Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Desarrollo de un Motor de Búsquedas para la recuperación de documentos académicos de la Facultad de Ciencias

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por el Bachiller
José Rafael Guevara Salazar
para optar al título de
Licenciado en Computación

Tutor: Sergio Rivas

Caracas, 25 de octubre de 2013

Acta

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado con el título "Desarrollo de un Motor de Búsquedas para la recuperación de documentos académicos de la Facultad de Ciencias", el cual es presentado por el Bachiller José Rafael Guevara Salazar, de Cédula de Identidad 17.704.098, a los fines de optar al título de Licenciado en Computación, dejamos en constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 25 de octubre de 2013, a las 2:00pm en el aula PB-III, para que el autor lo defendiera en forma pública, mediante una presentación oral de su contenido, luego de lo cual se respondió las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los 25 días del mes de octubre del año 2013.

Prof. Sergio José Rivas Atanacio

(Tutor)

Prof. María Elena Villapol Blanco

(Jurado)

Prof. Néstor Alejandro Méndez Darias

(Jurado)

Resumen

El siguiente Trabajo Especial de Grado (T.E.G.) describe el proceso de desarrollo de un Motor de Búsquedas de publicaciones digitales para la Facultad de Ciencias de la Universidad Central de Venezuela. Primeramente se exponen diversos tópicos relativos a la Recuperación de Información, disciplina capaz de proveer conceptos, métodos y modelos necesarios para poder plantear soluciones al problema de la recuperación de información en general. Seguidamente, se presentan casos de plataformas en producción que permiten la recuperación de documentos digitales de corte académico.

Asimismo, luego se describe el proceso de análisis, diseño e implementación de BUSCONEST 2, un Motor de Búsquedas especializado en la recuperación de documentos académicos generados por miembros de la comunidad académica de la Facultad de Ciencias. El desarrollo de BUSCONEST 2 involucró tomar en cuenta una serie de aspectos generales presentes en el desarrollo de todo Motor de Búsquedas, tales como lo son la construcción de los índices del Motor de Búsqueda, el Ordenamiento o *Ranking* de documentos y la recuperación de los mismos. Además varios aspectos específicos debieron ser tomados en cuenta para el desarrollo de BUSCONEST 2. Dichos aspectos específicos consistieron en la construcción de una interfaz de administración desde la cual se puede gestionar y cargar documentos académicos de forma interactiva, la incorporación de un algoritmo de Ordenamiento de documentos que no sólo toma en cuenta la ocurrencia de los términos de la consulta en el contenido del documento, sino que además toma en cuenta otros datos relevantes como lo son la calificación obtenida, la fecha de publicación y la cantidad de descargas, el desarrollo de una funcionalidad de indexado de documentos por lotes de forma no interactiva, la creación de una conexión entre el sistema BUSCONEST 2 y CONEST para que los estudiantes que aprobaron su T.E.G. puedan cargar la versión digital del documento, y el proceso de migración de datos desde un repositorio anteriormente puesto en producción, referido en este documento como BUSCONEST 1, hacia el repositorio de BUSCONEST 2.

Palabras claves

BUSCONEST, *Search Engines*, Motor de Búsqueda, Recuperación de Información, documentos académicos, CONEST, *web*.

Índice general

Índice de tablas

Índice de figuras

Índice de algoritmos

Capítulo 1

Introducción

Actualmente la Facultad de Ciencias de la Universidad Central de Venezuela dispone de un repositorio de publicaciones digitales que permite el almacenamiento y recuperación de los trabajos de investigación desarrollados por los miembros de la comunidad académica. El mismo surgió por la necesidad de facilitar el acceso a la información contenida en las publicaciones digitales y por la necesidad de difundir la producción intelectual académica de la Facultad de Ciencias. De esta forma estudiantes y docentes pueden estar al tanto de los temas de investigación que se desarrollan en las diferentes áreas de conocimiento dentro de la facultad, a saber Computación, Biología, Matemáticas, Física, Química y Geoquímica. El desarrollo de dicho repositorio dio como resultado a BUSCONEST 1, un Motor de Búsqueda basado en la *web*. Se diseñó un Motor de Búsqueda basado en la *web* ya que los mismos han influenciado en gran medida la forma en que los usuarios conciben la recuperación de información, y por la gran difusión y alcance que adquieren los documentos publicados por este medio.

Hoy en día los Motores de Búsqueda tales como Google o Bing representan la metáfora más popular para recuperar documentos. De hecho, es tan habitual que los usuarios realicen la búsquedas de información mediante Motores de Búsqueda, que estos sistemas informáticos establecen patrones de navegación e interacción que luego son reproducidos por multitud de sistemas informáticos. Sin embargo, aunque resulte beneficioso reproducir las mejores prácticas de los Motores de Búsqueda más populares, se debe tomar en cuenta las necesidades propias de la comunidad académica a la hora de diseñar un Motor de Búsquedas para la Facultad de Ciencias. Dicho Motor de Búsquedas debe generar un valor agregado a los usuarios miembros de la Facultad de Ciencias en términos de búsquedas académicas, con respecto a realizar la recuperación de los documentos en un Motor de Búsqueda de propósito general.

Considerando el contexto descrito anteriormente se desarrolla BUSCONEST 2, un nuevo Motor de Búsquedas para la Facultad de Ciencias que permite el almacenamiento, el Ordenamiento, la recuperación, y la carga de documentos académicos digitales generados por miembros de la comunidad académica.

Cabe destacar que el Ordenamiento realizado por BUSCONEST 2 toma en cuenta datos generados por los miembros de la comunidad académica que no son tomados en cuenta por los Motores de Búsqueda de propósito general, tales datos son: la calificación, un dato académico generado por los docentes responsables de calificar el documento (T.E.G. o seminario) luego de un proceso de evaluación específico, la fecha de publicación en que el documento comenzó a estar disponible para la comunidad académica, que no es necesariamente la fecha en que el documento ingresó a los índices de BUSCONEST 2 (Como

ocurriría en un Motor de Búsquedas de propósito general) y la cantidad de descargas, que permite estimar aproximadamente que tan consultado ha sido el documento por otros miembros de la comunidad. BUSCONEST 2 también permite la carga de T.E.G. desde el sistema CONEST por parte de los alumnos próximos a graduarse, presenta una interfaz de administración donde se puede visualizar el estado general del repositorio y donde todos los documentos digitales pueden ser gestionados por parte de un usuario Administrador.

A continuación se presentan el objetivo general, los objetivos específicos y el alcance del presente T.E.G.

1.1. Objetivo General

Desarrollar un Motor de Búsquedas que permita recuperar los documentos digitales académicos generados por los miembros de la comunidad de la Facultad de Ciencias.

1.2. Objetivos Específicos

- Investigar los principios, modelos y paradigmas relacionados con la disciplina de estudio Recuperación de Información.
- Aumentar la visibilidad de los documentos académicos digitales desarrollando un repositorio apto para ser indexado por Motores de Búsqueda basados en *web*.
- Mejorar la calidad de las búsquedas desarrollando criterios de búsqueda y formas de navegación de documentos digitales.
- Permitir a los estudiantes que hayan aprobado el T.E.G. la carga de su respectivo documento digital desde el sistema CONEST.
- Permitir a un usuario Administrador gestionar los documentos académicos digitales almacenados en el repositorio.
- Permitir a un usuario Administrador la visualización del estado general del repositorio.
- Adecuar la plataforma tecnológica requerida por BUSCONEST 2 incorporando tecnologías con activo mantenimiento y desarrollo.
- Llevar a cabo la construcción de BUSCONEST 2 realizando un conjunto de actividades planificadas bajo un Modelo de Desarrollo de *Software* Ágil generando los artefactos requeridos.

1.3. Alcance

El alcance del presente T.E.G. abarca la construcción de una plataforma de recuperación de publicaciones digitales accesible desde la *web* que presente las siguientes características:

- Los principales Motores de Búsqueda de propósito general puedan indexar los documentos almacenados en el repositorio de la plataforma.
- Se puedan almacenar distintos tipos de documentos digitales tales como T.E.G., seminarios, informe de pasantía o notas de docencia.
- Se permita la carga de documentos publicados antes del desarrollo de BUSCONEST 1.
- Se puedan realizar recuperación de documentos mediante los criterios de búsqueda: autor, tutor y jurado (si aplica), tipo de documento, rango de fechas y escuela.
- Se permita la navegación por las categorías: año, escuela y tipo de documento.
- Se puedan consultar los índices del Motor de Búsqueda y la información descriptiva de los documentos.

El presente Trabajo Especial de Grado (T.E.G.) está dividido en dos partes las cuales a su vez agrupan varios capítulos relacionados entre sí.

Introducción

Parte I - Marco Conceptual: Se exponen los fundamentos teóricos necesarios para emprender el análisis y diseño de un Motor de Búsquedas. Los capítulos que componen esta parte son:

- **Recuperación de Información**: Se exponen las definiciones, principios, modelos y paradigmas presentados por la disciplina Recuperación de Información.
- **Motores de Búsqueda**: Se exponen los fundamentos tecnológicos y las principales características de diseño presentes en los Motores de Búsqueda modernos basados en *web*.
- **Casos de Estudio**: Se exponen las características de usabilidad y navegación presentes en varios Motores de Búsqueda de corte académico basados en *web* en producción. También se describe el estado del repositorio de BUSCONEST 1 a la fecha.

Parte II - Marco Aplicativo: Se describe el proceso de desarrollo del sistema BUSCONEST 2. Los capítulos que componen esta parte son:

- **Método de Desarrollo de *Software***: Descripción del Método de Desarrollo de *Software* Ágil SCRUM y su respectiva modificación para poder ser aplicado en el contexto del presente T.E.G.
- **Desarrollo del Sistema BUSCONEST 2**: Descripción de las iteraciones requeridas para desarrollar el sistema BUSCONEST 2.

Conclusiones

Recomendaciones

Parte I

Marco Conceptual

Capítulo 2

Recuperación de Información

El presente capítulo describe los principales fundamentos teóricos relacionados con el área de estudio Recuperación de Información. Primeramente se describen conceptos teóricos básicos y luego se describen y analizan los modelos formales que plantea la Recuperación de Información. También se mencionan las conferencias más influyentes dentro de esta disciplina.

Una primera aproximación para definir la Recuperación de la Información es la planteada por Langville y Meyer [1]:

> "La recuperación de la información es el proceso de buscar dentro de una colección de documentos para una necesidad particular de información"

Una definición alternativa establecida por Manning enuncia que la Recuperación de Información es[2]:

> "La recuperación de la información es la acción de encontrar el material (usualmente documentos) de naturaleza no estructurada (usualmente texto) que satisfaga una necesidad de información dentro de una gran colección (usualmente almacenada en computadores)"

En un sentido general la Recuperación de Información o (*Information Retrieval*) es la disciplina encargada de investigar las técnicas relacionadas con la búsqueda de la información en repositorios, bien sea documentos en soporte físicos o electrónicos. La Recuperación de Información tiene un carácter interdisciplinario ya que su área de estudio puede abarcar distintas disciplinas tales como: bibliotecología, matemáticas, ciencias de la computación, teoría de la información, psicología cognitiva, probabilidad y estadística, lingüística y derecho, ésta última relacionada por los problemas legales que plantea el acceso a la información en contenidos protegidos por derechos de autor.

Por tanto podemos conceptualizar la Recuperación de Información como la disciplina encargada del estudio, diseño y desarrollo de los Sistemas de Recuperación de Información. Para ello plantea modelos teóricos y técnicas empíricas basadas en la experimentación, con el fin de plantear modelos formales que representen lo mejor posible la estructura de la información que se pretende recuperar, así como también se encarga de la investigación de técnicas y procedimientos necesarios para desarrollar dichos modelos.

2.1. Sistemas de Recuperación de Información

Un Sistema de Recuperación de Información es un programa informático que almacena información sobre documentos, usualmente documentos de texto pero posiblemente multimedia. El sistema asiste al usuario en encontrar la información que necesita. No necesariamente se retorna información contenida en el documento o respuestas a las consultas, sino que en su lugar el sistema puede retornar la existencia y localización de documentos que podrían contener la información solicitada.

2.1.1. Características de los Sistemas de Recuperación de Información

Un Sistema de Recuperación de Información en un primer nivel de abstracción está constituido conceptualmente por un conjunto de elementos relacionados sobre los cuales se realizan una serie de procesos. Los elementos y procesos que representan conceptualmente un Sistema de Recuperación de Información se muestran en la figura 2.1. Las características específicas de cada uno de los elementos representados y la forma que se llevarán a cabo los procesos que operan sobre los elementos, dependerán en gran medida del tipo de modelo seleccionado para implementar el Sistema de Recuperación de Información. Antes de analizar en profundidad dichos modelos es importante definir y explicar conceptos comunes a todos los Sistemas de Recuperación de Información.

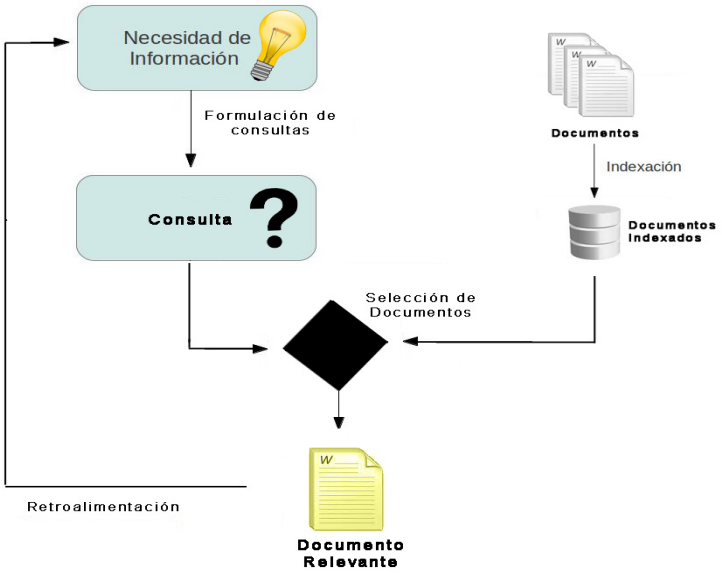


Figura 2.1: Sistema de Recuperación de Información: Representación Conceptual

2.1.1.1. Necesidad de Información y Consulta

Primeramente debemos diferenciar entre una necesidad de información o *information need* y una consulta, *query*. Una necesidad de información se refiere al asunto sobre el cual un usuario desea conocer más,

mientras que una consulta es una solicitud realizada por el usuario al sistema informático en un intento de satisfacer su necesidad de información. No necesariamente todas las consultas hechas por un usuario satisfarán su necesidad de información. Más aún la calidad de un Sistema de Recuperación de Información estará valorada en función de que los contenidos retornados al usuario sean relevantes para satisfacer su necesidad de información, independientemente si la consulta esté adecuadamente construida o no.

2.1.1.2. Documentos

Los documentos son abstracciones que permiten el almacenamiento y recuperación de la información. Típicamente están formados por un conjunto de frases y otros contenidos multimedia, además de presentar información que no está contenida directamente en ellos pero permiten describirlos (metadatos). Un documento indexado es aquel documento que se encuentra representado en la estructura interna del Sistema de Recuperación de Información. Un documento debe ser primeramente indexado para ser recuperado. El Sistema de Recuperación de Información no necesariamente tiene que almacenar el documento en sí, pero debe mantener un representación del mismo para poder almacenar por lo menos su ubicación. Se conoce como índice a la estructura de datos que alberga la información que relaciona los documentos con su ubicación en el repositorio. Al conjunto de todos los documentos contenidos en el repositorio se le denomina frecuentemente *corpus* o *corpora*, o también simplemente colección.

2.1.1.3. Tipos de Consultas

En general se pueden clasificar las consultas en las siguientes categorías:

- Consultas Informativas
- Consultas Navegables
- Consultas Transaccionales

Esto no quiere decir que una consulta siempre deba pertenecer a una de las categorías previamente enunciadas, ya que ésta es una clasificación muy general de las mismas. También una consulta puede pertenecer simultáneamente a varías de las categorías.

Un Sistema de Recuperación de Información debería ser capaz de poder calcular, por lo menos de un modo aproximado, a cual categoría puede pertenece una consulta dada, ya que esta información puede ser utilizada para darle un valor agregado a los resultados. Considérese que se infiere que una consulta es informativa, sería recomendable evitar mostrar al usuario información relacionada con publicidad y contenidos comerciales en los primeros resultados.

Consultas Informativas: Son aquellas consultas construidas por un usuario cuya necesidad de información es la de conocer más sobre un tópico particular. Este tópico bien podría ser música clásica, las pirámides de Egipto, o cualquier otro tema amplio. Usualmente toda la información relevante al usuario no estará contenida en un sólo documento sino posiblemente en un conjunto de documentos. Para procesar este tipo de consulta el Sistema de Recuperación de Información debe proveer de un sistema de Posicionamiento *o Ranking,* con la finalidad mostrar al usuario el conjunto de documentos que considere de mayor relevancia.

Consultas Navegables: Las consultas navegables son aquellas construidas por un usuario cuya necesidad de información es encontrar un documento en particular, podría ser un libro en línea en específico, en un contexto *web* podría ser la página de una empresa en particular, etc. Los resultados de estas consultas requieren una alta tasa de precisión, explicado en la sección 2.1.2.3. Sería recomendable mostrar al usuario directamente el documento requerido, para evitar que el mismo tenga que navegar a través de una lista, donde la mayoría de los resultados le serán poco relevantes.

Consultas Transaccionales: Las consultas transaccionales son aquellas que son realizadas por un usuario cuya necesidad de información es realizar un transacción. Entendemos por transacción, por ejemplo, descargar un archivo, realizar una reservación, realizar una compra electrónica, etc. En este contexto sí nos interesa mostrar al usuario como resultado los documentos que contengan información comercial. Por ejemplo éstos documentos podrían contener ofertas de compra, promociones, cupones y otros instrumentos comerciales para atraer más clientes.

2.1.1.4. Consideraciones Respecto a los Tipos de Consultas

Discernir a cuál categoría pertenece una consulta resulta ser una tarea no trivial, debido a que el usuario probablemente efectuará su consulta tan sólo ingresando unas pocas palabras. Poca información de entrada es generada directamente en la consulta para ser procesada por el Sistema de Recuperación de Información. Sin embargo cierta información subyacente puede ser extraída de la navegación de usuarios previos. Por ejemplo considere el caso que un usuario realiza la consulta American Airlines, es probable que dicha consulta sea transaccional o navegable, ya que la necesidad de información de los usuarios previos fue la de realizar una reservación de vuelo dada esa misma consulta. Sería razonable entonces mostrar directamente la página oficial de la mencionada aerolínea con los vuelos disponibles para el país del usuario, en vez de mostrar un documento donde se encuentre la historia de la aerolínea.

Una posible estrategia alternativa consiste en implementar Sistemas de Recuperación de Información específicos que se enfoquen en procesar con mayor precisión una categoría de consulta en particular. Se podría desarrollar por ejemplo Sistemas de Recuperación de Información enfocados a procesar consultas informativas, donde se excluyan de su repositorio documentos con fines comerciales. También se podrían desarrollar sistemas que sólo se centren en recuperar los documentos con los sitios *web* oficiales correspondientes a las compañías comerciales más buscadas por los usuarios, a modo de páginas amarillas. Sin embargo la forma de búsqueda que predomina actualmente consiste en entrar en un Sistema de Recuperación de Información que usualmente no toma en cuenta la naturaleza subyacente de la consulta, retornando todos los posibles resultados que considera relevante ordenados según una lógica interna.

2.1.1.5. Términos y Vocabulario

Los términos son las unidades que conforman los documentos y las consultas. Específicamente un término es una cadena finita de caracteres candidata a ser indexada por el Sistema de Recuperación de Información. Estas cadenas típicamente se corresponden con palabras empleadas dentro del lenguaje natural. Sin embargo códigos, abreviaciones, acrónimos y notaciones pueden ser considerados términos si se consideran relevantes. Incluso partes de una misma palabra también puede ser utilizadas como términos para mejorar el funcionamiento del motor de búsqueda. Por ejemplo en español la palabra *gato* varía dependiendo del

número y género gramatical (*gato, gatos, gata, gatas*). Sin embargo podría ser conveniente indexar todos los documentos con la ocurrencia del término *gat-* bajo una misma categoría.

Se conoce como vocabulario, también denominado *lexicon*, como conjunto que agrupa todos los términos indexados por el Sistema de Recuperación de Información.

2.1.1.6. Relevancia

La relevancia es una cualidad subjetiva que se le atribuye a un documento dependiendo si la información que representa es considerada de utilidad por el usuario. Aquellos documentos que contengan información que pueda satisfacer la necesidad de información del usuario se consideran documentos relevantes. Todos los que no posean esta cualidad se consideran irrelevantes. Sin embargo bajo un enfoque más general se pueden asignar diferentes grados de relevancia a los documentos que abarcan desde muy relevante hasta poco relevante.

2.1.1.7. Posicionamiento

El posicionamiento se define matemáticamente como una relación entre dos objetos, en la cual uno de dichos objetos puede ser catalogado como *mayor que*, *menor que*, o *igual que*, con respecto al otro objeto. Si se cumple esta relación formalmente se dice que existe una relación de orden parcial entre los dos objetos. Intuitivamente un posicionamiento de documentos consiste en mostrar un conjunto de documentos recuperados según su relevancia, donde los documentos considerados más relevantes son mostrados de primero al usuario. Para ello se necesita definir previamente algún tipo de escala que sea aplicable a todos los documentos, con el fin de poder poder comparar cuantitativamente la relevancia de los documentos entre sí. Formalmente se pueden ordenar los documentos de los resultados entre sí, si existe una una relación de orden parcial entre todos los documentos de la colección.

2.1.1.8. Retroalimentación

La retroalimentación *o feedback* se refiere al proceso mediante el cual un usuario puede evaluar la salida del sistema ingresando de forma explícita o implícita información adicional respecto a la apreciación de la calidad de los contenidos retornados. Con la finalidad que al sistema mejore las posibles salidas futuras. Usualmente la salida del sistema se corresponderá con un listado de los documentos que hayan sido seleccionados como relevantes, posiblemente con información descriptiva acerca de su ubicación o características.

La retroalimentación explícita se puede llevar acabo mediante el planteamiento de preguntas al usuario luego de realizar una consulta, preguntas tales como: ¿Le ha sido de utilidad este resultado?, ¿Ha encontrado la información que buscaba?, o puede ser ingresada de modo implícito al sistema, por ejemplo calculándose la frecuencia de uso de determinados términos para recuperar documentos específicos. El sistema podría inferir que términos son utilizados frecuentemente por el usuario para buscar información con respecto a un documento en particular. Por tanto sería conveniente mostrar en los resultados documentos con características similares cuando el usuario emplee los mismos términos en consultas futuras. También se podría determinar que tipos de documentos el usuario está consultando en los documentos sugeridos para mejorar la calidad de las sugerencias.

2.1.2. Tasa de Recuperación y Tasa de Precisión

Para cuantificar la calidad de los resultados retornados dada una consulta se define como tasa de recuperación *o recall* a la tasa de documentos relevantes retornados por una consulta con respecto al total de los documentos relevantes que se encuentran en el repositorio.

2.1.2.1. Definición Tasa de Recuperación

Sea DocRec el conjunto de todos los documentos recuperados por una consulta Q, y DocRel el conjunto de todos los documentos relevantes para el usuario. La tasa de recuperación de los resultados generados por Q viene dada por la expresión 2.1

$$\text{tasa\_recuperacion}_Q = \frac{|\text{DocRec} \cap \text{DocRel}|}{|\text{DocRel}|} \tag{2.1}$$

2.1.2.2. Ejemplo Tasa de Recuperación

Supongamos un universo de cinco documentos $U = \{A, B, C, D, E\}$ y que el usuario sólo considere relevantes C y D. Supongamos que una consulta Q_0 retorna el conjunto $\{D\}$ La recuperación de los resultados retornados por Q_0 viene dada por la expresión 2.2

$$\text{tasa\_recuperacion}_{Q_0} = \frac{|\{D\} \cap \{C, D\}|}{|\{C, D\}|} = \frac{|\{D\}|}{|\{C, D\}|} = \frac{1}{2} \tag{2.2}$$

La tasa de recuperación de Q_0 resulta ser un $\frac{1}{2} = 50\,\%$ es decir Q_0 sólo recuperó la mitad de todos los documentos relevantes para el usuario. Nótese que al usuario sólo recupera el documento D y posiblemente ignore la existencia del documento C. Es decir el Sistema de Recuperación de Información estaría de cierta forma *ocultando* información relevante al usuario.

Considérese el siguiente caso:

Supongamos que una consulta Q_1 retorna el conjunto $\{A, B, C, D, E\}$ la recuperación de los resultados retornados por Q_1 viene dada por la expresión 2.3

$$\text{tasa\_recuperacion}_{Q_1} = \frac{|\{A, B, C, D, E\} \cap \{C, D\}|}{|\{C, D\}|} = \frac{|\{C, D\}|}{|\{C, D\}|} = \frac{2}{2} = 1 \tag{2.3}$$

Como se puede observar la recuperación es 100 %, la máxima posible, debido a que todos los documentos que el usuario consideraba relevantes fueron devueltos por el sistema. Sin embargo nótese que todos los documentos del repositorio también fueron retornados, un comportamiento poco deseable ya que el usuario obtendría todos los documentos relevantes pero mezclados con muchos documentos irrelevantes, generando posibles molestias debido a la pérdida de tiempo y esfuerzo que implica obtener este tipo de resultados.

En líneas generales no es suficiente evaluar la calidad de un sistema de recuperación de información sólo teniendo en cuenta la tasa de recuperación de las consultas hechas al mismo. La tasa de recuperación sólo nos da una idea de cuántos documentos relevantes contenidos en el repositorio fueron retornados al usuario, sin tomar en cuenta cuántos documentos *irrelevantes* también fueron retornados.

2.1.2.3. Definición Tasa de Precisión

La expresión 2.4 es definida para solventar la limitación en la medición de la tasa de recuperación. Definimos como tasa de precisión o *precision*, a la tasa de documentos relevantes recuperados con respecto a los documentos recuperados en total.

$$\text{tasa\_precision}_Q = \frac{|\text{DocRec} \cap \text{DocRel}|}{|\text{DocRec}|} \tag{2.4}$$

2.1.2.4. Ejemplo Tasa de Precisión

Supongamos el mismo universo de cinco documentos $\{A, B, C, D, E\}$, y que el usuario sólo considere relevantes, de nuevo, los documentos C y D. Supongamos que se realiza la consulta Q_0 de nuevo retorna el conjunto $\{D\}$. La precisión de los resultados retornados por Q_0 viene dada por la expresión 2.5

$$\text{tasa\_precision}_{Q_0} = \frac{|\{D\} \cap \{C, D\}|}{|\{D\}|} = \frac{|\{D\}|}{|\{D\}|} = 1 \tag{2.5}$$

La tasa precisión para Q_0 resulta ser $1 = 100\,\%$ lo que se interpreta de la siguiente forma: del conjunto de los documentos recuperados todos fueron relevantes. Nótese que a pesar que C no fue recuperado, la precisión es máxima porque de todos los documentos recuperados, en este caso sólo D, todos son relevantes. Sin embargo la de tasa recuperación previamente calculada en 2.2sólo arrojó un resultado de $50\,\%$. Por tanto se *ocultó* la mitad de la información relevante en este caso.

Nuevamente se considera el conjunto $\{A, B, C, D, E\}$ Supóngase que la consulta Q_1 retorna el conjunto $\{A, B, C, D, E\}$de nuevo. La precisión de los resultados retornados por Q_1 viene dada por la expresión 2.6

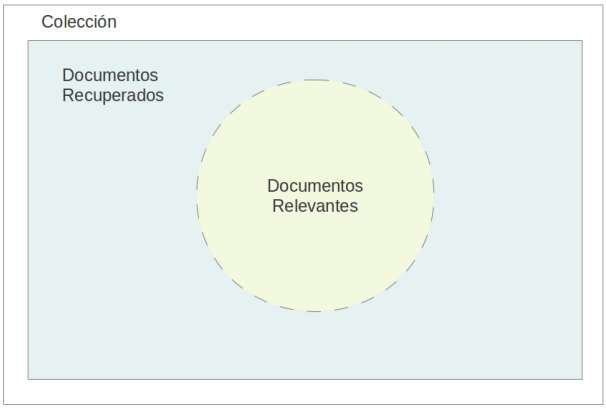
$$\text{precision}_{Q_1} = \frac{|\{A, B, C, D, E\} \cap \{C, D\}|}{|\{A, B, C, D, E\}|} = \frac{|\{C, D\}|}{|\{A, B, C, D, E\}|} = \frac{2}{5} \tag{2.6}$$

En este caso la precisión de la consulta sólo fue de $\frac{2}{5} = 40\,\%$ es decir del conjunto de los documentos recuperados menos de la mitad fueron relevantes. Compárese con el resultado de la medición de la tasa de recuperación en 2.3 que resultó ser $100\,\%$ para la misma consulta Q_1. Esto es debido a que muchos resultados irrelevantes son mostrados al usuario junto con los relevantes.

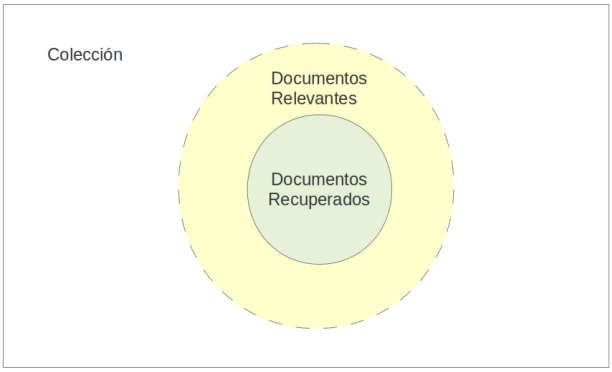
Por tanto la tasa de precisión tampoco resulta por sí misma, ser suficiente para evaluar la calidad de la información recuperada por el usuario. Por un lado la tasa de recuperación nos da una medida de

la proporción de documentos relevantes recuperados por una consulta, con respecto a la totalidad de documentos relevantes contenidos en el repositorio. Sin embargo esta medida no toma en cuenta todos los documentos irrelevantes obtenidos en el resultado. Si la tasa de recuperación es baja entonces se estaría omitiendo una cantidad considerable de información relevante en los resultados. Si la tasa de recuperación es alta se podrá asegurar que la mayoría de los documentos relevantes del repositorio está en el resultado, pero no se podrá determinar si hay muchos resultados irrelevantes interfiriendo.

Por otro lado la precisión nos da una medida de la proporción de documentos relevantes recuperados, con respecto a la totalidad de documentos relevantes e irrelevantes recuperados por la consulta. Sin embargo esta medida no toma en cuenta los documentos relevantes no recuperados del repositorio. Si la tasa precisión es baja entonces muchos documentos irrelevantes serán recuperados junto a los documentos relevantes interfiriendo con los resultados de la búsqueda. Si la tasa de precisión es alta se tendrá la certeza que la mayoría de los documentos retornados son relevantes, pero no se podrá tener certeza de la cantidad de documentos relevantes no recuperados del repositorio [3]. Ésta situación se ilustra mediante diagramas de Venn en la figura 2.2.



(a) Alta Tasa de Recuperación y Baja Tasa de Precisión: Todos los documentos relevantes son recuperados, pero también los irrelevantes.



(b) Alta Tasa de Precisión y Baja Tasa de Recuperación: Todos los documentos recuperados son relevantes, pero muchos documentos relevantes no son recuperados del repositorio.

Figura 2.2: Tasa de Recuperación y Tasa de Precisión

Para evaluar la calidad de un Sistema de Recuperación de Información, los desarrolladores deben tener en cuenta que los algoritmos y estructuras de datos que lo conforman estén diseñados con la finalidad de

maximizar tanto la tasa de recuperación como la tasa de precisión. No sólo basta con maximizar la tasa de recuperación, por ejemplo retornando todos los documentos del repositorio porque la precisión será mínima. Tampoco basta con maximizar la tasa precisión retornando un único documento relevante en los resultados porque la tasa de recuperación sería mínima (imagine a Google sólo con el botón *I'm feeling lucky*[1]), sino que se debe procurar maximizar ambas tasas. La situación óptima se puede observar en la figura 2.3.

Figura 2.3: Tasa de Precisión y Tasa de Recuperación: Caso óptimo.

2.1.2.5. F1-score

Para poder ponderar dos tasas, y por tanto tener una forma de cuantificar la calidad de los resultados en función de la de tasa precisión y la tasa de recuperación, la expresión conocida como F1-score 2.7 es frecuentemente utilizada[2].

$$\text{F1-score} = 2\,\frac{\text{tasa precision}\ .\ \ \text{tasa recuperacion}}{\text{tasa precision}\ + \text{tasa recuperacion}} \tag{2.7}$$

F1-score se puede interpretar como el promedio ponderado entre la tasa de recuperación y la tasa de precisión, donde F1-score alcanza su máximo valor en 1 y el el mínimo en 0.

[1] Opción que permite retornar sólo el documento considerado más relevante.
[2] F1-score corresponde a la media armónica entre la tasa de recuperación y la tasa de precisión.

2.1.2.6. Consideraciones adicionales

También hay que considerar que significa exactamente que un documento sea relevante para un usuario, el sistema debe ser capaz de inferir si un documento es relevante o no partir de la consulta del usuario para tomar la decisión de recuperar los documentos pertinentes.

Esta consulta muchas veces está expresada en lenguaje natural porque no se espera que el usuario del sistema posea conocimientos técnicos para construir consultas más precisas. Compárese el ingreso de palabras claves en lenguaje natural, empleado como método de entrada en los buscadores modernos, con el lenguaje de consultas estructuradas SQL utilizados para consultar los Sistemas Manejadores de Base de Datos Relacionales.

Existe también la posibilidad de que el usuario no haya expresado explícitamente lo que deseaba buscar pero analizando su consulta, consultas previas, y las consultas de otros usuarios se puede inferir lo que el usuario *trató* de expresar (*Did you mean?*). Se han propuesto modelos de recuperación de información que plantean soluciones a dicha situación. Finalmente se precisa de un método de posicionamiento para mostrar los resultados si éstos son numerosos o se tiene la certeza que determinados documentos son *más* relevantes que otros.

2.2. Modelos de Recuperación de la Información

Los Modelos de Recuperación de Información constituyen el pilar fundamental de la Recuperación de Información, los mismos han sido producto de la formalización de las soluciones planteadas al problema de recuperación de información.

2.2.1. Búsqueda Lineal

Supongamos que se presenta el problema básico de buscar dentro de una colección de libros la presencia de una palabra. Intuitivamente la búsqueda la podemos realizar de forma exhaustiva. Para ello primero tomamos un libro de la colección y recorremos cada una de las frases que componen el libro hasta encontrar la primera ocurrencia de la palabra buscada en cuestión. Si esto llegase a ocurrir apartamos el libro porque sabemos que contiene al menos una ocurrencia de la palabra. Por el contrario si no quedan más frases en el libro, y no se ha encontrado aún la primera ocurrencia, entonces se descarta dicho libro. Luego realizamos este procedimiento sucesivamente hasta inspeccionar todos los libros de la colección.

Este sencillo esquema de búsqueda es conocido como Búsqueda Lineal o *Linear Scanning,* y es el empleado por numerosas herramientas informáticas en la actualidad bajo ambientes Unix. Por ejemplo el popular programa de línea de comandos *grep* implementa dicho algoritmo. Su nombre es una mnemotécnico que les recuerda a los usuarios del sistema su funcionamiento, *Global/REgular expression/Print*, que se interpreta como: imprime por salida *standard* todas las ocurrencias de una expresión regular dentro de toda la colección. Su uso es tan común que ha dado origen al término *grepping* que, en contextos informáticos, significa buscar todas las ocurrencias de una expresión regular dentro de una colección de frases. Estas frases bien podrían estar almacenadas dentro de un conjunto de archivos o provenir de un flujo externo de datos, o ambas cosas.

Sin embargo la Búsqueda Lineal a pesar de ser simple y eficaz, no resulta siempre eficiente resolviendo el problema de recuperar información. A continuación se analizan las principales ventajas y desventajas de esta primera aproximación.

2.2.1.1. Ventajas

- Conceptualmente sencilla. La naturaleza sencilla de la búsqueda lineal resulta en una baja complejidad en la implementación de la solución correspondiente. Adicionalmente las estructuras de datos requeridas son fáciles de mantener en contraposición a, por ejemplo, mantener actualizado un índice global.
- Eficiente en consultas simples sobre repositorios con bajo volúmenes de datos. Supóngase la situación que un usuario desea saber todas las frases que contienen la palabra *hostname* en un conjunto de archivos de configuración. Supongamos que la colección de archivos no ocupa mas de 16KB. El recorrido lineal podría ser mucho más rápido que el tiempo requerido para la indexado, y el procesamiento de la consulta bajo otros modelos de recuperación de información mas complejos.

2.2.1.2. Desventajas

- No escala adecuadamente. A medida que el repositorio aumenta su volumen también aumenta proporcionalmente los tiempos de búsqueda bajo este enfoque, esto podría no ser tolerable en ciertos ámbitos dónde el usuario espera bajos tiempos de respuesta, típicamente fracciones de segundos.
- No permite el posicionamiento de los documentos. El usuario podría no estar interesados en toda la colección de documentos donde ocurra un término en la consulta, más bien podría estar interesado en un subconjunto de resultados valorados por orden de relevancia. Una Búsqueda Lineal no satisface esa necesidad de información en particular
- No permite el procesamiento de metadatos. Los metadatos, información descriptiva del documento que pudiera no estar contenida dentro del mismo, son una fuente importante de información para búsquedas complejas. Un usuario podría tener la necesidad de información *todos los documentos actualizados antes de determinada fecha*. La búsqueda lineal no está diseñada para valorar este tipo de información.
- No permite consultas en lenguaje natural. Para el usuario podría resultar confuso realizar consultas usando términos que involucren la formulación de expresiones regulares. Considérese la siguen te necesidad de información de un usuario hipotético *¿Cuáles documentos tratan sobre música clásica que no sean de historia?* La formulación de dicha interrogante en términos de expresiones regulares no es adecuada. No sólo basta con encontrar las ocurrencias de las palabras *música clásica* en todos los documentos, excluyendo aquellos que contengan al menos una ocurrencia de la palabra *historia*. Considérese un documento con la siguiente frase *la música clásica no es historia, aún es muy popular entre los jóvenes*. El documento en cuestión sería descartado pero su contenido nos sugiere que puede ser relevante para el usuario.

Las ventajas y desventajas de la Búsqueda Lineal se muestra en la tabla 2.1

Tabla 2.1: Búsqueda Lineal: Ventajas y Desventajas.

Ventajas	**Desventajas**
Conceptualmente sencillo	Poco Escalable
Eficiente procesando colecciones pequeñas	No hay posicionamiento
	No permite procesamiento de metadatos
	No permite consultas en lenguaje natural

2.2.2. Modelo Booleano

Una estrategia de búsqueda booleana recupera aquellos documentos que evalúan a verdad para dada una consulta. Esta información sólo tiene sentido si las consultas son expresadas en términos indexados, combinados con los conectivos lógicos usuales AND, OR, NOT[4].

El Modelo Booleano de Recuperación de Información representa cada documento como un conjunto de proposiciones lógicas. Como un documento está constituido por un subconjunto de términos contenidos en el vocabulario, cada documento se puede representar como un conjunto de proposiciones lógicas $D_i = \{p_1,p_2,p_3,...,p_n\}$[3] donde p_i representa la presencia o ausencia del término $t_i \epsilon V$ con $V = \{t_1, t_2, t_3, \ldots, t_n\}$.

2.2.2.1. Ejemplo

Dado el vocabulario $V = \{\text{computacion}, \text{informacion}, \text{algoritmos}, \text{musica}, \text{orquesta}, \text{concierto}\}$ supóngase que el documento D_1 contiene los siguientes términos: *computacion informacion algoritmos*. La representación booleana de D_1 será la expresión

$$D_1 = \{1, 1, 1, 0, 0, 0\}$$

Supóngase que el documento D_2 contiene los términos: *musica orquesta concierto*. La representación correspondiente a D_2 es

$$D_2 = \{0, 0, 0, 1, 1, 1\}$$

Supóngase el documento D_3 con contenido: *musica informacion*. Su representación será

$$D_3 = \{0, 1, 0, 1, 0, 0\}$$

A partir de esta información podemos construir un índice que relacione todas los términos del vocabulario V con los respectivos documentos D_1, D_2, D_3.

[3]Debido a la definición formal de conjunto, el conjunto D_i podría presentar cualquier otro orden, por ejemplo $D_i = \{p_3, p_2, p_n, \ldots, p_1\}$, pero para efectos de simplicidad, en los ejemplos se preservará el orden de las proposiciones lógicas del documento con respecto a los términos del vocabulario V.

Tabla 2.2: Índice Invertido

V	D_1	D_2	D_3
computacion	1	0	0
informacion	1	0	1
algoritmos	1	0	0
musica	0	1	1
orquesta	0	1	0
concierto	0	1	0

La estructura de datos en la tabla 2.2 se conoce cómo índice invertido ya que relaciona cada uno de los términos del vocabulario con los todos los documentos donde se presenta una ocurrencia de un término, en lugar de relacionar cada documento con sus respectivas ocurrencias de términos del vocabulario.

Si un usuario efectúa la consulta: *musica* AND *informacion*, entonces el sistema procesará las entradas correspondientes a *musica e informacion* en el índice invertido representado en la tabla 2.2. Lo cuál resulta en la expresión 2.8

$$011 \ \text{AND}\, 101 = 001 \tag{2.8}$$

El resultado en 2.8nos indica de forma inequívoca los documentos donde se encuentra la ocurrencia de las palabras *musica* e *informacion* simultáneamente, en este caso el documento D_3. Si el usuario ingresase la consulta: NOT *computacion*, sólo basta con procesar la entrada correspondiente en el vocabulario.

$$\text{NOT}\, 100 = 011 \tag{2.9}$$

Por tanto según la expresión 2.9 D_2 y D_3 son los documentos que no contienen el término *computacion.*

2.2.2.2. Modelo Booleano Extendido

El Modelo de Recuperación de Información Booleano, a pesar de su simplicidad conceptual, puede ser extendido para representar información adicional. Principalmente esta información adicional es utilizada por el sistema para inferir niveles de relevancia entre los documentos.

Por ejemplo, se puede diseñar una estructura de datos a partir del índice invertido que contenga cada una de las palabras del vocabulario ordenadas alfabéticamente. Luego cada entrada de dicha estructura puede corresponderse con una lista que capture información adicional. Además de la lista de documentos que contienen dicho término en el esquema básico.

Asimismo también se puede almacenar el número de ocurrencias de un término dentro de un documento, la posición de dicho término en el documento, o el número de documentos en el que aparece un término. Esta última información coincidiría con el tamaño de la lista de documentos que contiene el término, lo cual disminuiría el tiempo de ejecución de las consultas, ya que no se tiene que recorrer completamente la lista de documentos para calcular su tamaño. En la Figura se puede detallar la construcción de un índice invertido bajo el esquema booleano extendido.

En la figura 2.2 se muestra el proceso de construcción de un índice invertido. Primero se agrupan todos los término contenido es los documentos DOCID 1 y DOCID 2. Luego se ordena alfabéticamente el índice invertido. Finalmente a cada término se le hace corresponder la lista de documentos donde ocurre, POSTING LISTS, y se almacena adicionalmente la frecuencia de ocurrencias de dicho término, DOC.FREQ., en todos los documentos

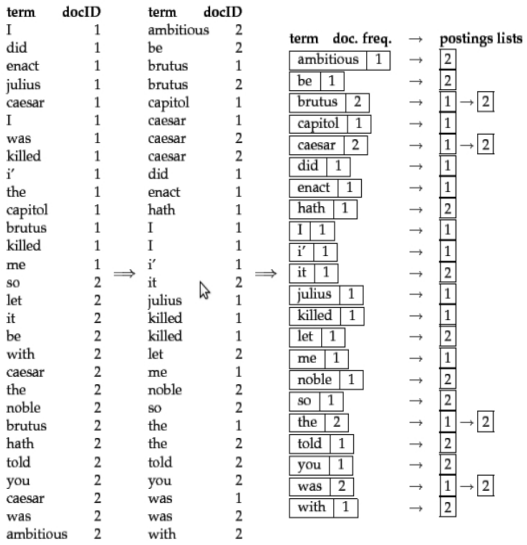


Figura 2.4: Construcción de un índice invertido extendido.

2.2.2.3. Ventajas

- Conceptualmente sencillo y ampliamente estudiado.
- Altamente eficiente e implementado con frecuencia como Sistema de Recuperación de Información.
- Permite la construcción de consultas expresivas y claras al estar basado en el Álgebra de Boole.
- Altamente eficaz para ejecutar consultas concretas y precisas.
- Debido a su eficiencia es adecuado para manejar grandes volúmenes de información.

2.2.2.4. Desventajas

- El usuario puede encontrar dificultades para construir consultas booleanas. Los operadores utilizados en el Álgebra de Boole no siempre corresponden con su uso en el lenguaje natural. Por ejemplo, en español es usual emplear la expresión *No sé nada*, a pesar de involucrar una doble negación desde un punto de vista estrictamente lógico. La doble negación en el español es utilizada realmente para enfatizar una declaración.

- Su naturaleza binaria evalúa los documentos como relevantes o no relevantes. El usuario no tiene la opción que los documentos sean clasificados en escalas intermedias como *muy relevante*, *poco relevante* en determinado contexto, pero *razonablemente relevante* en otro, etc.

- En su forma básica no contempla un sistema de posicionamiento, por tanto no se puede comparar la relevancia de los documentos entre ellos.

- Todos los términos empleados para construir la consulta tienen el mismo valor, peso o *weight* en la consulta, no hay forma de jerarquizar la relevancia de los términos entre ellos.

- Se asume que las consultas realizadas por el usuario no poseen niveles de incertidumbre. Cada término tiene un sólo significado y cada concepto es representado por un único término. Considérese el caso de palabras sinónimas: *escorpión* y *alacrán* son términos distintos para representar el mismo concepto, o la palabra *banco* con múltiples significados en el idioma español.

Tabla 2.3: Modelo Booleano: Ventajas y Desventas.

Ventajas	**Desventajas**
Conceptualmente sencillo	Asume experiencia del usuario para construir las consultas
Eficiente	No permite posicionamiento
Expresivo	No representa relaciones entre los documentos
Escalable	Todos los términos de la consulta tiene la misma relevancia
Eficaz con consultas precisas	No permite representar incertidumbre en las consultas
Extensible	No contempla retroalimentación

2.2.3. Modelo del Espacio Vectorial

Según Michael W. Berry, Murray Browne[5]:

> "En el modelo del espacio vectorial se plantea que ambos, términos y/o documentos son codificados como vectores en un espacio k-dimensional. La elección de k puede estar basada en el número de términos únicos, conceptos o quizás clases asociadas con el texto de la colección. Por tanto cada componente del vector (o dimensión) es usada para reflejar la importancia del correspondiente término/concepto/clase en representar la semántica o significado del documento"

El modelo del espacio vectorial fue ampliamente estudiado por primera vez por Gerald Salton y sus colegas durante el desarrollo del sistema SMART (*System for Mathematical Analysis and Retrieval of Text*) en la universidad de Cornell durante la década de 1960. Surgió en parte para solventar algunas limitaciones propias del Modelo Booleano de Recuperación de Información. Su diseño e implicaciones influyeron profundamente en posteriores investigaciones dentro del área de la Recuperación Información.

Un Modelo del Espacio Vectorial transforma datos textuales en vectores numéricos y matrices para emplear técnicas de análisis matricial con la finalidad de descubrir características y relaciones en una colección de documentos. Para ello el Modelo del Espacio Vectorial se emplea una matriz de *términosxdocumento*s *n-dimensional* donde cada columna representa un documento en particular y cada fila, en la forma mas simple del modelo, representa un término del vocabulario. Los valores almacenados en cada celda de la matriz representan el número de ocurrencias de un término en un documento en específico.

Sin embargo bajo un enfoque más general las celdas pueden corresponder a una función que calcula la relevancia de un término especifico en un documento en particular. Está función típicamente calcula un valor entero positivo conocido como peso o *weight*, y el proceso del cálculo de la relevancia de un término dentro de un documento (evaluación de la función de peso) es conocido como ponderación o *weighting*. La función de peso más simple calcula el número de ocurrencias del término dentro de un documento.

2.2.3.1. Ejemplo

En la tabla 2.4 podemos observar una matriz que representa cuatro documentos D_1, D_2, D_3, D_4 en cada columna y tres términos, *computacion, matematicas, musica*, en cada fila.

Tabla 2.4: Matriz *términosxdocumentos*

Vocabulario	D_1	D_2	D_3	D_4
computacion	1	0	1	0
matematicas	0	0	1	1
musica	0	1	1	0

Se han escogido sólo tres término en el vocabulario$V$ = {computacion,matematicas,musica} adrede para poder representar los vectores en un espacio vectorial *3-dimensional*. Sin embargo en la práctica los modelos del espacio vectorial suelen presentar dimensiones muy superiores. Según la tabla 2.4 el término *computacion* ocurre una vez en D_1 y D_3. El término *matematicas* ocurre una vez en el D_3 y una vez en D_4. Finalmente el término *musica* ocurre una vez en el D_2 y una vez en el D_3. Con esta información se construye el gráfico mostrado en la figura 2.5.

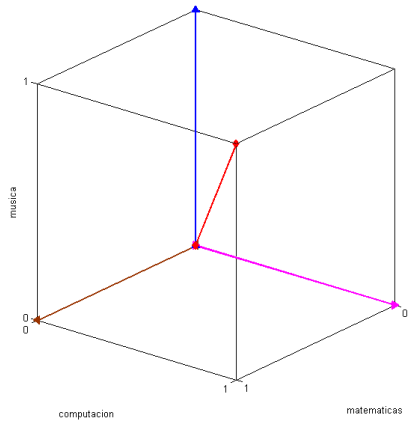


Figura 2.5: Cuatro vectores documentos en un espacio *3-dimensional*

2.2.3.2. Similitud

Peter Luhn en 1957 fue uno de los primeros investigadores en formular el criterio de similitud, según él:

> "Entre más parecidas sean dos representaciones dados sus elementos y su distribución, mayor será la probabilidad que ambas representen información similar"

La similitud es una propiedad presente entre dos o más vectores que permite cuantificar que tan parecidos son los vectores entre ellos. Para poder procesar una consulta la misma primero debe poder ser representada como un vector *k-dimensional* dentro de un espacio vectorial. Para su evaluación se calcula la similitud del vector consulta contra todos los vectores documentos. El cálculo de la similitud es un buen punto de referencia para empezar a estimar la relevancia de un documento con respecto a una consulta.

Resulta lógico pensar que entre más similares sean los vectores que representan las consultas y los vectores que representan los documentos, entonces es maś probable que la consulta y los documentos recuperados traten de información parecida o relacionada.

Como consecuencia, a diferencia del Modelo Booleano, en el Modelo del Espacio Vectorial se pueden ordenar los documentos en diferentes rangos de relevancia dependiendo su medida de similitud. Para el cálculo de la similitud se emplean operaciones propias del Álgebra Lineal.

2.2.3.3. Cálculo de la Similitud

Para calcular la Similitud entre dos vectores se pueden emplear varias funciones, entre ellas:

- Producto Escalar o Producto Punto

- Coseno del ángulo que separa dos vectores
- Índice de Dice
- Índice de Jaccard

Producto Escalar o Producto punto: La función 2.10 calcula el producto escalar entre el vector documento $\vec{d}$ y el vector consulta $\vec{q}$

$$\text{similaridad}(\vec{d},\vec{q}) = \sum_{k=1}^{m} d_k.q_k \tag{2.10}$$

Si cada componente d_k y q_k es igual a 1 cuando un término está presente en el documento o consulta, y 0 si no, entonces el producto vectorial mide el número de términos compartidos entre los vectores $\vec{d}$ y $\vec{q}$. Sin embargo una representación mas general puede utilizar el conjunto de los números naturales o reales no negativos como componentes de los vectores $\vec{d}$ y $\vec{q}$.

Coseno del ángulo que separa dos vectores: Se calcula con la expresión 2.11 y también puede ser utilizada como una medida de similitud[4]:

$$\cos(\alpha) = \frac{\sum_{k=1}^{m} d_k.q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2}.\sqrt{\sum_{k=1}^{m} (q_k)^2}} \tag{2.11}$$

Si los vectores $\vec{d}$ y $\vec{q}$ son normalizados la ecuación del cálculo del coseno entre dos ángulos es igual a la fórmula 2.10, es decir su producto escalar. Por tanto si normaliza el vector consulta $\vec{q}$ y el vector documento $\vec{d}$ y calculamos el ángulos entre ellos será equivalente a su producto escalar, que es la medida de similitud en 2.10. Ver expresión 2.12

$$\text{similaridad}(\vec{d},\vec{q}) = \sum_{k=1}^{m} n(d_k).n(q_k) \quad \text{donde} \quad n(v_k) = \frac{v_k}{\sqrt{\sum_{k=1}^{m} (v_k)^2}} \tag{2.12}$$

Otras medidas alternativas frecuentemente empleadas para el cálculo de la Similitud son[6]:

Índice de Dice:

$$\text{similaridad}\left(\vec{d},\vec{q}\right) = \frac{2.\sum_{k=1}^{m} d_k.q_k}{\sum_{k=1}^{m} d_k^2 + q_k^2} \tag{2.13}$$

Índice de Jaccard:

$$\text{similaridad}(\vec{d},\vec{q}) = \frac{\sum_{k=1}^{m} d_k.q_k}{\sum_{k=1}^{m} \left(d_k^2 + q_k^2 - d_k.q_k\right)} \tag{2.14}$$

[4]El coseno de un ángulo α es 0 si los vectores son ortogonales entre sí, y el coseno del ángulo α es 1 si el ángulo que separa los vectores es 0.

2.2.3.4. Agrupamiento

El agrupamiento o *clustering* consiste en la técnica de descubrir conjuntos de documentos relacionados de un conjunto universo. Para agrupar se necesita haber definido la medida de similitud entre los vectores previamente.

En el agrupamiento la medida de similitud es empleada para cuantificar el parecido entre objetos de tal modo que uno puede asumir que se pueden crear grupos de objetos. Si un objeto se encuentra en un grupo entonces será más parecido a los otros miembros del grupo, que cualquier otro objeto fuera del grupo. Las técnicas de agrupamiento se emplean a descubrir tales grupos. Además los agrupamientos podrían dar lugar a categorías o tópicos bajo los cuales clasificar los documentos.[4]

2.2.3.5. Clasificación

La clasificación o *classification* consiste en catalogar un documento dentro de una o varias categorías, o posiblemente ninguna. Para ello se toman en cuenta las propiedades compartidas de un documento, con respecto a las propiedades de los demás documentos dentro de un grupo específico. Si el grado de atributos compartidos en suficientemente alto, entonces se considera que el documento está dentro del grupo, de lo contrario no se considera que el documento pertenece al grupo.

Definir los atributos de los documentos, la cuantificación de dichos atributos, así como definir el criterio para saber si un documento pertenece a un grupo, resulta ser una tarea no trivial. Se han sugerido diversas técnicas para poder cumplir este propósito. La técnica más simple consiste en cuantificar la cantidad de términos comunes entre el documento analizado y los documentos dentro del grupo. Esta medida es conocida como Coeficiente Simple de Coincidencias (CSC)[19], expresión2.15:

$$\mathrm{CSC}(X, Y) = |X \cap Y| \tag{2.15}$$

Donde X es el conjunto de todos los términos dentro de un documento en particular, y Y es el conjunto que contiene los términos presentes en todos los documentos pertenecientes a un mismo grupo.

2.2.3.6. Ponderación

La ponderación consiste en la asignación de valores numéricos a los componentes de un vector consulta o documento mediante una función de ponderación. El criterio de frecuencias de términos es la función de ponderación más simple y consiste en calcular el número de ocurrencias de un término específico en un documento.

El cálculo de los valores correspondientes a las componentes de los vectores a generado investigaciones para mejorar el criterio de frecuencias de términos. Por ejemplo, la familia de funciones de ponderación *tf-idf* propone calcular el peso de un término dentro de un documento como el producto entre la frecuencia del término (número de ocurrencias de un término dentro de un documento) y una medida relacionada con el inverso de la frecuencia del documento, que es la cantidad de documentos en el repositorio donde ocurre el término.

Un ejemplo de una función *tf-idf* fue la propuesta por Salton durante el desarrollo del sistema SMART[7]:

$$d_k = q_k = tf(k,d).\lg\frac{N}{df(k)} \tag{2.16}$$

donde $tf(k,d)$es el número de ocurrencias del término k en el documento d, $df(k)$ es el número de documentos que contienen el término k, y N es el número total de documentos en la colección.

En la práctica la función 2.16 resultó tener un pobre desempeño. Sin embargo modificaciones a la función 2.16 han sido planteadas para salvar este inconveniente, como por ejemplo la función Okapi BM25.

2.2.3.7. Ventajas

- El modelo del espacio vectorial permite el posicionamiento de los documentos según su relevancia mediante el cálculo de similitud.
- Descubrimiento de grupos de vectores similares. El agrupamiento de vectores similares permite el descubrimiento de grupos de documentos relacionados y a su vez la definición de estos grupos puede ser utilizada en la clasificación de los documentos de forma automática. Organizar los documentos de éste modo permitiría la posibilidad de plantear búsquedas menos estrictas que aquellas efectuadas bajo el Modelo Booleano. El usuario sólo tendrá que construir una consulta parcial que genere un vector lo suficientemente similar a los vectores que representan los documentos que considera relevantes.
- La posibilidad de agrupar documentos permite implementar la técnica de retroalimentación, ya que un usuario luego de haber realizado una consulta se le podría devolver un conjunto de documentos agrupados y luego se le podría preguntar si considera relevantes el conjunto de documentos sugeridos. Si su repuesta es negativa se puede agrupar los documentos retornados en subconjuntos de documentos más estrechamente relacionados, dando lugar a descubrimientos de nuevos grupos.

2.2.3.8. Desventajas

- El modelo del espacio vectorial no plantea como se deben codificar las consultas o los documentos en vectores. La aproximación de considerar cada término en una dimensión independiente es poco realista ya que términos distintos podrían estar estrechamente vinculados en significado. Por ejemplo considérese el caso de de palabras sinónimas.
- Poco eficiente. El Modelo del Espacio Vectorial requiere el mantenimiento de una estructura de datos global donde se representen todos los vectores correspondientes a cada documento, y donde cada dimensión de dicho espacio vectorial corresponda posiblemente a un término. Como consecuencia el mantenimiento y representación de la estructura de datos global en el computador puede resultar altamente costosa.
- La adición de un nuevo documento implica la modificación de todos los vectores documento que contengan al menos un término compartido con el nuevo documento. Una solución para minimizar el impacto en el desempeño consiste en no representar términos como dimensiones del espacio vectorial,

sino como como conceptos que agrupen términos, o inclusive categorías de conceptos. Sin embargo la definición de estos conceptos resulta una tarea no trivial y que deseablemente debería realizarse de forma automática. Sin embargo este aspecto al estar estrechamente relacionado con las características semánticas propias del lenguaje natural.

- El Modelo del Espacio Vectorial no permite minimizar el impacto del cálculo de las operaciones de similitud que se deben efectuar sobre la estructura de datos global que representa el espacio vectorial. Considérese el caso del cálculo del producto escalar entre un vector consulta y miles de vectores documentos en un espacio *k-dimensional,* con k posiblemente un número muy grande, ya que cada dimensión representa un término. Los tiempos de respuesta resultantes, incluso para una consulta simple, podrían no ser aceptables.
- El Modelo del Espacio Vectorial no plantea un método preciso para codificar las consultas como vectores. Por ejemplo supóngase que un usuario quisiera realizar una búsqueda de un conjunto de documentos que no contengan un término. La representación de éste tipo de consultas como vectores no es trivial. Adicionalmente las consultas construidas por el usuario tendrán que ser lo más parecidas posibles al documento que se desea encontrar. Sin embargo puede que el usuario sólo posea un conocimiento limitado a unas pocas frases o palabras claves que describan al documento.

Tabla 2.5: Modelo del Espacio Vectorial: Ventajas y Desventajas.

Ventajas	**Desventajas**
Similaridad	Generalmente ineficiente
Ponderación	Poco escalable
Posicionamiento	Consultas deben ser similares a los documentos
Agrupamiento	Asume independencia de los términos
Clasificación	No considera aspectos semánticos del lenguaje natural
Retroalimentación	Expresividad limitada

2.2.4. Modelo Probabilístico

El Modelo del Espacio Vectorial aunque en principio plantea mejoras con respecto al Modelo Booleano, tales como permitir búsquedas más flexibles, clasificación automática de documentos, y descubrimientos de relaciones no triviales entre documentos mediante agrupación, está fuertemente basado en el criterio de similitud.

El criterio de similitud no siempre resulta adecuado para resolver una consulta. Supóngase el siguiente escenario: Existe un repositorio de 1000 documentos de los cuales sólo 100 están indexados con la palabra *computadora*. A su vez un usuario ingresa al sistema la consulta *computadora*. De de los 100 documentos indexados con la palabra computadora, sólo 1 es relevante para el usuario. De los 900 documentos no indexados con la palabra *computadora*, 10 sí son relevantes para el usuario (Por ejemplo los documentos relevantes fueron indexados con otros términos tales como *computador*, *ordenador*, etc).

Si un documento es tomado aleatoriamente del conjunto de documentos indexados con la palabra *computadora*, entonces el usuario tendrá una probabilidad de $\frac{1}{100} = 0,01 = 1\,\%$ de obtener un documento relevante. Por el contrario si el sistema retorna aleatoriamente un documento del conjunto de documentos

no indexados con la palabra *computadora*, entonces la probabilidad de que el documento retornado sea relevante será de $\frac{10}{900} = 0,011 = 1,1\,\%$. Esto claramente contrasta con el criterio de similitud explicado en la sección 2.2.3.2. En este caso la similitud no es una buena medida de la relevancia de un documento.

Debido a que el criterio de similitud no siempre resulta adecuado para para juzgar la relevancia de un documento, un criterio alternativo fue enunciado por Stephen Robertson en 1977. Robertson definió formalmente un criterio mas preciso para determinar la relevancia de un documento como se sigue:

> "Si la respuesta de un sistema de recuperación de información es una colección de documentos ordenados decrecientemente según la probabilidad de relevancia para el usuario que envió la solicitud, donde las probabilidades son estimadas de la forma más precisa posible, sobre la base de todos los datos que se han puesto a disposición del sistema para este fin, entonces la eficacia total del sistema con respecto a sus usuarios será la mejor que pueda ser obtenida con base a estos datos."

Un Modelo Probabilístico de Recuperación de Información es aquel que se basa en la teoría de probabilidades para poder estimar si un documento es relevante o no dada una determinada consulta. Los Modelos de Recuperación de Información Probabilística intentan estimar la probabilidad de que un usuario encuentre aleatoriamente un documento particularmente relevante de la colección. Un Modelo de recuperación de Información Probabilístico procura estimar lo mejor posible todas las relaciones entre los documentos y las consultas, y cualquier otra relación que se considere importante, a fin de poder estimar lo mejor posible el grado de relevancia de los documentos.

2.2.4.1. Aproximación Probabilística

La noción de probabilidad de algo, por ejemplo la probabilidad de relevancia denotada como $P(R)$, es usualmente formalizada a través del concepto de un experimento, donde un experimento es el proceso en el cual una observación es realizada. El conjunto de todas las salidas del experimento es llamada espacio muestral. En el caso del experimento *tomar aleatoriamente un documento del repositorio y constatar que éste sea relevante,* el espacio muestral podría ser el conjunto de eventos $\Omega_1 = \{\omega_1, \omega_2\}$ dónde $\omega_1 =$ el documento es relevante y $\omega_2 =$ el documento es irrelevante.

Luego se podría definir la variable aleatoria R que retorna los valores $\{0, 1\}$, donde R evalúa 0 si el documento es irrelevante, o 1 si el documento es relevante (evento Bernoulli). Finalmente podríamos calcular el valor de $P(R)$.

$$R(\omega) = \begin{cases} 1 & \text{si el documento es relevante} \\ 0 & \text{si el documento es irrelevante} \end{cases} \tag{2.17}$$

Sin embargo considérese el caso $P(D_k)$ donde D_k es la variable aleatoria con espacio muestral $\Omega_2 = \{\omega_1 =$ término k ocurre en D, $\omega_2 =$ término k no ocurre en $D\}$

con D_k retornando 0 si k no ocurre en el documento D, o 1 si k ocurre en el documento.

$$D_k(\omega) = \begin{cases} 1 & \text{si el término k ocurre en el documento} \\ 0 & \text{si el término k no ocurre en el documento} \end{cases} \tag{2.18}$$

Con las variables aleatorias 2.17 y 2.18 se podría calcular la función de distribución de probabilidad conjunta $P(R, D_k)$ cuyos dominio podría ser el conjunto $\{(0,0),(0,1),(1,0),(1,1)\}$, o se podría calcular $P(R = 1|D_k = 1)$, la probabilidad condicional de que un documento sea relevante dado que un documento fue indexado con el término k. Lo importante a destacar es que la interpretación de la función de probabilidad P puede variar dependiendo de como se realice la *observación*, incuso tratándose del *mismo* experimento.

Imagínese el experimento en el cual se lanzan dos dados y se observa la suma de sus caras. El mismo experimento se puede repetir pero ahora observando si la suma de los dados es par o impar. Dependiendo del aspecto que se esté observando del experimento cambiará el modelo probabilístico, esta característica contrasta notablemente con respecto al Modelo Booleano y al Modelo del Espacio Vectorial, donde las variables de dichos modelos ya están claramente planteadas. Sin embargo en el caso del Modelo Probabilístico se debe definir primero que aspecto se está observando del experimento, y el modelo es definido sólo después de un análisis meticuloso de todas las variables involucradas. Por supuesto esto añade niveles superiores de complejidad a su diseño pero puede aumentar en gran medida la efectividad del sistema.

2.2.4.2. Modelo de Indexado Probabilístico

Un modelo bastante estudiado fue el planteado por Bill Maron y Larry Kuhns en 1960. En aquel entonces Maron y Khuns no estaban diseñando un modelo de recuperación de información automatizado, sino que en su lugar estaban estudiando la clasificación manual de libros por parte de un bibliotecario hipotético. Ellos plantearon el siguiente experimento: Supóngase un bibliotecario que selecciona aleatoriamente un libro de una colección, luego de un conjunto finito de términos, el bibliotecario le asigna al libreo un subconjunto de términos para poder clasificarlo.

Por ejemplo si el bibliotecario toma un volumen de la novela *Don Quijote* entonces cuál sería la probabilidad que indexara dicho libro con la palabra: *Cervantes*. De manera formal ésta situación se expresa en 2.19

$$P(D|T) = \frac{P(T|D)P(D)}{P(T)} \tag{2.19}$$

Donde D denota un documento cualquiera de la colección y T un término cualquiera del conjunto total de términos.

A efectos de establecer un posicionamiento de los documentos según sus niveles de relevancia no es necesario calcular exactamente los valores de la función P sino los valores de una función P'(no necesariamente una función de probabilidad) que mantenga el posicionamiento correlativo de los documentos.

Analizando la expresión 2.19 podemos asumir que el denominador $P(T)$, la probabilidad que se use un término cualquiera T es constante (todos los términos del conjunto total de términos tienen igual probabilidad de ser utilizados para indexar documentos), por tanto se puede obviar en los cálculos.

Por su parte $P(D)$ no depende de T y se considera el peso del documento. En principio se sugirió que $P(D)$ fuese calculado como el cociente entre la cantidad de usos previos del documento entre la cantidad de uso de todos los documentos de la colección, véase la expresión 2.20. Este enfoque es el implementado por muchos buscadores *web*, dónde el documento HTML con mayor número de *clicks* es más probable a ser recuperado.

Incluso se han implementado formas alternativas altamente exitosas para el cálculo de $P(D)$, como por ejemplo el algoritmo PageRank, expresión 2.21, explicado en mayor detalle en la sección 3.2 página 60, algoritmo planteado por Sergey Brin y Lawrance Page para el popular motor de búsqueda Google.

$$P(D) = \frac{\text{cantidad de usos del documento } D}{\text{cantidad de usos de todos los documentos}} \tag{2.20}$$

$$P(D) = \text{PageRank}(D) \tag{2.21}$$

Nótese que el cálculo de $P(D)$, al no depender de la consulta, puede ser realizado antes de que se ingrese la misma al sistema, posiblemente al momento de ser indexados los documentos. Dicha técnica es comúnmente denominada posicionamiento Estático, ya que se puede calcular una ponderación de los documentos incluso antes de que el usuario ingrese la consulta al sistema.

Finalmente cabe destacar que el cálculo $P(T|D)$ depende del bibliotecario, ya que es el responsable de seleccionar un término aleatoriamente dado un libro de la colección. Para su estimar esta probabilidad se deben emplear técnicas adicionales, una de ellas consiste en almacenar cada uso de un documento junto a los términos que se emplearon para su recuperación.

2.2.4.3. Modelo de Recuperación Probabilística

En 1976 Stephen Robertson y Karen Spärck-Jones sugirieron que que se debía ordenar los documentos calculando $P(R|\vec{D})$ la probabilidad de relevancia R dado una descripción del documento $\vec{D}$. Sin embargo, a diferencia del Modelo de Indexado Probabilístico 2.2.4.2, el Modelo de Recuperación Probabilística plantea que $\vec{D}$ es un vector de componentes binarias, donde cada componente D_k representa ausencia o presencia de un término en la descripción $\vec{D}$. Esto contrasta con el Modelo de Indexado Probabilística 2.2.4.2 donde D representaba un documento relevante.

Por ejemplo supóngase 10 libros con la misma descripción $\vec{D}$ y 9 de ellos son relevantes, entonces $P\left(R|\vec{D}\right) = 0,9$. El modelo de recuperación probabilística plantea el cociente 2.22 como una buena medida para cuantificar la relevancia[5].

$$\text{relevancia}(D) = \frac{P\left(R|\vec{D}\right)}{P\left(\overline{R}|\vec{D}\right)} \tag{2.22}$$

[5] $\overline{R}$ denota irrelevancia. Los eventos R y $\overline{R}$ se consideran eventos mutuamente excluyentes.

Desarrollando la expresión 2.22con el teorema de Bayes se obtiene la expresión 2.23

$$\text{relevancia}\,(D) = \frac{P\left(\vec{D}|R\right) P\,(R)}{P\left(\vec{D}|\overline{R}\right) P\left(\overline{R}\right)} \tag{2.23}$$

Si D_k representa la componente *k-ésima* del vector $\vec{D}$ obtenemos la expresión 2.24a partir de la expresión 2.23

$$\text{relevancia}\,(D) = \prod_{k=1}^{m} \frac{P\,(D_k|R)\,P\,(R)}{P\left(D_k|\overline{R}\right) P\left(\overline{R}\right)} \tag{2.24}$$

Finalmente se realizan tres optimizaciones a la expresión 2.24. Primero, los elementos son ordenados por la suma de los logaritmos de las probabilidades, en vez del cálculo de la productoria de los cocientes de la probabilidades. Segundo, el cociente $\frac{P(R)}{P(\overline{R})}$ es ignorado. Tercero, se sustrae el *k-ésimo* término 2.25 para evitar calcular el peso de los términos no presentes en el documento (cuyo peso es 0).

$$\sum_{k=1}^{m} \left(\lg \left(\frac{P\,(D_k = 0|R)}{P\left(D_k = 0|\overline{R}\right)} \right) \right) \tag{2.25}$$

Finalmente se obtiene 2.26

$$\sum_{D_k \epsilon \text{terminos} \neq 0} \left(\lg \left(\frac{P\,(D_k = 1|R)\,P\left(D_k = 0|\overline{R}\right)}{P\left(D_k = 1|\overline{R}\right) P\,(D_k = 0|R)} \right) \right) \tag{2.26}$$

2.2.4.4. Ventajas

- En general la gran ventaja de los Modelos de Recuperación de Información Probabilísticos radica en su flexibilidad de poder tomar en cuenta toda la información disponible para estimar lo mejor posible qué documentos se consideran altamente probables de ser relevantes para el usuario.
- Los Modelos Probabilísticos de Recuperación de Información pueden ser sucesivamente refinados para mejorar sus tasas de recuperación y de precisión (Ver secciones 2.1 y 2.4, respectivamente) con técnicas que tomen en cuenta las propiedades del lenguaje natural, la retroalimentación con el usuario, la frecuencia de uso de los término presentes en la consulta, así como la frecuencia total de uso del documento. Se pueden tomar en cuenta los metadatos que describen un documento y asignarles una ponderación específica dentro del modelo. También se puede ponderar las relaciones que hay entre los documentos y tomar en cuenta la estructura interna del mismo. Por ejemplo, en un contexto *web* es muy común la práctica de asignarle una ponderación mayor a los términos que ocurren en los títulos, con respecto al asignado a los términos que ocurren en un párrafo.

2.2.4.5. Desventajas

- La calidad de los resultados retornados dependerá de cómo se plantee la observación del experimento, y por tanto la forma en que se defina el modelo. Por esta razón los modelos muy precisos o específicos pueden llegar a ser conceptualmente complejos de diseñar y costosos de desarrollar.
- La independencia supuesta de los términos, planteada en los Modelos de Indexado y Recuperación Probabilística es en muchos casos poco realista. Por ejemplo en el presente documento la palabra *información* tiene una alta probabilidad de ocurrir después de la palabra *recuperación*. Sin embargo estas particularidades puedes ser abordadas realizando estudios de distribución de términos en colecciones de documentos.

Tabla 2.6: Modelo Probabilístico: Ventajas y Desventajas.

Ventajas	**Desventajas**
Puede ser altamente eficaz	Alta complejidad conceptual
Flexible, modificable y extensible	Suposiciones poco realistas
Toma en cuenta los metadatos	Puede ser poco escalable
Permite retroalimentación	La definición del modelo depende de la observación
Permite posicionamiento	Difícil de estimar cuál es el mejor modelo

2.3. Conferencias

Las conferencias son mecanismos mediante el cual los investigadores presentan y discuten tópicos relacionados con el estado del arte[6] de las investigaciones. En particular TREC y ACM SIGIR se consideran las conferencias más importantes dentro del campo de la Recuperación de Información.

2.3.1. TREC

La Conferencia de Recuperación de Texto, Text REtrieval Conference, es una conferencia anual patrocinada por National Institute of Standards and Technology, y el Departamento de Defensa de los Estados Unidos. TREC empezó en 1992 como parte del proyecto de investigación militar TIPSTER. A pesar de que TIPSTER se consideró concluido, TREC continuó bajo un enfoque académico y comercial. Su propósito es apoyar la investigación dentro de la comunidad de investigadores dedicados al estudio de la Recuperación de de Información, brindado la infraestructura necesaria para la evaluación, a gran escala, de las metodologías de la recuperación de documentos de texto[10]. Según si sitio *web* oficial [8], sus objetivos son:

[6] Nivel más alto de desarrollo conseguido en una disciplina determinada.

- Promover la investigación en el campo de la Recuperación de Información.
- Incrementar la comunicación entre la industria, la academia y el gobierno creando un foro abierto para el intercambio de ideas.
- Acelerar la transferencia de tecnología desde los laboratorios de investigación a los productos comerciales por medio del desarrollo sustancial de mejoras en las metodologías de la recuperación de información aplicables a problemas reales.
- Incrementar la disponibilidad de técnicas de evaluación apropiadas para uso industrial y académico, incluyendo el desarrollo de nuevas técnicas de evaluación más aplicables a los sistemas actuales.

TREC está estructurado es una serie de talleres *workshops* dónde se discuten y exponen un conjunto de tópicos relacionados con diversas áreas dentro del campo de la Recuperación de Información. Cada tópico dentro del taller es denominado *track*. Algunos de los *tracks* más importantes son *Web Track, Microblog track, Genomic Track, Medical Records Track*, entre otros.

TREC es considerada una de las conferencias más importantes a nivel mundial porque ha generado un gran volumen de publicaciones que han influido sensiblemente en las tecnologías incorporadas en los motores de búsqueda comerciales modernos.

Cabe destacar que TREC también ha servido como modelo para otras conferencias de renombre mundial, como la iniciativa japonesa NTCIR, *NII Test Collection for IR Systems*, iniciada en 1999, y la iniciativa europea CLEF, *Cross Language Evaluation Forum*, iniciada en el año 2000.

2.3.2. ACM SIGIR

SIGIR corresponde a las siglas en inglés de Special Interest Group on Information Retrieval. SIGIR es un Grupo de Interés Especial dentro de la ACM (*Association for Computing Machinery)* cuyo ámbito de especialidad es la teoría y la aplicación del uso de computadoras para la adquisición, organización, almacenamiento, recuperación y distribución de la información. SIGIR hace énfasis en trabajar en información no numérica, abarcando desde el lenguaje natural, hasta base de datos altamente estructuradas [9].

La primera conferencia anual internacional SIGIR se llevó a cabo en 1978. SIGIR también trabaja en conjunto con otros SIG, tales como SIGWEB y SIGMOD, grupos de interés especial encargados de desarrollar el área relativa a la *World Wide Web* y el manejo de grandes volúmenes de datos (*Management of Data*).

Asimismo SIGIR ofrece premios y reconocimientos a los mejores trabajos de investigación, desarrollados tanto por profesionales como por estudiantes universitarios. El premio Gerard Salton, es concedido cada tres años al trabajo más influyente dentro del campo de Recuperación de Información. Al igual que las conferencias TREC, descritas en la sección 2.3.1, las conferencias ACM SIGIR se encuentran en un lugar destacado a nivel mundial dentro del campo de la Recuperación de Información por la alta calidad de sus publicaciones científicas.

Tabla 2.7: Evolución de los Sistemas de Recuperación de Información

Año	**Descripción**
1950	Calvin Mooers acuña el término Recuperación de Información
1951	Primeros Sistemas de Recuperación de Información, Modelo Booleano
1955	Kent define las medidas Tasa de Recuperación y Tasa de Precisión
1960	Salton sistema SMART, Maron y Kuhns Modelo de Indexado Probabilístico
1974	Bookstein y Swanson modelo Poisson
1976	Robertson y Spärck-Jones Modelo de Recuperación Probabilística
1978	Primera conferencia ACM SIGIR
1981	Robertson, van Rijsbergen y Porter estiman frecuencias de ocurrencias de términos
1989	Modelos acústicos, reconocimiento del lenguaje natural
1991	Howard Turtle, redes Bayesianas
1992	Primera conferencia TREC
1993	Primeros Motores de Búsqueda en la *web*
1994	Robertson y Walker Algoritmo Okapi BM25
1997	Navarro y Baeza-Yates, Modelo Booleano Extendido
1998	Motores de Búsqueda modernos MSN Search (Bing), Google, PageRank

Aquí concluye el capítulo **Recuperación de Información**. A continuación se presenta el capítulo **Motores de Búsqueda** donde se exponen los fundamentos tecnológicos y las principales características de diseño presentes en los Motores de Búsqueda modernos basados en *web*.

Capítulo 3

Motores de Búsqueda (*Search Engines*)

A continuación se exponen las características de la *web* que condicionan el diseño de los Motores de Búsqueda modernos y adicionalmente se describen algoritmos influyentes que analizan la estructura de las relaciones entre documentos relacionados. Finalmente se describen los Motores de Búsqueda propiamente y se explica su funcionamiento.

Un Motor de Búsqueda es una herramienta informática que ayuda a los usuarios a encontrar información digital. Dentro del área de la Recuperación de la Información (ver capítulo 2) los Motores de Búsqueda se consideran una implementación de los Sistemas de Recuperación de Información, explicados en la sección 2.1.

Bajo un contexto *web* un Motor de Búsquedas está diseñado para recuperar información contenida en los documentos HTML que conforman la *web*. Los resultados son generalmente presentados como una lista de páginas según su relevancia. La información suele consistir en documentos HTML, pero también imágenes, audio, video y otros tipos de archivos, incluso algunos Motores de Búsqueda pueden indexar información contenidas en Bases de Datos y Directorios Abiertos. A diferencia de los directorios *web* mantenidos por editores humanos, los Motores de Búsqueda mantienen actualizados los índices de forma automática ejecutando algoritmos llevados a cabo por plataformas informáticas llamados *web crawlers* o arañas. Asimismo como los documentos HTML se encuentran relacionados entre sí, muchos Motores de Búsqueda utilizan está estructura para mejorar la calidad de los resultados retornados.

3.1. Fundamentos Tecnológicos

La presente sección describe las generalidades de la *web* y sus características inherentes con el propósito de ser tomadas en cuenta a la hora de desarrollar cualquier plataforma que se apalanque de este medio.

3.1.1. Tecnologías en la *web*

El rápido crecimiento de la *web* se debe en gran medida a las características de diseño del protocolo HTTP. A pesar que a principios de la década de 1990 ya existían otros protocolos de comunicaciones como Gopher,

Usenet o los canales IRC, el protocolo HTTP encontró una mayor aceptación por parte de sus usuarios. Los características que hicieron a HTTP un protocolo tan popular fueron:

- Comunicación simple entre clientes y servidores: Gran variedad de cargas útiles (*payloads*[1]) podían ser transmitidas de forma asíncrona, codificadas dentro de un lenguaje de marcado.

- Clientes flexibles: Los navegadores ignoran todas las etiquetas inválidas del lenguaje de marcado, haciendo el mejor esfuerzo posible por mostrar los contenidos.

Asimismo el lenguaje de marcado HTML fue especificado poco después de HTTP. Se llamó en principio HTML *tags*, un conjunto de dieciocho sencillas etiquetas, donde la más importante era la relativa a los hipervínculos[2]. El funcionamiento de los hipervínculos está fuertemente ligado en el sistema DNS. Los hipervínculos consisten básicamente en el nombre de dominio del servidor *web* que aloja los documentos HTML y la ubicación del documento que se quiere recuperar dentro de dicho servidor, además de información opcional pasada con parámetros adicionales. Gracias al sistema DNS la *web* mantiene su consistencia, ya que aunque el servidor *web* cambie de dirección IP, su nombre de dominio no se modifica, debido a que las registros en el sistema DNS que relacionan los nombres de dominio con las direcciones IP, se actualizan automáticamente. Por tanto la estructura global de hipervínculos no se ve afectada por el cambio de direcciones IP de los servidores *web*.

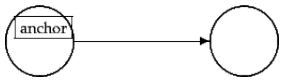
Estas características aunque simples en principio, contribuyeron enormemente al crecimiento de la *web*. Los desarrolladores de los primeros navegadores también permitieron que los usuarios pudieran visualizar fácilmente el marcado de los documentos HTML, por tanto miles de personas empezaron no sólo a consumir, sino a producir multitud de contenidos subiéndolos a la *web*. Como los navegadores simplemente ignoraban lo que no entendían, el programador no tenía porque preocuparse que el sistema colapsara por errores de forma en el documento HTML. Sin embargo esto tuvo como consecuencia que muchos documentos mal formados fueran subidos a la *web,* y con el tiempo surgieron muchas versiones de HTML con características distintas, y navegadores que aceptaban ciertas etiquetas que otros ignoraban.

3.1.2. La Forma de la *web*

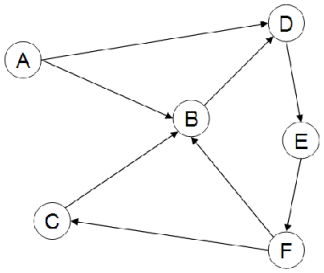
A medida que la *web* crecía muchos investigadores encontraron en ella una fuente de datos extensa y públicamente accesible para estudiar. Los primeros estudios se basaron en cuantificar el volumen de la colección global de documentos que la integraban, así como también entender la forma en que dichos documentos estaban relacionados. Los investigadores modelaron la *web* como un grafo dirigido donde cada vértice representaba un documento HTML estático y cada hipervínculo del documento correspondía a una arista dirigida al documento HTML relacionado (Ver figura 3.1).

[1] Cuerpo de un mensaje digital (paquete) transmitido entre dos dispositivos siguiendo un protocolo de comunicaciones común.

[2] Cadena de caracteres utilizada por el hipertexto para hacer referencia a contenido relacionado semánticamente.



(a) Los documentos representan vértices y los hipervínculos aristas.



(b) Un grafo no fuertemente conectado, no se puede navegar de la página B a la A siguiendo los hipervínculos.

Figura 3.1: La *web* como un grafo dirigido.

Un primer modelo ideal se podría definir estableciendo que si todos los hipervínculos salientes de un documento HTML hacen referencia a cualquier otro documento siguiendo una función de distribución de probabilidad uniformemente distribuida, entonces la cantidad de hipervínculos entrantes a un documento HTML seguirá una distribución Poisson. Sin embargo en la práctica los hipervínculos salientes de un documento no hacen referencia a otros documentos de forma uniformemente distribuida, sino que los mismos, al ser creados por los usuarios de los documentos, toman en cuenta las asociaciones semánticas que relacionan un documento con otro.

Como consecuencia directa resulta que el conjunto de hipervínculos que relacionan los documentos HTML, reflejan las correlaciones entre los conceptos del pensamiento humano. La estructura semántica de la *web* ha sido estudiada permitiendo categorizar las documentos HTML en tres categorías principales y tres secundarias según sus las características de sus hipervínculos[11] .

Páginas de entrada (IN): Los usuarios pueden navegar desde una página de entrada a una página SCC siguiendo los hipervínculos.

Páginas SCC: En las páginas SCC (*Strong Connected Components,* Componentes Fuertemente Conectados) los usuarios pueden navegar de una página SCC a una página de salida, o navegar entre páginas SCC siguiendo los hipervínculos. Los usuarios no pueden navegar de una página SCC a una página de entrada siguiendo los hipervínculos.

Páginas de salida (OUT): Los usuarios no pueden navegar de una página de salida a una página SCC, o a una página de Entrada siguiendo los hipervínculos.

La mayoría de los documentos *web* entran en estas tres categorías. Los pocos restantes se clasifican en tubos, tentáculos y páginas desconectadas.

Páginas tubos: Son páginas que no son páginas SCC y que permiten navegar desde páginas de entrada a páginas de salida directamente.

Páginas tentáculos: Son páginas que permiten navegar de las páginas de entrada a páginas que no hacen referencia a ninguna página, y páginas que ninguna otra página hace referencia que permiten navegar a páginas de salida.

Páginas desconectadas: Páginas que ninguna otra página referencia y que tampoco hacen referencia a ninguna página.

La figura 3.2 representa gráficamente como se organizan las páginas en la *web* según las características de los hipervínculos[11].

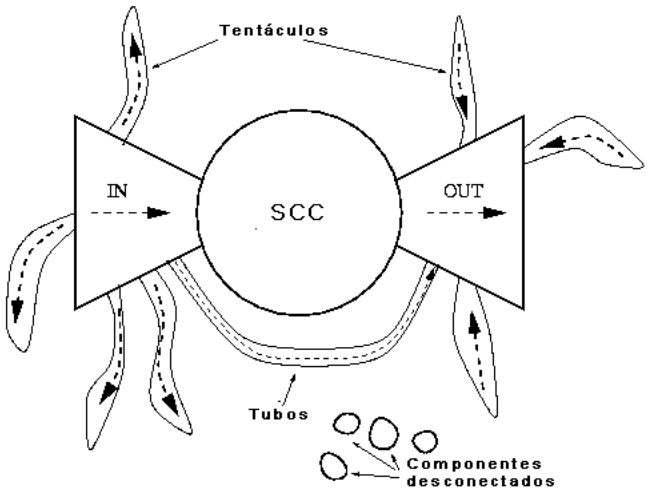


Figura 3.2: Estructura general de la *web*.

3.1.3. Algoritmos Frecuentes

El algoritmo HITS y el algoritmo PageRank son implementados frecuentemente por los Motores de Búsqueda para determinar la relevancia de los documentos. Usualmente los desarrolladores de los Motores de Búsqueda modifican los mismos para optimizar su rendimiento y aumentar la calidad de los resultados. HITS y PageRank son algoritmos que determinan los pesos de los nodos a partir de las aristas que los relacionan con otros nodos. Para poder aplicar ambos algoritmos es necesario concebir la *web* conceptualmente como un grafo dirigido, como se muestra en la figura 3.4 en la página 67.

3.1.3.1. HITS (*Hyperlink-Induced Topic Search*)

HITS fue desarrollado por Jon Kleinberg y precedió a PageRank. HITS permite encontrar concentradores (*hubs*) y autoridades (*authorities*) dentro de un grafo. Los concentradores son los nodos del grafo que hacen referencia a muchos otros nodos, y las autoridades son aquellos nodos que muchos otros nodos hacen referencia a ellos. Los concentradores actuarían como catálogos y las autoridades como páginas altamente confiables.

HITS calcula el valor de concentración y de autoridad para cada nodo de un grafo. Sin embargo el algoritmo es ejecutado al momento de realizarse la consulta, ya que sólo calcula los valores de concentración y autoridad en los documentos recuperados y no todos los de la colección. Los valores de concentración y de autoridad son definidos con la siguiente expresión mutuamente recursiva:

> "El valor de autoridad de un nodo debe ser igual a la suma de los valores de concentración de todos los nodos que apuntan a él, y el valor de concentración de un nodo será igual a la suma de todos los valores de autoridad de todos los nodos apuntados por él"

Lo cual matemáticamente se representa en el algoritmo 3.1. Sea el grafo $G(V, A)$ donde V es el conjunto de todos los vértices y A es el conjunto de todas las aristas. N es el conjunto de todos los nodos que hacen referencia al nodo v. M es el conjunto de nodos que v hace referencia.

Algoritmo 3.1 HITS

$$\begin{aligned} \forall v \epsilon V \quad \text{auth}(v) &= \sum_{k \epsilon N} \text{hub}(k) \\ \forall v \epsilon V \quad \text{hub}(v) &= \sum_{k \epsilon M} \text{auth}(k) \end{aligned} \tag{3.1}$$

Para el cálculo de los valores de concentración se implementa el método iterativo descrito a continuación:

1. Inicializar todos los valores de concentración y de autoridad con valor 1.
2. Calcular los valores de concentración y autoridad descritos en 3.1.
3. Normalizar[3] los valores dividiendo cada valor de concentración entre la raíz cuadrada de la suma de los cuadrados de todos los valores de concentración, y dividiendo cada valor de autoridad entre la raíz cuadrada de la suma de los cuadrados de todos los valores de autoridad.
4. Repetir el paso 2 si los valores de concentración y autoridad divergen.

[3]La normalización se hace necesaria ya que el valor de concentración y autoridad divergen para un número finito de iteraciones, a menos que dichos valores se normalicen después de cada iteración.

3.1.3.2. PageRank

PageRank fue implementado por Larry Page y Sergey Brin en 1998 para el popular Motor de Búsquedas Google. Inicialmente llamado Backrub, Google comenzó como una investigación doctoral en la Universidad de Stanford. PageRank es utilizado para ponderar el grado de importancia de los nodos de un grafo. En el caso de la *World Wide Web* PageRank pondera la relevancia relativa de una página *web* con respecto a todas las demás, asignándole un valor usualmente entre cero y uno. La relevancia de una página no depende del contenido de la misma, sino de la estructura general de la colección de documentos. Cada enlace que referencia a una página se interpreta como un voto a favor de la página referenciada porque esta contiene información de calidad. PageRank modela una técnica de ponderación de citas donde los documentos más citados serán considerados de mayor prestigio. Además cada voto no tiene la misma ponderación. Los votos de las páginas de alta calidad tienen un mayor peso que los votos de las páginas de menor calidad.

El valor PageRank de un documento HTML se puede calcular de la siguiente forma:

> "El valor PageRank de un documento es igual a la suma de todos los valores de los enlaces que hacen referencia al documento. El valor de un enlace individual es igual al valor PageRank del documento que lo contiene, dividido entre el total de enlaces salientes distintos que hay en el documento."
>
> La expresión 3.2 formaliza matemáticamente el cálculo del valor PageRank de un documento D. M es el conjunto de todas los documentos *web* con enlaces entrantes al documento D. D_j es el *j-ésimo* documento *web* perteneciente a M. $L(D_j)$ es una función que retorna la cantidad de enlaces salientes del documento D_j. $PR(D_j)$ corresponde al cálculo recursivo de la función PageRank del documento D_j. $d \epsilon [0, 1]$ es un factor de amortiguación y N la cantidad de documentos *web* en la colección.

Algoritmo 3.2 PageRank

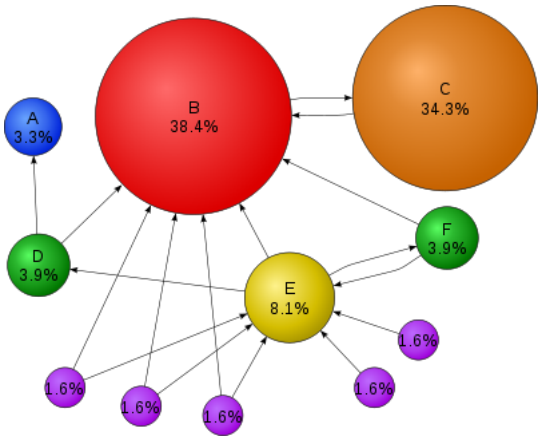
$$\text{PageRank}(D) = (1 - d)\,\frac{1}{N} + d \sum_{D_j \epsilon M} \frac{\text{PageRank}(D_j)}{L(D_j)} \tag{3.2}$$

La expresión 3.2 es interpretada comúnmente como el Modelo del Navegante Aleatorio[16], el cuál consiste en un usuario hipotético que navega por la *web* principalmente seleccionando los hipervínculos en las páginas de forma aleatoria y de vez en cuando utiliza el historial, la barra de navegación o los marcadores para acceder a un documento. En general PageRank se puede interpretar como la probabilidad que un usuario cualquiera visite una determinada página durante una sesión de navegación.

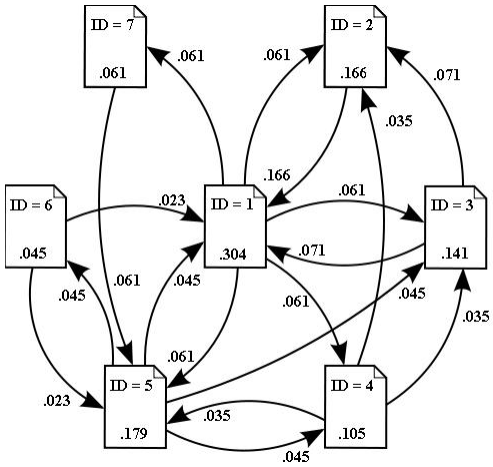
Estando en una página cualquiera el usuario bien puede seleccionar aleatoriamente cualquier enlace dentro de la página actual, o puede seleccionar una página aleatoriamente de la colección ingresando su URL (*Uniform Resource Locator*) .en la barra de navegación, utilizando el historial de navegación, o usando los marcadores del navegador *web*. La suma de los dos términos en 3.2 corresponde a este hecho. El término

$\sum_{D_j \epsilon M} \frac{\text{PageRank}(D_j)}{L(D_j)}$ es el cálculo del valor PageRank de la página en sí. El factor de amortiguación d es un valor que permite modelar si el usuario seleccionó un enlace o prefirió cualquier otra forma de navegación. Si $d = 0$ el usuario siempre usa marcadores, historial o barra de navegación para visitar otras páginas, si $d = 1$ el usuario siempre usa los hipervínculos para navegar. Se ha determinado empíricamente que $d \approx 0,85$ [17]. Finalmente $\frac{1}{N}$ representa la porción del PageRank que es transferida a la página si esta no es accedida mediante un hipervínculo.

Es importante destacar que la suma de todos los valores PageRank de las páginas D_j en la colección C siempre es igual a $\sum_{D_j \epsilon C} \text{PageRank}(D_j) = 1$, y que el PageRank de una página no es un valor que la página genera o elimina, sino que es un valor transferido de otras páginas por los enlaces entrantes y transferido hacia a otras páginas por sus enlaces salientes, más la cantidad de PageRank concedida equitativamente a todas las páginas, porque el usuario podría no haber utilizado un enlace para acceder a la página. La figura 3.3 muestra dos representaciones de la forma en que el algoritmo PageRank distribuye los valores PageRank entre los documentos de una colección.



(a) **Representación Subjetiva** La página B obtiene un mayor PageRank porque muchas páginas la referencian. B sólo referencia a una página, la C. Sin embargo aunque C sólo es referenciado una vez, obtiene mucho PageRank porque la página B es considerada de alta calidad. En este caso el factor de amortiguación $d \neq 1$



(b) **Representación Cuantitativa** El PageRank de una página es igual a la suma de los PageRank transferidos en los enlaces entrantes a la página. Una página transfiere cantidades iguales PageRank a otras páginas en sus enlaces salientes[18]. En este caso $d = 1$

Figura 3.3: PageRank

El cálculo de los valores PageRank de los documentos se puede realizar de diversas formas y no depende de la consulta realizada por el usuario (Posicionamiento Estático). Dependiendo de las características de la implementación del algoritmo PageRank, el desempeño del Motor de Búsqueda puede variar considerablemente. Una solución iterativa es descrita a continuación:

1. En la primera iteración $t = 0$ se asume que todas los documentos tienen un PageRank = $\frac{1}{N}$ como valor inicial.
2. Calcular para todas las páginas la expresión $\frac{1-d}{N} + d \sum_{D_j \epsilon M} \frac{\text{PageRank}(D_j)}{L(D_j)}$ explicada en 3.2.
3. Repetir el paso 2 hasta que los valores PageRank converjan.

3.2. Estrategias de Búsquedas

En general existen dos tipos de estrategias de búsquedas en la *web*, los Motores de Búsqueda propiamente dichos y los Directorios Temáticos. Aunque su finalidad es la misma, recuperar los documentos con la información solicitada por el usuario, su diseño y sus estrategias de búsqueda son bastante distintas.

3.2.1. Directorios Temáticos

Los Directorios Temáticos son Sistemas de Recuperación de Información que se organizan de un modo semejante a los páginas amarillas. Básicamente existe un estructura de categorías jerárquicas que tratan diverso tópicos. Dentro de cada categoría se encuentra un listado de documentos relacionados y posiblemente un conjunto de subcategorías también relacionadas. Los Directorios Temáticos son elaborados manualmente y su estructura depende del criterio de organización de sus editores. Aunque sus contenidos suelen tener mayor calidad que aquellos indexados automáticamente, ya que un editor profesional es el encargado de construirlos, sin difíciles de mantener y suelen presentar problemas de escalabilidad cuando la información es abundante, o se desconoce su estructura y naturaleza de antemano.

3.2.2. Motores de Búsqueda (*Search Engines*)

A diferencia de los Directorios Temáticos, los Motores de Búsqueda son sistemas informáticos capaces de indexar contenidos de forma automática en sus bases de datos. Usualmente una araña o *web crawler* es el módulo responsable de recolectar los documentos que se almacena en las Bases de Datos del Motor de Búsquedas. Luego un algoritmo es ejecutado para representar los contenidos de los documentos en estructuras de datos llamados índices, que son utilizados para recuperar el documento solicitado cuando el usuario realiza una consulta al sistema. Los Motores de Búsqueda manejan grandes volúmenes de información, y están constantemente actualizando sus contenidos. Sin embargo pudieran presentar resultados con baja tasa de precisión (ver sección 2.1.2.3) al no ser indexados por profesionales expertos.

3.2.3. Tipos de Motores de Búsqueda

Según Jorge Salas[12], en general los Motores de Búsquedas se pueden clasificar en:

Buscadores de Portal: Son aquellos Motores de Búsqueda que sólo recuperan documentos dentro del ámbito de un portal en específico.

Sistemas Mixtos: Sistemas que combinan un Motor de Búsqueda con las ventajas de un Directorio Temático. Se pueden realizar búsquedas automáticas dentro de cada una de las categorías del directorio.

Meta-buscadores: Motores de Búsquedas que realizan la búsqueda utilizando otros Motores de Búsquedas, retornando resultados de diferentes fuentes.

FFA: Los directorios FFA (*Free For All*) son directorios donde cualquier usuario puede registrar sus enlaces por un tiempo limitado.

Buscadores Verticales: Motores de Búsquedas que trabajan sobre colecciones de documentos específicas.

La tabla 3.1 enumera las principales ventajas y desventajas de los Motores de Búsquedas y los Directorios Temáticos.

Tabla 3.1: Motores de Búsquedas e Índices Temáticos: Ventajas y Desventajas.

Estrategia de Búsqueda	**Motores de Búsqueda**	**Directorios Temáticos**
Ventajas	Escalable	Alta Calidad de contenidos
	Actualización automática	Contenidos relacionados agrupados
	Indexado automático	Sugieren nuevos tópicos de búsqueda
Desventajas	Resultados poco precisos	Difícil de Mantener
		Difícil de Actualizar
		Indexado Manual

3.3. Funcionamiento de los Motores de Búsquedas

Todo Motor de Búsqueda en general realiza tres procesos relacionados entre sí que implican diferentes consideraciones de diseño y tecnológicas. En general estos tres procesos son:

- *Web Crawling*
- Indexado
- Búsqueda o Recuperación

Los algoritmos y técnicas empleados en la realización de cada uno de los procesos descritos puede variar enormemente de un Motor de Búsqueda o otro. Sin embargo cada uno de estos procesos realiza una tarea bastante específica que no suele variar en las implementaciones de los buscadores.

***Web Crawling*:** Consiste en la extracción de contenido de los documentos HTML por medio de un programa informático llamado *web crawler* o araña. Sin embargo este programa no recorre la *web* como una araña, ya que no se ejecuta en los servidores que contienen los documentos a indexar, como su nombre sugiere. En realidad las arañas son programas por lotes que se ejecutan en un servidor o conjunto de servidores (*clusters*), descargando sistemáticamente los documentos HTML alojados en los servidores *web*. Luego de descargados dichos documentos, estos son almacenados en un repositorio para su indexado. Los servidores *web* pueden crear un archivo llamado usualmente *robots.txt*. El mismo permiten a los administradores del sitio *web* especificar a las arañas explícitamente los documentos que se desean indexar en los Motores de Búsquedas. Asimismo las arañas, al igual que los navegadores, se identifican a sí mismas ante el servidor *web* transmitiéndole un identificador en la solicitud HTTP. El identificador consiste en una cadena de caracteres denominada *User-Agent*, donde un agente usuario pueden ser tanto navegadores *web* como arañas o cualquier otro sistema informático.

Indexado: Consiste en ordenar y procesar los documentos del repositorio para extraer información descriptiva de los mismos. Una vez calculados los datos, estos son almacenados en estructuras de datos denominadas índices. Los índices pueden tener diversos diseños, pero en general se denominan índices hacia adelante (*Forward Index*) a los índices que relacionan los documentos con la lista de términos que contiene, además de cualquier otra información descriptiva, e índices invertidos (*Backward Index*) a los índices que relacionan los términos con una lista de documentos que los contienen, posiblemente con información descriptiva adicional. Además varios cálculos adicionales se pueden realizar durante el proceso de indexado, como por ejemplo el cálculo de los valores PageRank (explicado en la sección 3.1.3.2) de los documentos indexados.

Búsqueda o Recuperación: En el contexto *web* la acción de recuperar los documentos relevantes del repositorio es conocida simplemente como *búsqueda*. En este proceso interviene las Interfaces de Usuario utilizadas por el usuario para ingresar la consulta. Una vez ingresada la consulta al sistema, los algoritmos de búsqueda permiten inferir los documentos relevantes procesando los datos descriptivos contenidos en los índices. Luego se elabora una lista con los enlaces a los documentos considerados más relevantes que se retorna al usuario. Los enlaces bien pueden referenciar a copias de documentos almacenados dentro del repositorio particular del Motor de Búsqueda (lo que se conoce como *búsqueda en caché*) o pueden referenciar directamente a documentos alojados es servidores *web* externos.

La estructura general de un Motor de Búsqueda es explicada en la figura adaptada 3.4[14]. En primer lugar se describirán los procesos que no involucran una interacción directa con el usuario.

- Las arañas (*robots*) descargan los documentos HTML de la *World Wide Web* para ser almacenados en una Colección de Datos (*Data Collection*) como *strings* [4] sin procesar. Las arañas descargan los documentos HTML a partir de una lista de URL (*URL list*) que es actualizada constantemente.

- El analizador sintáctico (*Parser*) identifica los elementos sintácticos (título, enlaces, meta-etiquetas, términos, etc) que conforman los documentos HTML a partir de las cadenas de *strings* almacenados en la Colección de Datos.

[4] Cadena de caracteres.

- Los elementos identificados por el analizador sintáctico son consumidos por el indexador (*Indexer*) para crear una asociación URL-lista de términos (*keywords*) que es almacenada en la Base de Datos, creándose un índice hacia adelante .

- Los datos del índice hacia adelante son utilizados por el indexador para calcular propiedades del documento partir los términos contenidos dentro de las páginas. A partir de las URL internas que apuntan a otros documentos, se analizan las relaciones con otras páginas también. Las nuevas URL encontradas son añadidas a la lista URL. Usualmente en esta parte se calcula la relevancia relativa del documento que no depende de la consulta (Posicionamiento Estático).

- Se procede a crear un índice invertido que relaciona los términos con la lista URL a partir del índice hacia adelante.

El proceso anteriormente descrito en principio no involucra ninguna interacción con el usuario, y usualmente se ejecuta continuamente con la finalidad de mantener los índices y las Bases de Datos actualizadas con los últimos contenidos de la *web*. La parte que involucra al usuario se describe a continuación, ella empieza con el ingreso de una consulta en la Interfaz de Usuario.

- El usuario ingresa una consulta mediante una Interfaz de Usuario. Se analiza la consulta ingresada para determinar si la misma consiste en palabras claves, o posiblemente en una lista de URL[5].

- Si la consulta es una lista de URL que identifican documentos HTML sin indexar, entonces se actualiza la lista de URL con las URL de los documentos no indexados. Si la consulta son un conjunto de palabras claves describiendo la información solicitada por el usuario se procede a buscar los documentos asociados a las palabras claves en el índice invertido.

- Una vez identificados los documentos relevantes, se retorna la lista de documentos HTML al secuenciador módulo que ordenará los documentos dependiendo de las características de la consulta y la relevancia relativa calculada al momento del indexado. Finalmente los resultados ordenados por relevancia son retornados en la Interfaz de Usuario.

[5]Algunos Motores de Búsqueda permiten al usuario indicar explícitamente una lista de URL para que sean indexadas, como el servicio *Sitemaps*[15] de Google.

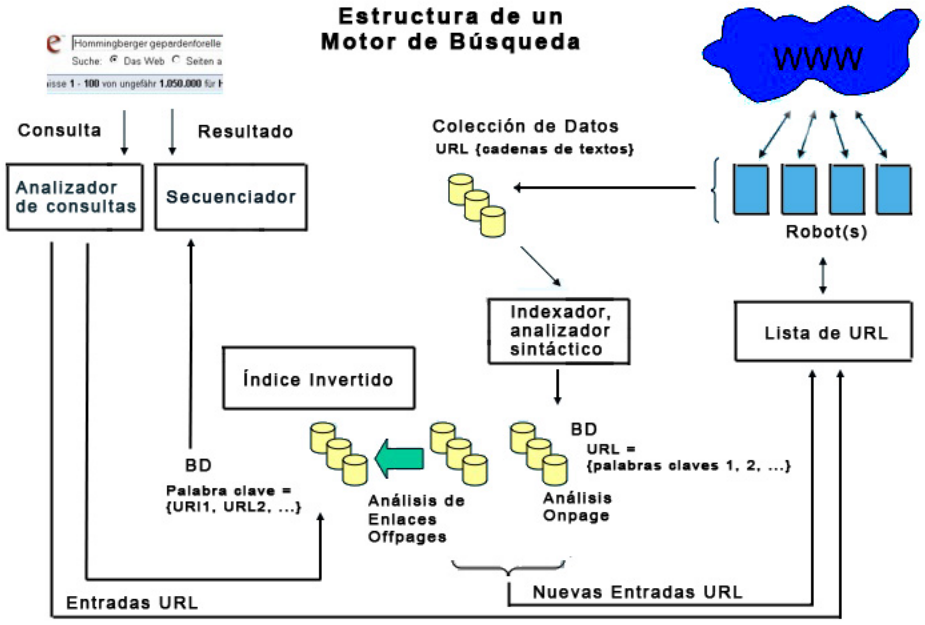


Figura 3.4: Funcionamiento General de un Motor de Búsquedas.

Aquí concluye el capítulo **Motores de Búsqueda**. A continuación se presenta el capítulo **Casos de Estudio** donde se exponen las características de diseño, usabilidad y navegación presentes en varios Motores de Búsqueda de corte académico basados en *web* en producción. También se describe el estado del repositorio de BUSCONEST 1 a la fecha.

Capítulo 4

Casos de Estudio

Los casos de estudios descritos a continuación son plataformas actualmente en producción que indexan contenido de naturaleza académica exclusivamente. Los mismos se exponen como ejemplos orientadores.

4.1. Algunos Buscadores *web* Académicos

A continuación se enumeran una serie de servicios publicados en la *web,* que permiten recuperar información académica, con el propósito de describir sus características más resaltan tes y aportes más significativos al momento de realizarse una búsqueda.

- Google Scholar
- Microsoft Academic Search
- Scirus
- Biblioteca de Recursos Universia

4.1.1. Google Scholar

Google Scholar es el servicio de Google publicado en la *web* para realizar búsquedas específicas sobre documentos de corte académico y de investigación. Google Scholar salió al público en versión beta a finales de 2004 y básicamente emplea las mismas estrategias que han convertido su motor de búsqueda de propósito general en un exitoso servicio, búsquedas sobre una amplia colección de documentos en múltiples fuentes, retorno de enlaces de alta calidad en los primeros lugares de los resultados de la búsqueda en un tiempo muy breve, y una interfaz de usuario muy sencilla, pragmática e intuitiva. Sin embargo Google Scholar conceptualmente se acerca más prototipo inicial de Google, Backrub, Motor de Búsquedas inicialmente diseñado para analizar las citas entre publicaciones, principalmente académicas, con el propósito de otorgarles una mayor importancia aquellas publicaciones que habían sido citadas un mayor número de veces por autores reconocidos.

Google Scholar a pesar de compartir algunas características con un Motor de Búsqueda de propósito general, por ejemplo permite buscar palabras clave o frases completas dentro de los documentos, además soporta el uso de operadores lógicos y operadores específicos en la consulta (OR, -, +, *intitle*, " ") presenta características específicas, tales como[21]:

- Permite buscar los artículos en la colección por autor, según su editorial y rango de fecha de publicación.
- Permite diferenciar la búsqueda por artículos, patentes y documentos legales.
- Contempla un sistema de citas donde un autor puede hacer un seguimiento de los documentos en los que ha sido citado.
- Permite generar automáticamente la referencia del documento buscado en formatos populares para crear citas automáticamente (BibTex, EndNote, RefMan, RefWorks).
- Presenta un directorio temático dónde los artículos están agrupados por disciplinas de investigación y áreas de interés ordenados según su índice h[1].
- Para realizar el posicionamiento de los resultados se emplea un algoritmo específico que toma en cuenta la ponderación de todos los términos en el documento, dónde fue publicado, su autor, y que tan frecuente y recientemente ha sido citado el documento en otras publicaciones académicas.
- Permite la visualización del documento completo y la posibilidad de descargarlo en formato PDF en un sitio externo sólo si si autor así lo permite, sino se presenta una descripción del documento en el sitio *web* donde se encuentra publicado.
- En la visualización de los resultados se muestra el número de veces que sido citado el documento.
- Permite una opción para incorporar bibliotecas digitales completas de centros académicos a la colección pero con restricciones de acceso.
- Para acceder a las bibliotecas digitales se debe poseer una cuenta especial, acceder desde el campus del centro académico, o mediante un *proxy* especializado.
- Provee soporte para publicaciones en múltiples lenguajes.

[1] El índice h o índice de Hirsch es un número utilizado para ponderar la relevancia de una publicación científica en función de su número de citas en otros documentos y la cantidad de documentos publicados por su autor.

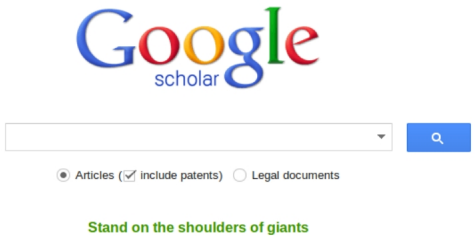


Figura 4.1: Google Scholar: Página de Inicio

Figura 4.2: Google Scholar: Búsqueda Avanzada

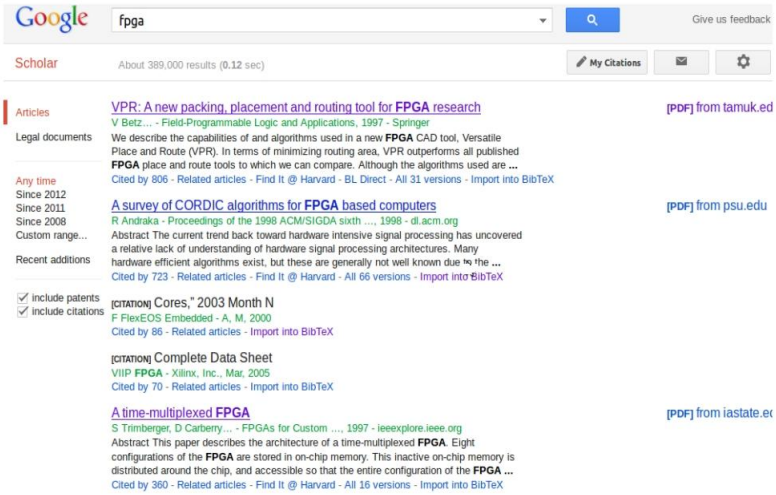
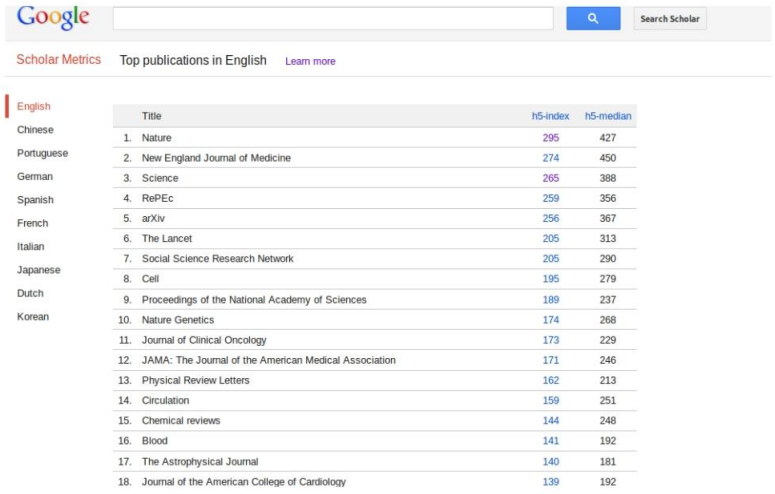


Figura 4.3: Google Scholar: Listado de Resultados.



	Title	h5-index	h5-median
1.	Nature	295	427
2.	New England Journal of Medicine	274	450
3.	Science	265	388
4.	RePEc	259	356
5.	arXiv	256	367
6.	The Lancet	205	313
7.	Social Science Research Network	205	290
8.	Cell	195	279
9.	Proceedings of the National Academy of Sciences	189	237
10.	Nature Genetics	174	268
11.	Journal of Clinical Oncology	173	229
12.	JAMA: The Journal of the American Medical Association	171	246
13.	Physical Review Letters	162	213
14.	Circulation	159	251
15.	Chemical reviews	144	248
16.	Blood	141	192
17.	The Astrophysical Journal	140	181
18.	Journal of the American College of Cardiology	139	192

Figura 4.4: Google Scholar: Directorio Temático.

4.1.2. Microsoft Academic Search

Microsoft Academic Search es un servicio gratuito de Microsoft para realizar búsquedas de documentos de corte académico en la *web*, empezando sus operaciones en 2009. Microsoft va un paso más allá y no sólo mantiene un índice con la información relativa a las publicaciones académicas, sino que además hace énfasis en indexar la información personal correspondiente a los autores de dichas publicaciones, información relativa a los centros de investigación dónde se realizaron los trabajos, las tendencias en las diferentes áreas de investigación en dichos centros a través del tiempo, y la relación que los autores de distintas publicaciones tienen entre sí (Si son colegas, investigan juntos, publicaron en la misma editorial, asisten a las mismas conferencias, son tutor y alumno, etc)[22].

Entre las características más resaltan tes de Microsoft Academic Search destacan:

- Permite filtrar la búsqueda por disciplina de investigación. Existen 15 disciplinas seleccionables, matemáticas, computación, medicina, química, física, etc. Dentro de cada disciplina existen subdominios, por ejemplo ciencias de la computación agrupa los subdominios gráficos, inteligencia artificial, recuperación de información, entre otros. Hay hasta 200 subdominios donde elegir.

- Presenta un directorio temático donde se pueden realizar búsquedas específicas según autor, publicaciones, conferencias, revistas científicas, palabras claves, organización y subdominios dada una disciplina, en lugar de retornar toda esta información mezclada en los resultados luego de ingresar una consulta. Las entradas del directorio se encuentran ordenadas por nivel de relevancia. Ver figura 4.7.

- La búsqueda avanzada permite buscar información por autor, conferencia, revista, organización, año y DOI[2]. Ver figura 4.6.

- Cada autor de una publicación que haya sido indexada dentro de Microsoft Academic Search posee un perfil creado automáticamente por el Motor de Búsquedas. Este perfil describe la trayectoria, impacto y frecuencia de publicación de las investigaciones realizadas por el autor. El perfil también describe la formación académica y centros de investigación a las que ha pertenecido el autor. Asimismo el autor puede editar el perfil para actualizar la información allí reflejada, la actualización se ejecuta luego que los cambios han sido comprobados directamente por Microsoft. Ver figura 4.10.

- Permite el uso de un lenguaje estructurado para realizar las consultas más potente que los disponibles en motores de búsquedas similares. En 4.1 se muestra el BNF[3] del lenguaje de consulta estructurado empleado por Microsoft Academic Search. Por ejemplo para buscar las publicaciones que contengan los términos *data mining* publicadas después del 2004, bastaría con ingresar la consulta *type: title:(data mining) year>=2004*.

[2]Digital Object Identifier, es un identificador unívoco que agrupa un conjunto de metadatos, entre ellos las posibles URL donde se encuentre la publicación. La idea es que aunque un objeto cambie su URL se mantenga constante su DOI actualizando el metadato URL con la nueva dirección.

[3]BNF por Backus-Naur Form, es una notación empleada para describir la gramática generativa de un lenguaje formal.

Algoritmo 4.1 BNF del lenguaje de consulta de Microsoft Academic Search.

```
<query> := <tokens>+
<token> := <normal query> | <field query>
<normal query> := (array of any non-white-space character)
<field query> := <key><oper><field query value>
<key> := 'author' | 'title' | 'conf' | 'jour' | 'year'
<oper> := '>=' | '<=' | ':' | '=' | '>' | '<'
<field query value> : <normal query> | '(' <normal query>+ ')'
```

- Ofrece la opción de listar las citas que referencian a una publicación particular o la citas que hace la publicación a otras publicaciones. El sistema en sí genera un grafo navegable donde se representan las conexiones entre los documentos a partir de citas. El grafo permite descubrimiento de información directamente relacionada sin tener que recurrir a una búsqueda basada en consultas y términos.
- Relaciona a los coautores de una misma publicación mediante un grafo, si dos autores publican muchos trabajos de investigación juntos los nodos que los representan aparecerán conectados con una arista más corta. Si se presiona la arista se listaran los trabajos en común que publicaron. Mediante este sistema también se puede visualizar el grado de separación entre dos autores cualesquiera que no han publicado juntos una publicación.
- Microsoft Academic Search no almacena los publicaciones indexadas, pero ofrece un enlace a un sitio externo donde puede ser descargada la publicación de forma parcial o completa si el autor así lo permite. La información retornada por el motor de búsqueda comprende datos desde el nombre del autor, resumen, citas, detalles bibliográficos, editorial, grafo de citas, hasta palabras claves relacionadas y artículos semejantes.
- Los artículos indexados provienen de fuentes especializadas tales como Cambridge University Press, IEEE, Association for Computing Machinery, MIT Press, Oxford University Press, CERN Document Server, entre otras instituciones.
- Se puede llevar un seguimiento de los resultados de las búsquedas suscribiéndose a canales RSS.
- Permite la búsqueda de actividad científica cerca de nuestra ubicación geográfica. Asimismo contempla un calendario donde se listan las fechas de eventos, simposios, conferencias y CFP (*Call for Papers*).
- Permite múltiples visualizaciones de los datos: grafo de citas por *papers* y por autor, grafo de genealogía (representa las relaciones entre colegas, o profesor-alumno como un árbol genealógico), mapa académico, tendencias de investigación dentro del subdominio y comparación de producción de conocimiento entre organizaciones.
- Microsoft Academic Search permite la publicación de las visualizaciones y los perfiles de los autores en sitios externos. También provee un API donde expone toda la información contenida en sus índices para ser reutilizada por terceros de forma autorizada sin fines de lucro. Algunas proyectos destacados construidos a partir de esta API es ScienceCard proyecto permite la creación y actualización automática de perfiles de investigadores que pueden ser utilizados en múltiples servicios o Eigenfactor un

proyecto creado para visualizar de forma innovadora los datos contenidos en los índices de Microsoft Academic Search para encontrar relaciones no triviales en ellos.

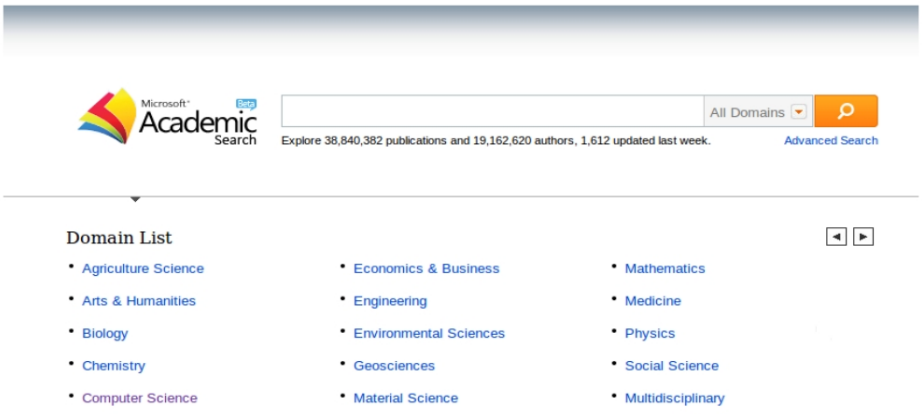


Figura 4.5: Microsoft Academic Search: Página de Inicio

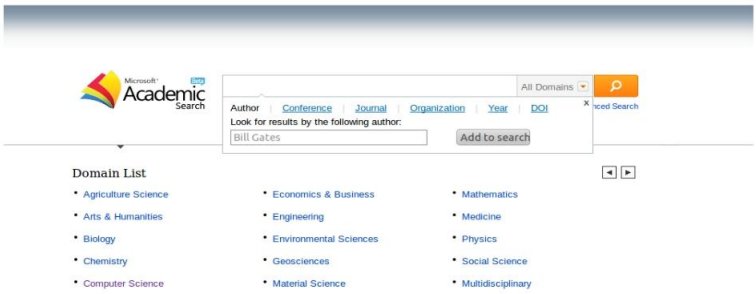


Figura 4.6: Microsoft Academic Search: Búsqueda Avanzada

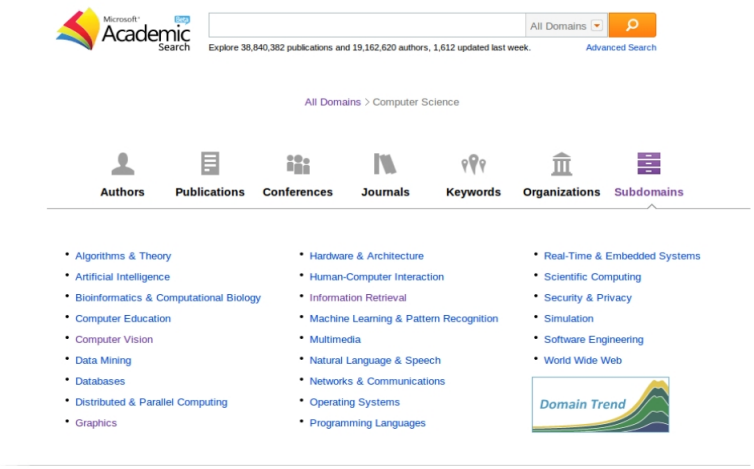


Figura 4.7: Microsoft Academic Search: Categorías

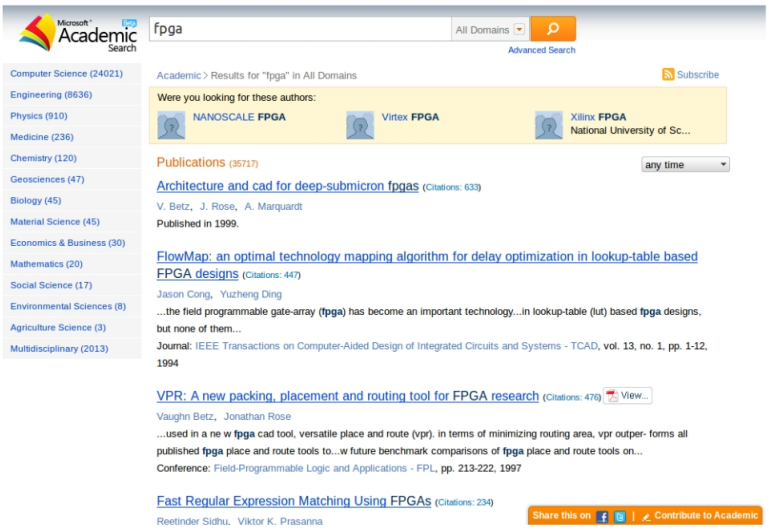


Figura 4.8: Microsoft Academic Search: Resultados

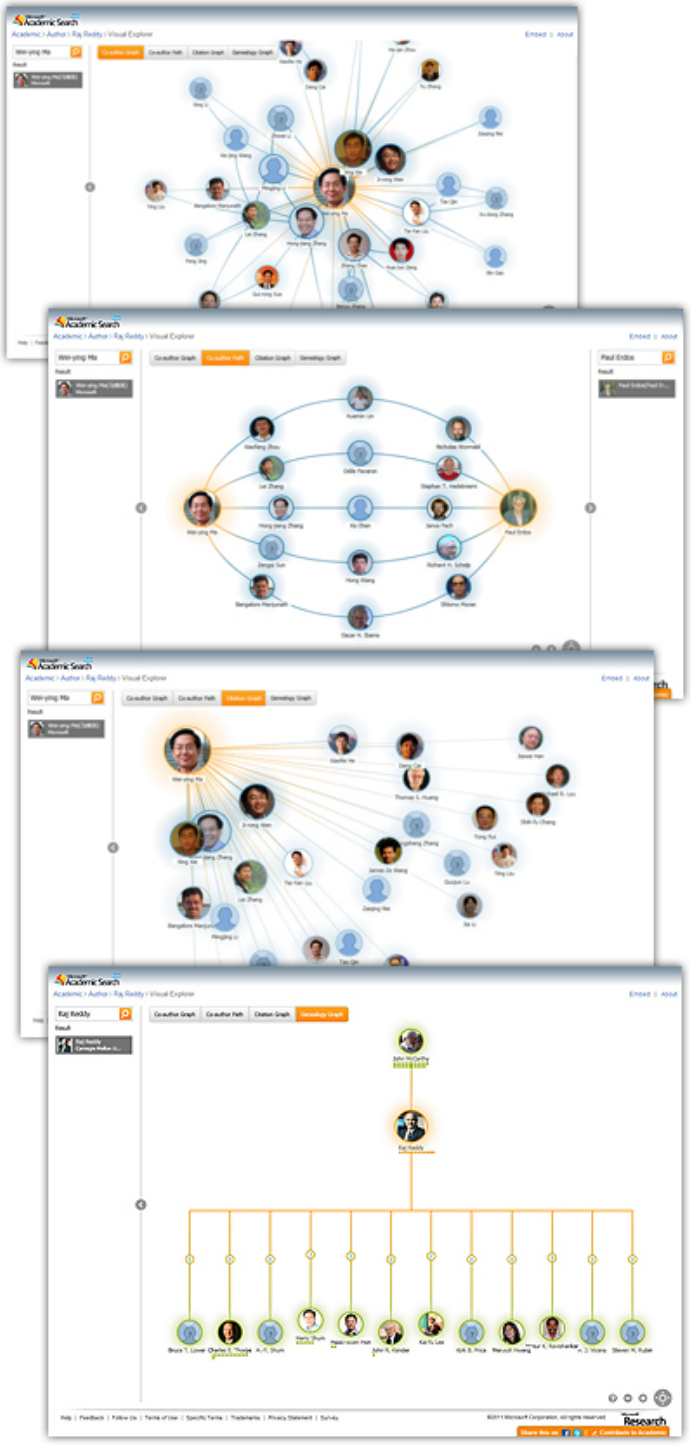


Figura 4.9: Microsoft Academic Search: Vistas

Figura 4.10: Microsoft Academic Search: Perfil de Autor

4.1.3. Scirus

Scirus es el Motor de Búsquedas académico accesible desde la *web* de la editorial de revistas holandesas Elsevier. Elsevier a su vez forma parte de del consorcio Anglo-Holandés Reed-Elsevier. Las plataformas Science Direct y Scopus son propiedad de Reed-Elsevier. Se estima que Science Direct posee alrededor del 25 % de todos los artículos e información bibliográfica relacionada con tópicos científicos, tecnológicos y médicos. Por su parte Scopus es una amplia base de datos de *abstracts*[4] y citas que es actualizada diariamente[23]. Scirus obtiene parte de sus resultados accediendo a estas dos plataformas. Scirus fue una iniciativa para ofrecer un servicio de búsqueda de información científica incluso antes que Google Scholar y Windows Academic Search ofrecieran sus servicios, empezando sus operaciones en 2001. Entre las características más resaltan tes de Scirus destacan:

- Indexa alrededor de 490 millones de páginas específicas con contenido científico verificado. Las arañas de Scirus hacen énfasis en indexar y mantener actualizada las páginas dentro de los dominios *.edu, .org, .ac.uk, .com*, y *.gov*.

[4]Resumen Documental, es una representación abreviada, objetiva y precisa del contenido de un documento.

- Además de información contenida en páginas *web* Scirus indexa información proveniente de bases de datos de revistas especializadas, reportes, repositorios de artículos *pre-print*[5] y *post-print*[6], patentes y revistas científicas.

- Organiza los resultados según su relevancia. Para estimar la relevancia asigna una ponderación de 50 % a la frecuencia y lugar donde aparecen los términos de la consulta en el documento y el otro 50 % es asignado por el análisis de las citas y autores que referencian el documento. Scirus explícitamente no hace uso de las meta etiquetas.

- Presenta un sistema de búsqueda avanzada altamente especializado donde se pueden aplicar filtros tipo de formato, tipo de información, fecha, fuentes específicas, disciplinas de investigación, ISSN[7], etc. Ver figura 4.12.

- Permite el uso de operadores lógicos y específicos al construir la consulta.

- Una descripción detallada referente a las características técnicas de la arquitectura interna de Scirus está descrita en detalle en el *white paper*[8] *How Scirus Works* disponible de forma gratuita en el sitio oficial [23].

Figura 4.11: Scirus: Página de Inicio

[5] Artículos que no han sido revisados aún por especialistas para su publicación oficial.
[6] Artículos *reviewed* o revisados por especialistas aptos para ser publicados.
[7] *International Standard Serial Number*, número de 8 dígitos utilizado para identificar unívocamente una publicación periódica impresa o electrónica.
[8] Documento oficial publicado por una organización con fines informativos.

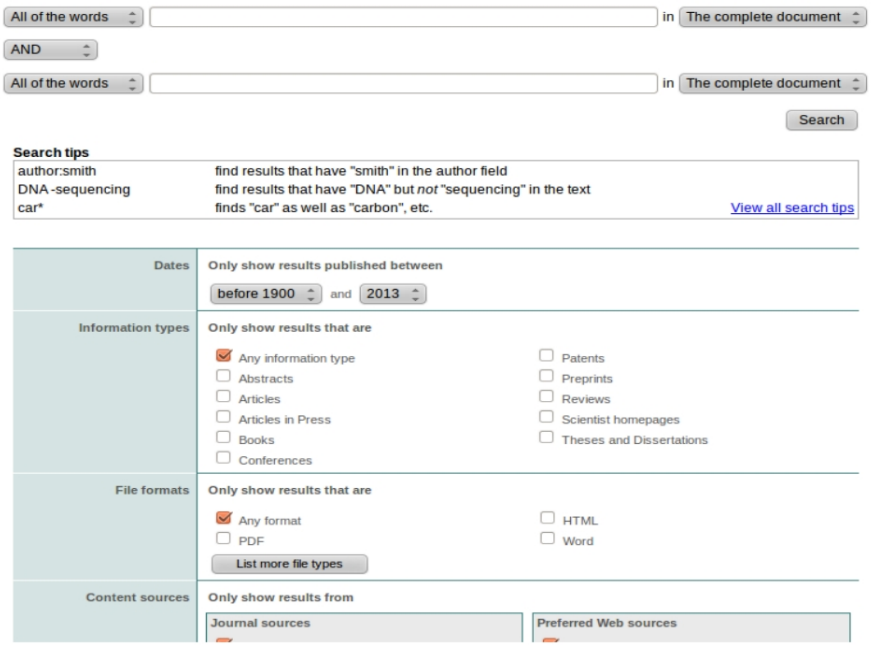


Figura 4.12: Scirus: Búsqueda Avanzada

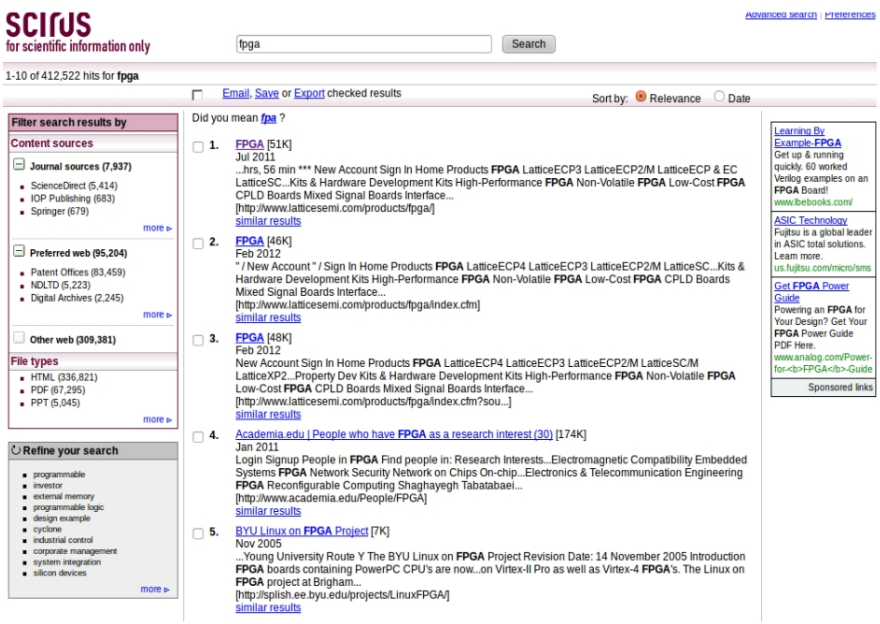


Figura 4.13: Scirus: Retorno de Resultados

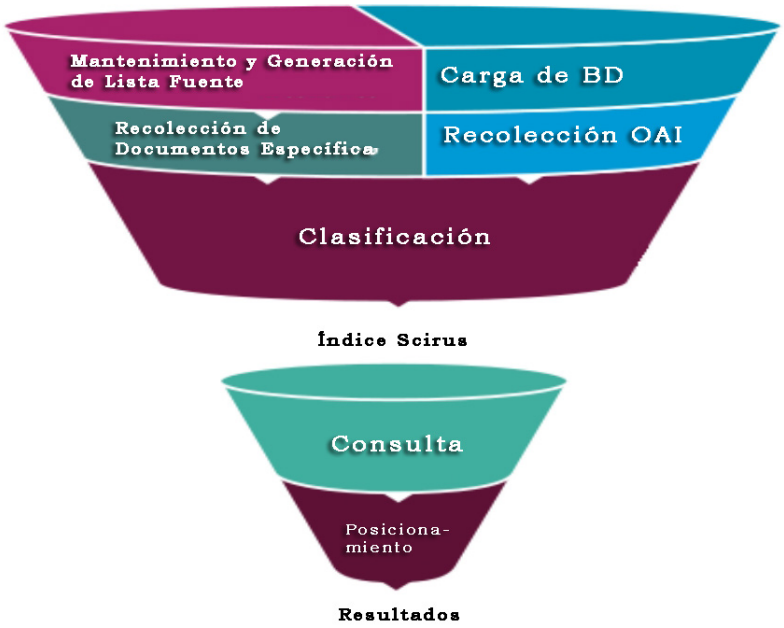


Figura 4.14: Scirus: Arquitectura interna

4.1.4. Biblioteca de Recursos Universia

Universia es una organización conformada por una red de universidades de habla hispana y portuguesa creada en 2000. Actualmente cuenta con la participación de 1.232 universidades de 23 países iberoamericanos. Universia provee un servicio en línea que permite la búsqueda de material académico en su portal *web*. Adicionalmente Universia provee un buscador específico de becas donde los estudiantes universitarios y profesionales pueden utilizar para conocer la oferta de becas de universidades pertenecientes a la red Universia y varias universidades adicionales. Las principales características del Motor de Búsqueda de la Biblioteca de Recursos Universia son[24]:

- En la página principal se muestran los últimos recursos añadidos a la biblioteca en orden cronológico además de mostrarse la institución académica a la que pertenecen y diferentes formas de realizar la búsqueda de documentos. Asimismo se muestra un listado con las colecciones de documentos destacados por la biblioteca y otra lista con los documentos más solicitados por los usuarios. Ver figura 4.15.

- La biblioteca de recursos ofrece tres formas de búsqueda. Búsqueda mediante el ingreso de una consulta y dos directorios. Un directorio es temático, es decir organiza el material por disciplina de estudio Física, Química, Historia, etc. Cada disciplina de estudio a su vez agrupa una serie de áreas de conocimiento específicas. Cada disciplina de estudio muestra el número de áreas de conocimiento que ésta agrupa. A su vez cada área de conocimiento muestra cuantos documentos del área se encuentran almacenados en el repositorio. Un segundo directorio organiza los documentos por colecciones, el Motor de Búsqueda entiende por colección todos los documentos provistos por una misma institución académica. Cuando se selecciona una colección en particular es mostrada una breve descripción de la colección junto a una lista con todos los documentos que agrupa organizados por categorías propias de la colección. Esto motivado a que las colecciones pueden alcanzar volúmenes de hasta tres millones de documentos.

- Los resultados de las búsquedas consisten en una ficha técnica (Figura 4.17) donde se puede consultar los metadatos que describen el documento buscado. Un enlace a un sitio externo es provisto para descargar el documento si el usuario se encuentra registrado y el autor del documento previamente autoriza la descarga del mismo. Adicionalmente también se despliegan dos listas relacionadas, una con documentos adicionales publicados por el mismo autor y otra con documentos similares pertenecientes a la misma colección. Asimismo está disponible la opción de compartir la información de la ficha técnica vía redes sociales.

- La opción de búsqueda avanzada (Figura 4.16) permite buscar por los campos título, autor y palabras claves, lo cual es un poco limitado comparado con la cantidad de metadatos disponibles para describir los documentos.

- Si el usuario se encuentra registrado se tiene la opción de poder valorar el documento en una escala de 1 a 5 estrellas, dejar un comentario y un asunto sobre el mismo además de poder consultar su historial de búsquedas.

- El protocolo OAI-PMH (*Open Archives Initiative-Protocol Metadata Harvesting*) permite el acceso y descarga de los metadatos de los recursos de una colección en formato Dublin Core, que a su vez son los que se muestran en la ficha técnica de cada recurso.

Figura 4.15: Biblioteca de Recursos Universia.

Figura 4.16: Biblioteca de Recursos Universia: Búsqueda Avanzada

Figura 4.17: Biblioteca de Recursos Universia: Ficha Técnica.

4.2. BUSCONEST 1

BUSCONEST 1 es un repositorio y un Motor de Búsquedas de trabajos de investigación desarrollado en la Facultad de Ciencias de la Universidad Central Venezuela como parte del T.E.G. *Elaboración de un prototipo de buscador de documentos académicos de la Facultad de Ciencias*[12]. BUSCONEST 1 está desarrollado como un sistema basado en *web* y presenta un diseño semejante a los más populares motores de búsqueda académicos modernos (véase sección 4.1). Asimismo BUSCONEST 1 se comunica con el sistema CONEST, sistema *web* empleado por los alumnos de la Facultad de Ciencias para realizar la gestión de petición de documentos, formalización de inscripción, revisión de estado académico, consulta de horarios, entre otras funciones. BUSCONEST 1 fue desarrollado como un sistema independiente de CONEST integrándose al mismo mediante Servicios Web, lo cual le confiere una gran independencia y flexibilidad a la hora de realizar modificaciones sobre el mismo.

4.2.1. Características Generales

- Trabaja sobre un repositorio de datos específico, se conoce detalladamente la naturaleza y características de los documentos que forman parte de la colección.
- Implementa el Modelo de Recuperación Booleano descrito en la sección 2.2.2 en la página 37, al incorporar un índice invertido a su lógica interna para recuperar los documentos.

- Orientado a *web* bajo paradigma cliente-servidor, los datos de entrada que procesa el sistema son obtenidos tanto desde clientes en navegadores *web* (consulta, carga y administración de documentos), como de otros servidores (obtención de metadatos del autor del documento y validación de sesión).

- Permite la realización de búsquedas generales y búsquedas avanzadas.

- El modelo Balance Lineal Simple es utilizado para determinar la relevancia de los documentos buscados.

- Provee un módulo de administración capaz de indexar documentos por lotes en segundo plano.

- La comunicación entre los diferentes componentes del sistema se lleva a cabo mediante Servicios Web, el sistema provee mecanismos para garantizar la confidencialidad e integridad de los datos transmitidos, así como también confirma la recepción exitosa de los mismos.

- Presenta un módulo de administración donde se puede monitorizar las tareas ejecutándose en segundo planos, así como también se puede administrar los documentos dentro del repositorio.

4.2.2. Metáfora del Sistema

El sistema BUSCONEST 1 involucra tres componentes bien diferenciadas, estos son:

- Módulo de carga de documentos

- Módulo de recuperación de documentos

- Módulo de posicionamiento de resultados

La figura 4.18 representa la metáfora que describe el sistema[12].

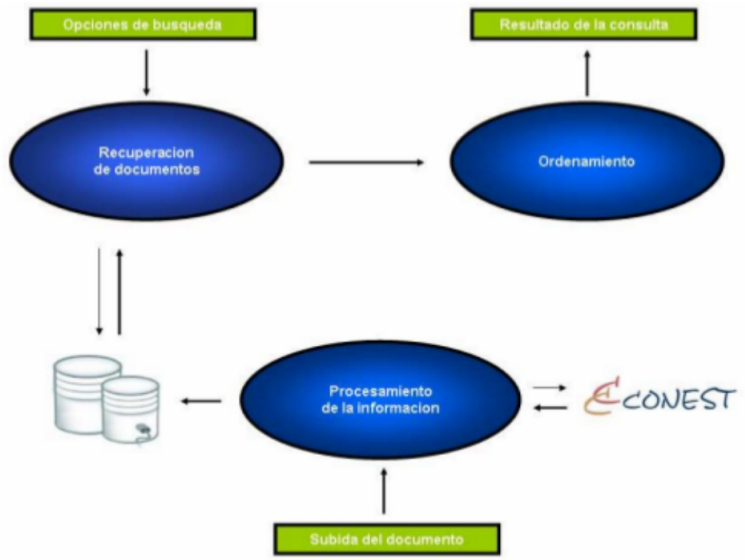


Figura 4.18: BUSCONEST 1: Metáfora del Sistema

Carga de documentos: Una vez aprobado el T.E.G. los estudiantes de la Facultad de Ciencias, inician sesión en el sistema CONEST. El sistema CONEST ofrece un enlace que permite la subida del documento. Cuando se accede a este enlace se transmiten, mediante el empleo del método POST y campos ocultos, datos cifrados que describen al autor del documento (calificación, méritos, datos personales y académicos, etc). La aplicación BUSCONEST 1 recibe los datos enviados por CONEST, los descifra y los valida. Una vez comprobado la validez del mismo el estudiante procede a cargar el documento, añadiendo información adicional (título resumen y palabras clave). Finalmente la aplicación procede al procesamiento de toda la información cargada para indexar el documento y su metainformación asociada en la base de datos del sistema y retorna el sistema CONEST un mensaje de confirmación avalando la carga exitosa del documento.

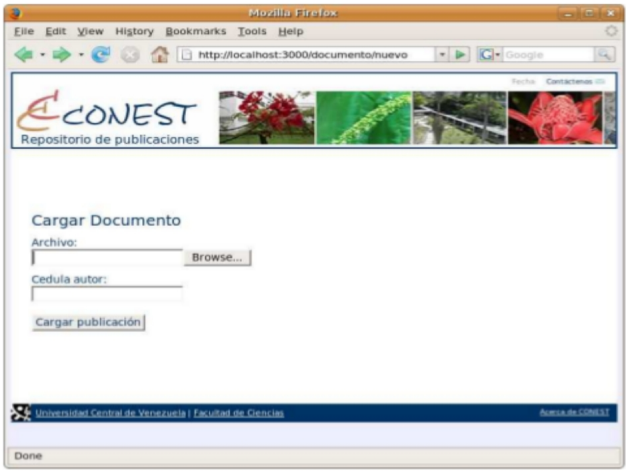
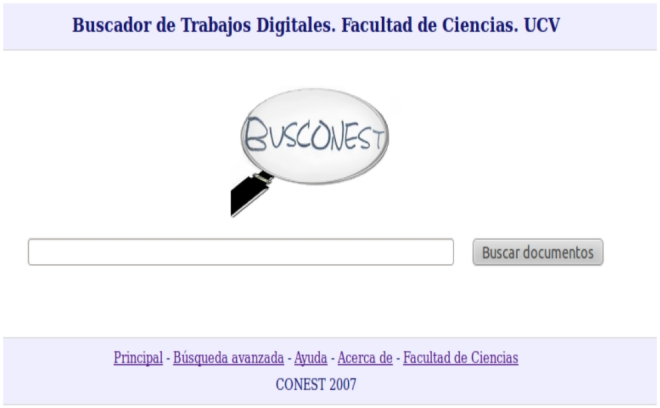


Figura 4.19: BUSCONEST 1: Carga de Documentos

Recuperación de documentos: La interfaz de inicio ofrece dos tipos de consulta, la básica y la avanzada. Asimismo el sistema permite el ingreso de palabras claves o frases exactas para realizar la búsqueda. La búsqueda avanzada permite especificar criterios como fecha de publicación o autores, entre otros criterios. La lista de resultados retorna el título del T.E.G. con información adicional tal como licenciatura, tipo de documento, fecha presentación, autores tutor, y extracto del resumen. Los resultados son mostrados al usuario de forma paginada en listas de a 15 documentos por página.

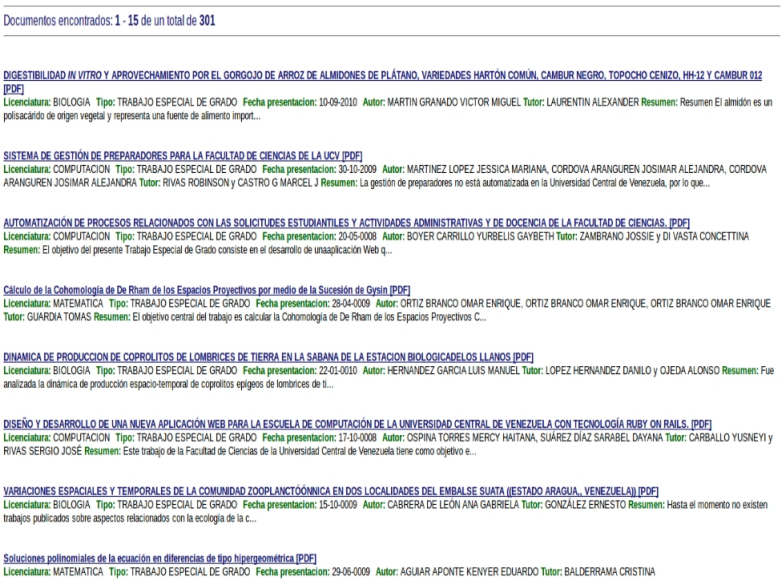


(a) Búsqueda sencilla

(b) Búsqueda avanzada

Figura 4.20: BUSCONEST 1: Recuperación de Documentos

Posicionamiento: Para realizar el posicionamiento BUSCONEST 1 emplea el modelo del Balance Lineal Simple, explicado en la sección 4.2.3, el cual consiste en asignar a cada documento un valor numérico que permita establecer una relación de orden parcial entre ellos (Explicado en la sección 2.1.1.7). Para calcular dicho valor se emplea un vector cuyas componentes ponderan la importancia de los criterios empleados para juzgar la relevancia del documento. Luego se calcula la distancia euclídea entre el vector que representa un documento en particular y el vector óptimo (vector cuyas componentes tienen todas el valor máximo establecido). Se repite el procedimiento para todos los documentos considerados relevantes según la consulta y finalmente se retornan el resultado ordenados de mayor a menor.


Documentos encontrados: 1 - 15 de un total de 301

DIGESTIBILIDAD *IN VITRO* Y APROVECHAMIENTO POR EL GORGOJO DE ARROZ DE ALMIDONES DE PLÁTANO, VARIEDADES HARTÓN COMÚN, CAMBUR NEGRO, TOPOCHO CENIZO, HH-12 Y CAMBUR 012 [PDF]
Licenciatura: BIOLOGIA Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 10-09-2010 Autor: MARTIN GRANADO VICTOR MIGUEL Tutor: LAURENTIN ALEXANDER Resumen: Resumen El almidón es un polisacárido de origen vegetal y representa una fuente de alimento import...

SISTEMA DE GESTIÓN DE PREPARADORES PARA LA FACULTAD DE CIENCIAS DE LA UCV [PDF]
Licenciatura: COMPUTACION Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 30-10-2009 Autor: MARTINEZ LOPEZ JESSICA MARIANA, CORDOVA ARANGUREN JOSIMAR ALEJANDRA, CORDOVA ARANGUREN JOSIMAR ALEJANDRA Tutor: RIVAS ROBINSON y CASTRO G MARCEL J Resumen: La gestión de preparadores no está automatizada en la Universidad Central de Venezuela, por lo que...

AUTOMATIZACIÓN DE PROCESOS RELACIONADOS CON LAS SOLICITUDES ESTUDIANTILES Y ACTIVIDADES ADMINISTRATIVAS Y DE DOCENCIA DE LA FACULTAD DE CIENCIAS. [PDF]
Licenciatura: COMPUTACION Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 20-05-0008 Autor: BOYER CARRILLO YURBELIS GAYBETH Tutor: ZAMBRANO JOSSIE y DI VASTA CONCETTINA
Resumen: El objetivo del presente Trabajo Especial de Grado consiste en el desarrollo de unaaplicación Web q...

Cálculo de la Cohomología de De Rham de los Espacios Proyectivos por medio de la Sucesión de Gysin [PDF]
Licenciatura: MATEMATICA Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 28-04-0009 Autor: ORTIZ BRANCO OMAR ENRIQUE, ORTIZ BRANCO OMAR ENRIQUE, ORTIZ BRANCO OMAR ENRIQUE
Tutor: GUARDIA TOMAS Resumen: El objetivo central del trabajo es calcular la Cohomología de De Rham de los Espacios Proyectivos C...

DINAMICA DE PRODUCCION DE COPROLITOS DE LOMBRICES DE TIERRA EN LA SABANA DE LA ESTACION BIOLOGICADELOS LLANOS [PDF]
Licenciatura: BIOLOGIA Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 22-01-0010 Autor: HERNANDEZ GARCIA LUIS MANUEL Tutor: LOPEZ HERNANDEZ DANILO y OJEDA ALONSO Resumen: Fue analizada la dinámica de producción espacio-temporal de coprolitos epígeos de lombrices de ti...

DISEÑO Y DESARROLLO DE UNA NUEVA APLICACIÓN WEB PARA LA ESCUELA DE COMPUTACIÓN DE LA UNIVERSIDAD CENTRAL DE VENEZUELA CON TECNOLOGÍA RUBY ON RAILS. [PDF]
Licenciatura: COMPUTACION Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 17-10-0008 Autor: OSPINA TORRES MERCY HAITANA, SUÁREZ DÍAZ SARABEL DAYANA Tutor: CARBALLO YUSNEYI y RIVAS SERGIO JOSÉ Resumen: Este trabajo de la Facultad de Ciencias de la Universidad Central de Venezuela tiene como objetivo e...

VARIACIONES ESPACIALES Y TEMPORALES DE LA COMUNIDAD ZOOPLANCTÓÓNICA EN DOS LOCALIDADES DEL EMBALSE SUATA ((ESTADO ARAGUA,, VENEZUELA)) [PDF]
Licenciatura: BIOLOGIA Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 15-10-0009 Autor: CABRERA DE LEÓN ANA GABRIELA Tutor: GONZÁLEZ ERNESTO Resumen: Hasta el momento no existen trabajos publicados sobre aspectos relacionados con la ecología de la c...

Soluciones polinomiales de la ecuación en diferencias de tipo hipergeométrica [PDF]
Licenciatura: MATEMATICA Tipo: TRABAJO ESPECIAL DE GRADO Fecha presentacion: 29-06-0009 Autor: AGUIAR APONTE KENYER EDUARDO Tutor: BALDERRAMA CRISTINA

Figura 4.21: BUSCONEST 1: Resultados

4.2.3. Balance Lineal Simple

El criterio del Balance Lineal Simple es el modelo utilizado para implementar el posicionamiento de BUSCONEST 1[20], la idea general es asignar un valor numérico a cada documento que permita cuantificar su grado de relevancia. El modelo del Balance Lineal Simple define un vector de criterios $\vec{v} = (cr_1, cr_2, \ldots, cr_{n-1}, cr_n)$, donde cr denota un criterio en particular comprendido entre los valores $[cr_{min}, cr_{max}]$. Los criterios son los valores numéricos necesarios empleados para calcular la ponderación de un documento. Estos criterios pueden emplear diferentes rangos de valores, por tanto para ponderar su importancia se deben normalizar los mismos estableciendo una cota mínima y una cota máxima en común. La normalización se calcula con la expresión 4.1.

$$crn = cota_{max} + \frac{(cota_{max} - cota_{min})\,(cr - cr_{min})}{cr_{max} - cr_{min}} \tag{4.1}$$

Si asumimos que el valor mínimo que puede tener cualquier criterio es cero y establecemos el rango de valores $[0, 1]$ para normalizar todos los criterios, la expresión 4.1 se simplifica a:

$$crn = \frac{cr}{cr_{max}}$$

Lo que resulta en el vector normalizado $\vec{v}_n = (crn_1, crn_2, \ldots, crn_{n-1}, crn_n)$. Luego podemos establecer el vector óptimo $\vec{O} = (1, 1, ..., 1, 1)$ para el rango de valores $[0, 1]$ y calculamos la distancia euclídea entre $\vec{O}$ y $\vec{v}_n$.

$$R = \sqrt{(1 - crn_1)^2 + (1 - crn_2)^2 + \ldots + (1 - crn_{n-1})^2 + (1 - crn_n)^2}$$

Sin embargo ciertos criterios podrían tener más importancia que otros. Para representar este hecho se establece un vector de importancia $\vec{i} = (imp_1, imp_2, \ldots, imp_{n-1}, imp_n)$ donde imp denota el valor que pondera la importancia de un criterio y R denota el grado de relevancia del documento. Finalmente, la expresión resulta en la expresión .

$$R = \sqrt{imp_1 (1 - crn_1)^2 + imp_2 (1 - crn_2)^2 + \ldots + imp_{n-1} (1 - crn_{n-1})^2 + imp_n (1 - crn_n)^2}$$

O resumidamente,

$$R = \sqrt{\sum_{i=1}^{n} imp_i (1 - crn_i)^2}$$

4.2.4. Tecnologías Empleadas

Para la implementación del sistema BUSCONEST 1 las siguiente tecnologías fueron empleadas:

- Lenguaje de programación Ruby, *framework* de desarrollo *web* Ruby on Rails.
- Servidor de Aplicación Mongrel.
- Sistema Manejador Base de Datos MySQL.
- *Plugins*: Google Calendar Date Select y Tiny MCE (Para enriquecer aspectos de usabilidad) y BackgrounDRb con Mongrel Upload Progress para soportar procesado de documentos por lotes con altas cargas de trabajo.
- Utlidades dependientes del sistema operativo: transformadores de formatos: pdftotext, htmltotext, y antiword.
- Action Web Service (Servicio Web integrado en Ruby on Rails). Action Web Service establece un API de comunicación entre CONEST (Estudiantes y Administradores) y BUSCONEST 1. Action Web Service esta diseñado siguiendo el protocolo SOAP (*Simple Object Access Protocol*).

4.2.5. Estado Actual

A la fecha (Semestre I-2013) el repositorio de BUSCONEST 1 cuenta con un total de 1004 documentos digitales en formato PDF. Todos ellos T.E.G. pertenecientes a estudiantes de las cinco escuelas de la Facultad de Ciencias (Computación, Biología, Química, Matemáticas y Física) que han egresado después de la puesta en producción del sistema, hecho ocurrido durante el semestre I-2008.

BUSCONEST 1 incorpora semestralmente los T.E.G. generados por los estudiantes que aprobaron su presentación de T.E.G. La versión digital del T.E.G. es cargada por los estudiantes próximos a graduarse desde el sistema CONEST para luego ser almacenada en el repositorio de BUSCONEST 1. Posteriormente, el administrador del sistema procede a ejecutar su indexado una vez concluido el período lectivo.

Aquí concluye la **Parte I - Marco Conceptual**. Seguidamente se presenta la **Parte II - Marco Aplicativo** en la cual se exponen los capítulos **Método de Desarrollo de *Software***, **Desarrollo del Sistema BUSCONEST 2**, **Conclusiones** y **Recomendaciones** respectivamente.

Parte II

Marco Aplicativo

Capítulo 5

Método de Desarrollo de *Software*

El Método de Desarrollo de *Software* seleccionado para la construcción del Motor de Búsquedas BUSCONEST 2 es un método de desarrollo ágil que toma características del método de desarrollo SCRUM. En el presente capítulo se describen los roles, actividades (Reuniones y procesos), y artefactos propuestos por SCRUM. y luego se presenta el método de desarrollo empleado durante el desarrollo del presente T.E.G.

5.1. SCRUM

SCRUM es un Método de Desarrollo de *Software* Ágil creado por Jeff Sutherland y Ken Schwaber en 1995[13]. SCRUM está centrado en la auto-organización de un equipo de trabajo bajo plazos de entregas quincenales o mensuales llamados *Sprints*. SCRUM a pesar de ser ampliamente utilizado para desarrollar *software* también es aplicable para el desarrollo de otros productos comerciales que requieren plazos de entregas breves.

En líneas generales SCRUM se puede conceptualizar en cuatro distintas partes bien diferenciadas. Ellas son:

- Roles
- Reuniones
- Artefactos
- Procesos *Sprints*

Básicamente en un equipo de desarrollo SCRUM sus miembros asumen roles, que determinan sus funciones, derechos y responsabilidades. Los miembros a su vez asisten a reuniones constantes donde se generan artefactos para planificar una iteración quincenal o mensual (*Sprint*), cuyo principal resultado es un prototipo de *software* funcional. Las iteraciones se repiten sucesivamente hasta lograr desarrollar un producto con las características deseadas por el cliente.

5.1.1. Roles

Un equipo organizado bajo el Método de Desarrollo de *Software* SCRUM está formado típicamente por cinco o nueve personas. Sin embargo SCRUM puede escalar a varios cientos de personas fácilmente, o puede ser usado por equipos individuales. En el equipo SCRUM no se asignan los cargos típicos contemplados tradicionalmente por la ingeniería de *software*, tales como programador, diseñador, arquitecto de *software*, *tester,* etc. En su lugar el equipo SCRUM trabaja colectivamente para ejecutar el volumen de trabajo que se han comprometido voluntariamente a completar durante una iteración. Este esquema de trabajo se basa en el compromiso que adquiere un miembro con respecto a los demás miembros. La idea es lograr que los miembros del equipo se sientan todos involucrados y comprometidos a trabajar por sus compañeros inmediatos, más que por ser presionados por un cliente o supervisor lejano que desconoce la dinámica interna del grupo de desarrollo. La idea es crear en el equipo la sensación de que *todos sus miembros se encuentran juntos en el proyecto,* en lugar de que cada individuo trabaje de forma aislada rindiendo cuentas a una autoridad superior. Los roles definidos por SCRUM son:

El Dueño del Producto: Es la persona que representa al cliente, bien puede ser un profesional en mercadeo, un gerente de ventas o productos, un usuario experto, etc. Es importante que el Dueño del Producto tenga una idea clara de las características del producto que desea que se desarrollen, tomando en cuenta las tendencias del mercado, la competencia y las tendencias futuras. El Dueño del Producto es el encargado de llenar una lista general con las características deseadas para el producto (Registro del Producto o *Product Backlog*) ordenadas según su prioridad. A pesar que el Dueño del Producto tiene el derecho a establecer las características del producto, es el equipo de trabajo quién selecciona el monto de trabajo a realizar para implementar las características solicitadas. El Dueño del Producto no puede por ejemplo, solicitar que se implemente un tercio del Registro del Producto quincenalmente si están planificadas tres iteraciones (un tercio por cada iteración quincenal). El equipo de trabajo es el que mejor conoce la carga de trabajo que pueden manejar por iteración. Del mismo modo si el equipo de trabajo adquiere el compromiso de implementar las características que ellos se asignaron voluntariamente, el Dueño del Producto tiene el compromiso de no cambiar los requerimientos una vez iniciada la iteración. Los cambios de requerimientos aunque constantes y promovidos por la metodología SCRUM, sólo se permitirán después de completada una iteración, no durante la ejecución de la misma. El Dueño del Producto tendrá el derecho de actualizar el registro del producto sólo antes del inicio de una nueva iteración.

El Maestro SCRUM: El Maestro SCRUM es el responsable de asegurarse que el equipo de trabajo sea lo más productivo posible. Para lograr este propósito el Maestro SCRUM, ayudará al equipo a seguir la metodología SCRUM realizando actividades como organizar las reuniones requeridas, asegurar el buen estado de los artefactos generados, servir de intermediario entre el Dueño del Producto y los miembros del equipo, entre otras actividades. El Maestro SCRUM puede ser un gerente de proyectos, pero también puede ser un miembro experimentado de algún equipo SCRUM anterior. Además el Maestro SCRUM se encargará de remover los factores que impidan que los miembros del equipo puedan realizar su trabajo adecuadamente. El Maestro SCRUM protege al equipo de trabajo de ser explotado por un Dueño del Producto excesivamente exigente. Pero también protege al equipo de sí mismo si en un momento determinado están rindiendo por debajo de sus capacidades. Por tanto el Maestro SCRUM no sólo debe conocer las capacidades de cada uno de los miembros del equipo sino también la capacidad general del grupo de trabajo. El rol del Maestro SCRUM también puede ser interpretado como un Dueño del Proceso, ya que el mismo ejerce autoridad sobre el *proceso* SCRUM y no sobre el *equipo de trabajo*. El Maestro SCRUM podría decidir por ejemplo cambiar el

plazo de las iteraciones de mensual a quincenal, pero no puede exigir qué trabajo deberá realizar un miembro del equipo durante la próxima iteración. Es sólo responsabilidad de los miembros del equipo comprometerse voluntariamente a realizar la carga de trabajo que consideren razonable dependiendo de sus propias aptitudes, habilidades y conocimientos.

Equipo de Trabajo: Son los principales actores dentro del equipo SCRUM, ya que son los encargados directos de desarrollar el producto. Los miembros del equipo suelen ser programadores, administradores de sistemas, diseñadores, *testers*, etc. Cada miembro del equipo se compromete con sus compañeros a asumir la carga de trabajo que el mismo considere capaz de realizar. Los miembros del equipo son los que mejor conocen sus propias habilidades, por tanto son ellos directamente los que se asignan las cargas de trabajo voluntariamente. Es fundamental que el equipo de trabajo conozca muy bien sus capacidades, o que el Maestro SCRUM los ayude a determinarlas. Un miembro del equipo tiene el derecho de implementar las características que considere necesarias y que sea capaz de implementar, pero es responsable ante sus compañeros de cumplir con su carga de trabajo durante la ejecución de la iteración.

Terceras Partes: Son agentes que aunque no intervienen directamente en el proceso de desarrollo del producto, se ven afectados por el éxito del desarrollo del mismo. Estos agentes pueden ser gerentes, auditores, representantes de otras organizaciones, etc. Básicamente la idea es que las terceras partes al no estar comprometidas en el desarrollo, ellas solamente están asociadas, no tienen derecho alguno sobre el proceso de desarrollo. Sin embargo las terceras partes pueden asistir como oyentes a las reuniones SCRUM planificadas, y pueden emitir sugerencias pero no pueden tomar decisiones. La finalidad es que las reuniones además de ser el medio mediante el cuál se planifican las actividades y se generan los artefactos SCRUM, sirva también como un mecanismo de difusión de información rápido y efectivo. La mejor forma de saber el estado acerca del estado de un proyecto SCRUM es asistiendo a las reuniones planificadas por el equipo de trabajo.

5.1.2. Reuniones

Reunión de Planificación *Sprint:* Es una reunión donde asisten el Dueño del Producto, el Maestro SCRUM, y el equipo de trabajo completo. Las terceras partes también pueden asistir, pero no es obligatorio. La Reunión de Planificación se realiza antes del inicio de cada *Sprint*. Durante la reunión de la planificación el Dueño del Producto describe las características generales que desea que el producto posea en orden de prioridad (Registro del Producto). Los miembros del equipo deben hacer la suficiente cantidad de preguntas como para que una lista detallada de tareas (Registro del *Sprint*) pueda ser elaborada a partir del Registro del Producto. No es requerido que el Dueño del Producto especifique exhaustivamente todas las características deseadas en el producto a partir de el primer *Sprint*, ya que en los *Sprint* siguientes se puede modificar el Registro del Producto . Una vez establecido el Registro del Producto los miembros del equipo de trabajo seleccionan aquellas características que se implementarán en el *Sprint* por comenzar. El equipo de trabajo también se debe poner de acuerdo con el Dueño del Producto para establecer una meta general para el *Sprint*. La meta general del *Sprint* debe ser breve y auto-descriptiva, por ejemplo: *Implementar un carrito de compras persistente entre sesiones con su respectivo CRUD de artículos*. Los artefactos generados por la Reunión de Planificación son: la meta general del *Sprint* y el Registro del *Sprint*. El Registro del *Sprint* estará compuesto a su vez por un conjunto de Historias de Usuario.

Reunión Diaria: La reunión diaria *Sprint* es llevada a cabo por todos los miembros del equipo, el Maestro SCRUM y el Dueño del Producto. También pueden asistir terceros, pero estos al no estar compro-

metidos sino solamente involucrados con el desarrollo usualmente sólo pueden asistir como oyentes, de esta forma también se disemina la información de forma rápida y poco costosa. Usualmente la reunión diaria es llevada a cabo en la mañana antes de cada jornada de trabajo y no debe durar más de 15 minutos. Su carácter es informal y sus asistentes deben ser breves y concisos. El Maestro SCRUM usualmente se limitará a preguntar a cada miembro del equipo tres preguntas: *¿Qué hiciste ayer? ¿Qué harás hoy?* y *¿Hay algún impedimento que no te permita avanzar?*. La idea de la reunión no es vigilar si cada miembro del equipo cumple con la planificación, sino que los miembros del equipo tengan una idea del volumen de trabajo que pueden realizar diariamente. Por ejemplo si alguien si compromete a realizar una tarea para el día de hoy, entonces los demás miembros del equipo sabrán que la día siguiente esa tarea debería estar completada. Es responsabilidad del Maestro SCRUM tratar de eliminar todos los impedimentos reportados por los miembros del equipo lo más rápido posible durante la jornada de trabajo. Si esto no es posible entonces el Maestro SCRUM es el responsable de consultar a otro miembro del equipo que tenga más dominio del problema, o en caso de que el problema persista buscar asesoría externa.

Reunión de Revisión *Sprint*: La reunión de Revisión del *Sprint*, es el resultado natural del mismo. Una vez concluido el *Sprint* los miembros del equipo preparan una presentación de sus resultados. Los resultados deben reflejar que la meta general del *Sprint* fue alcanzada. De no ser este el caso es responsabilidad del Maestro SCRUM analizar las razones por las cuales no se cumplió la meta al concluir el *Sprint*. Esta reunión es de carácter informal, no debe emplearse más de dos horas para su elaboración y no se deben utilizar láminas u otro soporte. Sólo está permitido presentar el prototipo, o la característica implementada de forma funcional y sin errores. La idea es que la elaboración de la presentación no consuma tiempo de trabajo valioso a los miembros del equipo. A la Reunión de Revisión pueden asistir terceras partes para que se pueda dar a conocer el progreso del desarrollo del producto.

Reunión Retrospectiva: La Reunión Retrospectiva es llevada a cabo por los miembros del equipo, el Dueño del Producto y el Maestro SCRUM. Esta reunión puede tener carácter privado si así se desea. La Reunión Retrospectiva es llevada a cabo usualmente inmediatamente después de concluida una Reunión de Revisión *Sprint*. Se recomienda que no se emplee más de una hora para su realización. Sin embargo su duración puede extenderse si así se cree necesario. Durante la Reunión Retrospectiva el equipo reflexiona acerca de los resultados obtenidos en el *Sprint* concluido con la finalidad de proponer mejores formas de mejorar su rendimiento y motivación. La dinámica de la reunión se puede llevar de varias maneras pero usualmente el Maestro SCRUM le pregunta a cada miembro del equipo *qué se debe empezar a hacer, qué se de debe dejar de hacer,* y *qué se debe continuar haciendo*. Es una buena práctica someter a votación las mejoras propuestas por todos los miembros del equipo de trabajo para determinar cuales se llevarán a cabo durante el próximo *Sprint*. Luego de concluido el *Sprint* se verificarán los resultados arrojados por las mejoras tomadas en la Reunión Retrospectiva anterior.

5.1.3. Artefactos

Registro del Producto El Registro del Producto es una lista de características ordenadas por prioridad. Cada entrada de la lista contiene un ítem con una descripción de las funcionalidades deseadas en el producto. El Registro del Producto se elabora mediante la realización de una tormenta de ideas entre el Dueño del Producto y el Equipo de Trabajo. Ambas partes escriben lo que desean del producto de forma flexible y natural. Esto permite que la primera iteración pueda ser iniciada sin tener que

invertir gran cantidad de tiempo y esfuerzo en documentación previa. Los ítems representados en las entradas de la lista pueden ser: características, errores, trabajo técnico (actualización de plataforma de trabajo, o de versión de bibliotecas, por ejemplo) y adquisición de conocimiento (aprender a manejar sistemas de control de versiones distribuidos, manejar determinado *framework* de desarrollo *web*, etc). La forma más usual de expresar cada ítem es mediante una Historia de Usuario. Una Historia de Usuario tiene la sintaxis: *Como <algún rol> puedo <algún objetivo> de modo que <alguna razón>*. Por ejemplo, *Como <cliente> puedo <revisar los productos en mi carrito de compras justo antes de realizar la compra> de modo que <pueda verificar los productos que he seleccionado>*. Las Historias de Usuario del Registro del Producto deben ser breves y descriptivas, en realidad es más importante la discusión que el Dueño del Producto y el Equipo de Trabajo hagan sobre las mismas, siendo la Historia de Usuario en sí un recordatorio de dicha discusión.

Meta *Sprint* Una Meta *Sprint* es una descripción breve que representa lo que el equipo de desarrollo desea lograr durante una iteración. La Meta *Sprint* es redactada de forma colaborativa entre el Dueño del Producto y el equipo de desarrollo. La Meta *Sprint* puede ser utilizada como una descripción resumida de las actividades que está realizando el equipo de desarrollo. Terceras partes pueden ser informadas rápidamente de las actividades del grupo de trabajo mediante la Meta *Sprint,* sin tener que describirse todos los detalles. El éxito de la iteración se evaluará contra la Meta *Sprint*, en lugar de contra cada uno de los ítems específicos seleccionados en el Registro del Producto.

Registro del *Sprint* El Registro del *Sprint* consiste en una lista de tareas identificadas por el el Equipo de Trabajo durante la planificación del *Sprint*. Durante la planificación del Sprint el equipo selecciona una cantidad de características a desarrollar del Registro del Producto, expresadas en la forma de Historias de Usuario, e identifica las tareas necesarias para completarlas. El registro del *Sprint* puede ser actualizado con una frecuencia diaria por los miembros del equipo a medida que se vayan desarrollando las características, pero se debe tratar de minimizar las veces que se actualiza el registro durante un día. Las tareas usualmente son mantenidas en una tabla, como la tabla mostrada en 5.1 dónde se listan y se estiman las horas requeridas para completar las tareas.

Tabla 5.1: Ejemplo de planificación de Iteración SCRUM

Historia de Usuario	**Tareas**	**Día 1**	**Día 2**	**Día 3**	**Día 4**	**Día 5**
Como miembro puedo ver perfiles de otros miembros de manera que pueda encontrar otros miembros	Codificar...	8	8	8	6	6
	Diseñar...	2	1	1	0	0
	Investigar...	4	4	2	0	0
	Revisar ...	2	1	1	1	0
	Automatizar...	3	1	0	0	0
	Estimar...	2	1	0	0	0
Como miembro puedo actualizar mi información de facturación	Actualizar...	1	0	0	0	0
	Codificar...	8	8	6	6	4
	Analizar...	2	2	2	1	0
	Diseñar...	2	1	1	0	0
	Probar...	1	1	1	1	1

Lista de Impedimentos La Lista de Impedimentos es la lista donde el Maestro SCRUM lleva un seguimiento de todos los bloqueos, impedimentos, inconvenientes, y decisiones pendientes que pudieran representar una amenaza al buen desarrollo del proyecto. El Maestro SCRUM deberá actualizar esta lista diariamente y es su responsabilidad tratar de identificar y resolver todos los problemas que surjan durante el desarrollo.

Incremento El incremento es el prototipo funcional o *demo* que se planteó desarrollar en la Meta *Sprint*. El mismo puede ser una nueva versión del producto o una funcionalidad específica lista para ser entregada, totalmente probada y documentada.

Retroalimentación Visual Se trata de todos los artefactos generados durante el proceso de desarrollo que son necesarios para implementar el producto. En esta categoría se agrupan los diagramas entidad-relación, los diagramas físicos de Bases de Datos, flujogramas, diagramas arquitectónicos, diagrama de objetos de dominio, de clases, Casos de Uso, etc. Sin embargo SCRUM define un nuevo artefacto llamado Gráfico de Progreso (*Burndown Chart*) el cual representa el progreso del equipo de desarrollando el producto. El Gráfico de Progreso representa inicialmente la suma de las horas requeridas para completar todas las tareas del *Sprint* actual, luego diariamente a medida que el equipo de desarrollo completa las tareas, el Maestro SCRUM actualiza el gráfico que va descendiendo progresivamente. El Maestro SCRUM puede estimar la tasa de horas de trabajo completadas por día y calcular de antemano si el equipo a ese ritmo completará la Meta *Sprint* dentro del intervalo acordado (quincenal o mensual). El Maestro SCRUM puede apoyar al equipo sugiriendo que se retiren o añadan tareas a las cargas de trabajo, previo acuerdo con el Dueño del Producto, para lograr el mejor desempeño del equipo de desarrollo. El Gráfico de Progreso también puede emplearse para representar información más general, al representar la cantidad de Historias de Usuario completadas durante cada Sprint para estimar la fecha en que el equipo de desarrollo culminará de desarrollar el producto definitivo. Un Gráfico de Progreso general es mostrado en la figura 5.1.

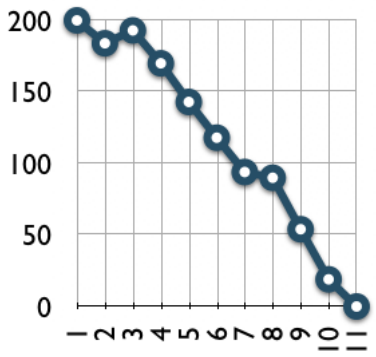


Figura 5.1: Gráfico de Progreso *Sprints*

5.1.4. Procesos *Sprint*

Los procesos *Sprint* consisten en la realización de un conjunto de actividades (reuniones, tareas, resolución de impedimentos, etc) llevadas a cabo por agentes que asumen roles. Dichas actividades generan artefactos

necesarios para desarrollar el incremento del *Sprint*. La figura 5.2 identifica los elementos de un proceso *Sprint* y representa como se relacionan dichos elementos durante el ciclo de desarrollo.

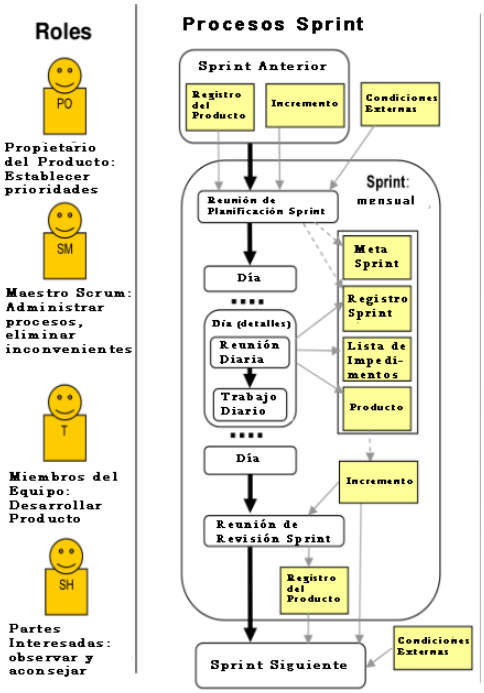


Figura 5.2: Proceso *Sprint*

5.2. Modificación del Método de Desarrollo de *Software*

Debido a que el Método de Desarrollo de *Software* SCRUM no especifica con precisión que artefactos deben ser generados en cada iteración, y a que la naturaleza del desarrollo está enmarcada dentro de un contexto completamente académico se hace necesario definir un Método de Desarrollo de *Software* Ágil que permita establecer la planificación de las actividades y artefactos necesarios para lograr la construcción del Motor de Búsqueda BUSCONEST 2. Para ello se toman en cuenta aspectos considerados por SCRUM, pero también se simplifican otros aspectos y se incluyen nuevos. Los principales cambios en el método empleado se describen a continuación.

Roles Los roles del Dueño del Producto y el maestro SCRUM para el desarrollo del presente T.E.G. son ejercidos por el tutor. El equipo de trabajo está compuesto por un sólo miembro, a saber el estudiante desarrollando el T.E.G. No se consideran terceras partes involucradas.

Artefactos El principal artefacto que se tomará de SCRUM es el Registro del Producto ya que el mismo plantea los requerimientos funcionales a desarrollar necesarios para completar los objetivos específicos planteados por el presente T.E.G (ver capítulo 1.2 en la página 20). También se fijará una

meta *Sprint* por cada iteración la cual consistirá en una descripción breve de la meta a lograr en cada iteración. Además, cada iteración generará un documento de control a modo de resumen cuyo contenido consistirá en una tabla con el formato presentado en la tabla 5.2. En la misma se listan la tareas propuestas para completar la iteración. Adicionalmente por cada iteración se describirán cuatro apartados en: Planificación, Diseño, Implementación, y Pruebas, en los casos que aplique. En estos apartados se explicará como se logró implementar cada una de las tareas pertenecientes a la iteración respectiva. Cada tarea posiblemente generará artefactos adicionales descriptivos del incremento desarrollado. Los artefactos descriptivos del sistema (Tablas, diagramas, capturas de pantalla, etc) generados durante el desarrollo son incluidos dentro de la retroalimentación visual correspondiente a la iteración que correspondan.

Reuniones De las reuniones planteadas por SCRUM sólo se llevaran a cabo la Reunión de Planificación *Sprint* y la Reunión de Revisión *Sprint* (en la misma se puede realizar la evaluación retrospectiva). Debido a que el equipo de trabajo está compuesto solamente por un integrante se prescinde las reuniones diarias. Además, la Reunión de Planificación *Sprint* de la siguiente iteración, puede realizarse inmediatamente después de concluida la Reunión de Revisión *Sprint* de la iteración anterior.

Tabla 5.2: Tabla de Control de Iteración

N#	**Fecha**	**Tareas**	**Tipo**
1	DD/MM/AAAA	Descripción de la tarea 1	Módulo desarrollado
2	DD/MM/AAAA	Descripción de la tarea 2	Ejemplo: Módulo de autenticación
⋮	⋮	⋮	⋮
n	DD/MM/AAAA	Descripción de la tarea n	Ejemplo:Módulo de administración

Aquí concluye el capítulo **Método de Desarrollo de *Software***. El próximo capítulo **Desarrollo del Sistema BUSCONEST 2** describe detalladamente el proceso de análisis, planificación y desarrollo del Motor de Búsquedas BUSCONEST 2 organizado por iteraciones.

Capítulo 6

Desarrollo del Sistema BUSCONEST 2

A continuación se describe las iteraciones necesarias para lograr el desarrollo de BUSCONEST 2. En cada una de ellas se define una meta *Sprint* (ver capítulo 5.1.3 en la página 99) y una tabla de control con las tareas necesarias para cumplir dicha meta. Luego de cada tabla de control se muestran los apartados planificación, diseño, implementación y pruebas que describen las actividades llevadas a cabo para completar las tareas que conforman cada iteración.

6.1. Iteración 0

Meta *Sprint:* Definir los requerimientos funcionales del sistema BUSCONEST 2 y establecer las iteraciones necesarias para su implementación.

En esta iteración del proyecto se definieron las características generales requeridas para el desarrollo del Motor de Búsquedas propuesto. Las mismas fueron registradas en el Registro del Producto presentado a continuación.

- Como Usuario puedo ingresar términos de modo que pueda recuperar documentos relacionados con los términos ingresados.
- Como Usuario puedo ingresar datos descriptivos de documentos de modo que pueda recuperar los documentos relacionados con esos datos.
- Como Usuario puedo seleccionar la forma que los resultados son devueltos de modo que pueda visualizar los documentos más relevantes primero.
- Como Usuario puedo navegar por categorías de modo que pueda recuperar documentos asociados con dichas categorías.
- Como Usuario puedo visualizar la organización del repositorio de modo que pueda tener una visión general de los documentos almacenados.

- Como Administrador puedo acceder a un *Dashboard*[1] de modo que pueda visualizar el estado general del repositorio.
- Como Administrador puedo gestionar los documentos de modo que pueda incorporar nuevos documentos o modificar los ya existentes.
- Como Administrador puedo indexar todos los documentos generados en un semestre de modo que puedan ser recuperables por los usuarios.
- Como Sistema Externo CONEST puedo cargar documentos de modo que el usuario Administrador pueda indexar dichos documentos posteriormente.
- Como un Sistema Externo puedo indexar la información descriptiva de los documentos de modo que puedan ser visibles en Motores de Búsqueda externos.

Tomando en consideración los requerimientos especificados en el Registro del Producto se propone un plan de iteraciones para completar los requerimientos especificados en el Registro del Producto. Las mismas son representadas en la figura 6.1.

Figura 6.1: BUSCONEST 2: Iteraciones

Se proponen 5 iteraciones sucesivas de plazos de duración aproximada de dos a tres semanas. En cada iteración se desarrollaran las siguientes actividades.

1. Diseño e Implementación de la Base de Datos

[1] Un *Dashboard* es una interfaz de usuario donde se puede visualizar el estado general de un sistema.

2. Implementación de Interfaz de Administración (Admin Panel)

3. Indexado de documentos

4. Implementación Interfaz Pública

5. Comunicación con sistema CONEST

6. Migración de datos del sistema BUSCONEST 1 al sistema BUSCONEST 2.

La figura 6.2 representa la arquitectura general del sistema BUSCONEST 2. En el mismo se representan los distintos actores que interactúan con el sistema.

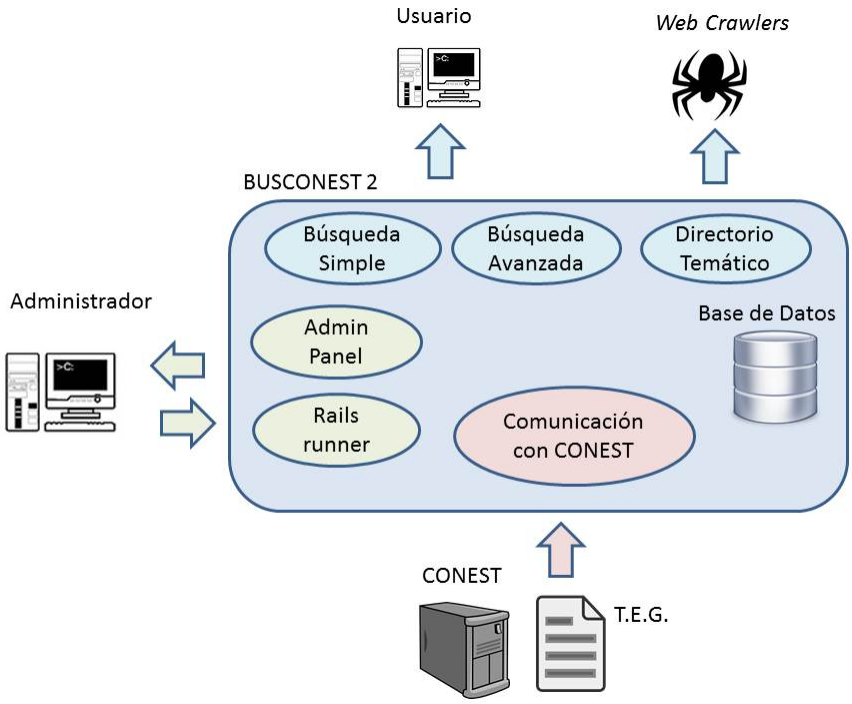


Figura 6.2: BUSCONEST 2: Arquitectura

6.2. Iteración 1

Meta *Sprint:* Diseñar el Modelo Datos del Sistema BUSCONEST 2.

La iteración 1 consistió en el diseño e implementación de un modelo de datos relacional. La tabla 6.1 lista las tareas necesarias para llevar a cabo la actividad.

Tabla 6.1: Tabla de Control - Iteración 1

N#	Fecha	Tareas	Tipo
1	04/02/2013	Definir modelo de datos de la aplicación	Modelo de Datos
2	11/02/2013	Implementar el modelo físico de la Base de Datos	Modelo de Datos

Planificación: El sistema BUSCONEST 1, puesto en producción en el año 2008, presenta una estructura consistente en cuatro tablas. El diagrama entidad-relación correspondiente es mostrado en la figura 6.3 [12].

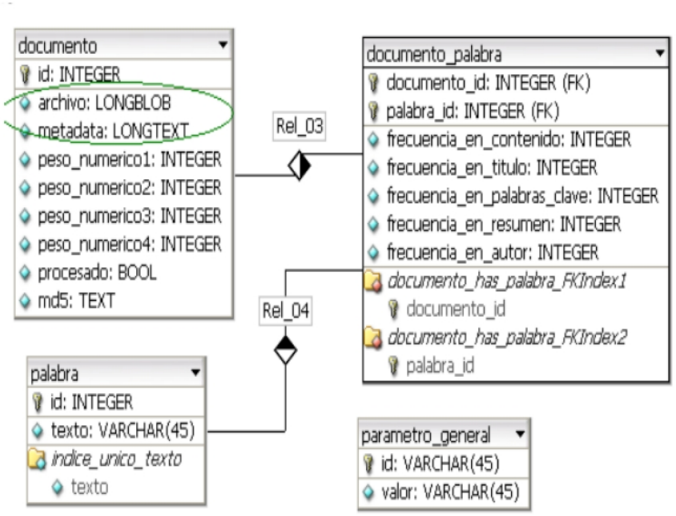


Figura 6.3: BUSCONEST 1:Diagrama Entidad-Relación

El esquema presenta la característica de almacenar todo el documento en formato binario en el campo *archivo* de la tabla *documento*. Del mismo modo la información correspondiente a la descripción del documento es almacenada en el campo *metadata* en un *string* en formato XML. La información correspondiente a la relación de un término con un determinado documento se almacena en la tabla *documento_palabra*. Se propone la creación de un modelo de datos que permita capturar la información asociada a los documentos directamente en la Base de Datos, sin estar representada ésta en un archivo XML. Con dicho cambio se logra un manejo más flexible de los datos ya que la aplicación puede realizar consultas SQL para recuperar la información almacenada.

Diseño: El nuevo esquema de Base de Datos propuesto para el Sistema BUSCONEST 2 presenta una estructura que captura mayor información descriptiva de los documentos indexados, tal como cantidad de descargas de un documento. Asimismo, el nuevo esquema separa la carga generada por almacenar en una sola tabla todo el índice invertido. Para ello se divide el índice en cuatro subíndices. Un índice almacena las asociaciones entre los términos y los documentos si el término ocurre en el título (*tindex*). Análogamente, se crea otro índice para almacenar las ocurrencias de un término en el resumen (*rindex*), otro índice si el término ocurre en el contenido (*cindex*) y otro índice si el término ocurre en las palabras claves (*clindex*).

La información descriptiva del estudiante ya no se encuentra almacenada en un *string* en formato XML, sino que dicha información ahora se almacena en forma de registros en las tablas correspondientes, permitiendo consultar la información descriptiva del documentos y las personas directamente mediante consultas SQL.

Del mismo modo se define una tabla persona de forma separada a los roles que una persona puede asumir con un documento (autor, tutor o jurado). Esto permite un manejo flexible de las personas en el sistema al separar los datos personales con respecto a la relación que tienen estos con los documentos. El diagrama Entidad-Relación correspondiente se visualiza en la figura 6.4.

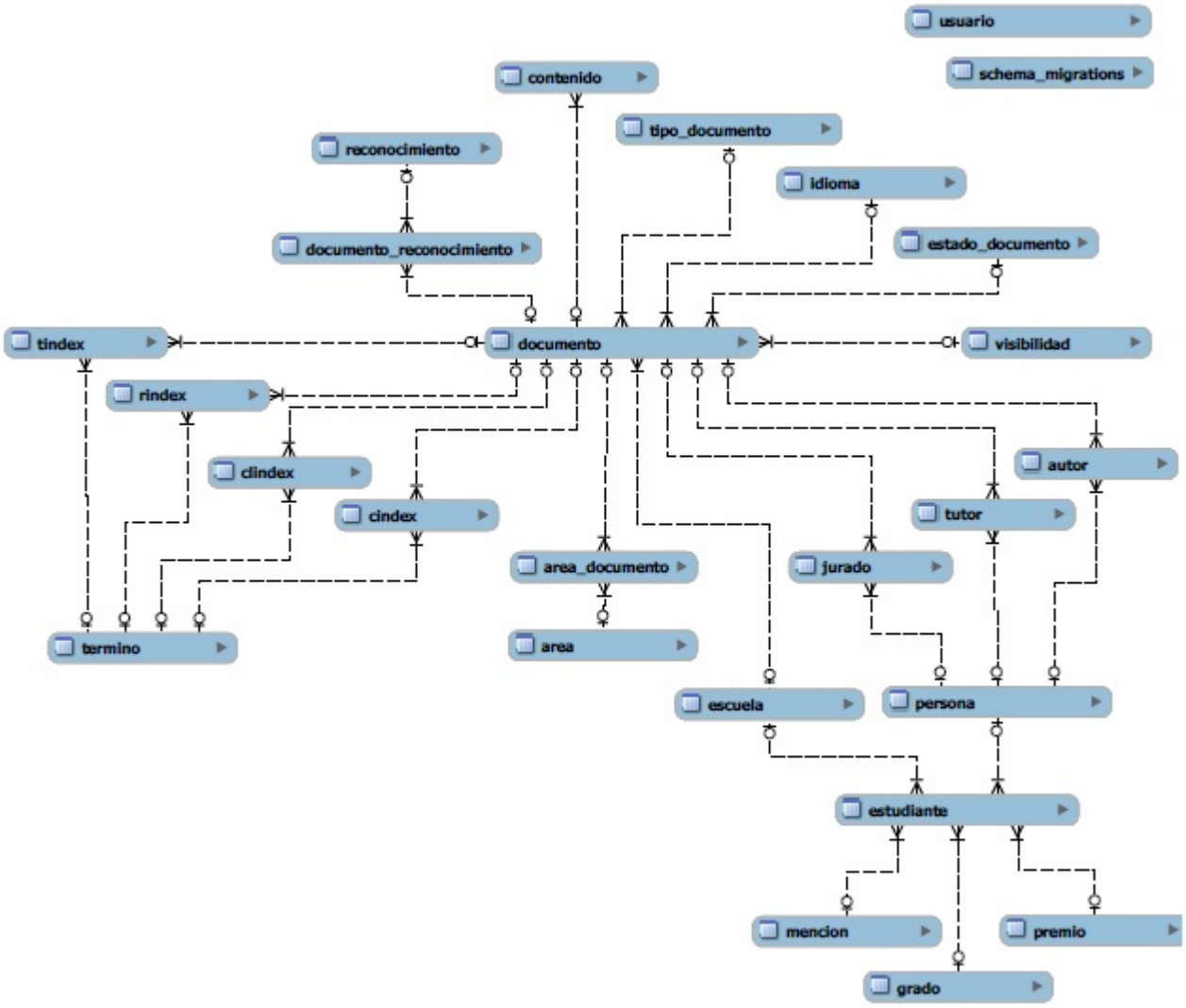


Figura 6.4: BUSCONEST 2: Diagrama Entidad-Relación

Implementación: La Base de Datos se implementó mediante el empleo de migraciones provistas por los generadores del *framework* de desarrollo *web* Ruby on Rails. Para garantizar la integridad de los datos se aplicaron restricciones a nivel de base de datos ya que las migraciones provistas por Rails no generan directamente la claves foráneas a nivel de base de datos. Asimismo algunos campos de la tabla Documento debieron ser cambiados manualmente mediante consultas del tipo ALTER TABLE para poder cambiar los tipo de datos por defecto creado por las migraciones, en los casos necesarios. Por ejemplo el tipo de dato string corresponde al tipo de datos varchar(256) en MySQL pero para el campo título y resumen se necesitan tipos de datos LONGTEXT y MEDIUMTEXT.

6.3. Iteración 2

Meta *Sprint:* Implementar la Interfaz de Administración (Admin Panel)

Se implementó una interfaz de usuario para que un usuario con rol de administrador manejara toda la información concerniente a los documentos, personas y estudiantes y datos asociados.

Tabla 6.2: Tabla de Control - Iteración 2

N#	Fecha	Tareas	Tipo
1	18/02/2013	Implementación de Recursos principales y secundarios	Interfaz de Administración
2	04/03/2013	Integración sistema de autenticación con gema Devise	Interfaz de Administración
3	11/03/2013	Validaciones y *Callbacks*	Interfaz de Administración

Planificación: La arquitectura del sistema BUSCONEST 2 está orientada a recursos. Un recurso es una entidad con un identificador unívoco que soporta cuatro operaciones elementales: Crear, Actualizar, Recuperar y Eliminar (CRUD). En BUSCONEST 2 los recursos principales son Persona y Documento. A su vez estos recursos se encuentran relacionados con otros recursos secundarios que permiten capturar información adicional. Se debe proveer una interfaz de usuario que permita la ejecución de estas operaciones básicas, y aquella adicionales que se consideren pertinentes.

Diseño: Para diseñar el recurso documento se debe tomar en cuenta los datos descriptivos del documento a indexar, comúnmente llamados metadatos. El documento consta de cuatro campos principales: el título, el resumen, el contenido y las palabras claves y un conjunto de campos adicionales, algunos de ellos opcionales. Los campos principales tienen una mayor importancia porque a partir de ellos se generará un índice invertido que nos permitirá recuperar los documentos según la ocurrencia de términos en dichos campos. El segundo recurso principal se trata del recurso Persona que está relacionado con los documentos según un rol (autor, tutor y jurado). Dependiendo del tipo de documento (si es T.E.G. o seminario) un documento debe estar relacionado con uno o dos autores, uno o dos tutores y siempre dos jurados asignados, de otro modo se considera un documento general sin calificación asociada que pudiese tener de uno a tres autores, por ejemplo un *paper*, unas notas de docencia o cualquier otro documento de interés. Adicionalmente se crea el recurso Estudiante, que está relacionado con el recurso Persona, esto permite capturar información adicional como promedio ponderado, promedio general y eficiencia, así como los premios otorgados al estudiante por su desempeño a lo largo de sus estudios universitarios (Summa Cum Laude, Magna Cum Laude, Premio Especial de Graduación).

Implementación: Se utilizaron los generadores provistos por Ruby on Rails para la creación de recursos. En total ellos son:

- Persona
- Estudiante

- Documento
- Reconocimientos
- Tipo de documento
- Idioma
- Mención
- Área
- Premio

Cada recurso se puede manejar desde un panel de administración llamado Admin Panel. Asimismo para acceder a dicho panel se requiere garantizar la autenticación por parte del administrador ya que sólo este podrá cambiar el estado de la información de los recursos del sistema. La gema de autenticación Devise fue utilizada con este propósito. Devise consta de once módulo activables de los cuales se utilizaron cuatro: *Timeoutable*, *Trackable*, *Database Athenticatable*, *Validatable*. *Timeoutable* desautentica automáticamente al usuario después de un periodo de tiempo determinado de inactividad. *Trackable* registra el número de autenticaciones que ha llevado a cabo el usuario con su respectiva fecha y dirección IP. *Database Athenticatable* y *Validatable* son usados para el almacenamiento de los datos del usuario en una tabla llamada *usuario* y para verificar que los datos ingresados a dicha tabla sean correctos, respectivamente.

Los reconocimientos, tipo de documento, los idiomas, las menciones, las áreas y premios, pueden ser creadas a conveniencia por el administrador. El tipo de documento se refiere a si el documento es un *paper*, una nota de docencia o de cualquier otro tipo que el administrador considere conveniente. Sólo los documentos tipo T.E.G. y Seminario contemplan evaluación y asignación de tutores y jurados. Cabe destacar que los reconocimientos son otorgados al Documento y no a la Persona, para ese caso se maneja otra abstracción denominada Premio. Mención Honorífica es un ejemplo de un Reconocimiento relacionado a un Documento. Summa Cum Laude es un premio otorgado a un Estudiante. Las menciones están relacionadas con los estudiantes y las áreas con los documentos, además las áreas no necesariamente tienen que coincidir con las menciones. Por ejemplo puede existir un estudiante cuya mención sea Inteligencia Artificial, pero también puede existir un documento relacionado con el área Minería de Datos (una disciplina dentro de la Inteligencia Artificial).

Los roles Autor, Tutor y Jurado se manejan con tres tablas asociativas que relacionan un recurso Persona con un recurso Documento. El CRUD sobre estos roles se realiza automáticamente al realizar las operaciones básicas sobre los recursos Persona, Estudiante y Documento. Las mismas se muestran en la figura 6.5.

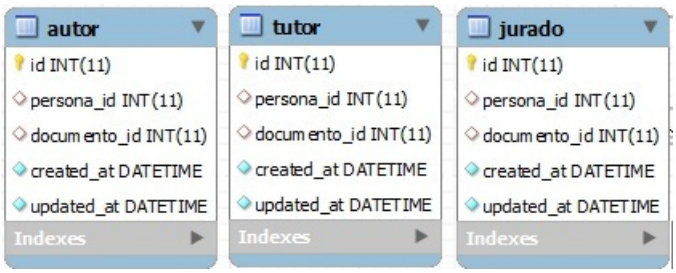


Figura 6.5: Esquema: Tablas Autor, Tutor y Jurado

Adicionalmente se crearon entidades que no son sujetas a modificación, ya que se esperan que no cambien o representan datos internos al sistema. Estas entidades son:

- Escuela
- Grado
- Premio
- Visibilidad
- Estado del documento

Escuela corresponde a las escuelas de la Facultad de Ciencias, a saber: Biología, Computación, Física, Matemática y Química. Se creó una escuela Geoquímica[2], a pesar que a la fecha la misma no exista propiamente en la Facultad de Ciencias de la UCV, para simplificar el diseño de la Base de Datos y la recuperación de documentos pertenecientes a esta licenciatura. GRADO es un atributo relacionado a un estudiante y corresponde al valor PREGRADO y POSTGRADO. Cabe destacar que una Persona puede tener más de un estudiante asociado, ya que esa persona puede haber cursado varios pregrados y/o varios postgrados en la Facultad de Ciencias.

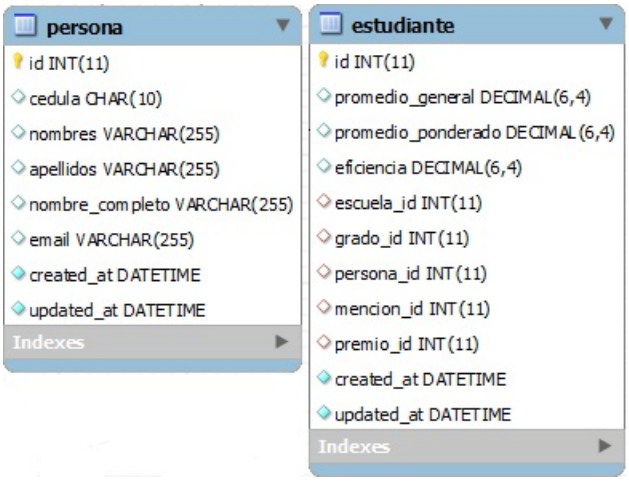


Figura 6.6: Esquema: Tablas Persona y Estudiante

Premio corresponde a los reconocimientos concedidos por la Universidad Central de Venezuela a los alumnos destacados. La visibilidad puede tomar valores de *publico* si el autor desea que su documento se pueda descargar desde la *web* o *privado* en caso contrario. El estado del documento permite saber si el documento

[2]La Licenciatura de Geoquímica es otorgada por la Escuela de Química.

es *nuevo* o ya se encuentra presente es los índices (*indexado*). Finalmente el campo *estado_carga* indica si el respectivo archivo PDF se encuentra asociado al documento.

Las campos y tipos de datos de la tabla Documento se muestran a continuación:

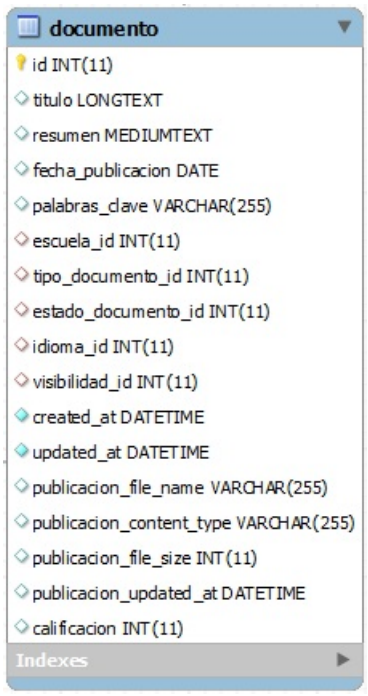


Figura 6.7: Esquema: Tabla Documento

Pruebas: Para realizar las pruebas en esta iteración se integraron declaraciones nativas soportadas por Ruby on Rails para validar elementos de datos, conocidas como *validators,* en cada uno de los Recursos. Los *validators* son directivas declarativas en los Modelos que indican las restricciones a aplicar en los campos donde se ingresarán datos. Los *validators* poseen diversos mecanismos de evaluación para impedir el ingreso datos incorrectos, tales como de campos vacíos o fuera de rango, entre otros. Además los *validators* permiten el despliegue de mensajes de error, no sólo en las vistas HTML sino en formas de mensajes en texto plano que pueden ser desplegados por un terminal durante el proceso de pruebas . Asimismo fueron integrados *callbacks*[3] que permitieron procesar los datos antes que estos fuesen validados. Por ejemplo los datos como nombres y apellidos de una persona son almacenados siempre en mayúscula sin acentos aunque sean ingresados en minúsculas y acentos. Adicionalmente se utilizó la opción :*restric* en varios *validators*, esta opción impide que un registro sea eliminado si existe algún otro registro asociado a él, esto con el fin de conservar la integridad referencial de recursos relacionados entre sí.

[3]Funciones pasadas por parámetros invocadas en el contexto de la función que la recibe.

A continuación se muestra la creación de un documento mediante la instrucción p = Documento.new que una instancia documento con los campos vacíos. Cuando se trata de bajar está instancia a la base de datos con la llamada p.save el sistema rechaza la ejecución de esta instrucción y efectúa un *rollback* en la base de datos. Luego se muestran los mensajes de error asociados. Estos mensajes pueden ser procesados posteriormente por ejemplo imprimiéndolos en una vista HTML.

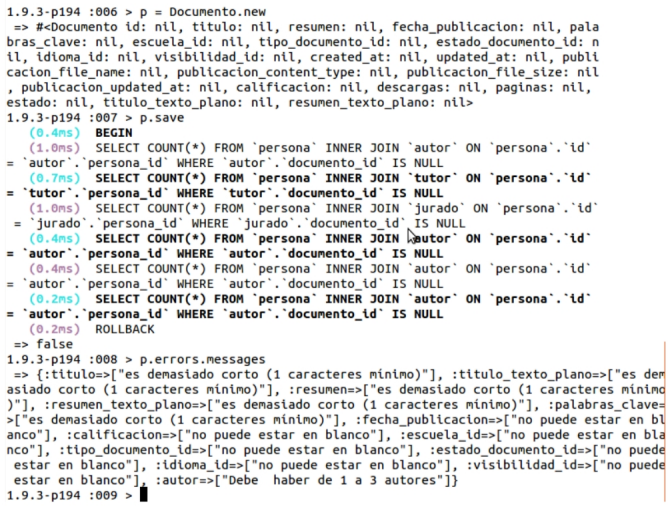


Figura 6.8: Prueba: *Rollback* y mensajes de error

Para poder implementar la validación mostrada en 6.8 se necesito declarar el respectivo *validator* en el modelo Documento. El mismo se muestra en 6.1.

Algoritmo 6.1 *Validator*

```
validates :titulo, :length => { :in => 1..4294967295 }
validates :titulo_texto_plano, :length => { :in => 1..4294967295 }
validates :resumen, :length => { :minimum => 1 }
validates :resumen_texto_plano, :length => { :minimum => 1 }
validates :palabras_clave, :length => { :in => 1..255 }
```

De forma análoga se implementaron los *validators* requeridos para todos los recursos implementados en el sistema. En general la no admisión de campos vacíos o con determinado formato fue empleada reiteradamente, como por ejemplo al momento de validar un nombre o una calificación. Sin embargo, algunos recursos requirieron validaciones más complejas. Por ejemplo el recurso Documento requiere que todos los autores, tutores u jurados sean diferentes entre sí, si el documento es un T.E.G o seminario.

Adicionalmente se implementaron *callbacks* que permiten cambiar el formato de los datos ingresados antes de ser procesados por los *validators*. Por ejemplo, si un usuario ingresa alguna descripción en minúsculas la misma será cambiada mayúscula antes de la inserción en la base de datos. La definición de un *callback* es mostrada en el algoritmo 6.2.

Algoritmo 6.2 *Callback*

```
before_validation do |persona|
    eliminar_acentos!(persona.nombres).upcase!
    eliminar_acentos!(persona.apellidos).upcase!    eliminar_acentos!(persona.nombre_completo).upcase!
eliminar_acentos!(persona.email).upcase!
end
```

6.4. Iteración 3

Meta ***Sprint:*** Implementar un módulo que permita extraer las palabras contenidas en un documento y que permita el indexado de un documentos mediante ejecución no interactiva.

Tabla 6.3: Tabla de Control - Iteración 3

N#	Fecha	Tareas	Tipo
1	18/03/2013	Integración de la gema Paperclip para carga de archivos	Carga de documentos
2	25/03/2013	Integración de la gema PDF-Reader para extraer palabras	Extractor de palabras
3	01/04/2013	Indexado de los documentos en los índices de BUSCONEST 2	Indexado de Documentos
4	08/04/2013	Implementación de una rutina de indexado por lotes	Indexado de Documentos

Planificación: Una vez implementada la Interfaz de Administración, se requiere que el usuario administrador pueda subir documentos usando ese medio. Luego se implementará una rutina que permita el indexado del documento extrayendo cada uno de los términos que contiene el documento. Seguidamente se actualizarán 4 índices según la sección donde ocurren los términos en el documento (*tindex* en el título, *cindex* en el contenido, *rindex* en el resumen y *clindex* en las palabras clave). Finalmente se implementará la funcionalidad de invocar esta rutina sobre todos los documentos de forma no interactiva.

En la figura 6.9 se puede observar los cuatro índices del sistema BUSCONEST 2.

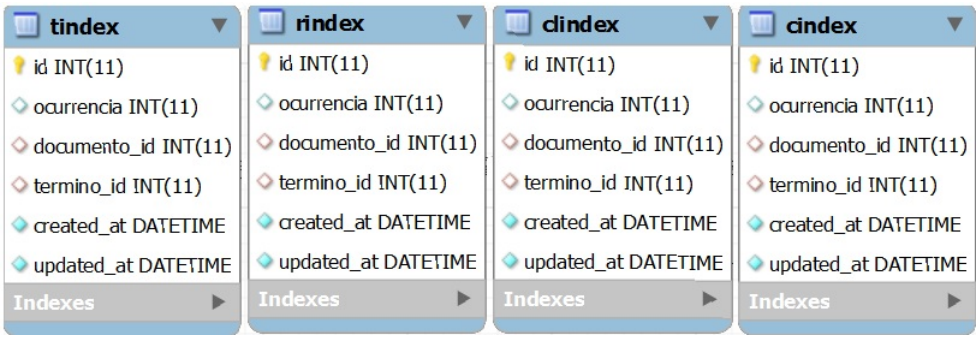


Figura 6.9: BUSCONEST 2: Índices invertidos

Diseño: Para poder resolver el problema de indexado se debe pensar el problema en 4 subproblemas bien diferenciados. Ellos son:

1. Cargar el documento
2. Extraer los términos contenidos dentro del documento
3. Indexar un documento
4. Indexar todos los documentos no indexados en el repositorio de forma no interactiva

El primer problema se solucionó utilizando la gema Paperclip, esta gema provee varios opciones para la subida de documentos, tales como verificación del tamaño y tipo de archivo. Su utilización resulta conveniente ya que permite integrar los archivos subidos con un modelo previamente implementado, en nuestro caso el modelo Documento.

El segundo problema se solucionó utilizando la gema PDF-Reader la cual provee una rutina donde se le indica la ruta o *path* de un documento en el sistema de archivos y devuelve el contenido de dicho documento en forma de *string*. Luego de extraído el respectivo *string* se requirió la implementación de una rutina que recibiera de entrada un *string* y devolviera un arreglo donde cada uno de sus elementos fuese un término. Sin embargo si queremos indexar un documento este arreglo debe ser procesado de nuevo para eliminar los términos que generan ruido en las búsquedas (esto es, términos que aparecen en todos los documento con una frecuencia muy alta pero que no tienen significado en sí mismos, tales como artículos, pronombres, preposiciones, etc). Es ideal que sólo aquellas palabras que representasen adjetivos, sustantivos, y adverbios sean la mayoría de los términos asociados a un documento, esto aplica para cualquier idioma. Una vez obtenido este arreglo depurado se necesita generar a partir de él un *hash* de la forma $\{t_0 \Rightarrow o_0, \ldots, t_n \Rightarrow o_n\}$donde t_i representa un término y o_i representa la ocurrencia de ese término en ese documento. Por ejemplo el *hash* $\{dato \Rightarrow 10, \ldots, sistema \Rightarrow 5\}$indica que el documento contiene el término dato 10 veces y la palabra sistema 5 veces. Una vez obtenido este *hash* se procede al indexado.

El tercer problema consiste en el indexado del documento. Por indexado nos referimos a la acción de asociar todos los términos contenidos en un documento con el identificador del documento. Realizando esta operación permitimos que el documento en cuestión sea recuperable al ingresar algunos de los términos que lo describen. Esta estrategia es conocida como Modelo Booleano de recuperación de información y se basa en la construcción de un índice invertido, ver figura 2.4 en la página 39. En nuestro caso se implementarán cuatro índices invertidos cada uno correspondiente a la sección del documento donde ocurrieron los términos (título, contenido, resumen y palabras clave). Es decir por cada sección del documento existe un índice invertido representado por una tabla en la base de datos.

Una vez construido el índice invertido, se puede proceder a resolver el cuarto problema implementando una rutina en el modelo Documento que consista en invocar la rutina para resolver el problema tres pero sobre cada uno de los documentos. Esta rutina debe ser invocada externamente a la aplicación, por ejemplo sería deseable que se pudiese ejecutar como una tarea periódica *cron*[4].

Implementación: Para resolver los cuatro problemas analizados en el apartado de diseño se implementaron las siguientes funciones.

[4] *cron* es el planificador de tareas periódicas para sistemas Unix.

Tabla 6.4: Funciones para indexar un documento

Problema	Función	Entrada	Salida	Ubicación
1	*extraer_texto*	*string*	*string*	Módulo Utilidades
2	*generar_terminos*	*string*	arreglo	Módulo Utilidades
2	*agrupar_terminos*	arreglo	*hash*	Módulo Utilidades
3	*indexar*	*hash*	nil	Acción: Documento#Indexar
4	*indexar_todo*	void	nil	Modelo: Documento

La función *extraer_texto* recibe un parámetro *string* que es la ubicación del documento en el sistema de archivos (*path*). La función extrae en un *string* todo el contenido de un documento que esté en formato PDF mediante la utilización de la gema PDF-Reader.

Algoritmo 6.3 Extracción de términos de un documento PDF

```
def self.extraer_texto(archivo)
    texto = ""
    lector = PDF::Reader.new(archivo)
    lector.pages.each do |pagina|
        if !pagina.text.nil?
            texto = texto + pagina.text
        end
    end
    return texto
end
```

La función *generar_terminos* crea un arreglo de términos a partir de un *string*. En particular localiza los elementos léxicos dentro de un *string* según la ocurrencia de espacios en blanco y caracteres especiales. Cada uno de los elementos léxicos encontrados son insertados como los elementos de un arreglo.

Algoritmo 6.4 Identificación de términos en un *string*

```
def self.generar_terminos(texto="")
    texto = texto.upcase

    # Dividir el texto en sus terminos constituyentes
    terminos = texto.split(/\W+|\s+/).reject{ |e| e.empty? }

    # Eliminar los terminos que contengan sólo números
    terminos.reject! {|word| word =~ /\A[0-9]+\z/ }

    # Eliminar los terminos de longitud 1
    terminos.reject! {|word| word.length == 1 }

    # Eliminar todos los terminos innecesarios
    terminos.reject! {|word| word =~ palabras_ruido }       #Eliminar terminos con más de 255 caracteres

    terminos.reject! {|word| word.length > 255 }

    return terminos
end
```

La función *agrupar_terminos* crea un *hash* a partir de un arreglo de términos, para ello se cuenta el número de veces que ocurre un término y almacena esa cantidad junto al término en cuestión.

Algoritmo 6.5 Agrupación de términos y conteo de ocurrencias

```
def self.agrupar_terminos(arr_terminos)
    hash = Hash.new(0)
    arr_terminos.each {|palabra| hash[palabra] += 1 }
     return hash
end
```

La función *indexar* es la encargada de crear la asociación término documento y de construir el índice invertido. En particular BUSCONEST 2 cuenta con cuatro tablas las cuales capturan esta asociación. Los índices son: *tindex* (almacena la asociación documento-término si esta ocurre en el título), *cindex* (almacena la asociación documento-término si esta ocurre en el contenido), *rindex* (almacena la asociación documento-término si esta ocurre en el resumen), y *clindex* (almacena la asociación documento-término si esta ocurre en las palabras clave). Además las asociaciones guardan el número de ocurrencias del término para poder ser utilizados en los algoritmos de Ordenamiento de resultados.

Algoritmo 6.6 Indexado de términos de un documento dado

```
def indexar
    @documento = Documento.find(params[:id])
    # Si el documento tiene el estado NUEVO y esta COMPLETO (tiene archivo PDF asociado)
    if @documento.estado_documento_id == NUEVO && @documento.estado == 'COMPLETO'
        # Almacenar el nuevo contenido en la BD
        @documento.build_contenido
        @documento.contenido.texto=Utilidades::extraer_texto(@documento.publicacion.path)
        @documento.contenido.save
        Utilidades::llenar_tindex(@documento)
        Utilidades::llenar_cindex(@documento)
        Utilidades::llenar_rindex(@documento)
        Utilidades::llenar_clindex(@documento)
        @documento.estado_documento_id = INDEXADO
    end
                                                    ⋮
end
```

La función *indexar_ todo* indexa todos los documentos del repositorio que no hayan sido indexados, es decir itera sobre todos los documentos con estado *nuevo*. Un documento tiene estado *nuevo* al ser creado o al ser modificado. Esta rutina se define de forma estática para que pueda ser invocada mediante la utilización del comando *rails runner co*n la finalidad que pueda ser ejecutada por el planificador de tareas periódicas *cron* de forma externa a la aplicación.

Algoritmo 6.7 Indexado de todos los documentos en el repositorio

```
def self.indexar_todo
    documentos=Documento.where("estado_documento_id='NUEVO'
                                        AND estado like 'COMPLETO'")
    documentos.each do |documento|

        # Almacenar el nuevo contenido en la BD
        documento.build_contenido
        documento.contenido.texto=Utilidades::extraer_texto(documento.publicacion.path)
        documento.contenido.save

        # Llenar los indices del buscador
        Utilidades::llenar_tindex(documento)
        Utilidades::llenar_cindex(documento)
        Utilidades::llenar_rindex(documento)
        Utilidades::llenar_clindex(documento)

        documento.estado_documento_id = INDEXADO
        documento.save
        puts "#{documento.titulo} #{documento.estado_documento.descripcion}"
     end
end
```

Pruebas: Las pruebas en esta sección se realizaron mediante la subida de diferentes archivos de tamaño y contenido controlados con la respectiva inspección de términos en los índices *tindex*, *cindex*, *rindex* y *clindex*. Se comprobó que sólo archivos en formato PDF e inferiores a 100 MB se pudieran cargar por la interfaz de administración.

La figura 6.10 muestra un intento de carga de archivo con formato incorrecto, y en la figura 6.11 muestra un intento de carga de archivo demasiado grande.

Figura 6.10: Carga de archivo con formato incorrecto

Figura 6.11: Carga de archivo con tamaño incorrecto

Adicionalmente se indexó un documento de prueba cuyo título, resumen, palabras claves y contenido en el archivo PDF eran los términnos *documento* y *prueba*. Se revisaron los índices respectivos en la base de datos a continuación (*tindex*, *rindex*, *clindex* y *cindex*). Ver figura 6.12.

```
mysql> SELECT titulo as documento,descripcion as termino,ocurrencia FROM tindex JOIN documento as d ON(documento_id=d.id) JOIN termino as t ON(
termino_id=t.id) WHERE d.titulo LIKE "%Documento prueba%";
+------------------------+-----------+------------+
| documento              | termino   | ocurrencia |
+------------------------+-----------+------------+
| <p>Documento prueba</p> | DOCUMENTO |          1 |
| <p>Documento prueba</p> | PRUEBA    |          1 |
+------------------------+-----------+------------+
2 rows in set (0.00 sec)

mysql> SELECT titulo as documento,descripcion as termino,ocurrencia FROM rindex JOIN documento as d ON(documento_id=d.id) JOIN termino as t ON(
termino_id=t.id) WHERE d.titulo LIKE "%Documento prueba%";
+------------------------+-----------+------------+
| documento              | termino   | ocurrencia |
+------------------------+-----------+------------+
| <p>Documento prueba</p> | DOCUMENTO |          1 |
| <p>Documento prueba</p> | PRUEBA    |          1 |
+------------------------+-----------+------------+
2 rows in set (0.00 sec)

mysql> SELECT titulo as documento,descripcion as termino,ocurrencia FROM clindex JOIN documento as d ON(documento_id=d.id) JOIN termino as t ON
(termino_id=t.id) WHERE d.titulo LIKE "%Documento prueba%";
+------------------------+-----------+------------+
| documento              | termino   | ocurrencia |
+------------------------+-----------+------------+
| <p>Documento prueba</p> | DOCUMENTO |          1 |
| <p>Documento prueba</p> | PRUEBA    |          1 |
+------------------------+-----------+------------+
2 rows in set (0.00 sec)

mysql> SELECT titulo as documento,descripcion as termino,ocurrencia FROM cindex JOIN documento as d ON(documento_id=d.id) JOIN termino as t ON(
termino_id=t.id) WHERE d.titulo LIKE "%Documento prueba%";
+------------------------+-----------+------------+
| documento              | termino   | ocurrencia |
+------------------------+-----------+------------+
| <p>Documento prueba</p> | DOCUMENTO |          1 |
| <p>Documento prueba</p> | PRUEBA    |          1 |
+------------------------+-----------+------------+
```

Figura 6.12: Términos indexados

6.5. Iteración 4

Meta *Sprint:* Desarrollar una Interfaz Pública que permita la recuperación de documentos por parte de los usuarios sin privilegios de administración.

Tabla 6.5: Tabla de Control - Iteración 4

N#	Fecha	Tareas	Tipo
1	15/04/2013	Implementación Búsqueda Simple	Interfaz Pública
2	29/04/2013	Implementación de *Dashboard*	Interfaz de Administración
3	06/05/2013	Implementación Búsqueda Avanzada	Interfaz Pública
4	20/05/2013	Implementación Directorio Temático	Interfaz Pública
5	27/05/2013	Contar número de descargas	Interfaz Pública

Planificación: El Sistema BUSCONEST 2 requiere que un usuario no autenticado pueda recuperar los documentos almacenados en el repositorio. Para ello se proponen tres paradigmas distintos para efectuar dicha recuperación. El primero consiste en el ingreso de términos para recuperar los documentos asociados a éstos. Adicionalmente se requiere la implementación de un formulario donde se ingrese las características descriptivas de los documentos y se recuperen los documentos que las presentan. Asimismo se dispondrá de un directorio temático que permitirá visualizar los documentos agrupados por categorías. También se dispondrá de una vista adicional en el módulo administrador que despliegue el estado general del repositorio con información descriptiva del mismo incluyendo la información correspondiente al número de descargas de un documento.

Diseño: Para implementar la recuperación vía términos se divide el problema en dos subproblemas bien diferenciados. El primero es la recuperación de los documentos asociados a los términos ingresados en la consulta del usuario y el segundo es el ordenamiento de dichos documentos dependiendo su importancia. Se diseña el formulario mostrado en la figura 6.13.

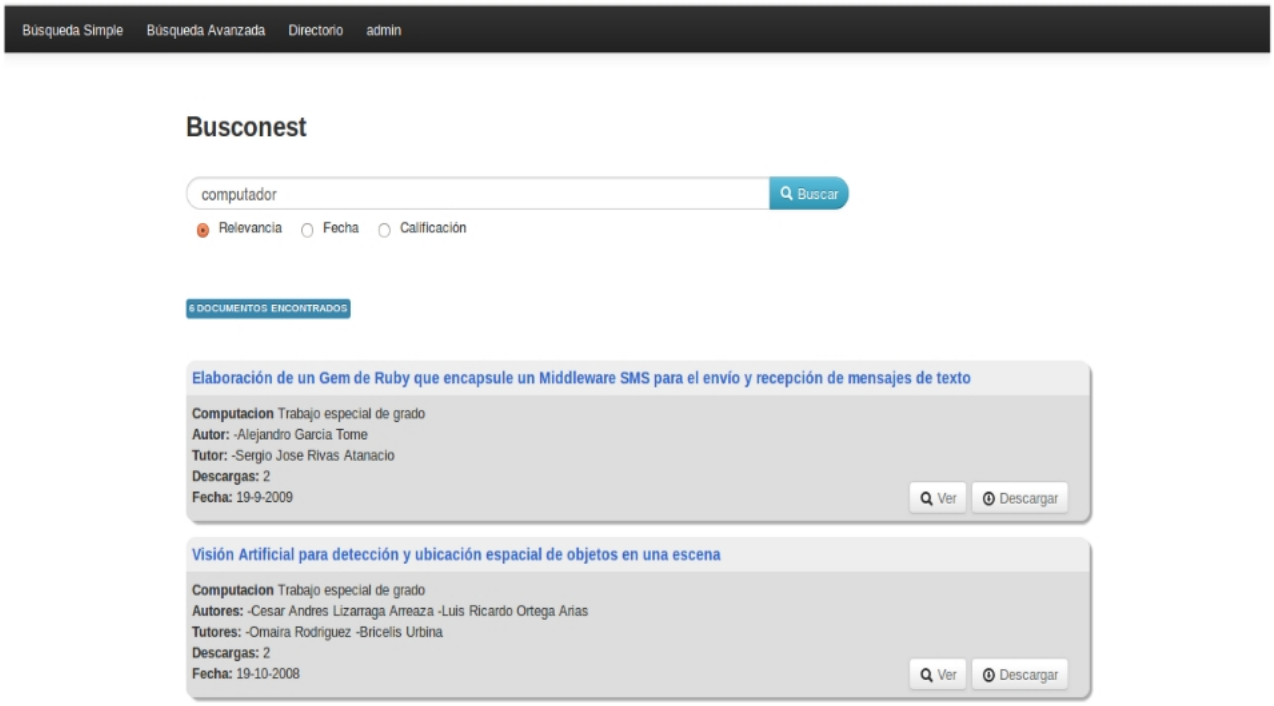


Figura 6.13: Búsqueda Simple

Se establecen tres criterios para realizar el Ordenamiento o *Ranking* de los documentos: por fecha, por calificación y por relevancia. El ordenamiento por fecha listará en los primeros lugares de forma decreciente todos los documentos que contengan al menos una ocurrencia de algunos de los términos ingresados en la consulta en alguna sección del documento. Es decir de mayor fecha a menor fecha. Análogamente si el usuario selecciona la opción calificación se listarán los documentos asociados a los términos ingresados mostrándose de primero los de mayor calificación, pero como es posible que muchos documentos coincidan en su calificación, adicionalmente los documentos que coincidan en calificación serán ordenados a su vez por fecha mostrándose de primero los más recientes. El tercer criterio, llamado relevancia, corresponde a un valor asignado a cada documento dependiendo de varias características en el mismo. Primeramente se analiza la ocurrencia de los términos ingresados en la consulta en las diferentes secciones del documento. La sección del título y las palabras clave tienen una importancia mayor que la sección resumen, y ésta a su vez tiene mayor importancia que la ocurrencia de los términos en la sección contenido. Seguido de esto también se toma en cuenta en la ponderación la calificación del documento, la fecha de publicación y el número de descargas. Con esta información se realiza un cálculo que determinará un valor numérico al documento y que permitirá establecer el ordenamiento de los documentos según este criterio.

Para implementar la búsqueda avanzada se establecen los siguientes parámetros:

- Autor

- Tutor
- Jurado
- Titulo
- Resumen
- Contenido
- Palabras clave
- Tipo de documento
- Año de publicación
- Idioma
- Escuela
- Reconocimiento

Los mismos sin ingresados en el formulario presentado en la figura 6.14.

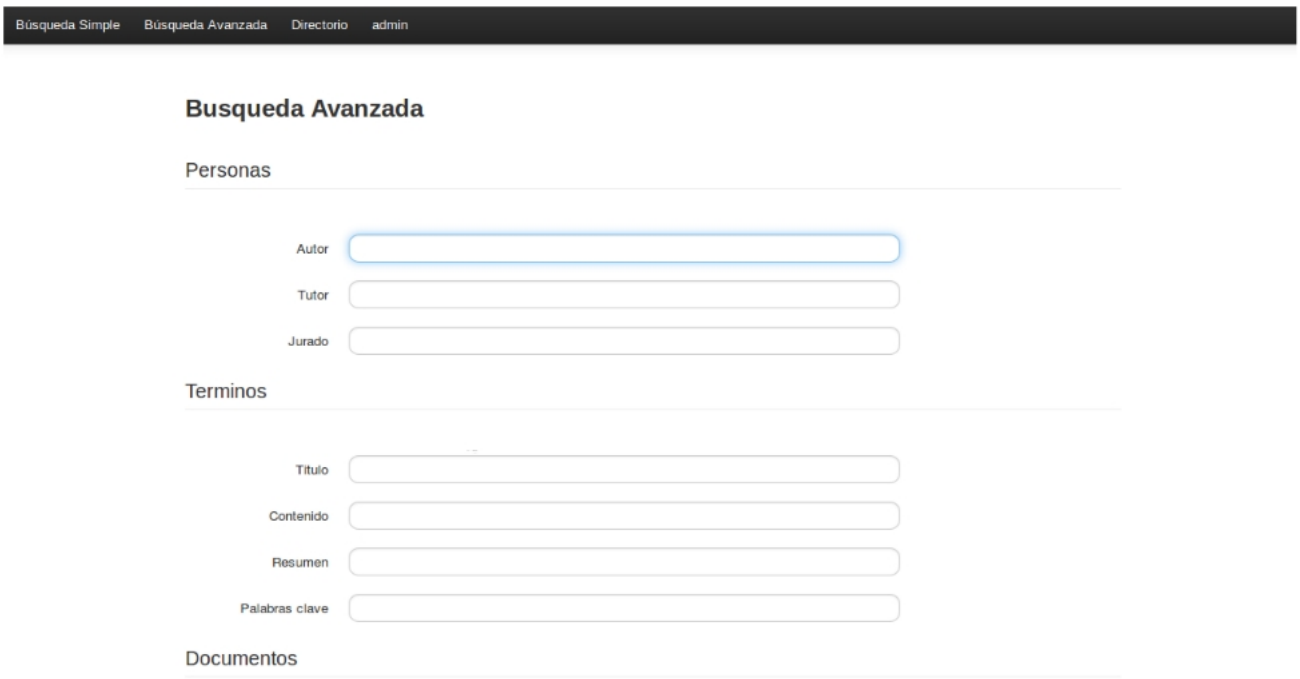


Figura 6.14: Búsqueda Avanzada

El directorio temático organizará los documentos por tres categorías, estas son: escuela, tipo de documento y año de publicación. Debido a que muchos documentos pueden coincidir en cada una de estas categorías, los documentos se ordenan a su vez por fecha de publicación descendentemente. La visualización del directorio temático es mostrada en la figura 6.15.

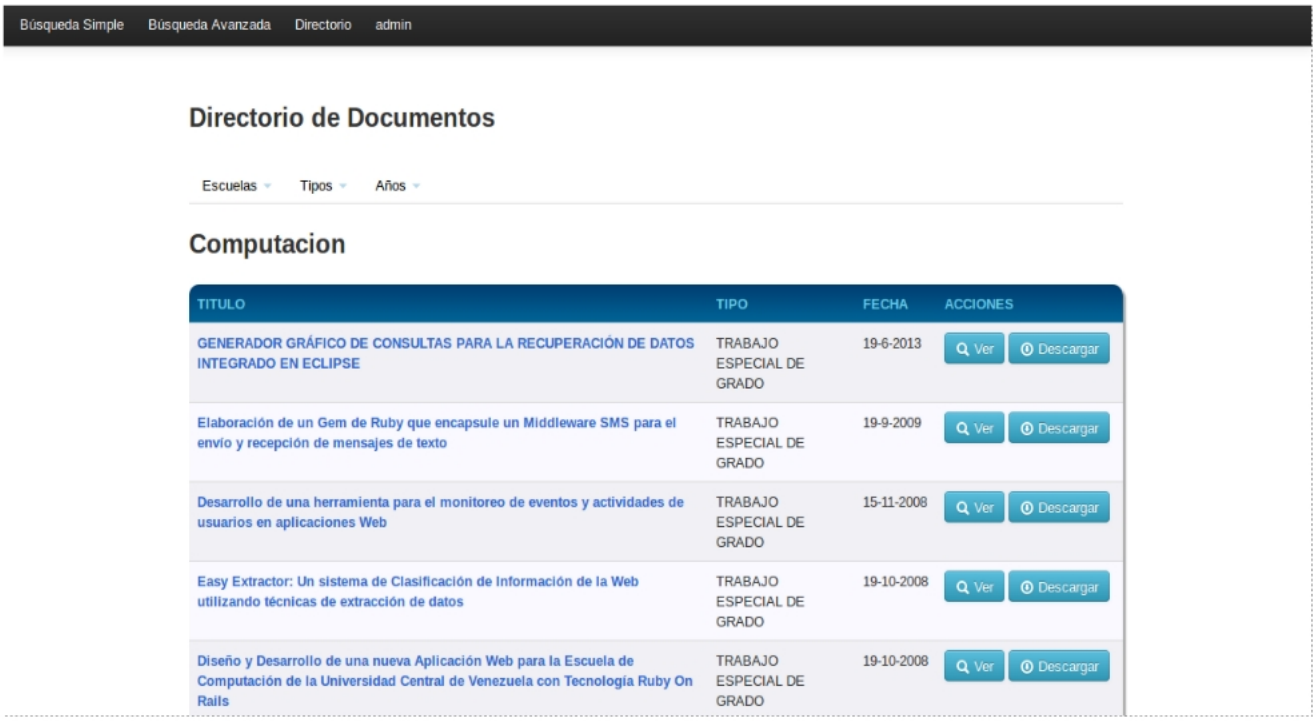


Figura 6.15: Directorio Temático

El *Dashboard* consiste en una serie de tablas que permiten visualizar el estado general del repositorio de forma rápida y práctica. En el mismo se puede verificar información concerniente a la cantidad total de documentos almacenados en el repositorio, la cantidad de documentos indexados, la cantidad de documentos asociados a cada escuela, entre otros.

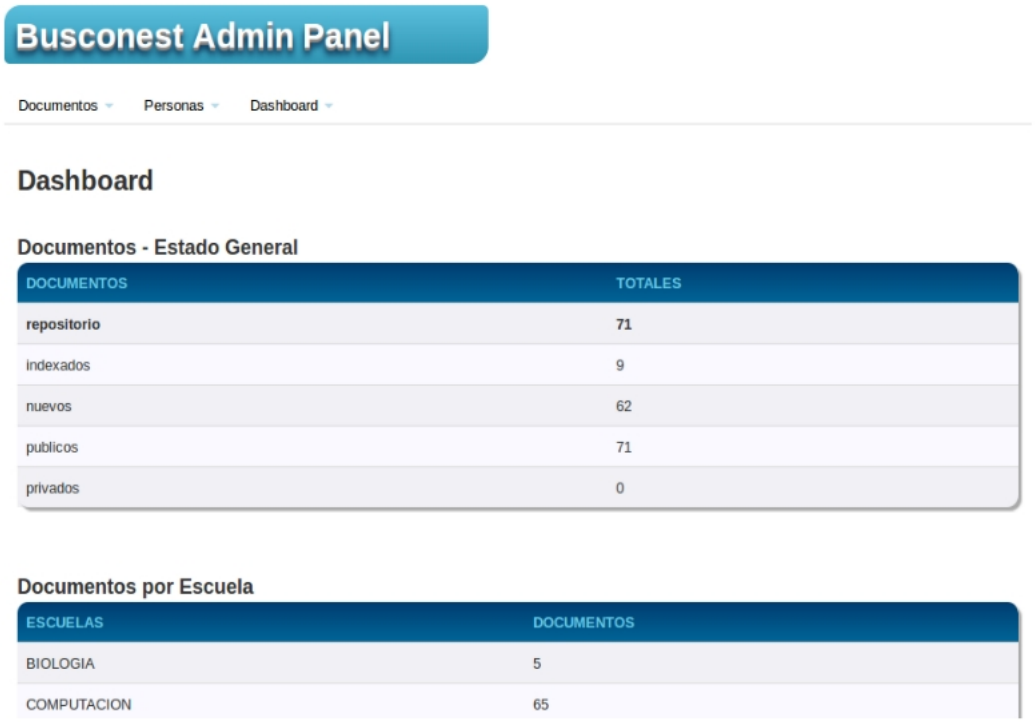


Figura 6.16: *Dashboard*

Implementación: La implementación de la Búsqueda Simple implicó el desarrollo de la función RECUPERAR_DOCUMENTOS(TERMINOS_ARRAY) ubicada en el directorio /lib/utilidades/utilidades.rb. Esta función recibe de parámetros de entrada un arreglo donde cada elemento es un término de la consulta del usuario y retorna un arreglo con la forma:

[[doc_id$_1$, [[doc_id$_1$, titulo, fecha, termino$_1$_descripcion, o_titulo, o_contenido, o_resumen, o_palabras_clave, termino$_1$_id, (Freq.doc$_1$)],...]], ..., [doc_id$_n$, [[doc_id$_n$, titulo, fecha, termino$_n$_descripcion, o_titulo, o_contenido, o_resumen, o_palabras_clave, termino$_n$_id, (Freq.doc$_n$)],...]]

Este arreglo puede ser interpretado de la siguiente manera. Suponiendo una consulta con la estructura $t_1 t_2 \ldots t_m$, donde t_i indica un término cualquiera:

[[doc_id_1, [[Descripción del documento$_1$ con respecto al término t_1],..., [Descripción del documento$_1$ con respecto al término t_n]] , ... , [doc_id_n, [[Descripción del documento$_n$ respecto al término t_1], ... , [Descripción del documento$_n$ respecto al término t_m]]]

Las descripciones del documento permiten saber los términos que ocurrieron en un documento particular y su frecuencia de ocurrencia en las diferentes secciones de dicho documento. Una descripción tiene el formato:

[doc_id, titulo, fecha, termino_descripcion, o_titulo, o_contenido, o_resumen, o_palabras_clave, termino1_id, *Freq*]

Donde los campos o_titulo, o_contenido, o_resumen y o_palabras_clave representan la cantidad de ocurrencias de los términos en las diferentes secciones del documento y *Freq* es un cálculo basado en estos valores como se sigue:

$$Freq = 10 * \text{o\_titulo} + \log_{10}(1 + \text{o\_contenido}) + \log_2(1 + \text{o\_resumen}) + 10 * \text{o\_palabras\_clave} \quad (6.1)$$

La frecuencia *Freq* pondera linealmente un término que ocurre el título y las palabras claves mientras que pondera el término con un crecimiento logarítmico si este ocurre en el resumen y en el contenido, evitándose de este modo que un documento con muchas ocurrencias de ese término en su contenido y/o resumen pueda escalar excesivamente en el *Ranking*.

La función RECUPERAR_DOCUMENTOS es particularmente útil ya que el arreglo retornado contiene los identificadores de los documentos que se encuentran asociados a los términos ingresados por el usuario. Por tanto, para implementar el *Ranking* por fecha y por calificación sólo basta con obtener dichos identificadores recorriendo el arreglo respuesta para instanciar los objetos ActiveRecord definidos en el modelo Documento. Luego este conjunto de objetos pueden ser ordenados por fecha o por calificación según sea necesario y ser pasados a la vista respectiva.

Algoritmo 6.8 Ordenamiento por fecha

```
# Almacenar los terminos de la consulta en un arreglo de terminos
@terminos = params[:terminos].split(/\W+|\s+/).reject{ |e| e.empty? }
consulta = recuperar_documentos(@terminos)

# Agrupar el resultado de la consulta por documentoID
documentos = consulta.to_a.group_by { |relacion| relacion[0] }

if params[:busqueda][:tipo] == 'fecha'
    tmp_docs = ((documentos.sort_by { |doc_id,doc| doc.first.third }).reverse).uniq
    @documentos = tmp_docs.paginate(:page => params[:page], :per_page => PER_PAGE)
else params[:busqueda][:tipo] == 'calificacion'
    tmp_docs=((@documentos_c.sort_by{|doc|[doc.calificacion,doc.fecha_publicacion]}).reverse).uniq
end
```

Para poder implementar la recuperación de documentos por relevancia se tomó en cuenta la calificación del documento, la fecha de publicación, el valor frecuencia (*Freq*) de cada uno de los términos que ocurrieron en el documento y la cantidad de descargas del documento. Los mismos están relacionados bajo la siguiente ecuación:

$$R\left(doc\right) = \left(\sum_{k=1}^{n} Freq_k\right) * \left(C/10\right) * \left(\log_{10}\left(D\right)\right) * F \tag{6.2}$$

Donde R denota la relevancia del documento *doc*. $\sum_{k=1}^{n} Freq_k$ es la sumatoria de cada uno de los valores *Freq* de cada término ocurrido en el documento. C es la calificación del documento $C\epsilon\left[0,20\right]$. Si el documento no es calificado (*papers*, notas de docencias,etc) entonces $C = 10$. D denota el número de descargas del documento, si $D < 10$ entonces a D se le asigna el valor 10 para evitar que $\log_{10}\left(D\right) < 0$. F denota un valor *Fecha* que se calcula como sigue:

$$F = 1 + \frac{F_p - F_{min}}{F_a - F_{min}}$$

Donde F_p es el año de publicación del documento. $F_a$es el año actual y F_{min}es el año de publicación del documento más antiguo en el repositorio (Por ejemplo $F = 2000$). La función F devuelve valores en el intervalo $[1, 2]$. Nótese que $F = 1$ si $F_p = F_{min}$ y $F = 2$ si $F_p = F_a$.

La Búsqueda Avanzada implicó el desarrollo de un formulario donde cada campo ingresado por el usuario se traduce en una cláusula WHERE construida para la recuperación del documento con las restricciones requeridas. Primeramente se verifican un conjunto de banderas que indican si el usuario ingresó la restricción en el campo del formulario de Búsqueda Avanzada, y a partir de estas banderas se construye la cláusula WHERE respectiva. Finalmente se ejecuta la consulta SQL necesaria para recuperar los documentos con dicha característica.

Algoritmo 6.9 Búsqueda Avanzada

```
clausula_where = []
@documentos = Documento

# Revisión de Banderas
autor_flag = true if params[:autor_id] != " " && params.has_key?(:autor_id)
tutor_flag = true if params[:tutor_id] != " " && params.has_key?(:tutor_id)
                                                    ⋮
# Construcción de la clausula where
if autor_flag
 clausula_where.push("autor.persona_id = #{(params[:autor_id])}")
 @documentos = @documentos.joins(:personas_autor)
end
                                                    ⋮
# Ejecución de la consulta
tmp_docs = @documentos.where(clausula_where).order('fecha_publicacion DESC').uniq

# Retornar documentos a la vista
@documentos = tmp_docs.paginate(:page => params[:page])
```

Para implementar el Directorio Temático se recuperan los documentos pertenecientes a una categoría específica ordenándolos respectivamente.

Algoritmo 6.10 Directorio Temático

```
def directorio_escuela
 @escuela = params[:escuela].upcase
 @documentos_escuela = Documento.where('escuela_id = ? AND estado_documento_id = ?', @escuela, INDEXADO ).order('fecha_publicacion DESC').paginate(:page => params[:page])
end

def directorio_tipo
 tipo_doc = TipoDocumento.find_by_descripcion_corta( params[:tipo] )
 @tipo = tipo_doc.descripcion
 @documentos_tipo = Documento.where("tipo_documento_id = ? AND estado_documento_id = ?", tipo_doc.id, INDEXADO ).order('fecha_publicacion DESC').paginate(:page => params[:page])
end

def directorio_fecha
 @fecha = params[:fecha]
 fecha_inicio = params[:fecha]+'/01/01'
 fecha_fin = (fecha_inicio.to_date + 1.year).strftime("%Y/%m/%d")
 @documentos_fecha = Documento.where("fecha_publicacion >= ? AND fecha_publicacion < ? AND estado_documento_id = ?", fecha_inicio, fecha_fin, INDEXADO).order('fecha_publicacion DESC').paginate(:page => params[:page])
end
```

La implementación de *Dashboard* implicó el cálculo de los siguiente datos:

- Cantidad total de documentos almacenados en el repositorio
- Cantidad total de documentos indexados
- Cantidad total de documentos nuevos
- Cantidad total de documentos por escuela
- Escuela con mayor cantidad de documentos asociados
- Cantidad de documentos por Áreas
- Área con más documentos
- Cantidad de documentos por Tipo de Documento
- Tipo de Documento con mayor cantidad de documentos asociados
- Autores, tutores y jurados con más documentos asociados
- Calificación promedio de los T.E.G.

- Calificación promedio de T.E.G. por escuela
- Calificación promedio de los seminarios
- Calificación promedio de los seminarios por escuela
- Agrupación de T.E.G por rangos de calificación
- Agrupación de seminarios por rangos de calificación
- Documento más descargado

Un extracto de algunos cálculos efectuado en el *Dashboard* se presentan a continuación:

Algoritmo 6.11 *Dashboard*

```
@documentos_indexados_totales=Documento.where("estado_documento_id=?", INDEXADO).size
@documentos_nuevos_totales=Documento.where("estado_documento_id = ?", NUEVO).size
                                                    ⋮
@documentos_publicos_totales=Documento.where("visibilidad_id=?",PUBLICO).size  @documentos_privados_totales=Documento.where("visibilidad_id = ?", PRIVADO).size
                                                    ⋮
@autores_maximo = max_autor.select { |persona| persona.second == maximo_a }
@tutores_maximo = max_tutor.select { |persona| persona.second == maximo_t }
@jurados_maximo = max_jurado.select { |persona| persona.second == maximo_j }
                                                    ⋮
#Calcular promedio de los seminarios
sem_ciencias=Documento.where("calificacion>0ANDtipo_documento_id=?",SEM).select("calificacion")
@promedio_SEM_ciencias=sem_ciencias.inject(0.0){|sum,nota|sum+nota.calificacion
}/sem_ciencias.size.to_f
                                                    ⋮
```

Finalmente para implementar el conteo de descargas de un documento se necesita crear una acción dentro del controlador documentos_controller.rb que permita incrementar en una unidad el contador que mantiene el número de descargas del documento solicitado y luego envíe este documento al usuario en lugar de una vista HTML.

Algoritmo 6.12 Conteo de descargas

```
# Acción Documento#descargar
def descargar
 if @documento.visibilidad.descripcion=="PUBLICO"||usuario_signed_in?
  unless @documento.descargas.nil?
   @documento.update_attributes(:descargas=>@documento.descargas+=1)
  else
   @documento.update_attributes(:descargas => 1)
  end
  send_file("#{Rails.root}/documentos/#{@documento.escuela.descripcion}/
             #{@documento.fecha_publicacion.year}/documento_#{@documento.id}.pdf",
             :filename=>"documento_#{@documento.id}",:type => 'application/pdf')
 else
 redirect_to '/busqueda_simple'
 end
end
```

Pruebas: Para la realización de las pruebas se procedió a indexar un conjunto de documentos y se realizó su recuperación en las vistas búsqueda simple, búsqueda avanzada y directorio. Asimismo se comprobó que el documento se pudiera descargar desde la ventana modal mostrada al usuario y desde el botón de descargas, y que adicionalmente el contador de descargas se incrementara en una unidad cada vez que ejecutásemos dicha acción. En la figura 6.17 podemos observar el documento asociado con las etiquetas *documento* y *prueba* indexado previamente siendo recuperado desde la interfaz de usuario pública de búsqueda sencilla usando el criterio Relevancia.

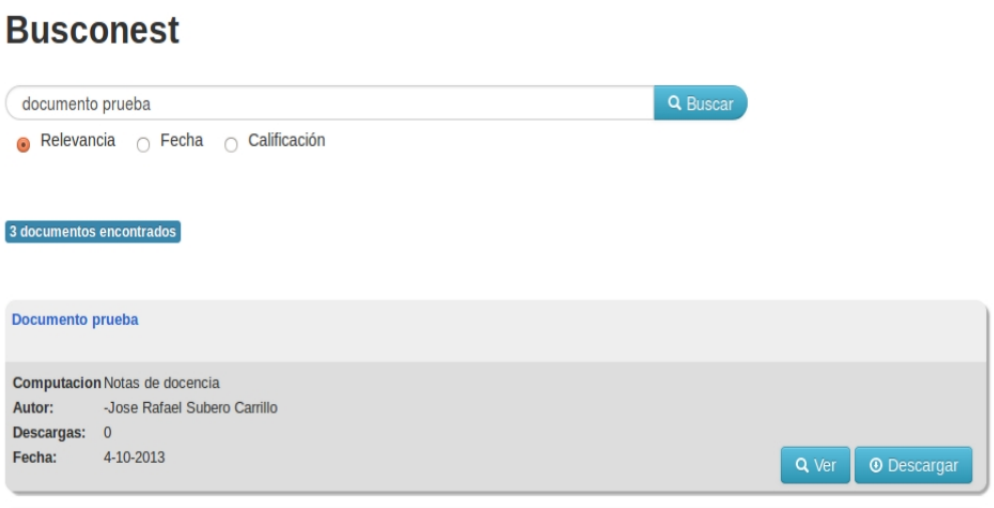


Figura 6.17: Recuperación por etiquetas

6.6. Iteración 5

Meta *Sprint:* Desarrollar una Interfaz que permita la carga de documentos por parte de un usuario autenticado en el sistema CONEST.

Tabla 6.6: Tabla de Control - Iteración 5

N#	Fecha	Tareas	Tipo
1	03/06/2013	Implementación Módulo CONEST	Interfaz CONEST
2	24/06/2013	Implementación Módulo BUSCONEST 2	Interfaz CONEST
3	01/07/2013	Implementación protocolo CORS	Interfaz CONEST
4	08/07/2013	Integración utilidad TinyMCE	Interfaz CONEST
5	15/07/2013	Consideraciones de seguridad	Interfaz CONEST

Planificación: Se requiere implementar un módulo en el sistema CONEST capaz capturar datos asociados al usuario autenticado en el momento de subir un documento T.E.G. Asimismo se requiere que se envíen estos datos junto con datos descriptivos del documento al sistema BUSCONEST 2. BUSCONEST 2 debe ser capaz de capturar los datos para generar su respectiva representación en el sistema.

Diseño: Una vez aprobado el T.E.G. se requiere que el graduando ingrese al sistema CONEST para cargar su respectivo documento T.E.G. El usuario debe llenar un formulario en dos etapas. En la primera etapa el estudiante indicará la siguiente información explícitamente en el formulario:

- Título
- Resumen
- Palabras clave
- T.E.G. individual/pareja

La información adicional requerida será extraída automáticamente de la base de datos del sistema CONEST y la misma será enviada mediante el empleo de campos ocultos en el formulario. Está información corresponde a:

- Autor
- Tutores
- Jurados
- Escuela

- Fecha de presentación
- Calificación
- Premios

La primera etapa del formulario puede observarse en la figura 6.18.

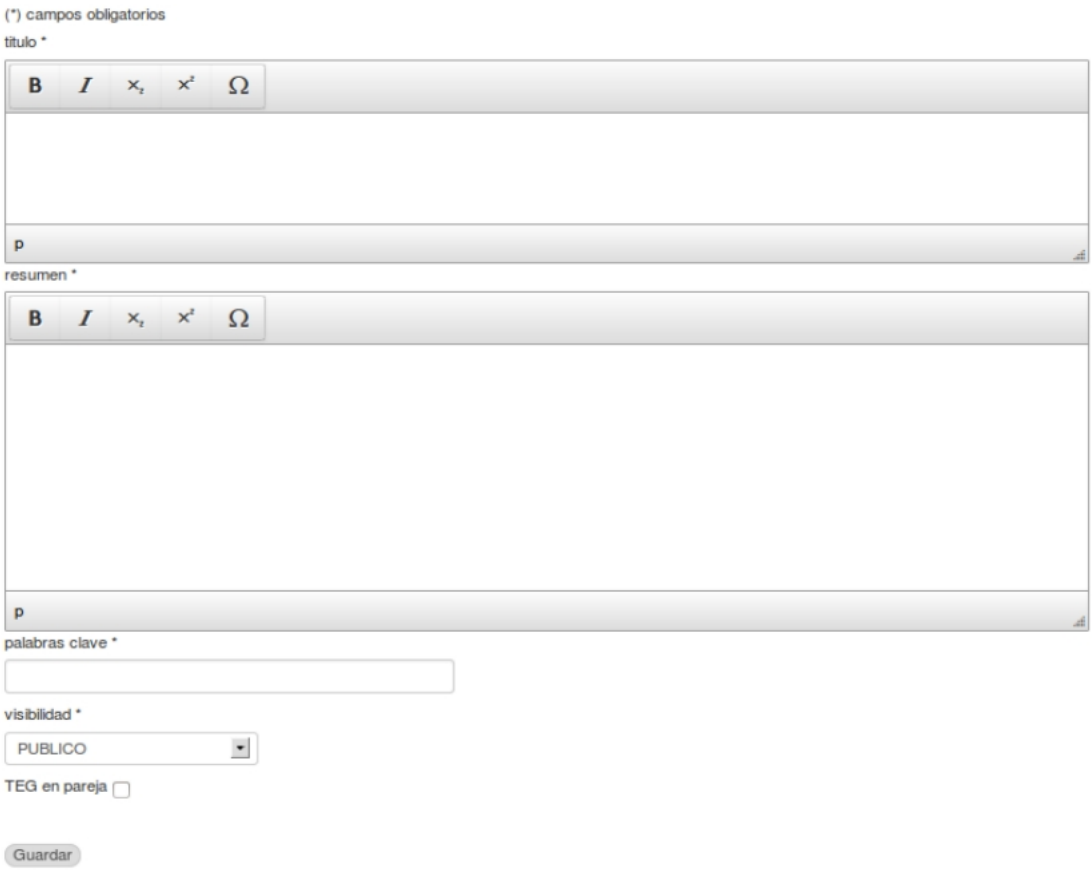


Figura 6.18: Formulario carga de datos T.E.G.

Luego de completado el ingreso de datos de la primera parte del formulario, el estudiante es dirigido a una segunda etapa donde se le solicita la carga del documento en formato PDF.

La segunda etapa puede observarse en la figura 6.19.

Figura 6.19: Formulario carga de documento

Luego de enviado el documento, el sistema CONEST debe cambiar el estado de la base de datos para notificar el hecho que el estudiante ya ha cargado los datos requeridos.

También se necesita que el el estudiante pueda almacenar el texto con un formato que permita el despliegue de subíndices, superíndices, letras griegas, números, texto en negrita, entre otros. Del mismo modo se requiere que se contemplen los casos en que dos alumnos presentan el mismo T.E.G. para evitar que el mismo documento sea cargado dos veces. También deben ser considerada la integridad de los datos durante el proceso de comunicación entre el sistema CONEST y BUSCONEST 2.

Implementación: Para implementar los requerimientos descritos se deben crear dos módulos que se comunican entre sí. Primeramente se desarrolló el módulo que se añade al sistema CONEST, el cual a partir de la cédula del estudiante almacenada en la sesión de usuario puede extraer los datos concerniente a la presentación del T.E.G. Un extracto del código se muestra a continuación.

Algoritmo 6.13 Módulo carga de T.E.G. - CONEST

```
# Recuperar estudiante_en_licenciatura de la sesión
@estudiante_en_licenciatura=session[:estudiante_en_licenciatura]
@cedula = session[:estudiante_en_licenciatura][0]

# Recuperar datos personales del estudiante
my_query = "SELECT primer_nombre, segundo_nombre, primer_apellido, segundo_apellido, correo
            FROM estudiante
            WHERE cedula LIKE #{@cedula}"

datos = ActiveRecord::Base.connection.execute(my_query).to_a.first
@nombres = datos[0]+' '+datos[1]
@apellidos = datos[2]+' '+datos[3]
@correo = datos[4]

my_query = "SELECT promedio_general, promedio_ponderado, eficiencia
FROM estudiante_datos_academicos
WHERE estudiante_cedula LIKE #{@cedula}"

datos = ActiveRecord::Base.connection.execute(my_query).to_a.first
@promedio_general = datos[0] @promedio_ponderado = datos[1] @eficiencia = datos[2]

# Recuperar la fecha de la planilla_individual (presentacion de TEG)
my_query = "SELECT fecha_presentacion
            FROM planilla_individual
            WHERE estudiante_cedula LIKE #{@cedula} AND es_entrega_tesis = 1"
datos = ActiveRecord::Base.connection.execute(my_query).to_a @fecha_presentacion = datos[0]
                                                    ⋮
```

El segundo módulo implica la creación de una acción adicional en el controlador de documentos en el sistema BUSCONEST 2 que permita crear el documento empleando los datos enviados desde el formulario generado en CONEST. Sin embargo durante el proceso de carga de documento también se debe contemplar el hecho de que no sólo se carga información respecto al documento en sí, sino también información relacionada con el documento pero que está almacenada en otras tablas de la base de datos. Por ejemplo, si un estudiante carga un documento cuyos tutores y jurados no están almacenados en BUSCONEST 2 entonces el sistema deberá crearlos y crear las asociaciones respectivas. Si los jurados y los tutores ya están el sistema sólo basta con crear las asociaciones pertinentes. Lo mismo aplica para el estudiante en sí. En caso de éxito el sistema BUSCONEST 2 realiza un *redirect* al módulo CONEST para que se despliegue la segunda etapa del formulario, en caso contrario el *redirect* vuelve solicitar la primera etapa del formulario.

En la segunda etapa del formulario se carga el archivo en formato PDF y el mismo se envía a BUSCONEST 2. La carga del documento implicó la creación de una barra de progreso que mediante la abstracción XMLHttpRequest realiza llamadas AJAX (*Asynchronous Javascript and XML*) al sistema BUSCONEST 2 para subir el documento por partes. Esto con la finalidad de evitar que la ventana del navegador se

quedara congelada si el documento a cargar tuviese un gran tamaño. Sin embargo debido a la Política del Mismo Origen presente en la mayoría de los navegadores el objeto XMLHttpRequest de la barra de progreso no podía realizar llamadas asíncronas a un origen[5] distinto al origen donde se encontraba alojado el *script* que implementa la barra. Es decir que el *script* que implementa la barra de progreso al encontrarse alojado en CONEST no podía realizar las llamadas asíncronas necesarias para cargar el documento en BUSCONEST 2.

Para poder flexibilizar esta política de seguridad el protocolo CORS (*Cross-Origin Resource Sharing*) permitió la realización de llamadas asíncronas a un dominio distinto pero con la condición de efectuar un intercambio de cabeceras HTTP previo conocido como solicitud *preflight*. En este intercambio de cabeceras el cliente se identifica a si mismo enviando una solicitud con la cabecera Origin <origen del cliente> en el mensaje HTTP para que el servidor verifique la identidad del cliente (en este caso el método HTTP empleado es OPTIONS). El servidor debe mantener una lista con todos los clientes permitidos (se puede utilizar el comodín *). Si el cliente está autorizado entonces se devuelve un mensaje HTTP en cuya cabecera se encuentra los métodos que puede realizar el cliente en el servidor y una cabecera de la forma Access-Control-Allow-Origin: <origen del cliente>. Seguidamente el navegador verifica esta información y procede a realizar la llamada asíncrona. A continuación se muestra el código en el sistema BUSCONEST 2 que permite la autorización del cliente ejecutando la barra de progreso.

Algoritmo 6.14 Implementanción CORS

```
# Cross-Site Resource Sharing (CORS)
before_filter :cors_preflight_check
after_filter :cors_set_access_control_headers

# Para todas las respuestas en este controlador, retornar las cabeceras de control de acceso CORS.
def cors_set_access_control_headers
    headers['Access-Control-Allow-Origin'] = '*'
    headers['Access-Control-Allow-Methods']='POST,GET,OPTIONS'
    headers['Access-Control-Max-Age'] = "1728000"
end

# Si esto es una solicitud preflight con el método OPTIONS, returna sólo las cabeceras necesarias y el
cuerpo en text/plain vacío
def cors_preflight_check
    if request.method == "OPTIONS"
        headers['Access-Control-Allow-Origin'] = '*'
        headers['Access-Control-Allow-Methods']='POST'
        headers['Access-Control-Max-Age']='1728000'
        render :text => '', :content_type => 'text/plain'
    end
end
```

[5]Combinación de esquema, *host* y puerto.

Para implementar el caso que el T.E.G. haya sido realizado en pareja el estudiante debe marcar el *checkbox* correspondiente a esta situación e indicar el nombre de su compañero. Seguidamente se efectúa el mismo procedimiento que para el caso individual pero se asocia un autor adicional al documento. Una vez cargado el documento el módulo CONEST debe registrar esta situación en su base de datos para evitar que se cargue el documento dos veces.

El título del T.E.G. debe también ser almacenado en la base de datos de CONEST además de ser enviado junto al resto de los datos al sistema BUSCONEST 2. La forma de representación del título debe soportar el despliegue de caracteres y formatos especiales. La herramienta TinyMCE transforma un campo TEXTAREA o similar de un documento HTML en un editor de texto configurable con las características requeridas. Una vista del editor se muestra en la figura 6.20.

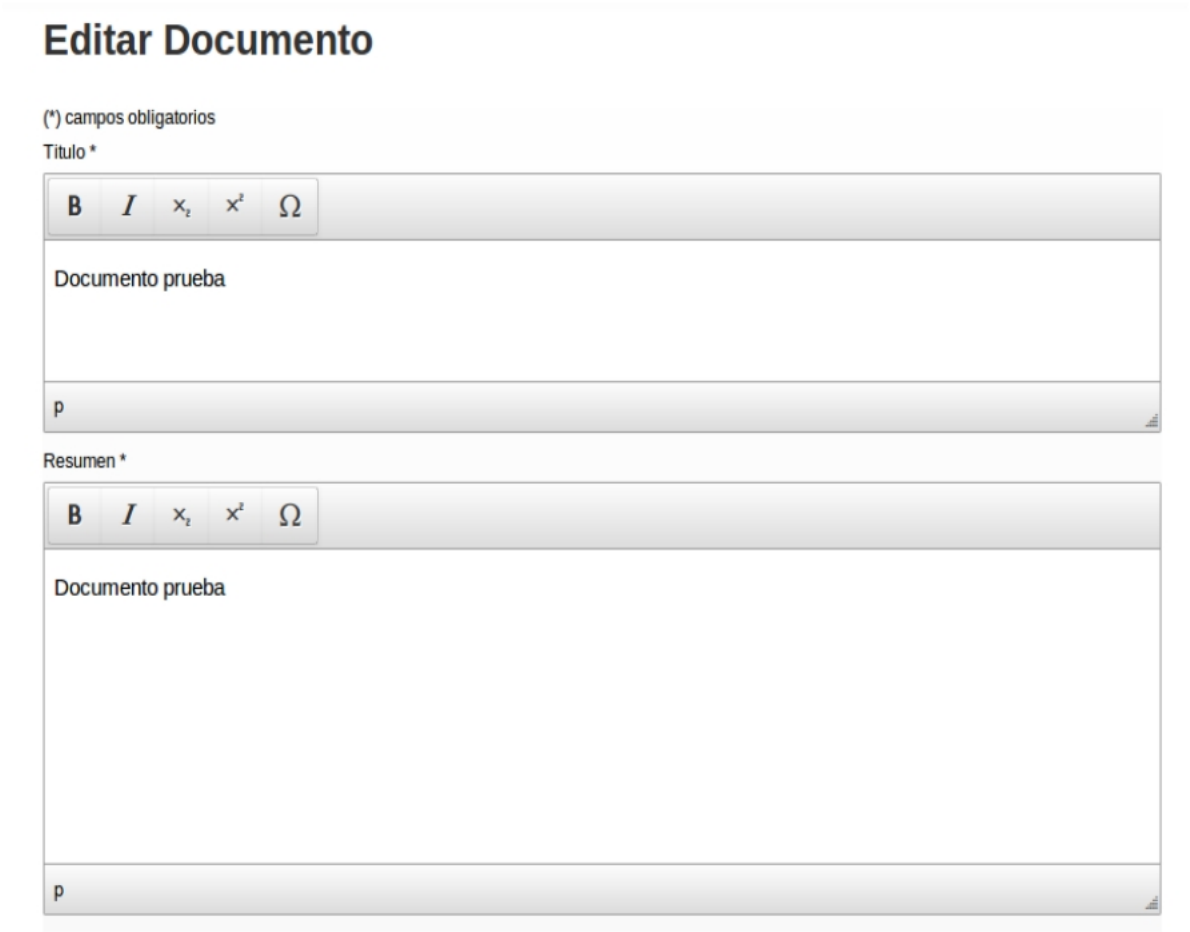


Figura 6.20: Editor de texto para el título del T.E.G.

Para garantizar la integridad y autenticidad de los datos se concatenó los datos suministrados en la primera etapa del formulario con una frase previamente compartida entre CONEST y BUSCONEST 2 generando un nuevo *string*. La idea es que en el lado del emisor (CONEST) se calcule el valor MD5 (*hash*) de este *string* y se envíe el mismo de forma oculta en el formulario. En el lado del receptor (BUSCONEST 2) se vuelven a efectuar la concatenación de los valores enviados por el emisor y la frase previamente compartida para efectuar el respectivo cálculo del valor MD5. Si este valor *hash* coincide con el valor *hash* enviado de forma oculta en el formulario es porque los datos no fueron cambiados en tránsito y el emisor está en conocimiento de la frase compartida, garantizándose de esta forma no sólo la integridad sino también la autenticidad de los datos. Nótese que aunque una tercera parte pueda obtener el valor *hash* en tránsito (por ejemplo con un ataque tipo hombre en el medio) este *hash* es sólo válido para esa combinación de datos. Por tanto cambiar los datos y enviar el mismo *hash* resultaría en un rechazo de parte del receptor porque se necesita la concatenación de los valores *y la frase compartida* para generar el *hash* correspondiente.

La generación del valor *hash* del lado del emisor y comprobación del *hash* del lado del receptor (BUSCONEST 2) se muestra a continuación:

Algoritmo 6.15 Generación y comparación de valores MD5

```
salt="aqui_debe_haber_un_string_compartido_aleatorio"
hash = Digest::MD5.hexdigest(estudiante+datos_comunes+tutor1+jurado1+jurado2+salt)

# verificar si es CONEST quien manda la solicitud
if hash == params[:hash]
  #Crear Personas y Documentos
else
  #Devolver mensajes de error
end
```

Pruebas: Para la realización de las pruebas se cargaron un conjunto de documentos de contenido conocido desde la interfaz de carga de T.E.G. de CONEST. Sin embargo, como ya se ha mencionado anteriormente en el apartado de desarrollo, debido a que la aplicación CONEST se encuentra en un origen distinto (en este caso 127.0.0.1:3000) a la aplicación BUSCONEST 2 (127.0.0.1:3001) las llamadas Javascript asíncronas en principio se encuentran deshabilitadas por la Política del Mismo Origen del Navegador. Sin embargo, al implementando el protocolo CORS en el módulo de cargas de T.E.G. en CONEST se flexibilizó dicha política. En la figura 6.21 se puede observar el módulo en cuestión realizando la petición *preflight* de solicitud de permisos a la aplicación BUSCONEST 2 mediante el método OPTIONS. Seguidamente se observa que dicha solicitud es respondida con código de éxito 200 OK con lista de las operaciones permitidas en el servidor para ese cliente.

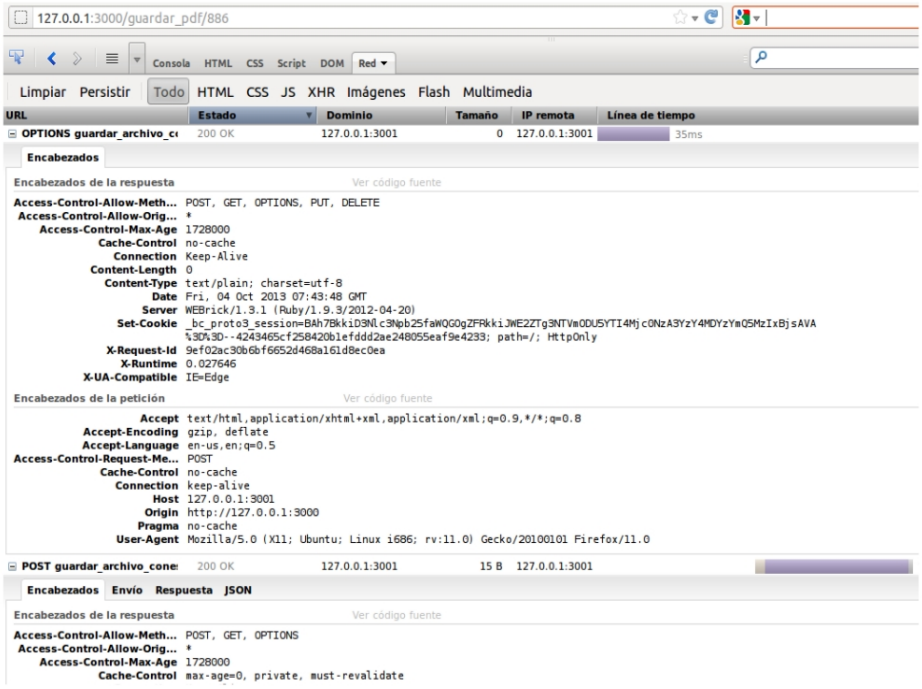


Figura 6.21: Solicitud *preflight*

Finalmente se verificó que los formularios no permitiesen el ingreso de datos incorrectos en sus campos, como por ejemplo un T.E.G. sin título, resumen o palabras claves.

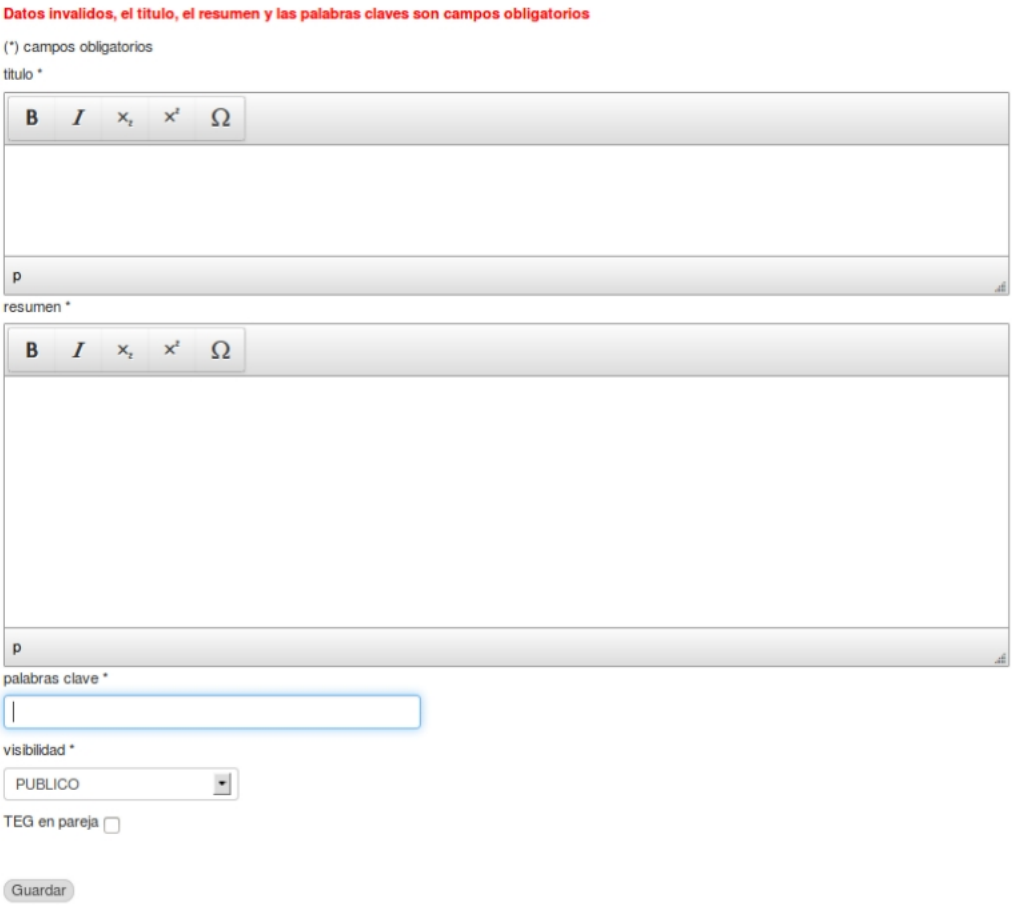


Figura 6.22: Prueba: Verificación Formulario

6.7. Iteración 6

Meta *Sprint:* Migrar los datos contenidos en el repositorio BUSCONEST 1 al sistema BUSCONEST 2.

Tabla 6.7: Tabla de Control - Iteración 6

N#	Fecha	Tareas	Tipo
1	22/07/2013	Análisis de Bases de Datos: CONEST y BUSCONEST 1	Migración de datos
2	29/07/2013	Implementación y ejecución de *script* de migración	Migración de datos

Planificación: Para que el sistema BUSCONEST 2 cumpla su función principal como plataforma que facilita la difusión de la producción intelectual académica de la Facultad de Ciencias, se hace necesario que antes de su salida a producción BUSCONEST 2 cuente con los documentos previamente indexados en el sistema BUSCONEST 1. Sin embargo los documentos en formato PDF almacenados en BUSCONEST 1 están representados como columnas tipo LONGBLOB dentro de una tabla, y la información descriptiva de los mismos (los metadatos) están también almacenados en formato XML en una sola columna tipo LONGTEXT como se puede observar en la figura 6.3 en la página 106. Adicionalmente la información representada en estos metadatos no es suficiente para completar todos los datos solicitados por el sistema BUSCONEST 2 (por ejemplo BUSCONEST 2 exige asociar los autores, tutores y jurados pero BUSCONEST 1 no almacena esa información) por tanto se hace necesario consultar también al sistema CONEST directamente para lograr extraer los metadatos requeridos.

Diseño: Existen varios aspectos importantes a considerar al momento de realizar la migración de datos. El primero consiste en tomar en cuenta que no todos los datos se encuentran en la base de datos del sistema BUSCONEST 1. De hecho, de esa base de datos sólo se necesita extraer el número de cédula de los autores, el título, el resumen, y las palabras clave del T.E.G. y por supuesto el documento PDF. Los demás datos se encuentran almacenados en la base de datos del sistema CONEST. Es realmente en la base datos de CONEST donde se extraen los datos académicos de los autores, los datos personales de los autores, tutores y jurados, los reconocimientos académicos y en general la mayoría de la información faltante necesaria para completar los campos restantes en las tablas de la base de datos de BUSCONEST 2.

El segundo aspecto a tomar en cuenta es que BUSCONEST 2 almacena la información relativa a las personas y estudiantes en tablas distintas (ver figura 6.6 en la página 111) por tanto cada vez que se migra exitosamente un documento también se debe crear la persona y el estudiante asociados al documento PDF, pero en caso de que ya exista esa persona o el estudiante entonces se debe crear solamente las asociaciones para evitar duplicados inconsistentes. Para el caso de los autores, tutores, jurados solo basta con crear la asociación documento-persona en la tabla respectiva (ver esquemas en la figura 6.5 en la página 110).

Implementación: La implementación del *script* de migración se llevó a cabo implementando una aplicación específica Ruby on Rails que abre tres conexiones hacia tres bases de datos distintas. Es decir el *script* se encuentra almacenado en un modelo ActiveRecord que al ser invocado extrae los datos y los documentos PDF del sistema BUSCONEST 1 y los datos restantes del sistema CONEST. Luego, el *script* formatea dichos datos y crea otros predeterminados para posteriormente almacenar toda los datos procesados en la base de datos de BUSCONEST 2. Sin embargo, los metadatos almacenados en la tabla documento de la base de datos BUSCONEST 1 se encuentra representados en formato XML, por tanto el *script* precisó de un analizador sintáctico (*parser*) que permitió la extracción de los datos contenidos en el documento XML. El algoritmo 6.16 muestra un extracto de código del *script* de migración, con la invocación del *parser* XML.

Algoritmo 6.16 *Script* de migración con *Parser* XML

```
def self.migrar_TEG
  # Conexión con la base de datos de BUSCONEST 1
  # Extraer los ids de todos los documentos
  busconest_1_darabase = ActiveRecord::Base.establish_connection "development"
  result = busconest_1_database.connection.execute("SELECT id FROM documento")
  ids = result.to_a
  # GRAN LOOP (desde el primer documento hasta el ultimo)
  ids.each do |id|
    doc_id = id.first
    # Conexion con la base de datos de BUSCONEST 1
    # Parsear el XML con la data del documento
    busconest_1_database = ActiveRecord::Base.establish_connection "development"
    result = busconest_1_database.connection.execute("SELECT metadata FROM documento
WHERE id=#{doc_id}")
    xml_data = result.to_a.first[0] unless xml_data.nil?
    teg = XmlSimple.xml_in(xml_data)
    autores_persona_id = []
# CREAR EL PRIMER GRADUANDO ...
                                        ⋮
```

Pruebas: Las pruebas del *script* consistieron en ejecuciones sucesivas del mismo en las cuales se iban migrando lotes de documentos y luego desde la interfaz de administración de BUSCONEST 2 se inspeccionaban los mismos para corroborar la coherencia de los datos migrados y que estos correspondieran con el documento PDF. Cuando el *script* consigió migrar exitosamente un total de 793 documentos de una muestra de 802 se consideró finalizado.

Aquí concluye la **Parte II - Marco Aplicativo** y el capítulo **Desarrollo del Sistema BUSCONEST 2**. A continuación se presentan los capítulos **Conclusiones** y **Recomendaciones** respectivamente.

Capítulo 7

Conclusiones

Para la Facultad de Ciencias la Recuperación de Información representa una valiosa herramienta que permite potenciar la misión de dicha facultad, la cual es la producción y difusión de la producción intelectual académica, ya que docentes, estudiantes e investigadores pueden consultar los documentos digitales generados por miembros de la Facultad de Ciencias al realizar consultas al sistema, y del mismo modo, se puede hacer recuperable su producción intelectual haciendo disponible a los demás miembros de la comunidad los documentos respectivos.

También, la disponibilidad de un repositorio central de documentos permite apreciar el volumen de la producción intelectual académica generada por los miembros de la Facultad de Ciencias. Más aún, esta colección de documentos generan a su vez datos adicionales que pueden ser visualizados por el administrador para recabar información adicional relacionada con los documentos que de otro modo sería desconocida. Por ejemplo: cantidad de documentos por escuela, por tipo de documento, por año de publicación, cantidad de documentos totales en el repositorio, promedio de notas de T.E.G., cantidad de T.E.G. organizadas por rango de notas, autores, tutores y jurados asociados a mayor cantidad de documentos.

Cabe destacar que la disponibilidad de un repositorio de publicaciones digitales basado en un Motor de Búsqueda accesible desde la *web* aumenta la difusión de los documentos indexados ya que los Motores de Búsqueda modernos pueden agregar en sus índices las páginas *web* que permiten la visualización y descarga de dichos documentos. El estudiante o docente puede consultar el repositorio desde cualquier lugar con acceso a la *web*, en lugar de ir físicamente a una biblioteca consultando cotas y autores para encontrar los documentos relacionados con la información buscada.

Adicionalmente proveer de varias formas de navegación aumenta las posibilidades que un documento en particular sea recuperado cuando sólo se tiene un conocimiento parcial del mismo (sólo se conoce el autor, el año de publicación o algunas palabras relacionadas a su contenido).

La incorporación de tecnologías actuales ayudan en gran medida en a éste propósito, por ejemplo es más rápido visualizar el resumen descriptivo de un documento en una ventana modal o en la lista de resultados que realizar consultas sucesivas para descargar documentos y luego leer sus respectivos resúmenes.

Por otro lado si el usuario tiene sólo un conocimiento parcial del documento podría navegar por el directorio por escuela, por tipo de documento o por año de publicación, o podría consultar la búsqueda avanzada

ingresando los datos que conoce, descubriendo a su vez nuevos documentos relacionados. También se provee de una característica de Ordenamiento (*Ranking*) que permite ubicar en las primeras posiciones aquellos documentos con mayor cantidad de ocurrencias entre los términos de la consulta y su título, resumen, contenido y palabras clave, pero también se toman en cuenta datos específicos generados por los miembros de la comunidad académica de la Facultad de Ciencias, datos tales como: la calificación, que es asignada por el personal docente pertinente (el los documentos que aplique) luego de un proceso de evaluación específico, la fecha de publicación, que nos indica cuando el documento comenzó a estar disponible a los miembros de la comunidad académica, y la cantidad de descargas que permite estimar que tanto a sido recuperado un documento. Estos datos serían descartados por un Motor de Búsquedas de propósito general porque no son generados por la estructura del documento en sí.

La incorporación de una interfaz de administración que provee un funcionalidad de carga de documentos interactiva y una funcionalidad de indexado de documentos por lotes no interactiva contribuyen a la automatización del procedimiento de alimentar de forma constante el repositorio, manteniéndose la información actualizada y organizada. Asimismo, la conexión con el sistema CONEST permite mantener el repositorio de BUSCONEST 2 alimentado periódicamente con los últimos T.E.G. realizados en la Facultad de Ciencias. Dicha conexión se implementó con el protocolo CORS (*Cross-Origin resource sharing*), protocolo que flexibiliza la Política del Mismo Origen (*Same-origin policy*) sin comprometer la seguridad de los sistemas en comunicación.

El proceso de migración de datos también añade un valor agregado al sistema BUSCONEST 2 porque ahora se encuentran incorporados al repositorio de BUSCONEST 2 los documentos que habían sido publicados anteriormente en el repositorio de BUSCONEST 1. De hecho, desde la interfaz de administración se puede cargar interactivamente cualquier documento publicado antes de la puesta de producción de BUSCONEST 1 (semestre I-2008), aumentándose de esta forma la colección de documentos potencialmente disponibles en los índices de BUSCONEST 2.

La elección de una plataforma de *software* equipada con una gama de herramientas y plantillas facilita la tarea del desarrollador de mantener los componentes de *software* ordenados, al mismo tiempo que acelera el proceso de crear características nuevas. Por ejemplo, Ruby on Rails provee de mecanismos de generación de formularios, mantenimiento de sesión, autenticación, analizador sintáctico XML, generador de plantillas, *validators* y *callbacks* para este propósito. Bootstrap, tecnología basada en Javascript y CSS utilizada del lado del cliente, incorpora ventanas modales, barras de progreso, sistema de autocompletado de formularios y paginación de resultados.

Finalmente, el empleo de un Método de Desarrollo de *Software* ágil facilita la construcción de un sistema de *software* con requerimientos cambiantes, o conocidos sólo parcialmente al comienzo del desarrollo. Sin embargo, el método de desarrollo SCRUM plantea explícitamente sólo los artefactos relacionados con la planificación temporal de las tareas, y la forma en que los miembros del equipo cooperarán entre sí. Por ello se hace necesario definir artefactos específicos adicionales que reflejen las actividades específicas realizadas para llevar a cabo las tareas que componen cada una de las iteraciones, en el presente T.E.G. las actividades contempladas fueron: planificación, diseño, implementación y pruebas.

Capítulo 8

Recomendaciones

- Diseñar nuevas visualizaciones para los datos mostrados en el *Dashboard*, por ejemplo: gráficos de torta, histogramas, gráficos de líneas. Calcular nuevos datos relevantes tales como histórico de notas generales, cantidad de documentos generados semestralmente, promedio de notas por semestre. Estos datos podrían permitir una estimación más fina del volumen de la producción intelectual académica generada en la Facultad de Ciencias.
- Permitir el almacenamiento de referencias para generar nuevas tipos de búsqueda aparte de la búsqueda simple, búsqueda avanzada y directorio temático. Por ejemplo se podría visualizar un grafo de documentos relacionados por referencias o por cercanía de autores.
- Implementar soporte para otros formatos de documentos.
- Implementar mecanismos que permitan la interconexión del sistema BUSCONEST 2 con bibliotecas digitales externas mediante el empleo de protocolos y formatos estandarizados. Por ejemplo: OAI-PMH (*Open Archives Initiative-Protocol Metadata Harvesting*)[25] y el formato de descripción de metadatos de documentos *Dublin Core*[26].

Referencias

[1] Langville, A., y Meyer, C. *Google's PageRank and Beyond: the Science of Search Engine Rankings.* Princeton, N.J.: Princeton University Press, 2012.

[2] Manning, C. Prabhakar Raghavan, y Hinrich Schütze. *Introduction to Information Retrieval.* New York: Cambridge University Press, 2008.

[3] Belew, R. *Finding Out About: A Cognitive Perspective On Search Engine Technology and the WWW.* Cambridge University Press, 2008.

[4] Rijsbergen, J. *Information Retrieval.* 2da ed. London: Butterworth-Heinemann, 1979.

[5] Berry, M., y Browne M. *Understanding Search Engines: Mathematical Modeling and Text Retrieval.* 2da ed. Philadelphia, PA.: SIAM, Society for Industrial and Applied Mathematics, 2005.

[6] López, A. *Modelos de sistemas de recuperación de información documental basados en información lingüística difusa.* Tesis doctoral. Universidad de Granada, 2006

[7] Goker, A., and Davies, J. *Information Retrieval: Searching in the 21st Century.* John Wiley and Sons, Ltd. , noviembre 2009

[8] TREC. (2010, agosto 10). En *Text REtrieval Conference Overview.* Recuperado abril 20, 2012, en http://trec.nist.gov/overview.html

[9] ACM SIGIR. (2012, abril 4). En *ACM SIGIR Special Interest Group on Information Retrieval Home Page.* Recuperado abril 20, 2012, en http://www.sigir.org/

[10] Voorhees, E. y Harman, D., *Text REtrieval Conference.* Gaithersburg, Maryland: Department of Commerce, National Institute of Standards and Technology, noviembre 9-11 1998.

[11] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A. & Wiener J. (2003). *Graph structure in the web.*

[12] Sánchez, J., *Elaboración de un prototipo de buscador de documentos académicos de la Facultad de Ciencias* (Trabajo Especial de Grado). Enero, 2008.

[13] Schwaber, K. y Sutherland J., *The Scrum Guide* (2011). En Read the SCRUM guide. Recuperado 09 abril de 2012 en https://www.scrum.org/.

[14] Anlehnung, A., *Searching the web*, ACM Transactions on Internet Technology, Vol. 1/1, Agosto 2001, S. 4

[15] Google Support. (2012, enero). En *About sitemaps - Webmaster's tools help.* Recuperado abril 25, 2012, en http://support.google.com/webmasters/bin/answer.py?hl=en&answer=156184

[16] Huuhka, P. (2007). *Google: data structures and algorithms.* Helsinki, Finlandia: University of Helsinki.

[17] Brin, S., Page, L. (1998). *The anatomy of a large-scale hypertextual web search engine.* Computer Networks and ISDN Systems 30: 107–11

[18] VIC Consulting. (2010). *Google PageRank explained.* En Google PageRank general description. Recuperado abril 28, 2012, from http://www.vicconsult.com/seo/

[19] Spoerri, A. (1995). *InfoCrystal: a visual tool for information retrieval.* Tesis doctoral, Massachusetts Institute of Technology, Cambridge.

[20] Rivas, S., Villapol, M., y Zambrano, J. (2010). *BUSCONEST: Una aplicación para búsquedas de documentos académicos electrónicos.* Caracas, Venezuela: Universidad Central de Venezuela.

[21] Google. (2012). *About Google Scholar.* En Google Scholar. Recuperado junio 10, 2012, en http://scholar.google.com/intl/es/scholar/about.html

[22] Microsoft Research. (2012). *Overview.* En Academic help center. Recuperado junio 11, 2012, en http://academic.research.microsoft.com/About/Help.htm

[23] Elsevier. (2012). *About Scirus.* En About us. Recuperado junio 11, 2012, en http://www.scirus.com/srsapp/aboutus/

[24] Universia. (2012). *Ayuda.* En Biblioteca.Net. Recuperado junio 14, 2012, en http://biblioteca.universia.net/ayuda.htm

[25] Open Archives Initiative. (2013). *Standards for Web Content Interoperability.* En Home. Recuperado octubre 04, 2013, en http://www.openarchives.org/pmh/

[26] Dublin Core Metadata Initiative. (2013). *The Dublin Core Metadata Initiative.* En Home. Recuperado octubre 04, 2013, en http://dublincore.org/