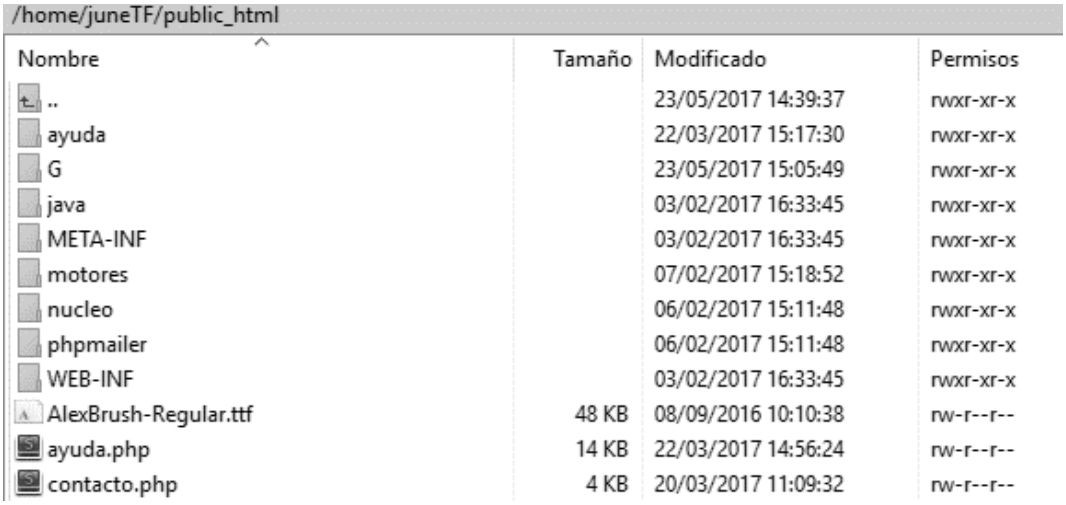


Figura 90 - Instalación Parte PHP - Crear Carpeta "june"

Luego se debe copiar todo el contenido que se encuentra dentro de la carpeta “php” ubicada en “D:\DVD_JUNE\june\”, como se muestra en la Figura 91, cuya ruta original es “D:\DVD_JUNE\june\june_php”, en el directorio “public_html” (/home/juneTF/public_html/), como se muestra en la Figura 92.

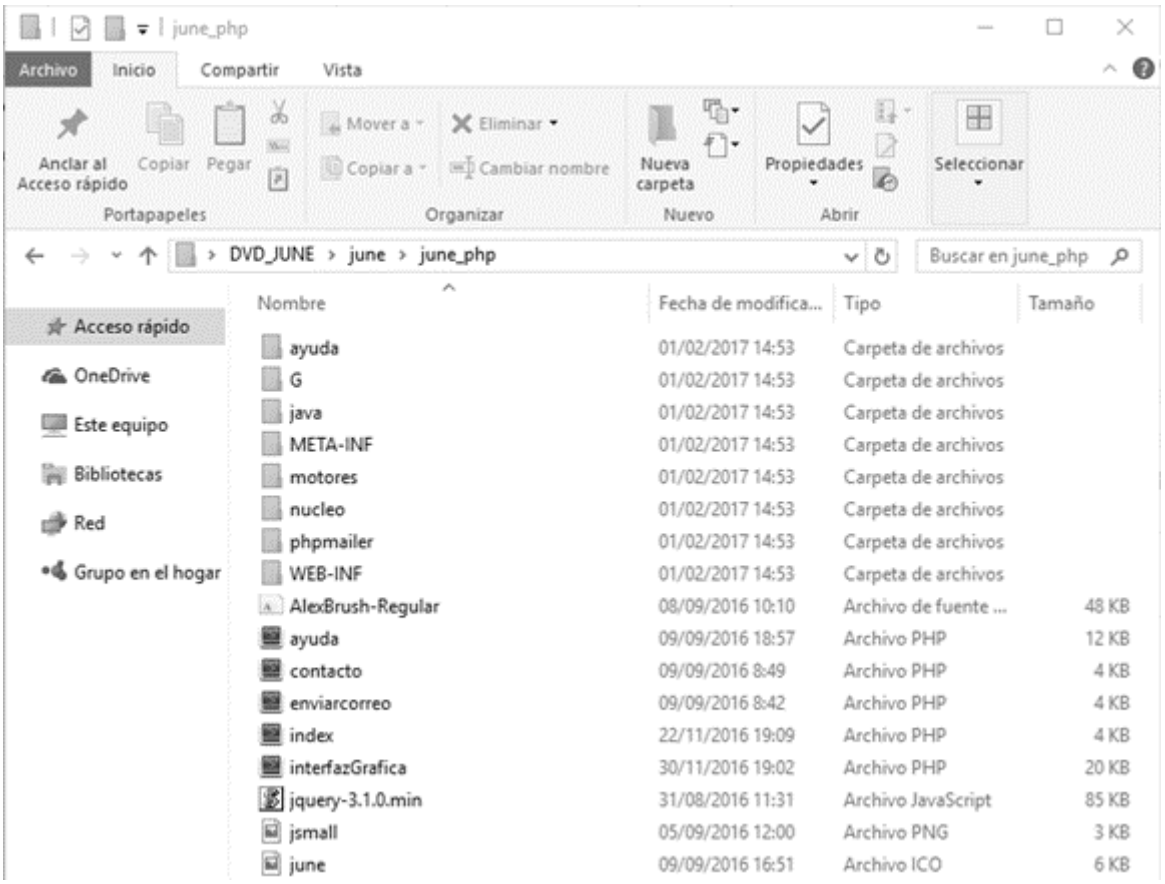


Figura 91 - Instalación Parte PHP - Contenido de "june" de “june_php”

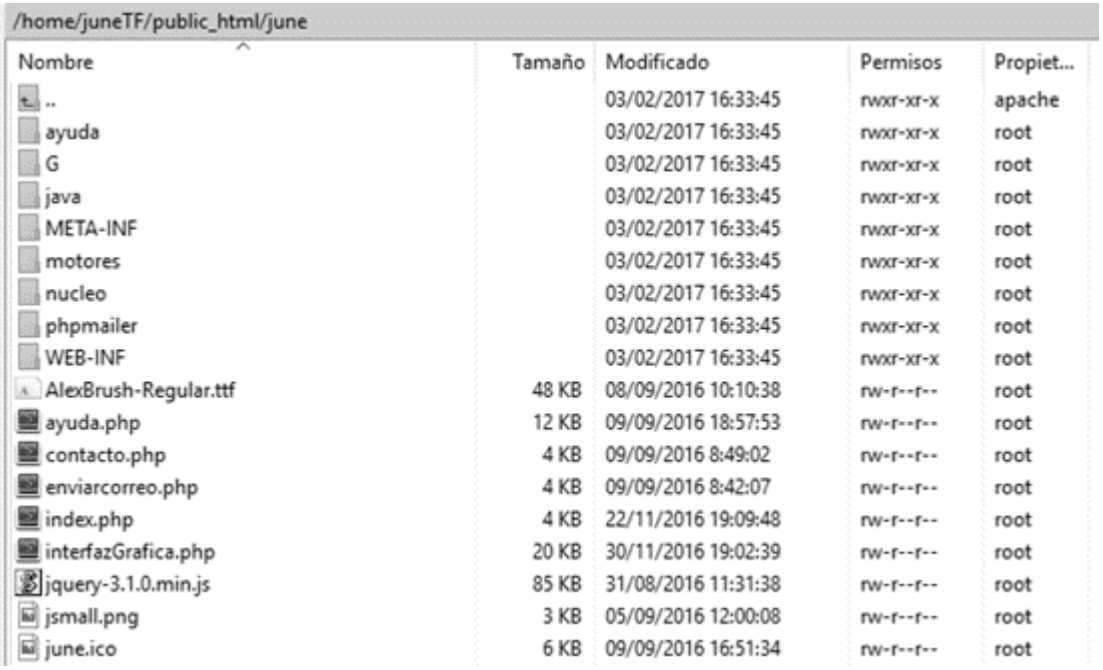


Figura 92 - Instalación Parte PHP - Contenido de "june" de "public_html"

Cabe aclarar que para copiar y pegar los archivos se puede utilizar una herramienta conocida como WinSCP que es un cliente SFTP gráfico para Windows que emplea SSH y

cuya función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos como ser uno local y el otro remoto.

El hosting que se contrate o un servidor propietario deben tener un gestor para transferencia de archivos o al menos un servicio como SSH. El S.O. Linux CentOS de la máquina virtual tiene habilitado el servicio de SSH y el puerto 33133 para dicho servicio cuyo usuario tiene como nombre y contraseña:

Usuario: juneTF Contraseña: 04091983

El **segundo** paso es editar el archivo "Java.inc" situado en "/home/juneTF/public_html/java", como se muestra en la Figura 93.

Figura 93 - Instalación Parte PHP - Edicion de archivo "Java.inc"

La edición se la puede hacer con cualquier editor de texto. Lo importante es que se modifique una línea de dicho archivo. Debe editarse la línea 7, como se muestra en la Figura 94.

De la línea 7 debe modificarse el número (puerto) "8080" situado después de los dos puntos (":") por el puerto utilizado por el servicio de TOMCAT 8.

define ("JAVA_HOSTS", "127.0.0.1:**8080**");

El **valor subrayado y en negrita** indica el puerto que utiliza TOMCAT 8. Si ese número "8080" no es el puerto utilizado por Tomcat, borrarlo y agregar el que SI utiliza. Normalmente el puerto 8080 es por defecto, es decir que si es ese el puerto utilizado entonces **NO** modificar el archivo.

```
<?php
# Java.inc -- The PHP/Java Bridge PHP library. Compiled from JavaBridge.inc.
# Copyright (C) 2003-2009 Jost Boekemeier.
# Distributed under the MIT license, see Options.inc for details.
# Customization examples:
# define ("JAVA_HOSTS", 9267); define ("JAVA_SERVLET", false);
  define ("JAVA_HOSTS", "127.0.0.1:8080");
# define ("JAVA_HOSTS", "ssl://my-secure-host.com:8443");
define ("JAVA_SERVLET", "/june/servlet.phpjavabridge");
# define ("JAVA_PREFER_VALUES", 1);

if(!function_exists("java_get_base")) {
1.0E512;
function java_get_base() {
$ar=get_required_files();
$arLen=sizeof($ar);
if($arLen>0) {
$thiz=$ar[$arLen-1];
return dirname($thiz);
} else {
return "java";
}
```

Figura 94 - Instalación Parte PHP - Edición de 2 líneas del archivo "Java.inc"

Una vez finalizada la edición deben guardarse los cambios.

V.3. INSTALACIÓN DE LA PARTE JAVA DEL PROTOTIPO

Aclaración: La dirección o ruta de Tomcat 8 puede variar según donde se haya instalado en el servidor propietario o hosting contratado. En el S.O. Linux CentOS de la máquina virtual, Tomcat 8 está instalado en la ruta: /usr/local/tomcat8.

No es necesario, pero para evitar inconvenientes, se recomienda copiar el archivo "mysql-connector-java-5.1.39-bin.jar" situado en "D:\DVD_JUNE\june\", en la carpeta "lib" de Tomcat 8 ("/usr/local/tomcat8/lib/"). Una vez copiado el archivo, la ruta al mismo debe ser: /usr/local/tomcat8/lib/mysql-connector-java-5.1.39-bin.jar

Para instalar la parte JAVA del Prototipo de Metabuscador "June" se debe ingresar al Administrador de Tomcat 8. Para ello ingresamos la siguiente dirección en la barra de direcciones del navegador:

http://192.168.1.123:8080/

Nota: Tener en cuenta que la dirección IP 192.168.1.123 es la correspondiente a la que viene en la máquina virtual. Si se ha cambiado dicha dirección por una nueva, entonces esta nueva IP es la que debe ingresarse en la barra de direcciones del navegador. Por ejemplo:

http://XXX.XXX.XXX.XXX:8080/

Donde XXX.XXX.XXX.XXX es la nueva dirección IP de la máquina virtual.

Hecho esto aparecerá en el navegador lo que muestra la siguiente Figura 95. Para instalar la parte java de June debemos hacer click en el botón “Manager App”.

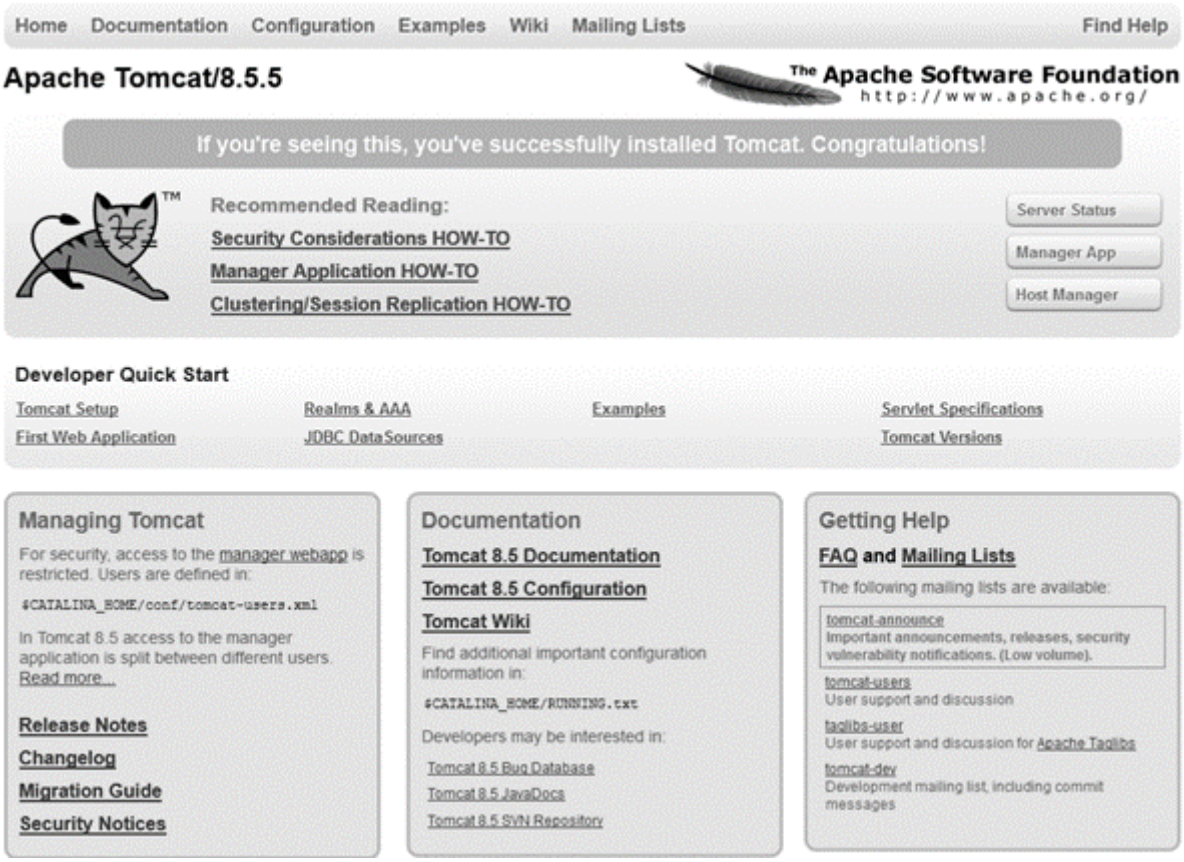


Figura 95 - Instalación Parte JAVA - Administrador de Tomcat 8

Luego de hacer click en el botón “Manager App”, se solicitará que ingrese el nombre de usuario y la contraseña del administrador de Tomcat 8. Se completa con los siguientes datos y luego se hace click en el “Aceptar”, como se muestra en la Figura 96.

Nombre de Usuario: **admin** Contraseña: **manager**

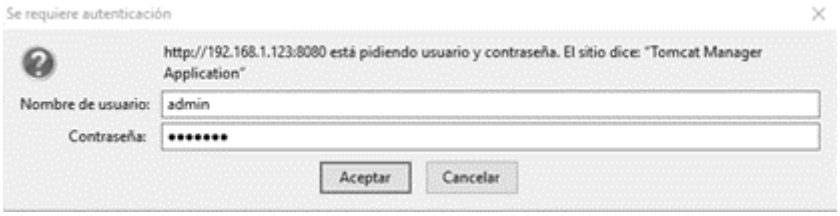


Figura 96 - Instalación Parte JAVA - Usuario y Contraseña del Administrador de Tomcat 8

Luego de hacer click en “Aceptar” aparecerá, como se muestra en la Figura 97, el Gestor de Aplicaciones Web de Tomcat 8.

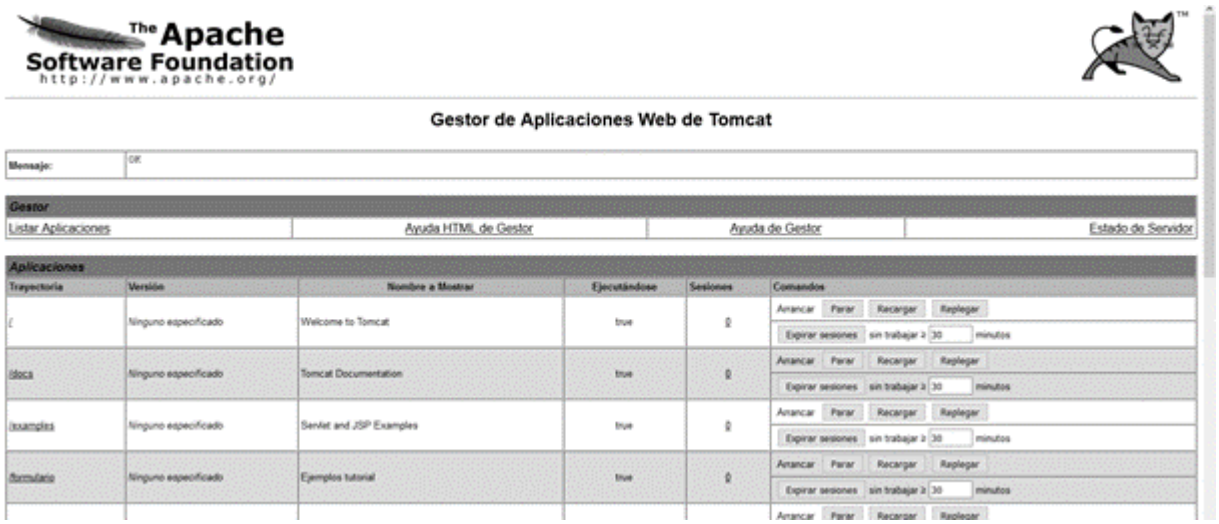


Figura 97 - Instalación Parte JAVA - Gestor de Aplicaciones Web de Tomcat 8

Desplazarse por la página hacia abajo hasta encontrar la parte de “Archivo WAR a desplegar”, y hacer click en el botón “Examinar”, como se muestra en la Figura 98.

Figura 98 - Instalación Parte JAVA - Archivo WAR a desplegar

Luego de hacer click en “Examinar”, se solicitará seleccionar el archivo war a cargar. Se debe seleccionar el archivo “june.war” de la ruta D:\DVD_JUNE\june\june_java\, y hacer click en “Abrir”, como se muestra en la Figura 99.

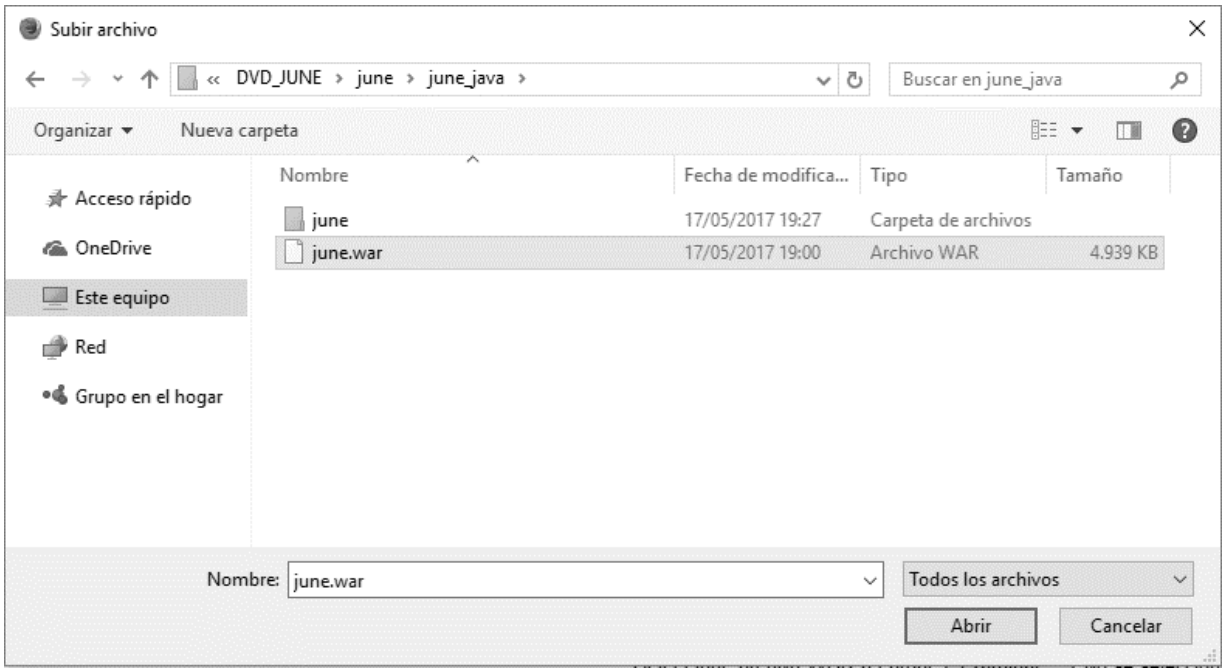


Figura 99 - Instalación Parte JAVA - Subir archivo june.war

Una vez clickeado el botón “Abrir”, se mostrará nuevamente el Gestor de Aplicaciones y en la parte de “Archivo WAR a desplegar” aparecerá el nombre del archivo seleccionado “june.war”, como se muestra en la Figura 100.

Figura 100 - Instalación Parte JAVA - Archivo WAR a desplegar seleccionado

Al hacer click en “Desplegar” se debe esperar un momento hasta que se termine de instalar la parte JAVA del Prototipo de Metabuscador “June”. Una vez finalizada la instalación, se mostrará el listado de aplicaciones del Gestor de Aplicaciones Web de Tomcat 8 incluía la aplicación “june”, como se muestra en la Figura 101.

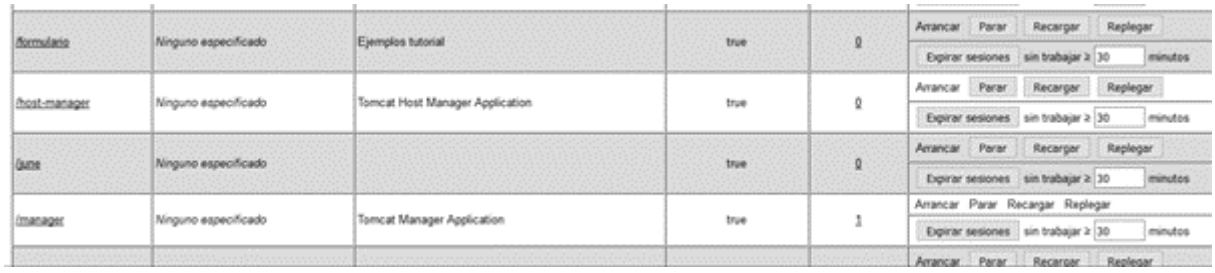


Figura 101 - Instalación Parte JAVA - Listado de Aplicaciones Web con "june" incluído

Hecho esto, finaliza la instalación de la parte JAVA del Prototipo de Metabuscador “June”.

ANEXO III

MANUAL DE CONFIGURACION

- JUNE -

INDICE

INDICE DE FIGURAS

I. INTRODUCCIÓN

El siguiente documento corresponde al manual de configuración de June, el metabuscador colaborativo basado en agentes para grupos de estudiantes. El objetivo de este manual es ofrecer a los propietarios o encargados del metabuscador las operaciones básicas para su administración, las cuales consisten en iniciar los agentes, crear los grupos y definir los usuarios integrantes de cada grupo, y por último apagar el servidor.

II. METABUSCADOR JUNE

June nace como propuesta al trabajo final presentado por quienes redactaron este mismo documento. Es un metabuscador colaborativo basado en agentes para apoyar las tareas de grupos de estudiantes. De manera general, el metabuscador permite realizar búsquedas de material en la web, revisando, calificando y/o comentando los resultados mostrados en el listado de resultados de búsqueda devuelto. Dicho listado está rankeado de acuerdo con la calificación general de cada resultado, la cual se obtiene como promedio de las calificaciones individuales de los integrantes del grupo. También puede consultar la ayuda sobre cómo se maneja el metabuscador o contactar con los desarrolladores ante cualquier consulta. Por último, puede consultar el historial de resultados de búsquedas del grupo.

III. ACLARACION IMPORTANTE

Al momento de realizar el presente documento, el metabuscador June, se encuentra instalado en dependencias de la Facultad de Exactas y Tecnologías, de la Universidad Nacional de Santiago del Estero, puede ser accedido desde fuera de la red LAN y posee el siguiente dominio para las urls que serán utilizadas para acceder al metabuscador y a sus funciones:

june.unse.edu.ar

Ahora bien, si el metabuscador June fuese instalado para ser accedido únicamente desde una red interna LAN (Red de Área Local) entonces sería necesario realizar un pequeño cambio en las urls que se muestran en el presente documento. Dicho cambio se realizaría sobre el dominio de las urls, reemplazándolo por la dirección IP del Servidor donde fue instalado June.

En el “Manual de Instalación” se proponen alternativas de instalación. La “Alternativa I” es la más sencilla y que además ya viene con la siguiente configuración de red por defecto:

Dirección IP del Servidor: **192.168.1.123**

Mascara de Subred: **255.255.255.0**
Puerta de Enlace: **192.168.1.1**
DNS Principal: **192.168.1.1**

Entonces a modo de simplificar la lectura de este manual, toda url cuyo dominio sea “june.unse.edu.ar”, estará seguidamente acompañada por una url entre paréntesis pero con el dominio reemplazado por la dirección IP 192.168.1.123.

IV. CONFIGURACION DE JUNE

IV.1. INICIALIZACIÓN DE LOS AGENTES

June es una aplicación web basada de agentes de software. Para poner en funcionamiento estos agentes debe ejecutarse una serie de pasos consecutivos una vez que el servidor esté completamente encendido y se haya cargado en su totalidad el Sistema Operativo (SO).

Aclaración: A lo largo de este manual se procederá a explicar la configuración de June implementada sobre un Servidor Linux CentOS 6.8.

IV.1.1. Iniciar Sesión en el SO con el usuario específico para June

Se inicia sesión en el servidor seleccionando el usuario “juneTF” e ingresando como contraseña “04091983”, como se muestra en la Figura 102.

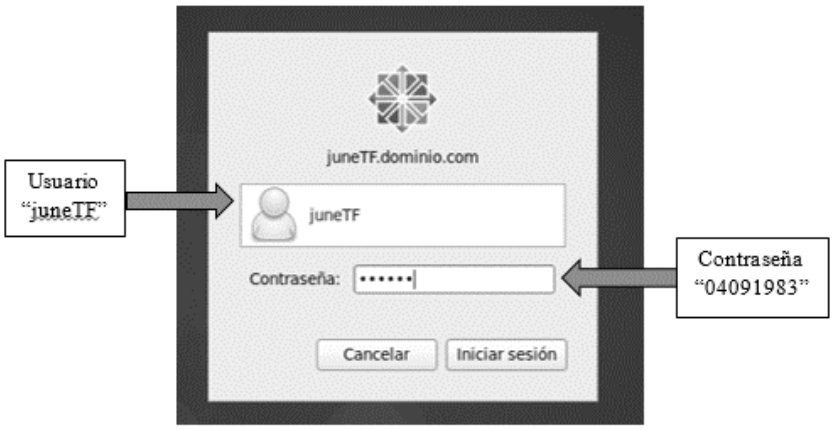


Figura 102 - Inicio de sesión en el usuario juneTF

IV.1.2. Iniciar la plataforma Jade

La plataforma Jade es en donde se van a cargar e iniciar los agentes de software. Una vez iniciada la sesión del usuario June, se debe ingresar en la carpeta "Jade" como se muestra en la Figura 103, haciendo doble click.

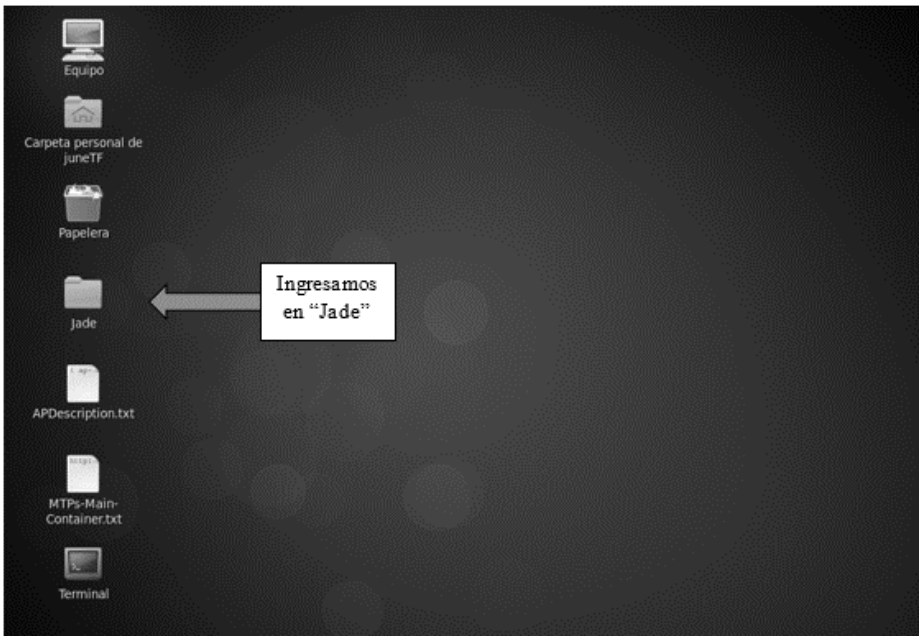


Figura 103 - Ingreso en la carpeta Jade

Una vez dentro de la carpeta "Jade", se debe ejecutar el archivo "jade.sh" haciendo doble click sobre él. Cuando se haga doble click, aparecerá una pantalla como la que se muestra en la Figura 104.

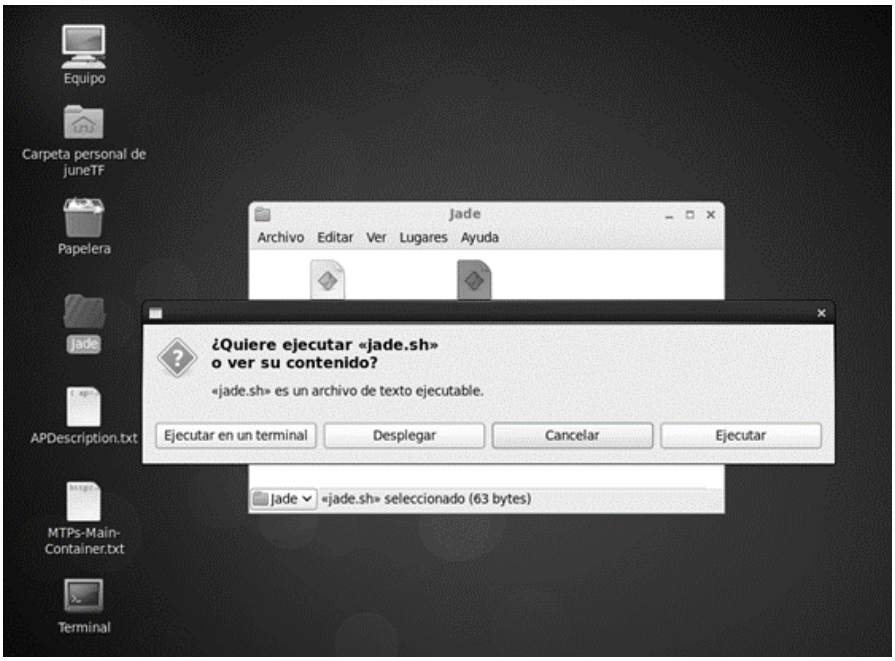


Figura 104 - Ejecución del archivo "jade.sh"

Se debe hacer click en “Ejecutar en un terminal”.

June trabaja con dos agentes de software, uno de “Consulta” y el otro de “Actualización”. Cuando se ejecute el archivo “jade.sh”, se abrirá una ventana con la plataforma JADE Remote Agent Management GUI iniciada (como se muestra en la Figura 105), así como también ya estará cargado e iniciado el agente de “Consulta” de manera automática en la plataforma. Luego de manera manual debe cargarse e iniciarse el agente de “Actualización” en la plataforma.

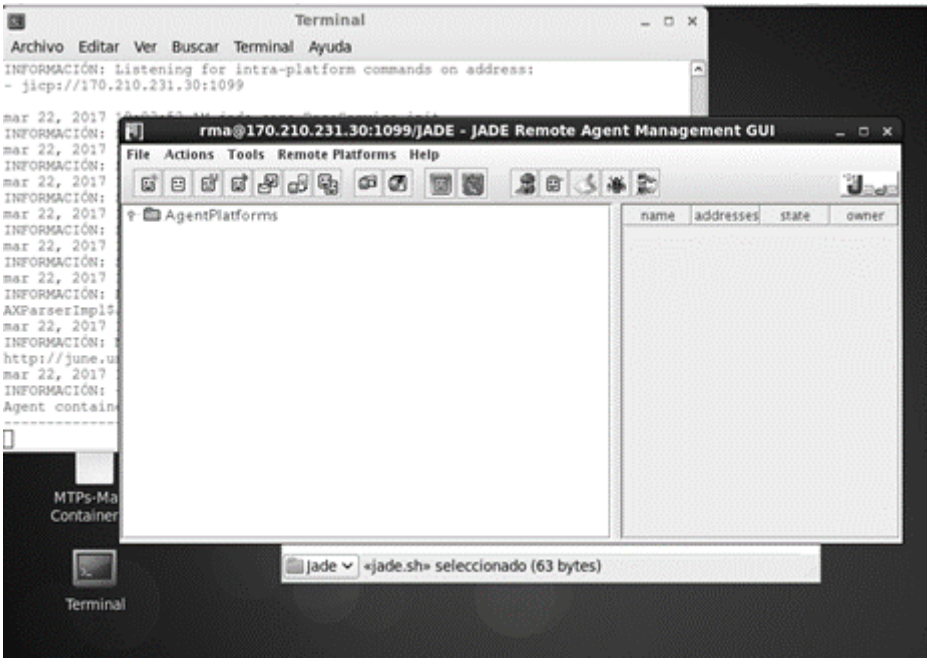


Figura 105 - Finalización de la ejecución del archivo "jade.sh"

Nota: Para mayor información sobre el uso de todos los recuros del “JADE Remote Agent Management GUI”, ir a la página oficial de JADE

En la parte izquierda de la consola, se puede ver un árbol que muestra que dentro hay una plataforma llamada “170.210.231.30: 1099/JADE”, que está compuesta por un Containter llamado “Main-Container”. Dentro de dicho Container, existen ejecutándose 4 agentes: 3 que vienen por defecto (ams, df y rma) y el agente de consulta cargado e iniciado para el metabuscador. Todo esto puede observarse en la Figura 106.

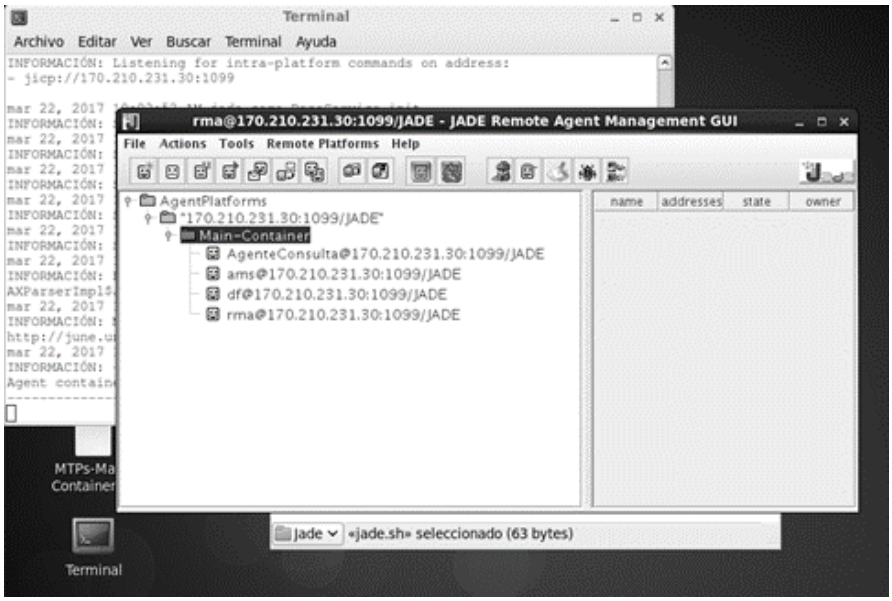


Figura 106 - Agentes funcionando en el Container

IV.1.3. Iniciar el Agente de Actualización en la plataforma Jade

Para cargar e iniciar el agente de Actualización en la plataforma, se debe hacer click derecho sobre “Main-Container” y seleccionar “Start New Agente”. Se abrirá la ventana “Insert Start Parameters”. En el parámetro “Agent Name” colocamos el nombre “AgenteActualizacion”. Se debe respetar el nombre como muestra la Figura 107, porque de otro modo el metabuscador no funcionará. Luego de ingresar el nombre, debemos seleccionar el “Class Name”. Para ello hacemos click en el botón “…” y aparecerá la ventana “Select Agent Class”. Allí seleccionamos “agentes.AgenteActualizacion” y luego presionamos “Ok”.

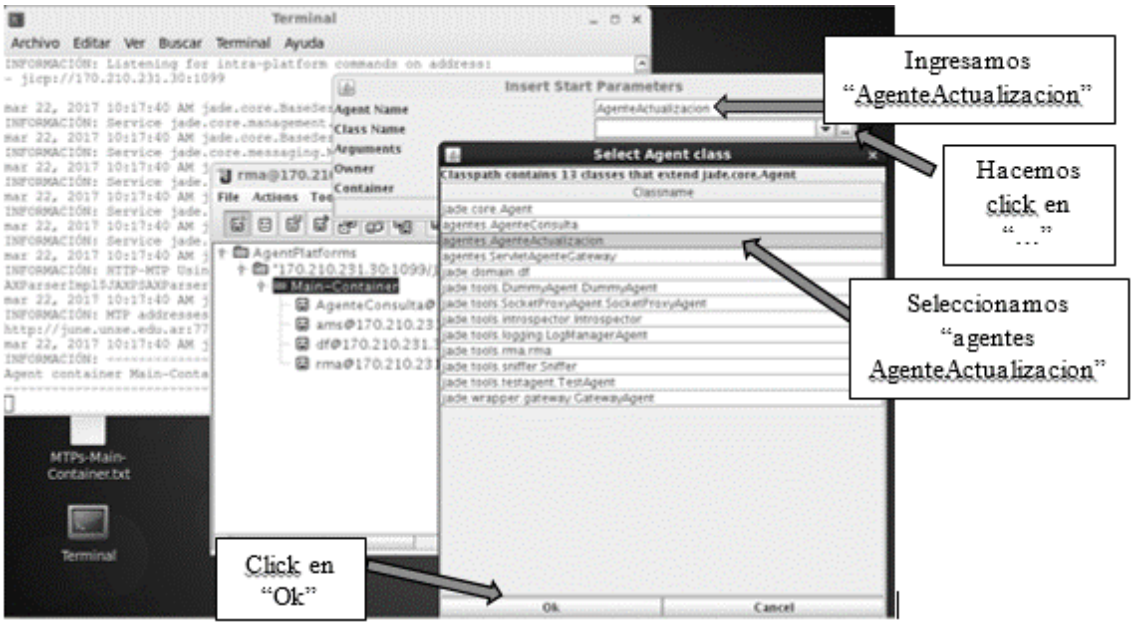


Figura 107 - Creación del Agente Actualizacion

Luego de realizar estas acciones, se hace click en “Ok” en la ventana de “Insert Start Parameters”. A continuación se puede observar que en el “Main-Container” se cargó e inició el Agente de Actualización. A partir de aquí **ya se puede utilizar el Metabuscador**.

Nota: No se deben cerrar ni la interfaz gráfica ni la terminal

IV.2. CREACIÓN DE GRUPOS Y USUARIOS PARA IDENTIFICACIÓN EN EL METABUSCADOR

Como los desarrolladores presentan un prototipo de Metabuscador, la creación de grupos y usuarios se hace de manera manual, incluyendo archivos en una carpeta del Metabuscador.

Para ello, luego de iniciar sesión en el servidor como se explica en el punto anterior, debemos ingresar a las siguientes carpetas: “Carpeta personal de juneTF/public_html/G”. Pasos a seguir para llegar a la carpeta “G”:

1º - Entrar en la carpeta llamada “Carpeta personal de juneTF” ubicada en el escritorio.

2º - Luego, en la ventana que aparece, entrar en la carpeta llamada “public_html”.

3º - Por último, en la nueva venta que aparece, entrar en la carpeta llamada “G”.

“G” es la carpeta asignada que contendrá todos los grupos. Como se puede ver en la Figura 108, ya se encuentran creados los grupos “1” y “2”.

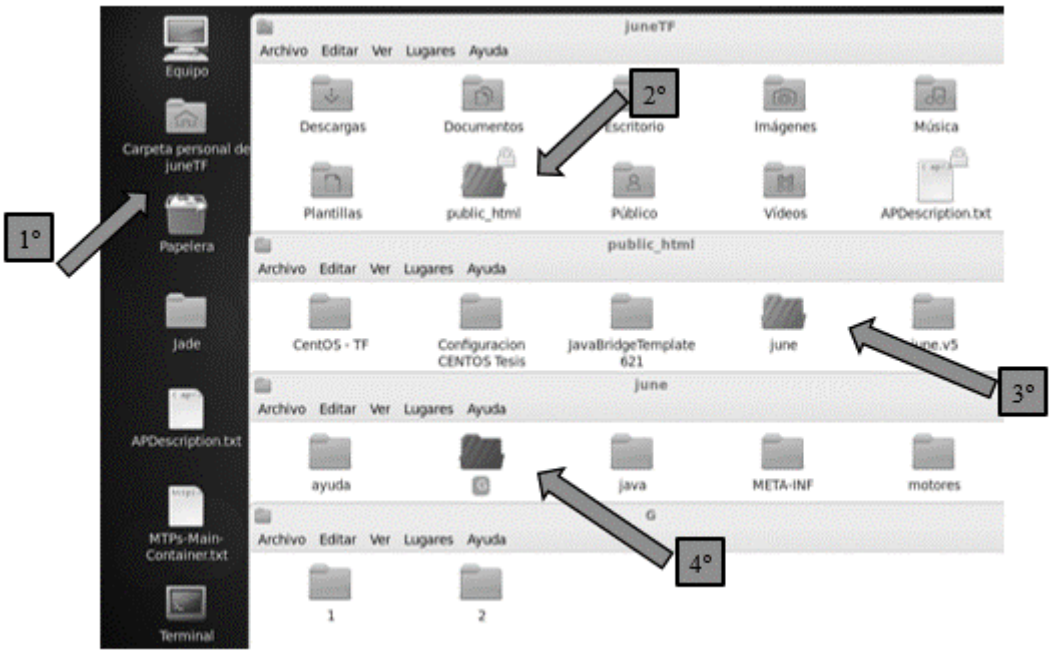


Figura 108 - Conjunto de Grupos Existentes

Para agregar un nuevo grupo es solo necesario crear una nueva carpeta con el número de grupo que se quiera agregar. Para ello se debe hacer click derecho dentro de la carpeta, seleccionar “Crear un folder” y luego colocarle el número que represente al grupo. En la Figura 109 se puede ver como se crearon los grupos 3, 4 y 5.

Figura 109 - Creación de grupos 3, 4 y 5

Nota: A modo de ejemplo se crearon grupos identificados únicamente por números, pero también es posible crear carpetas y nombrarlas de manera que identifiquen a los grupos por materia y número de grupo, es decir, por ejemplo: IA-G1 haciendo referencia al grupo número uno de la materia “Inteligencia Artificial”

Si se ingresa a la carpeta 3, se pueden ver los archivos de los usuarios (en este caso, estudiantes) que están asignados a dicho grupo como se muestra en la Figura 110. Estos archivos identifican al usuario dentro del grupo, en este caso dentro del grupo 3.

Es necesario que los archivos respeten la estructura del nombre:

numeroLegajo_añoLegajo.php

Para agregar nuevos usuarios solo basta con crear un nuevo archivo cuyo nombre será el número de legajo del estudiante y la extensión “.php”, además de incorporar dentro del archivo las siguientes líneas:

```
<?php
$grupoUsuario = explode(“/”, substr(dirname($_SERVER[‘PHP_SELF’]),1))[2];
$idUsuario = explode(“.”, array_pop(explorer(‘/’, $_SERVER[‘PHP_SELF’])))[0];
header(“Location:
../../index.php?key=”.base64_encode($grupoUsuario.”|”,$idUsuario.”|june@30030_2008&
june@30089_2003”));
?>
```

También se puede agregar nuevos usuarios copiando el archivo de otro existente cambiándole el nombre por el legajo que corresponda.

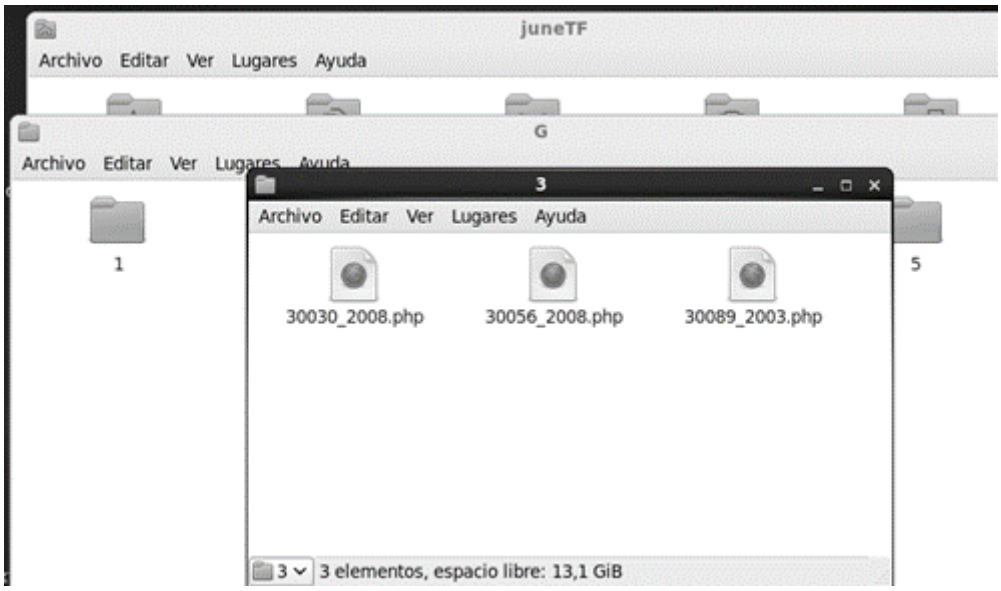


Figura 110 - Creación de Usuarios por Grupo

Ahora bien, esto se maneja de esta manera para la identificación del estudiante y su grupo en el Metabuscador, pero si se observa atentamente los nombres de los archivos .php, en ningún lado se proporciona el nombre de usuario, apellido, ni ningún otro dato. Esta parte debe ser cargada en la Base de Datos.

Para cargar la Base de Datos, se debe contar con conexión a internet e ingresar, mediante cualquier navegador (Google Chrome, Mozilla Firefox, Safari, etc.) a la siguiente dirección:

http://june.unse.edu.ar/phpmyadmin
(http://192.168.1.123/phpmyadmin)

En la página de inicio de dicha dirección, aparecerá el inicio de la interfaz para el manejo de la Base de Datos. Se deben ingresar el usuario (root) y contraseña (root@june) en los campos correspondientes, y luego presionar “Continuar” como se muestra en la Figura 111.

Figura 111 - Inicio de sesión en la Base de Datos

Luego de iniciar sesión, se tendrá a disposición un conjunto de Bases de Datos para trabajar, como se muestra en la columna izquierda de la Figura 112.

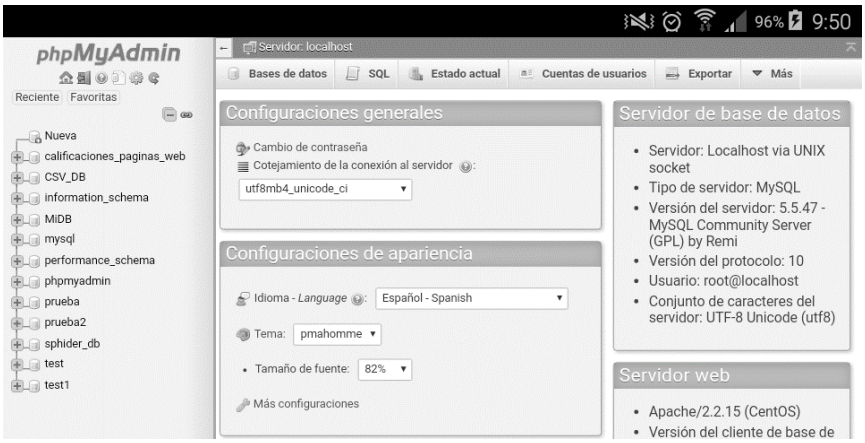


Figura 112 - Inicio de phpMyAdmin para el usuario "root"

La Base de Datos creada para el Metabuscador es la que recibe el nombre de "calificaciones_paginas_web". Al hacer click en el enlace de esta Base de Datos, se abrirá el conjunto de tablas relacionadas con el Metabuscador en la columna derecha. La que incumbe en este momento para la creación de estudiantes es aquella tabla que recibe el nombre de "estudiante", como se muestra en la Figura 113.

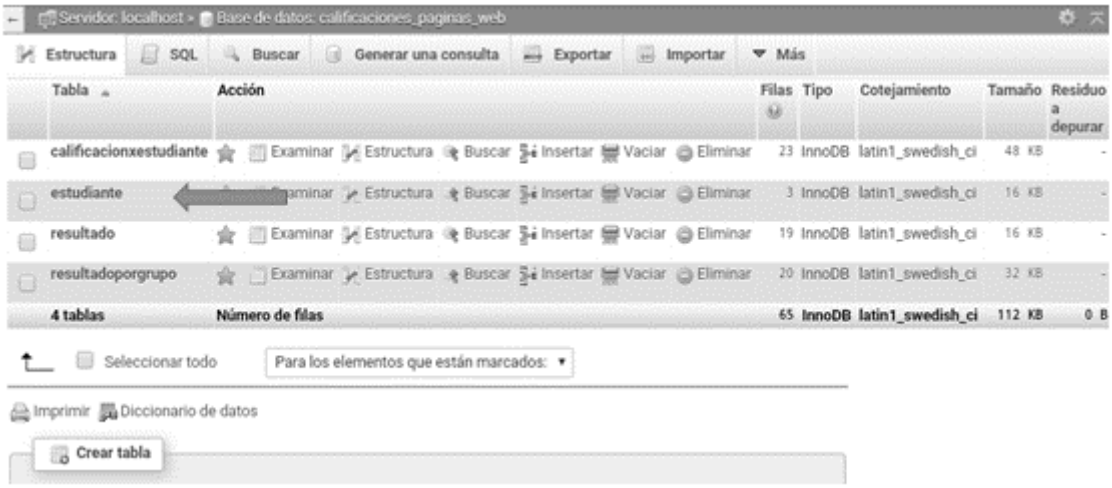


Figura 113 - Tablas de la Base de Datos "calificaciones_paginas_web"

Para ingresar un estudiante en la tabla estudiante, se debe hacer clic sobre ella y luego cuando se accede a la tabla, hacer click en “Insertar” en la barra superior de dicha imagen, como se muestra en la Figura 114.

Figura 114 – Insertar estudiante en la tabla "estudiante"

Como se puede observar en la Figura 115, para ingresar un nuevo estudiante solo basta con ingresar los datos requeridos: Legajo, Apellido y Nombre. En dicha figura se puede observar un ejemplo.

Figura 115 - Ingreso de datos de un estudiante

Luego de ingresar los datos del estudiante, se presiona “Continuar” y para ver que correctamente se cargó el nuevo estudiante, se debe presionar sobre “Examinar”, ubicado en la barra superior de la columna derecha, como se muestra en la imagen anterior. Los resultados se muestran en la Figura 116.

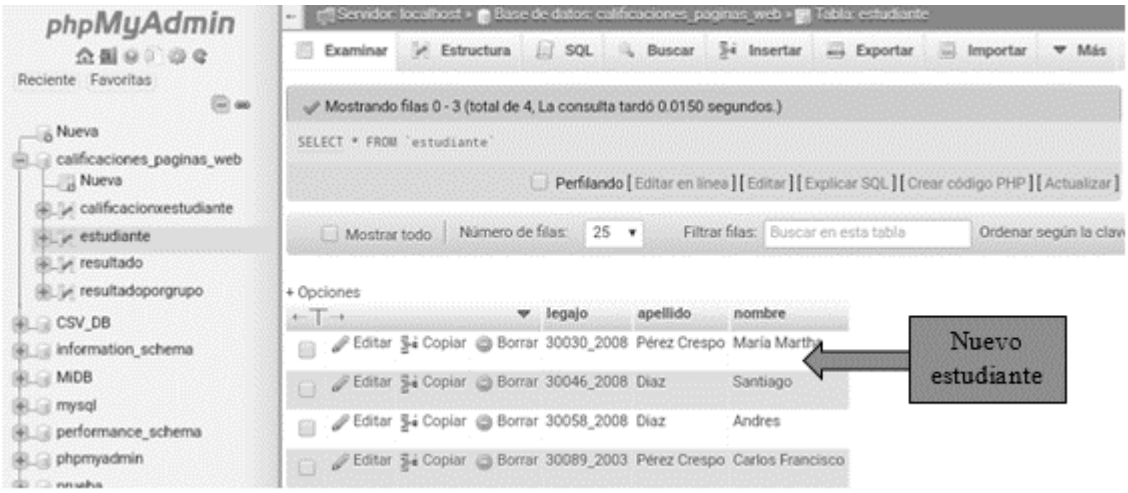


Figura 116 - Tabla actualizada de estudiante

IV.3. APAGAR EL SERVIDOR

Si por alguna eventualidad es necesario apagar el Servidor, es importante primero apagar los agentes. Para ello hay que volver al entorno de trabajo de los agentes “JADE Remote Agent Management GUI” y en la barra de menú seleccionamos “File/Shut Down Agent Plataform”. Luego se hace click en “Si”, como se muestra en la Figura 117.

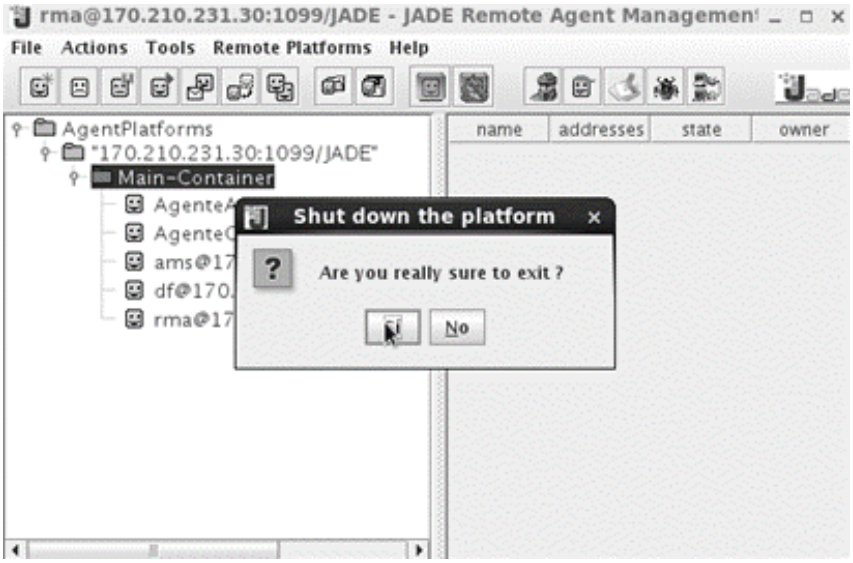


Figura 117 - Apagado de Agentes

Para apagar el servidor vamos a la barra de herramientas del sistema, se hace click en “Sistema/Apagar…”, como se muestra en la Figura 118.

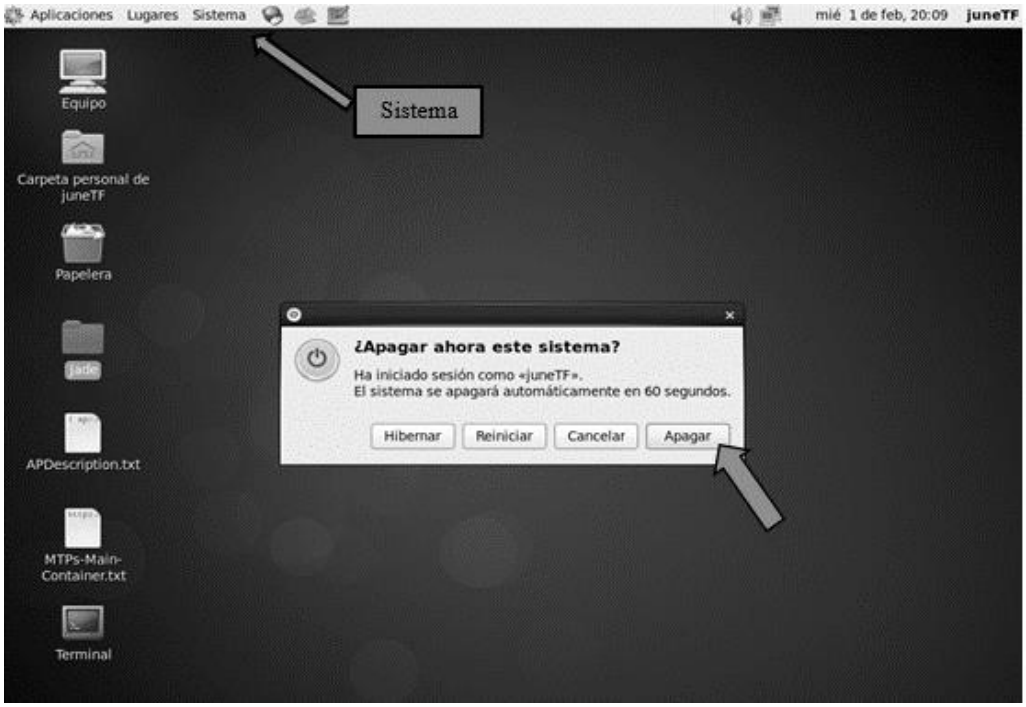


Figura 118 - Apagado del servidor

Cuando se hace click en “Apagar”, por cuestiones de seguridad, el servidor pide el ingreso de la contraseña del usuario root para corroborar que realmente es él quien desea apagar el servidor. Para ello, se debe ingresar “root@root” en el recuadro, como se muestra en la Figura 119.

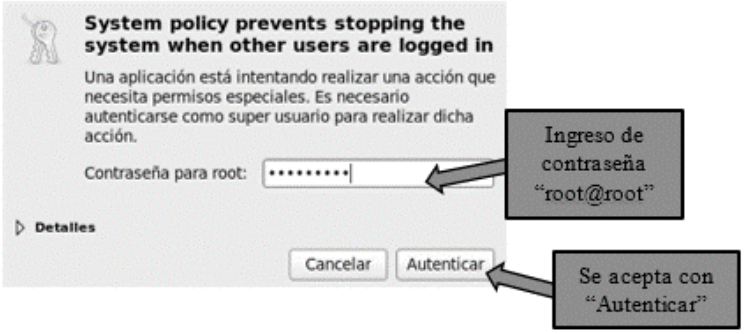


Figura 119 - Confirmación de apagado del servidor

ANEXO IV

MANUAL DE USUARIO

- JUNE -

INDICE

INDICE DE FIGURAS

I. INTRODUCCIÓN

El siguiente documento corresponde al manual de usuario para el manejo de June, el metabuscador basado en agentes para grupos de estudiantes colaborativos. El objetivo de este manual, es que se aprovechen todos los recursos disponibles que propone el metabuscador.

Este documento ofrece una guía básica que permite, a los estudiantes/profesores que interactúen con dicho metabuscador, familiarizarse con el manejo de las opciones que proporciona.

II. METABUSCADOR JUNE

June nace como propuesta al trabajo final presentado por quienes redactaron este mismo documento. Es un metabuscador basado en agentes para apoyo de grupos de estudiantes colaborativo. De manera general, el metabuscador permite realizar una búsqueda de material en la web, revisando, calificando y/o comentando los resultados mostrados en el listado de resultados de búsqueda devuelto por el metabuscador. Dicho listado estará rankeado de acuerdo con la calificación general de cada resultado, la cual se obtiene como promedio de las calificaciones individuales de los integrantes del grupo. También puede consultar la ayuda sobre cómo se maneja el metabuscador o contactar con los desarrolladores ante cualquier consulta. Por último, puede consultar el historial de resultados de búsquedas del grupo de trabajo.

III. ACLARACION IMPORTANTE

Al momento de realizar el presente documento, el metabuscador June, se encuentra instalado en dependencias de la Facultad de Exactas y Tecnologías de la Universidad Nacional de Santiago del Estero, puede ser accedido desde fuera de la red LAN y posee el siguiente dominio:

june.unse.edu.ar

Ahora bien, si el metabuscador June fuese instalado para ser accedido únicamente desde una red interna LAN (Red de Área Local) entonces es necesario realizar un pequeño cambio en las urls que se muestran en el presente documento. Dicho cambio se realiza sobre el dominio de las urls, reemplazándolo por la dirección IP del Servidor donde fue instalado June.

En el “Manual de Instalación” se proponen alternativas de instalación. La “Alternativa I” es la más sencilla y que además ya viene con la siguiente configuración de red por defecto:

Dirección IP del Servidor: **192.168.1.123**

Mascara de Subred: **255.255.255.0**
Puerta de Enlace: **192.168.1.1**
DNS Principal: **192.168.1.1**

Entonces a modo de simplificar la lectura de este manual, toda urls cuyo dominio es “june.unse.edu.ar”, estará seguidamente acompañada por una url entre paréntesis pero con el dominio reemplazado por la dirección IP 192.168.1.123.

IV. ¿CÓMO ACCEDER AL METABUSCADOR?

Para acceder a la aplicación web June se recomienda utilizar los navegadores Google Chrome y Mozilla Firefox, ya que con otros no se garantiza una buena experiencia de uso.

En la barra de direcciones del navegador, el usuario debe ingresar una url (dirección web) que lo identifique a él y al grupo con el que va a trabajar. Dicha url debe ser solicitada a los propietarios o encargados de June.

Esa url es la única manera que un usuario tiene de acceder a June, y debe tener la siguiente estructura:

http://june.unse.edu.ar/G/**GRUPO**/NUMERO_LEGAJO.php
(http://192.168.1.123/G/**GRUPO**/NUMERO_LEGAJO.php)

A modo de ejemplo, a continuación se presenta un ejemplo de la url que debe ingresarse en la barra de dirección del navegador.

http://june.unse.edu.ar/G/LSI-IA/30089_2003.php

(http://192.168.1.123/G/LSI-IA/30089_2003.php)

En el caso de que se ingrese incorrectamente la url proporcionada por los propietarios/encargados de June, o se quiera acceder a otra sección de June o una vez accedido a June haya ocurrido un tiempo de inactividad (tiempo que no se use el metabuscador) aparecerá el siguiente mensaje que se muestra en la Figura 120.

Figura 120 - Inicio del Metabuscador

V. INGRESANDO AL METABUSCADOR

Una vez que el metabuscador identificó al estudiante (quién es y en qué grupo está), éste puede comenzar a utilizar la aplicación. Como muestra la Figura 121, en la pantalla de inicio se da la opción de ingresar la "Frase de búsqueda" o "Ver Resultados Calificados". Si ninguno de los integrantes calificó resultados anteriormente, entonces la página de "Ver Resultados Calificados" se mostrará vacía.

Figura 121 - Página Inicial del Metabuscador con usuario registrado

VI. METABUSCADOR

Dentro del metabuscador, propiamente dicho, se ofrece la opción de ingresar la frase de búsqueda y tildar el/los motor/es de búsqueda para realizar la búsqueda de material en la

web. El metabuscador requiere que al menos uno de los motores esté seleccionado. Luego del ingreso y la selección, se puede presionar el botón de “Buscar” (Figura 122).

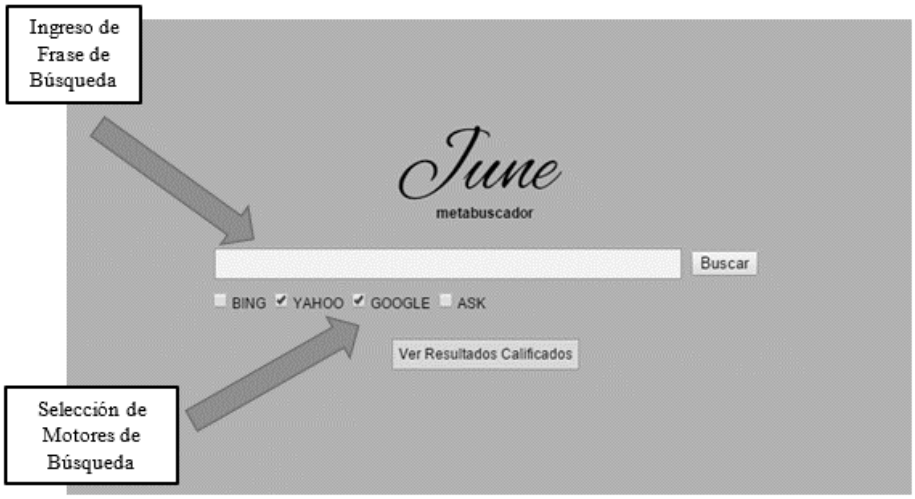


Figura 122 – Búsqueda

Una vez que la lista de resultados es presentada en pantalla, como se muestra en la Figura 123, se habilitan nuevas opciones. El estudiante puede entrar a cada uno de los resultados de la lista, y una vez que realizó el análisis correspondiente, puede calificar y/o comentar dicho resultado.

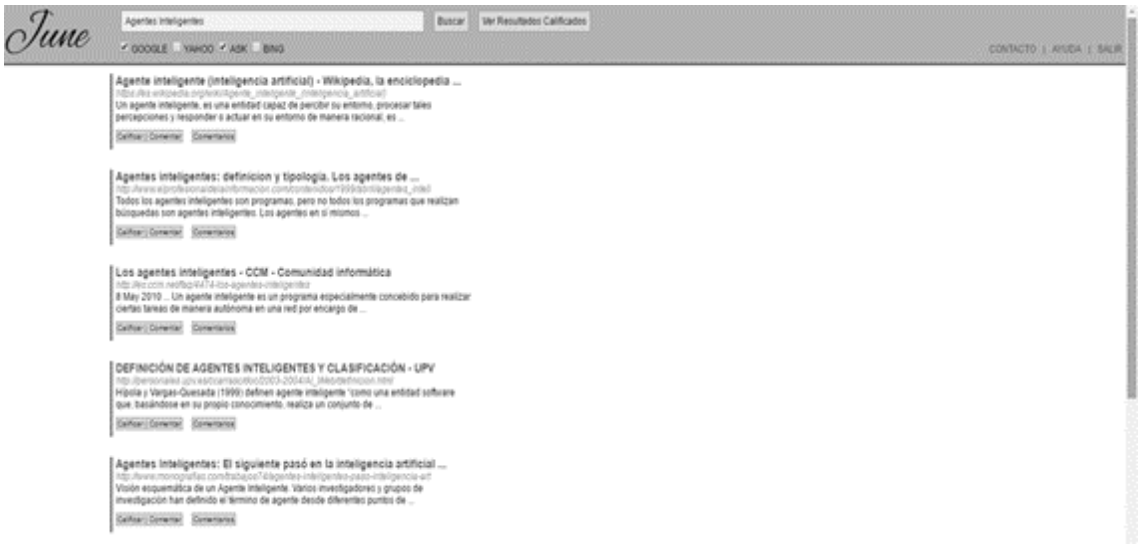


Figura 123 – Ejemplo de Búsqueda

Para calificar un resultado se debe presionar el botón “Calificar/Comentar”. Al hacer esto ser desplegarán las opciones en dos columnas, como se puede observar en la Figura 124. La izquierda columna de la izquierda cuenta con un conjunto de estrellas acorde a la puntuación que el estudiante desee ingresar para el resultado, y la columna derecha permite

ingresar el comentario. Una vez realizada/s la/s operación/es, el estudiante debe seleccionar la opción “Finalizar Valoración”.

Figura 124 - Opciones de Calificar y/o Comentar

Cada integrante del grupo puede calificar y/o comentar los resultados de la búsqueda realizada, sin importar que otro integrante haya calificado y/o comentado anteriormente. Además, cada resultado contará con una calificación general la cual resultará de ir promediando el conjunto de calificaciones individuales dadas sobre un mismo resultado. A la derecha de cada resultado, como se muestra en la Figura 125, se pueden observar las calificaciones individuales de los integrantes que calificaron dicho resultado. La calificación de cada resultado es obligatoria, mientras que el comentario que se realice sobre el mismo, es opcional. Si no se desea ingresar un comentario, es suficiente con dejar el recuadro en blanco.

Figura 125 - Búsqueda de otro integrante

El listado de resultados de búsqueda además cuenta con la particularidad de presentar los resultados ordenados de acuerdo con las calificaciones generales de los mismos, mostrando primero los "calificados" y luego los "sin calificar", como se muestra en la Figura 125.

> Nota: La calificación general es por grupo. Puede ser que otro grupo también califique un determinado resultado, pero eso no influirá en la calificación general de resultado para el grupo en cuestión.

El estudiante también puede rever los comentarios de un aporte presionando en el botón "Comentarios", como se muestra en la Figura 126.

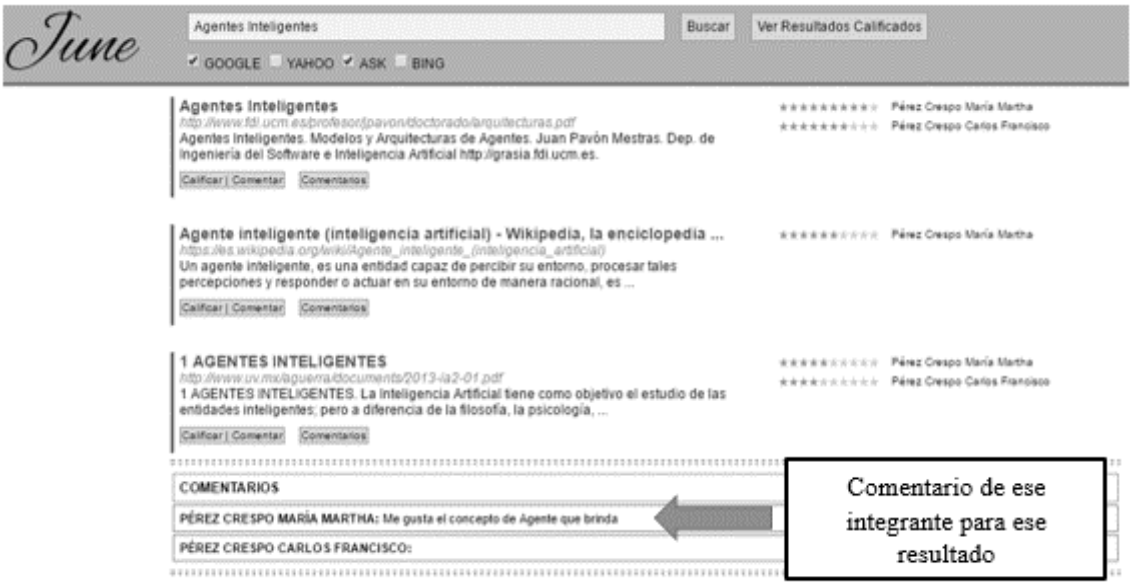


Figura 126 - Comentario de un integrante a un resultado

Además de calificar y/o comentar, el usuario también puede modificar dichos ingresos. El estudiante puede presionar nuevamente el botón de “Calificar/Comentar” y la aplicación le brindará nuevamente las dos columnas: estrellas para la valoración y un recuadro para ingresar el comentario. Como puede observarse en la Figura 127, el metabuscador devuelve los aportes ingresados con anterioridad por el usuario, de manera que el estudiante puede reverlos y determinar los cambios apropiados que quiera realizarle (ya sea modificar su calificación, su comentario o ambos). Luego, se pueden ingresar nuevos parámetros y, por último presionar “Finalizar Valoración” para terminar la modificación.

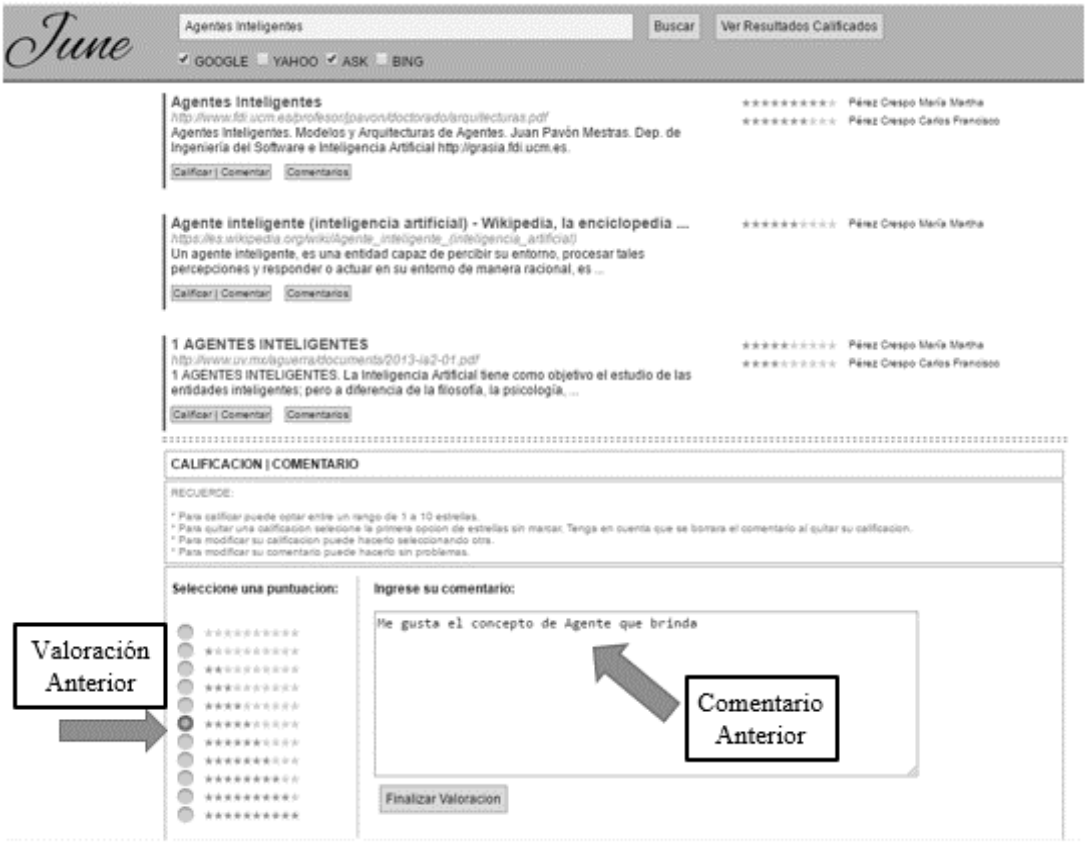


Figura 127 - Modificación de Calificación y/o Comentarios

El metabuscador ofrece la opción de eliminar una calificación (y con ello un aporte a un resultado). Para eso, se debe seleccionar la calificación en “0”, como se muestra en la Figura 128.

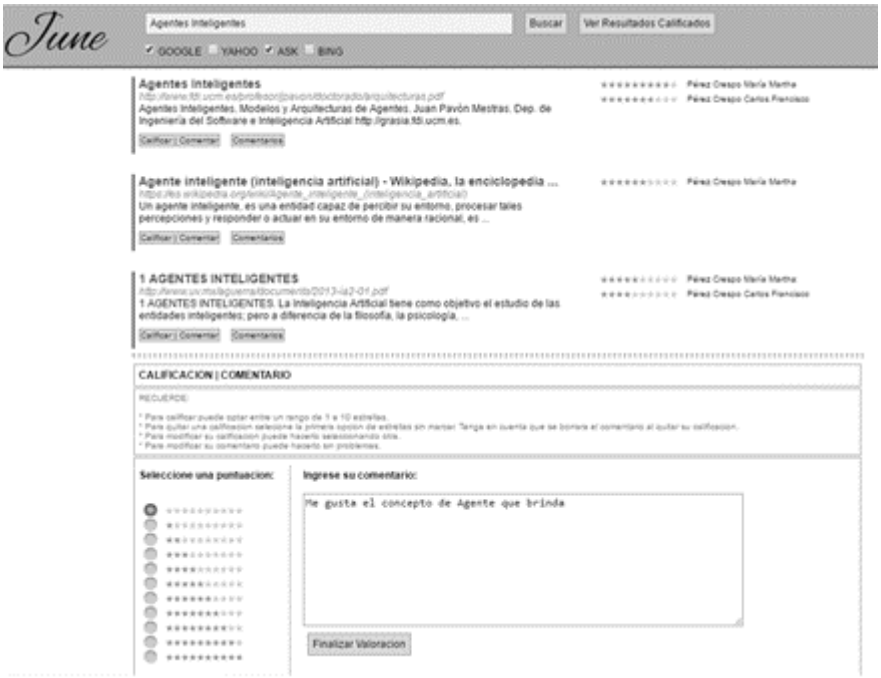


Figura 128 - Recuperación del aporte de un integrante.

Una vez que se presiona “Finalizar Valoración” el aporte realizado por el integrante desaparece para ese resultado, como se muestra en la Figura 129.

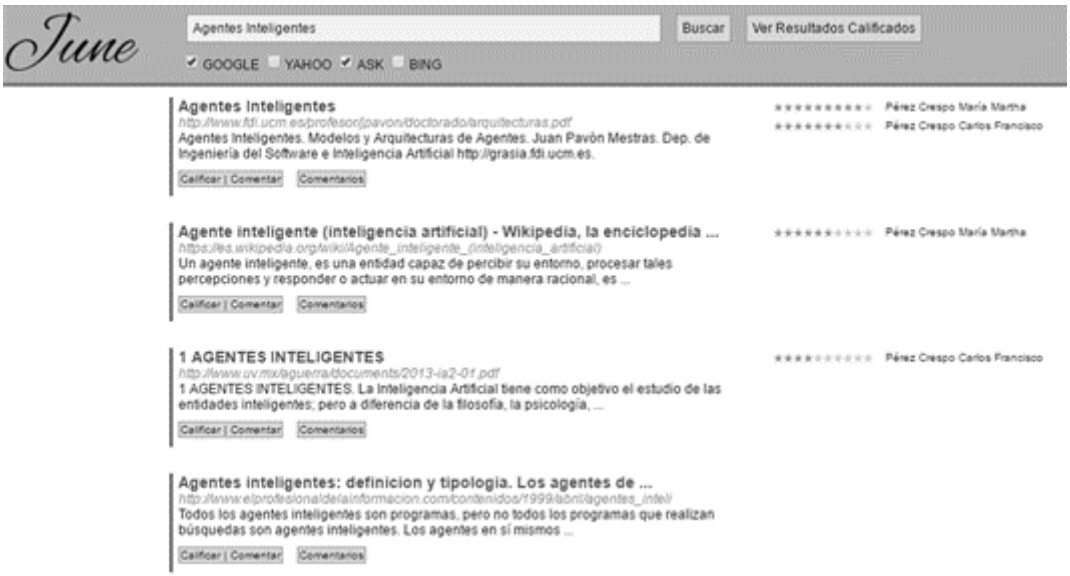


Figura 129 - Eliminación del aporte de un integrante

Dado que el listado de resultados devuelto por el metabuscador en cada búsqueda es demasiado extenso, a manera de organización se ofrece, en el margen inferior, una barra de navegación organizada por el número de listado de resultados más una letra. De ese modo se puede acceder a todos los resultados de búsqueda de la primera búsqueda realizada para una determinada “frase de búsqueda”. Si se desea hacer una segunda búsqueda con la misma “frase de búsqueda”, solo basta con tocar el número siguiente al del listado de búsqueda. Por ejemplo: Si se está navegando en el listado de búsqueda “1”, se mostrará el listado 1 en sublistados “1.A”, “1.B”, “1.C”… y para realizar una nueva búsqueda, basta con presionar el “2” en la barra de navegación. Esto se puede apreciar en la Figura 130.

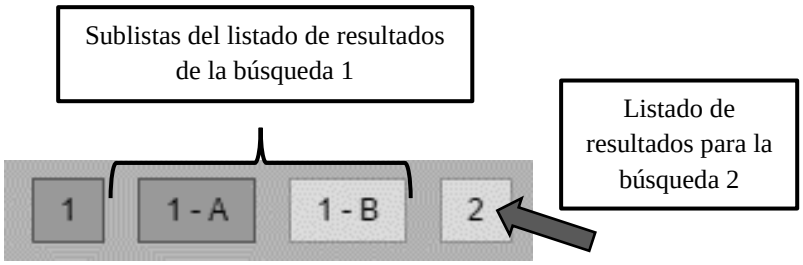


Figura 130 - Barra de Navegación

Nota: No se rankea el listado entero. En pantalla, el metabuscador ofrece los resultados de la búsqueda ordenados para un determinado sublistado.

VII. AYUDA

En esta sección del metabuscador se encuentra en el margen superior derecho de la aplicación, como puede mostrarse en la Figura 131.

Figura 131 - Opción de Ayuda

Los desarrolladores ofrecen un conjunto de recomendaciones para que el estudiante pueda entender el manejo del metabuscador y optimizar, si corresponde, la manera en la que realiza la búsqueda. Una breve introducción puede mostrarse en la Figura 132


June

Ayuda

Para mejorar su experiencia al momento de utilizar la aplicación se recomienda leer los siguientes parrafos.

Acceso a JUNE

Como JUNE surge de un proyecto de Trabajo Final de Graduación, solo fue prototipado con sus funciones principales que hacen a su funcionamiento el correcto para el proyecto.

Entre las limitaciones que JUNE posee, las más relevantes son en cuanto a la identificación del estudiante y los filtros de búsqueda.

En cuanto a los filtros, debe tenerse en cuenta que cada buscador opera la frase de busqueda a su manera pudiendo o no coincidir entre ellos. JUNE no opera sobre la frase de busqueda, pero da la posibilidad de elegir aquellos buscadores (de los que dispone JUNE) de los que el estudiante tiene conocimiento sobre como operan la frase de busqueda, y en la caja de texto donde se ingresa la frase de búsqueda puede ingresarse algún filtro que sea conocido por el estudiante. Es decir, se selecciona el buscador o los buscadores que trabajan con el mismo filtro, ingresan la frase de búsqueda y el filtro y por ultimo hacen la búsqueda de esa forma.

Es el caso de google y yahoo que utilizan el filtro "ext:pdf" al final de la frase de búsqueda para dar un listado de resultados referido a sitios con PDF. Entonces, por ejemplo, si queremos para buscar PDFs sobre Inteligencia Artificial podemos ingresar la siguiente frase de busqueda en el metabuscador: "Inteligencia Artificial ext:pdf"

En cuanto a la identificación del estudiante, se realiza a través del ingreso de la URL de JUNE.

Cada estudiante que desee utilizar la aplicación JUNE debe darse de alta comunicandose con los desarrolladores de la aplicación. Ellos le darán de alta en el sistema y le proveeran a cada estudiante que lo solicite, una dirección url única para acceder a la aplicación. Dicha dirección está compuesta por el dominio donde esta alojada la aplicación, por la identificación y/o numero de grupo y por el numero de legajo del estudiante. Es decir, la dirección url de cada estudiante debería tener la siguiente form genérica:

www.dominio.com/GA/idGrupo_nGrupo/legajo_estudiante.php

- www.dominio.com: Ubicación donde se encuentra el metabuscador JUNE.
- legajo_estudiante: Numero de Legajo de un Estudiante.
- idGrupo_nGrupo: Es el nombre y número que identifica a un grupo de estudiantes colaborativos, donde idGrupo puede ser siglas de una materia y nGrupo es el número de grupo.

Figura 132 - Inicio de la sección ayuda

VIII. CONTACTO CON EL DESARROLLADOR

En esta sección, el estudiante puede contactar con los desarrolladores de la aplicación ante cierto error que pueda surgirle, alguna consulta relacionada con su uso, o alguna cuestión que no esté disponible en la sección de "*Ayudar*". Dicha opción se encuentra ubicada en el margen superior derecho de la aplicación, como se muestra en la Figura 133.

Figura 133 - Opción de Contacto

Para utilizar esta opción se debe proporcionar a la aplicación los datos de nombre, apellido, correo electrónico del remitente y el asunto, para luego explicar de manera precisa cuál es el inconveniente, como se muestra en la Figura 134.

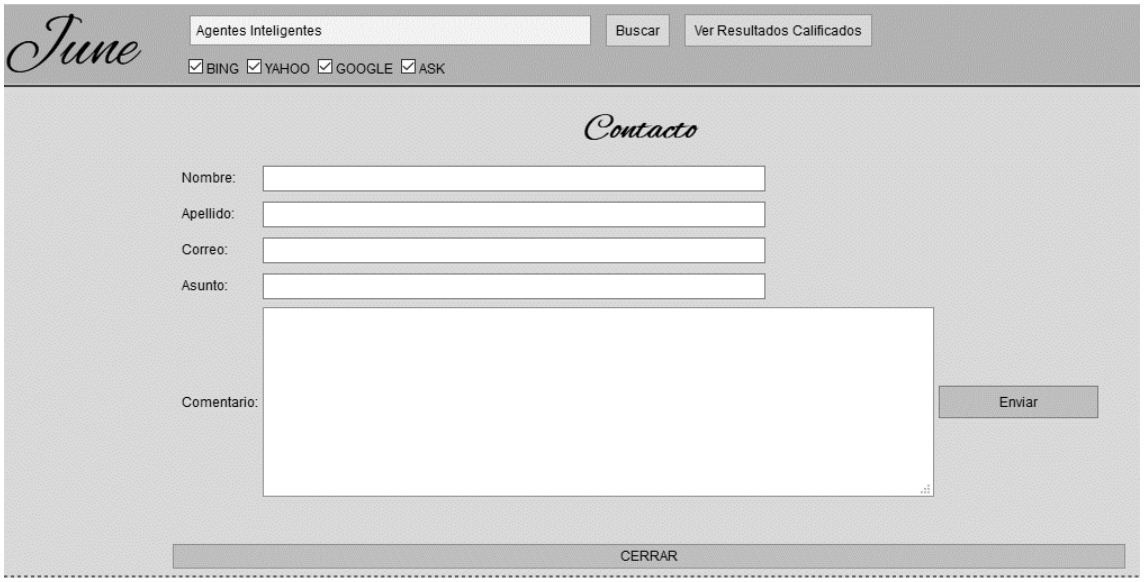


Figura 134 - Datos para contactar a los desarrolladores

IX. HISTORIAL DE BÚSQUEDAS

La web es muy grande y la forma de buscar que cada integrante es única. Esto da la posibilidad de que no siempre puedan coincidir las búsquedas realizadas por cada integrante. Para ello la aplicación ofrece un historial de búsqueda, ingresando al presionar el botón de "Ver Resultados Calificados". En esta sección se ofrecen todos los resultados de búsqueda calificados, con o sin comentarios, de todos los integrantes del grupo, rankeados por su calificación general. Los estudiantes podrán revisar cada resultado, así como también calificarlos y/o comentarlos, además de modificar y eliminar dichas calificaciones/comentarios, tal cual como se puede realizar en el metabuscador. Para acceder a él, existen dos formas: en la página principal de June, antes de realizar cualquier búsqueda, luego de realizar la búsqueda, al lado de la barra de búsqueda.

Un ejemplo de este historial de búsqueda, puede verse en la Figura 135 que se muestra a continuación:

Figura 135 - Historial de Búsqueda ranqueados por su calificación general

X. SALIR

Una vez que se haya terminado de realizar la búsqueda, se debe cerrar sesión. Para ello, se debe hacer click en la barra superior derecha, en la opción "SALIR", como se muestra en la Figura 136.

Figura 136 - Opción de Salir

ANEXO V

CUESTIONARIOS

- JUNE -

CUESTIONARIO PARA DOCENTE

1) ¿Cree que June es fácil de usar para los estudiantes?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……

……

……

2) ¿Cree que June es útil para trabajos en grupo?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……

……

……

3) El hecho de que los estudiantes puedan observar las calificaciones, con o sin comentarios, ¿Cree que impacta en su motivación y/o productividad?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……

……

……

4) El hecho de que los estudiantes puedan calificar y agregar su comentario a los resultados de búsqueda ¿Cree que impacta en su motivación y/o su productividad?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……

……

……

5) ¿Cree que los documentos que devolvió el metabuscador contribuyeron a mejorar la calidad del trabajo en los grupos?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……
……
……

6) ¿Cree que el uso del metabuscador permitió a los estudiantes disponer de fuentes de información en cantidad apropiada?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……
……
……

7) ¿Cree que deberían tenerse en cuenta otras consideraciones? Es decir, ¿Mejoraría algún aspecto de June?

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……
……
……

CUESTIONARIO PARA ESTUDIANTE

PARTE 1

1) Materia a la que pertenece

☐ Base de Datos ☐ Criptografía ☐ Inteligencia Artificial

2) ¿Qué navegador utiliza con mayor frecuencia?

☐ Google ☐ Bing ☐ Yahoo ☐ Ask ☐ Otros

Si escogió Otros, escriba cuales: …….

3) ¿La organización del contenido del metabuscador le parece correcto?

☐ SI ☐ NO

4) ¿La presentación de los resultados es de su agrado? Es decir, como los resultados están ordenados priorizando aquellos con mayor calificación.

☐ SI ☐ NO

Si su respuesta es NO, explique por qué

……
……
……

Respecto a los Resultados que han sido calificados

5) ¿Es de su agrado que el valor de las calificaciones realizadas sobre los resultados se representen con estrellas?

☐ SI ☐ NO

Si su respuesta es NO, indicar cuál sería la mejor representación de los valores de las calificaciones para usted.

……
……
……

6) ¿La ubicación de las calificaciones hacia la derecha de los resultados es de su agrado?

☐ SI ☐ NO

Si la respuesta es NO, indicar cuál sería la mejor ubicación para usted.

……
……
……

7) ¿Es de su agrado la forma para acceder a los comentarios de cada resultado?

☐ SI ☐ NO

Si la respuesta es NO, indicar que otra forma sería posible.

……
……
……

8) ¿Es suficiente la información que se proporciona en los comentarios de cada resultado?

☐ SI ☐ NO

Si la respuesta es NO, indicar que otra información sería necesaria.

……
……
……

9) ¿La ubicación en la que aparecen los comentarios es de su agrado?

☐ SI ☐ NO

Si la respuesta es NO, indicar cuál sería la mejor ubicación para usted.

……
……
……

Respecto a la calificación individual de cada resultado

10) ¿Es de su agrado la forma en la que se accede a la opción de calificar y comentar cada resultado?

☐ SI ☐ NO

Si la respuesta es NO, indicar que otra forma sería posible.

……
……
……

11) ¿Es de su agrado que la calificación sobre un resultado se realice mediante la selección de estrellas?

☐ SI ☐ NO

Si la respuesta es NO, indicar que otra forma sería posible.

……
……
……

12) ¿La ubicación de las estrellas para calificar situadas a la izquierda, le parece agradable?

☐ SI ☐ NO

Si la respuesta es NO, indicar cuál sería la mejor ubicación para usted.

……
……
……

13) ¿La ubicación en la que aparece la caja de texto para ingresar su comentario hacia la derecha, es de su agrado?

☐ SI ☐ NO

Si la respuesta es NO, indicar cuál sería la mejor ubicación para usted.

……
……
……

PARTE 2

14) ¿Cree que utilizaría este metabuscador con frecuencia?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

15) ¿Encuentra al metabuscador innecesariamente complejo?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

16) ¿El metabuscador es fácil de utilizar?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

17) ¿Cree que necesita ayuda de otra persona para utilizar el metabuscador?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

18) ¿Encuentra las funciones del metabuscador bien integradas?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

19) ¿Considera que existen inconsistencias en el metabuscador?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

20) ¿Considera que la mayoría de los estudiantes aprenderían muy rápidamente a utilizar el metabuscador?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

21) ¿Encuentra que el metabuscador es muy grande para recorrelo?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

22) ¿Se sintió confiado al manejar el metabuscador?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

23) ¿Cree que es necesario aprender muchas cosas antes de manejar el metabuscador?

☐ En completo de desacuerdo ☐ Desacuerdo ☐ Tal vez ☐ De acuerdo ☐ Completamente de acuerdo

PARTE 3

24) ¿Considera usted que el metabuscador "June" se ajusta a su finalidad que es la de realizar búsquedas e indicar aquellos resultados calificados (con o sin comentarios) entre los compañeros de grupo, mostrando nombre, apellido, calificación y comentario de quienes calificaron?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……

25) ¿Considera usted que utilizando el metabuscador "June" es más rápido comunicar, a sus compañeros de grupo, los resultados que son importantes o no para usted, en lugar de utilizar WhatsApp, Facebook, Mensaje de Texto, una red social, etc.?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……

26) El hecho de que aparezcan los resultados de búsqueda con las calificaciones y los comentarios de sus compañeros de grupo ¿Lo incentiva a usted a buscar otro resultado que no haya sido calificado?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……

27) El hecho de que aparezcan los resultados de búsqueda con las calificaciones y los comentarios de sus compañeros de grupo. ¿Considera usted que el metabuscador "June" incentiva a que sus compañeros de grupo no ingresen a un resultado que ha sido

calificado (con o sin cometario) por usted, evitando tener que buscar y analizar la información dentro del sitio vinculado a ese resultado?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……

28) ¿Es suficiente la información brindada en la sección de ayuda?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……

29) ¿Le parecen suficiente los campos solicitados para contactarse con los desarrolladores?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……

30) ¿Considera usted que los resultados brindados por JUNE le fueron de utilidad para desarrollar la tarea?

☐ SI ☐ NO

Si la respuesta es NO, explicar por qué.

……
……
……