UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Desarrollo de una Aplicación Web para la Biblioteca Alonso Gamero que permita el almacenamiento, gestión y búsqueda de la Producción Intelectual de la Facultad de Ciencias de la Universidad Central de Venezuela

Trabajo Especial de Grado presentado ante la Ilustre Universidad Central de Venezuela por los Bachilleres

Bolívar Morín Juan Carlos C.I. 16924510
Páez Román César Andrés C.I. 16988987

para optar por el título de Licenciado en Computación

Tutora:

Prof. Ospina Mercy

Octubre.2010

ACTA

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado titulado “Desarrollo de una Aplicación Web para la Biblioteca Alonso Gamero que permita el almacenamiento, la gestión y búsqueda de la Producción Intelectual de la Facultad de Ciencias de la Universidad Central de Venezuela” y presentado por los Brs. César Páez de C.I. 16.988.987 y Juan Bolívar de C.I. 16.924.510, a los fines de optar al título de **Licenciado en Computación**, dejamos constancia de lo siguiente:

Leído como fue dicho trabajo, por cada uno de los miembros del jurado, se fijó el día ____ de Octubre de 2010, a las horas, para que los autores lo defendieran en forma pública, lo que estos hicieron en _____ de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobar con la nota de ____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día ___ de _____ de ____.

_____

Profa. Mercy Ospina

(Tutora)

_____ _____

Profa. Yusneyi Carballo

(Jurado Principal)

Profa. Carmen Marrero

(Jurado Principal)

No hay palabras para describir el gran sentimiento de agradecimiento por aquellas personas que me apoyaron, Pina, Chuo, mis hermanos, mis amigos, mi novia, mi gran compañero de tesis y sobre todo a Mercy, por su confianza y su apoyo constante….

Gracias Eternas

Juan Bolívar

A mi familia, en especial a mi grandiosa mamá que tanto se ha preocupado y ha luchado por mis hermanos y por mí. A mi amor flor que siempre estuvo a mi lado dándome apoyo y ánimos para seguir adelante. Muchas gracias a Mercy por ayudarnos y depositar su confianza en nosotros, sin ella lograr el objetivo no hubiese sido posible. Al Prof. Sergio por ayudarnos cuando lo necesitamos.

A la Profesora Carmen Marrero por su dedicación y tiempo ofrecido, y por último pero no menos importante a Juan por su trabajo, amistad incondicional y dedicación.

César Páez

Resumen

El siguiente Trabajo Especial de Grado aborda la localización y el difícil acceso que tiene la Producción Intelectual de la Facultad de Ciencias de la Universidad Central de Venezuela, más específicamente de los Trabajos Especiales de Grado (T.E.G.) de pregrado. En el marco teórico se explican los conceptos asociados a la gestión del conocimiento y su relación con las aplicaciones Web, el formato de documento electrónico más común y los motores de búsqueda que manipulan estos documentos electrónicos. En el marco aplicativo se muestran las etapas del desarrollo de una aplicación Web con un motor de búsqueda embebido que gestiona los T.E.G. de la Facultad de Ciencias, utilizando el método de desarrollo de software Programación Extrema (XP).

El sistema desarrollado realiza una integración con el sistema CONEST para aprovechar los datos allí almacenados y los procesos de validación de usuarios, y así poder alcanzar y cubrir por completo los requerimientos de los usuarios de éste sistema al que se le denominará BUBAG.

Palabras clave: Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación, Biblioteca Alonso Gamero, Gestión de Documentos, Aplicación Web, Búsqueda de Documentos Electrónicos, Ruby on Rails, Programación Extrema (XP), BUBAG.

Índice de Contenido

Índice de Figuras

Ejemplos de Código

Índice de Tablas

Introducción

En la Facultad de Ciencias de la Universidad Central de Venezuela, como en muchas universidades reconocidas a nivel mundial, se desarrolla una gran cantidad de investigaciones importantes, las cuales, si el investigador no publica en medios tales como congresos o revistas solo son conocidas de manera local, tal es el caso de los trabajos especiales de grados que son almacenados en la Biblioteca Alonso Gamero (B.A.G.) solamente en forma física, para su consulta. Las publicaciones de estos trabajos en medios electrónicos no es constante y está muy dispersa en las páginas de la facultad, lo que hace difícil para los investigadores tanto de la facultad como externos a esta conocerlos.

Para que la actividad investigativa de cualquier universidad sea eficiente necesita estar a la vanguardia en el manejo del conocimiento, por lo que es necesario conocer las investigaciones relacionadas, sin lo cual además se puede repetir trabajo de manera innecesaria, y hacer uso de las tecnologías presentes para la publicación y gestión del conocimiento, entre ellas destacan las tecnologías del área Web, por permitir el acceso general a la información.

El objetivo de este Trabajo Especial de Grado es el desarrollo de una aplicación Web que posea un motor de búsqueda embebido para la Biblioteca Alonso Gamero, la cual se encargará de almacenar y gestionar la Producción Intelectual de la Facultad de Ciencias de la Universidad Central de Venezuela, más específicamente de los Trabajos Especiales de Grado, permitiendo consultarlas a través del motor de búsqueda.

El presente documento se encuentra estructurado de la siguiente manera:

En el capítulo I, se describe el Problema de Investigación donde se exponen y explican las dificultades que se presentan al momento de acceder a los trabajos

de investigación en la Facultad de Ciencias, así como el objetivo general y los objetivos específicos de este Trabajo Especial de Grado, su justificación y alcance.

En el capítulo II, se presentan las bases que fundamentan el desarrollo de este trabajo. Se define Gestión del conocimiento, destacando sus características y la forma de trabajar basado en la gestión del conocimiento. Además se describe las tecnologías usadas para el desarrollo de la aplicación Web. También se definen los formatos de documentos más usados para almacenar electrónicamente los trabajos de investigación, se describe detalladamente Ruby on Rails y los conceptos fundamentales de los tipos de Bases de Datos.

En el capítulo III, se describe como se realizó la adaptación de la Programación Extrema (XP) para el desarrollo de la aplicación planteada, haciendo énfasis en las fases de planificación, diseño, programación y pruebas que se encuentran dentro de cada iteración realizada.

Finalmente las referencias bibliográficas y digitales, las conclusiones y recomendaciones presentes en éste trabajo.

Capítulo I
Planteamiento del Problema

En este capítulo se describe el contexto del problema, luego se expone el objetivo general, objetivos específicos, metodología para el desarrollo de la solución, el alcance, las tecnologías y plataformas de esta investigación, así como la justificación de realizar una aplicación Web con un motor de búsqueda embebido.

1.1 Planteamiento del Problema

Al realizar un análisis de los inconvenientes actuales en cuanto a la publicación y manejo de los documentos, producto de las investigaciones, en medios electrónicos en la Facultad de Ciencias de la Universidad Central de Venezuela se observa que la información es poca y se encuentra muy dispersa, es decir, para encontrar documentos de algún tema de interés se debe navegar por incontables páginas pertenecientes a los centros de información de las Escuelas de la Facultad de Ciencias de la Universidad Central de Venezuela, y en muchos casos, según investigadores consultados, la búsqueda resulta infructuosa.

Otro inconveniente de suma importancia es la falta de información sobre los trabajos que han sido desarrollados, lo que ocasiona que si algún investigador requiere alguna información sobre éstos le resulte complejo conseguirla, por ejemplo, debe buscar a los profesores que han sido tutores o investigadores de los trabajos para poder consultar un área específica desarrollada, lo cual genera pérdida de tiempo y esfuerzo para el investigador, también puede ocurrir que varios proyectos muy similares estén siendo desarrollados al mismo tiempo y no se tenga conocimiento alguno de esa situación.

Adicionalmente no hay una herramienta en la Facultad que facilite la publicación (por parte del autor), la administración, la búsqueda y la difusión del Producto Intelectual[1].

Para conocer el alcance que puede tener el problema planteado, se consultó un estudio realizado en el 2009 por Ranking of World's Universities (ranking de universidades del mundo) cuyo objetivo es medir la visibilidad en Internet de las investigaciones realizadas por las Universidades a nivel mundial para lo cual se utilizaron cuatro (4) parámetros, descritos a continuación.

- **Tamaño**, número de páginas recuperadas desde Google, Yahoo, Live Search and Exalead.

- **Visibilidad**, el número total de enlaces externos únicos recibidos por un sitio, que puede ser sólo obtenido confidencialmente desde yahoo search.

- **Archivos**, después de la evaluación de la importancia académica de las actividades de publicación y considerando el volumen de diferentes formatos de archivos, fueron seleccionados Adobe Acrobat (.pdf), Adobe PostScript (.ps), Microsoft Word (.doc) y Microsoft PowerPoint (.ppt)., esos archivos fueron extraídos usando Google, Yahoo Search, Live Search y Exalead.

- **Scholar**, Google Scholar provee el número de publicaciones y citas por cada dominio académico, esos resultados desde la base de datos de Scholar representan publicaciones, reportes y otros datos académicos.

Este estudio coloca a la Universidad Central de Venezuela con poco visibilidad,

[1] (Biblioteca Alonso Gamero). Documentos o trabajos de investigación producidos por los investigadores de la Facultad de Ciencias de la Universidad Central de Venezuela que genera nuevo conocimiento en un área específica.

ésta se encuentra en el puesto noventa y dos (92/100) a nivel latinoamericano y mil cuatrocientos diez y seis (1416/8000) a nivel mundial (Ver Figura 1), esto es una consecuencia de la falta de divulgación de la información sobre los productos de las investigaciones realizadas en la Universidad, y de que sus conocimientos son solo locales. Existe poca visualización nacional e internacional de las publicaciones de la Universidad Central de Venezuela.

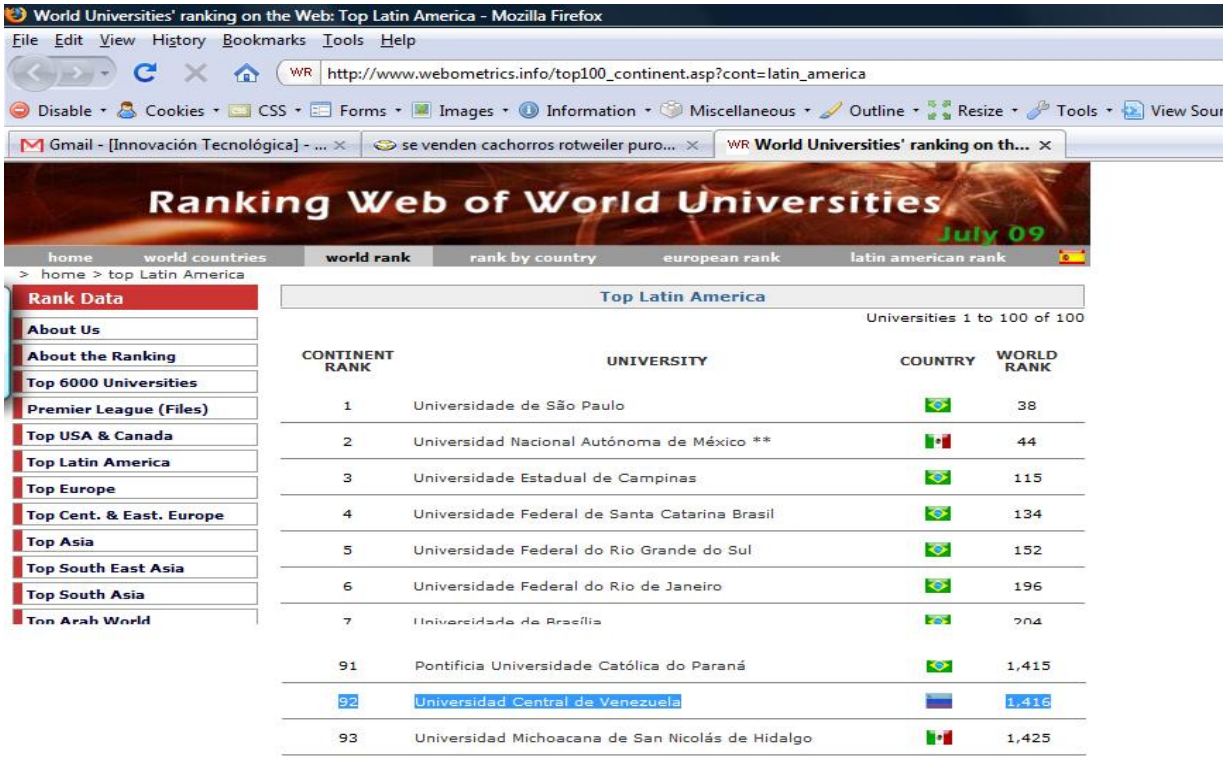

Figura 1. Ranking Mundial de Universidades

La propuesta consiste en desarrollar una aplicación Web que será llamada BUBAG, que permita el almacenamiento, gestión y búsqueda de las publicaciones producto de las investigaciones de la Facultad de Ciencias, específicamente de los Trabajos Especiales de Grado de Pregrado, con un motor de búsqueda embebido

de documentos electrónicos. Esta aplicación estará integrada con CONEST[2] debido a que este sistema ya cuenta con los datos de la comunidad (estudiantes de la Facultad de Ciencias que hayan presentado y aprobado su Trabajo Especial de Grado de Pregrado) que podrá publicar en BUBAG, lo cual permite lograr una solución al problema planteado.

1.2 Objetivo General

Desarrollar una Aplicación Web para la Biblioteca Alonso Gamero (B.A.G.) que permita el almacenamiento, gestión y búsqueda de la Producción Intelectual contenida en las Tesis de Pregrado de la Facultad de Ciencias de la Universidad Central de Venezuela.

1.3 Objetivos Específicos

1. Analizar los requerimientos de los usuarios de la Biblioteca Alonso Gamero (B.A.G.) en cuanto a la recuperación de la información de los Trabajos Especiales de Grado de Pregrado.
2. Estudiar el funcionamiento del sistema BUSCONEST[3] y su relación con BUBAG.
3. Diseñar e implementar un modelo de datos que permita almacenar la información de la Producción Intelectual de la Facultad de Ciencias.
4. Utilizar un método de desarrollo de software ágil e implementar el sistema en Ruby on Rails.

[2] CONEST (Control de Estudio): Sistema de Gestión Académica de la División de Control de estudio de la Facultad de Ciencias de la Universidad Central de Venezuela.

[3] BUSCONEST (Control de Estudio): Es una aplicación Web dependiente de CONEST la cual ofrece a cualquier usuario la posibilidad de ubicar y consultar trabajos académicos realizados por la comunidad de la Facultad de Ciencias de la Universidad de Venezuela.

5. Desarrollar un módulo para el almacenamiento, gestión y búsqueda de la Propiedad Intelectual de la Facultad de Ciencias.
6. Desarrollar un módulo para consultar en texto completo la Producción Intelectual de la Facultad de Ciencias.
7. Realizar pruebas del sistema, principalmente pruebas de aceptación.

1.4 Alcance

Desarrollar una Aplicación Web para la Facultad de Ciencias de la Universidad Central de Venezuela que facilite la tarea de almacenar, administrar y publicar la Producción Intelectual de la Facultad de Ciencias de la Universidad Central de Venezuela, con una primera fase que se limitará a los Trabajos Especiales de Grado de los estudiantes de pregrado de ésta Facultad a partir del año 2009.

Esta Aplicación permitirá:

1. Realizar búsquedas por palabras claves y filtros (tales como: título, resumen, fecha de creación, tutor, autor, palabra clave del documento) de los Trabajos Especiales de Grado de Pregrado y su acceso al texto completo.
2. Seleccionar los resultados de las búsquedas y enviarlos por correo electrónico.
3. Publicar los Trabajos Especiales de Grado preparados por los estudiantes de Pregrado.

1.5 Metodología a seguir para el desarrollo de la solución

Se tomó como metodología de desarrollo una adaptación de Programación Extrema XP porque el proceso de integración es contínuo, se atienden las

necesidades de los usuarios con mayor exactitud y además se consigue un producto usable con rapidez. (Ambler, 2005)

1.6 Tecnologías y plataformas

- Ruby 1.8.7: Lenguaje de programación orientado a objetos, multiplataforma e interpretado.
- Rails 2.3.8: Framework para desarrollo de aplicaciones Web de código abierto y basado en el patrón de diseño Modelo Vista Controlador.
- MySQL Server 5.1: Manejador de base de datos de código abierto, multiplataforma y multihilo.
- Webrick 1.3.1: Servidor Web encargado de recibir las peticiones por parte de un cliente (navegador Web) y responder a esas peticiones adecuadamente.

Capítulo II

Marco Teórico

El objetivo de éste capítulo es presentar las bases teóricas en las cuales está fundamentada la investigación. Aquí se define Gestión del conocimiento, destacando sus características y la forma como se trabaja, se describen las tecnologías usadas para el desarrollo de la aplicación Web. También se definen los formatos en que se almacena la Producción Intelectual, se describe detalladamente Ruby on Rails y los conceptos fundamentales de los tipos de Bases de Datos.

Para finalizar el capítulo se detalla la forma como los motores de búsqueda trabajan, los tipos de indexación de las bases de datos y el patrón de diseño MVC (Modelo-Vista-Controlador) el cual se utiliza para desarrollar aplicaciones Web.

"Todo saber es por naturaleza efímero y temporal; pero ello no se opone a la idea que para generar un conocimiento nuevo, se requiere de otro que le antecede" (PIAGET, 1980)

2.1 Gestión del conocimiento

Se conoce como conocimiento a la recopilación de información adecuada, de tal manera que la intención es la de ser útil. (Bellinger, 2004)

La gestión del conocimiento es un concepto que busca transferir el conocimiento y la experiencia existente entre los miembros de cualquier organización y así poder utilizarlo como un recurso entre las personas involucradas en dicha organización, en otras palabras la gestión del conocimiento no es más que la gestión del flujo de información y llevarla a los que la necesiten.

Existen técnicas y herramientas que se utilizan para preservar la disponibilidad de la información, así se podrá dar un trato personalizado a la información para lograr convertirla en conocimiento, y de ahí lograr generar activos intelectuales que presten beneficios y además se pueda compartir.

En la administración del conocimiento encontramos tres (3) directrices que son bastante importantes y difícilmente pueden ser ignoradas o dejadas a un lado, a continuación se explican brevemente:

- Identificar, recoger y organizar el conocimiento existente.
- Facilitar la creación de nuevo conocimiento.
- Reutilizar el conocimiento existente para así mejorar notablemente el rendimiento y habilidades.

La gestión del conocimiento dentro de la investigación educativa tiene principalmente dos motivos:

1. La tendencia actual está cambiando las organizaciones, y en términos educativos significa la llegada de nuevas investigaciones, nuevas habilidades y competencias que han de ser cubiertas, ante las necesidades del mercado laboral.

2. La gestión del conocimiento interviene en espacios donde la población adulta representa una gran parte de los usuarios, y se requiere de estrategias específicas para la elaboración de aprendizajes en dicha población.

Existen innumerables sitios Web con información pobre o de muy mala calidad, y eso conlleva a disminuir la probabilidad de conseguir algún conocimiento, ejemplos de dichos sitios son aquellos que no son actualizados constantemente o la falta de un buscador orientado a contenidos, por lo tanto se requieren herramientas informáticas que ayuden a facilitar la información bajo los nuevos criterios digitales. (Rodríguez, 2009)

2.1.1 Características de la Gestión del Conocimiento

Entre las características de un sistema ideal de gestión de conocimiento podemos encontrar: (Rodríguez, 2009)

1. Cada persona debería tener acceso a la base de conocimiento de la organización.

2. Cada usuario debería poder ingresar sus propios documentos.

3. El sistema debería funcionar las 24 horas y mantenerse actualizado en todo momento.

2.2 Definición de Aplicaciones Web:

Aplicaciones Web son las aplicaciones que pueden ser accedidas por los usuarios a través de un servidor de internet o intranet, utilizando un navegador. Estas aplicaciones son ligera, actualizables en tiempo real y si es por internet el cliente puede acceder desde cualquier parte del mundo. (Aplicacion Web, 2009)

Las aplicaciones Web se utilizan para tiendas virtuales, blogs, compartir información, correo Web, wikis, foros, redes sociales, entre otros. (ALEGSA, 2008)

2.3 Definición Arquitectura Cliente-Servidor

Es una arquitectura para el desarrollo de aplicaciones web, donde básicamente el cliente le hace peticiones al servidor que le da respuesta, en otras palabras, es un entorno en el que las transacciones se dividen en procesos independientes para el intercambio información, recursos o servicios. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos tales como manejo de la interfaz de usuario, captura y validación de los datos de entrada, generación de consultas e informes

sobre las bases de datos y el servidor al proceso que responde a las solicitudes. (Cliente-Servidor, 2010)

La arquitectura cliente-servidor se ha desarrollado con la finalidad de propiciar un ambiente de computación en el que un número de computadores personales, servidores de archivos, impresoras y otros equipos estén todos conectados en una red. La idea es definir servidores especializados en funciones específicas.

Cuando los procesos cliente y servidor están en uno o más computadores en forma independiente en la red, entonces, el servidor puede proporcionar servicios a más de un cliente, y un cliente puede solicitar servicios a varios servidores que se encuentren disponibles en la red sin importar la ubicación y las características físicas que este posea. La red es la que permite unir los servidores y clientes, por lo tanto, es el medio de comunicación entre clientes y servidores. (VALLE, 2005)

2.3.1 Componentes de la Arquitectura Cliente-Servidor

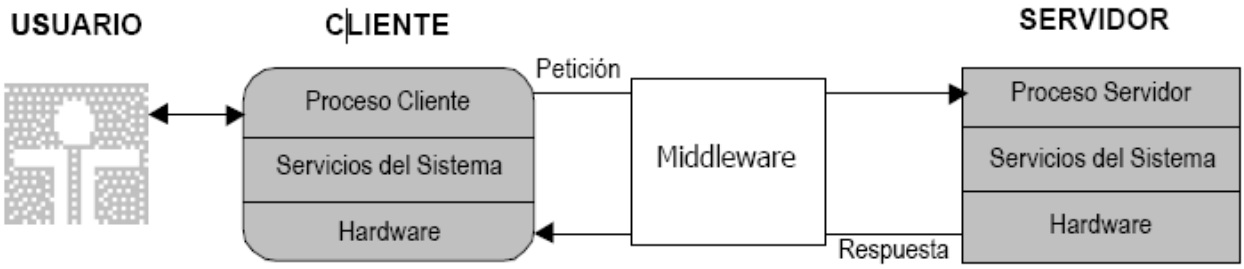

Figura 2. Componentes del Modelo Cliente-Servidor

Cliente: El cliente es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN.

Middleware: Es un término que abarca a todo el software distribuido necesario para el soporte de interacciones entre Clientes y Servidores. Este se inicia en el modulo de API de la parte del cliente que se emplea para invocar un servicio real; esto pertenece a los dominios del servidor.

Tipos de Middleware:

Middleware general: Este tipo permite la impresión de documentos remotos, manejos de transacciones, autenticación de usuarios, etc.

Middleware especifico: Generalmente trabajan orientados a mensajes. Trabaja uno sola transacción a la vez.

Servidor: Cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs.

Tipos de Servidores: Servidores de archivos, Servidores de Base de Datos**,** Software de Grupo, Web, Correo, Objetos, Impresión y Aplicación.

2.3.1.2 Ventajas de la arquitectura Cliente-Servidor

- Los usuarios pueden usar herramientas conocidas, como hojas de cálculo y herramientas de acceso a bases de datos.
- Los usuarios pueden construir soluciones particularizadas mediante la integración con las aplicaciones de uso habitual.
- Los recursos, accesos e integridad de los datos son administrados por el servidor. De manera que un sistema defectuoso no puede dañar el sistema.

- Es de fácil mantenimiento ya que se puede modificar un cliente sin afectar a los otros, o inclusive, reparar, actualizar o mover un servidor sin afectar a los clientes.
- Se puede aumentar el número de clientes y/o servidor(es) por separado, dependiendo de las necesidades.

2.3.1.3 Tecnología del lado del Cliente

Lenguaje de Marcas de Hipertexto (HTML): Es un lenguaje de programación que permite describir la estructura y el contenido en forma de texto, mediante etiquetas, no es un lenguaje compilado por lo tanto los errores de programación no son detectados. Su principal base son los hipervínculos o los link, que permiten el enlace entre páginas HTML o documentos, permitiendo así la navegación por el Word Wide Web.

El Lenguaje HTML está compuesto por dos partes, cabecera y cuerpo. En la cabecera va la identificación de la pagina HTML y se reconoce porque comienza con" <head>" y termina con "</head>", de igual manera el cuerpo que lleva contenido del documento, comienza con "<body>" y termina con "</body>".

Lenguaje extensible de marcado de hipertexto (XHTML): es el lenguaje de programación pensado para reemplazar al HTML como estándar de las páginas Web. Los documentos XHTML están diseñados en el trabajo en conjunto con aplicaciones de usuario XML, también se puede definir como una extensión del HTML. El XML (Extensible Markup Language**),** es un lenguaje de programación que permite crear etiquetas propias, permitiendo así, la validación, definición e interpretación de datos entre aplicaciones.

Las principales diferencias del XML con respecto al HTML, se basan en:

- Los elementos vacios o no vacios siempre deben cerrarse.
- Los elementos anidados deben mantener el orden de apertura.

- Los valores de los atributos siempre van en comillas.
- Las etiquitas, nombre de elementos y atributos son en minúscula.

(XHTML vs HTML, 2005)

Hojas de Estilo en Cascada (CSS): es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML), se usa para separar la estructura del documento con su presentación. (Definicion de CSS, 2009)

Las Hojas de estilo se definen entre el “<head>” y el “</head>”, y permite cambiar las características de una página Web sin modificar su contenido, sus principales usos son para cambiar imágenes, fondos, tamaño de letras, textos, organización del documento. Sus Ventajas primordiales:

- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.
- Reduce las operaciones de actualización y mantenimiento.
- Aumenta la compatibilidad mediante el cumplimiento de las recomendaciones W3C[4] sobre estándares de Internet.

Las Hojas de estilo aparte de ser muy beneficiarias, presentan grandes inconvenientes que deben ser tomados en cuenta, tales como:

[4] W3C siglas de World Wide Web Consortium, es un consorcio internacional donde las organizaciones que son miembro, el personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web. (W3C, 2008 a).

- No todas las propiedades de las CSS son reconocidas igualmente por todos los navegadores.
- Algunas propiedades de las CSS, como las que aceptan la posición o visibilidad de los elementos, pueden provocar que una parte del contenido de la página resulte inaccesible desde ciertos navegadores.

JavaScript: Es un lenguaje de utilizado para la construcción de páginas Web, haciéndolas más interactivas. Al correr del lado del cliente, y poder interactuar con el HTML permite a los creadores de las páginas responder acontecimientos dados.

Sus principales ventajas:

- Las características de Interfaz son gestionadas por el navegador y el código HTML.
- No incluye sintaxis complejas, lo cual lo hace fácil de manejar.
- Los programas javascript tienden a ser pequeños y compactos, los cuales no necesitan mucha memoria, ni recursos para ser ejecutados.

Las principales desventajas.

- El código debe incluirse dentro del HTML, lo cual lo hace visible por cualquier usuario.
- En algunos navegadores puede ser desactivada la opción de ejecutar javascript.

2.3.1.4 Tecnología del lado del Servidor

Hypertext Pre-processor (PHP): Es un lenguaje de programación que corre del lado del servidor, su principal función es crear sitios Webs dinámicos (Ver Figura 3). En otras palabras PHP es una extensión de servidores Web y es posible

utilizar los protocolos de red más famosos como IMAP, SMTP, POP3 e incluso HTTP, o utilizar los sockets. (PHP, 2010)

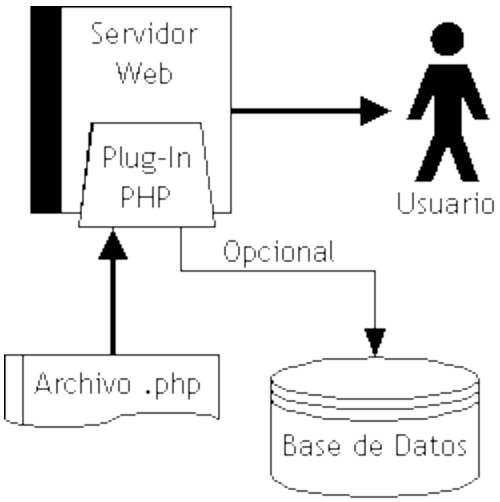

Figura 3. Comportamiento de PHP

Básicamente lo que hace PHP es tomar el código de la Web, lo ejecuta en el servidor y retorna el resultado al cliente. En general se usa frecuentemente para procesar información de formularios, generar contenidos dinámicos, enviar o recibir cookies, conexión con múltiples base de datos (MySQL, Oracle, ODBC, etc.), entre otras.

Las principales características de PHP:

- Permite la integración de bibliotecas externas.
- Es un lenguaje multiplataforma.
- El código es invisible para los usuarios, esto lo hace confiable.
- Es software libre.
- Tiene manejo de excepciones.
- Capacidad para la conexión de múltiples motores de base de datos.
- Permite la programación orientada a objetos.
- Biblioteca de funciones bastante amplia.

- Es un ligeramente lento, debido a que consume recursos llamando y ejecutando una función entre líneas del script.

PáSgina de servidor Java (JSP): Es una tecnología de Java que permite crear contenido dinámico, en respuesta a los requerimientos de un cliente Web. Es un lenguaje de programación que se procesa en línea para finalmente retornar el resultado al usuario de forma HTML.

La sintaxis de JSP agrega llamadas a funciones incorporadas, etiquetas XML, llamada a acciones JSP, para ser usadas para invocar funcionalidades incorporadas. También permite la creación de bibliotecas de etiquetas JSP, que actúan como extensiones a las etiquetas HTML y XML estándares. En el MVC, JSP se encarga de las vistas al usuario.

Java Servlets: son módulos que extienden los servidores petición-respuesta, que corren dentro de un contenedor de servlets o un servidor de aplicaciones (ver Figura 4). Estos módulos reciben los requerimientos del cliente y dinámicamente generan una respuesta que es devuelta regularmente en forma de documento HTML.

Cada Servlet tiene su propio ciclo de vida:

- Un servidor carga e inicia el servlet.
- El servlet maneja cero (0) ó muchas peticiones.
- El servidor elimina el servlet.

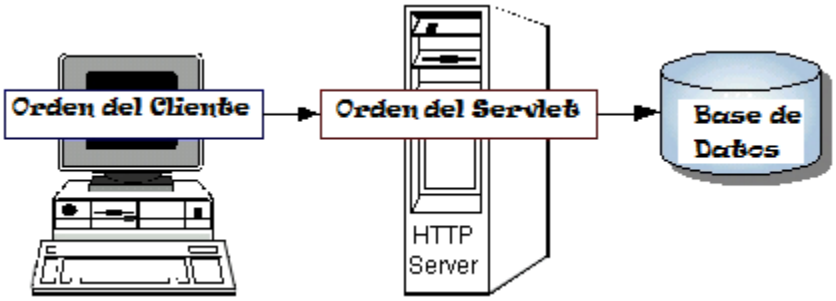

Figura 4. Java Servlets

- **Inicializar un Servlet (primera fase)**

  Cuando el servidor carga un servlet (lo hace ejecutando el método "init"). La inicialización se completa antes de manejar peticiones de clientes y antes de que el servlet sea destruido (muchos servlets pueden ser llamados al mismo tiempo ya que no presentan problemas de concurrencia durante su inicialización).

- **Interactuar con Clientes (segunda fase)**

  Después de la inicialización, el servlet puede manejar peticiones de clientes para el procesamiento necesario de respuesta a las peticiones recibidas. (Hay que tener en cuenta los problemas entre requerimientos concurrentes con acceso a variables compartidas).

- **Destruir un Servlet (tercera fase)**

  Los servlets se ejecutan hasta que el servidor los destruye, por cierre el servidor o bien a petición del administrador del sistema. Cuando un servidor destruye un servlet, ejecuta el método destroy del propio servlet.

Al momento de la implementación del MVC, tenemos a los servlets en la parte del controlador.

2.4 Patrón Modelo-Vista-Controlador

Modelo Vista Controlador (MCV) es un patrón de diseño, comúnmente utilizado en aplicaciones Web, que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres (3) componentes distintos (Ver Figura 5). (Modelo Vista Controlador, 2009)

Las actividades asociadas a los componentes se describen a continuación:

- **Modelo:** Define las reglas del negocio (funcionalidad de la aplicación) y representa las estructuras de los datos.
- **Controlador:** Gestiona las acciones del usuario actuando como intermediario entre las vistas y el modelo.
- **Vistas:** es la información presentada al usuario del sistema.

A continuación se presenta una secuencia de tareas del MCV las cuales se combinan para ofrecer una respuesta satisfactoria y coherente con respecto a la acción realizada por el usuario de la aplicación:

1. El usuario interactúa con la interfaz e introduce un evento cualquiera.
2. El controlador recibe y gestiona el evento introducido por el usuario.
3. El controlador se comunica con el modelo para realizar la tarea solicitada por el usuario. En algunas situaciones se deben modificar los datos del modelo.
4. Después de recibir la respuesta del modelo el controlador la entrega a la vista para que sea mostrada al usuario. En algunos casos el modelo se comunica con la vista para entregarle los resultados de la solicitud.
5. La vista espera nuevas interacciones por parte del usuario, si éstas ocurren se repite el ciclo nuevamente. (Modelo Vista Controlador, 2010)

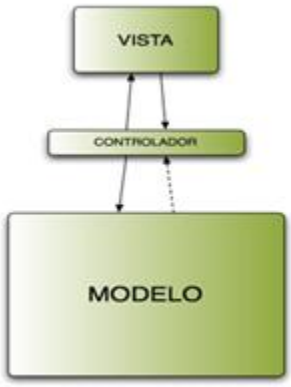

Figura 5. Modelo Vista Controlador

2.5 Ruby

Ruby es un lenguaje de programación creado en el año 1993. Es interpretado, reflexivo y orientado a objeto y su distribución es bajo una licencia de software libre. (Ruby, 2009)

El objetivo primordial de Ruby es ser un lenguaje de programación fácil de leer y escribir, siguiendo el primicia "principio de la menor sorpresa", lo que significa que el sistema debe comportarse de tal manera que minimice la confusión de los usuarios más expertos. Los paquetes de aplicación o librerías de Ruby son llamadas RubyGems y se consiguen en el repositorio RubyForge. (Ruby, 2009)

Las principales características de Ruby son:

- Procesamiento de texto (Text processing**)**, archivos de Ruby, Strings y clases ayudan al procesamiento de los datos rápido y limpiamente.
- Programación CGI (CGI programming**)**, Ruby incluye clases a las librerías CGI e interfaz de base de datos.
- Programación de red (network programming), incluye socket clases.

- Programación de Interfaz Grafica de Usuario (GUI programming), Ruby cuenta con kits de interfaz tal como Ruby/tk.
- Programación XML (XML programming), permite la programación manual del XML
- Prototipos (Prototyping), permite aumentar la productividad permitiendo hacer uso de Ruby en prototipos o plantillas.
- Programación educacional (Programming education), es fácil de aprender y de ensenar.

2.6 Ruby on Rails

Ruby on Rails, RoR como también se le conoce, es un framework para desarrollar aplicaciones Web el cual sigue el paradigma MVC (modelo-vista-controlador) y es de código abierto. Una de las filosofías en la que está basado RoR es la de *DRY* (no te repitas); la idea es la de no repetir lo que ya se ha definido en otro lugar. La segunda filosofía nos habla de que la convención predomina sobre la configuración, esto es que, si estamos acostumbrados a realizar siempre las mismas configuraciones para lograr un objetivo específico RoR nos ofrece una convención para alcanzar dicha meta lo más rápido y fácil posible. (Ruby on Rails, 2009)

La correspondencia que se genera con el patrón MVC viene dada por: Modelo → ActiveRecord, Vista → ActionView y el Controlador → ActionController.

- **ActiveRecord:** Ésta clase implementa el patrón de diseño *ActiveRecord*, el cual provee una interfaz entre las tablas de una base de datos relacional, y el código de la aplicación escrita en Ruby que manipula los registros de la base de datos.

- **ActionView:** Ésta clase es la encargada de recibir los datos de las clases del controlador, y por otro lado, existe una vista para cada acción del

controlador y son éstas las que definen la presentación que tendrán los datos en los archivos HTML que se enviarán al navegador.

- **ActionController:** Invocan a la lógica de la aplicación, que a su vez manipula los datos de la clase *ActiveRecord* y muestra los resultados usando la clase *ActionView*. Cada controlador es una clase de Ruby, y cada método dentro de dicha clase representa una acción.

2.7 PDF

El formato de documento portátil (PDF) fue creado por Adobe Systems hace ya quince (15) años, permite entre otras cosas obtener y poder visualizar información desde cualquier aplicación y en cualquier sistema informático. La ventaja que se asoma de lo antes mencionado es que dicha información se puede compartir prácticamente con cualquier persona sin tener que preocuparse del lugar donde se encuentre y tampoco en que sistema este trabajando. (Adobe, 2010)

Características del formato de documento portátil (PDF):

- **Multiplataforma:** Los archivos PDF se pueden ver e imprimir en cualquier plataforma. Ejemplo: Mac OS, Microsoft Windows, Unix y una gran variedad de plataformas móviles.
- **Extensible:** Alrededor de 1800 empresas alrededor del mundo ofrecen asesoría y soluciones basadas en PDF.
- **Sofisticado en cuanto a la integridad de la información:** Los archivos PDF muestran la misma información que los archivos originales. Ejemplo: texto, dibujos 3D, fotos, hipervínculos, etc.
- **Más seguro:** Los archivos PDF que se crean pueden ser firmados electrónicamente o también se les puede proteger con contraseña.

La gran aceptación que ha obtenido PDF a través de los años se debe a la posibilidad de imprimir exactamente lo que se ve en la pantalla sin la necesidad de adaptar márgenes o la visualización de las imágenes, como ocurre con otros formatos.

2.7.1 Estándares de los archivos PDF

- **PDF/A:** Aprobado por la ISO a finales de 2005, se principal objetivo es facilitar la preservación de los documentos electrónicos. Un aspecto importante de los archivos PDF es la metadata (datos sobre los datos) contenida en las propiedades, cualquier cambio que se realice en el documento PDF se ve reflejado en la metadata de dicho documento. Como la metadata se encuentra en formato XML se puede modificar, así las empresas pueden crear su propio estándar para que los documentos electrónicos perduren a través del tiempo. (Adobe, 2010)

- **PDF/E:** Aprobado por la ISO en 2007, se caracteriza por proporcionar directrices para el intercambio fiable de documentos y diseños. Los grandes beneficiados de este estándar son los ingenieros, arquitectos, y profesionales dedicados a lo construcción ya que permite el intercambio de eficiente de documentos que soportan el contenido complejo en 3D. (Adobe, 2010)

- **PDF/X:** Fue desarrollado para facilitar el intercambio de páginas finales y listas para impresión. Los profesionales que más se benefician son los que se encuentran en el mundo de las artes graficas. (Adobe, 2010)

2.8 Creación de Documentos PDF con Ruby on Rails.

PDF::Writer

Es la biblioteca de Ruby on Rails más utilizada para la creación de PDF, no soporta todas las versiones de PDF (en especial las más antiguas), y es de licencia gratuita.

Para su instalación es necesario la descarga de Ruby PDF TOOLS, desde la página oficial de Ruby, y escribiendo la siguiente línea de comando:
"% ruby setup.rb". (Ver Ejemplo de Código 1)

```
"require "pdf/writer"
pdf = PDF::Writer.new
pdf.select_font "Times-Roman"
pdf.text "Hello, Ruby.", :font_size => 72, :justification => :center
File.open("hello.pdf", "wb") { |f| f.write pdf.render }"
```

Ejemplo de Código 1. Crear PDF

Cabe destacar que en el Ejemplo de Código 1 se creó un documento PDF llamado *hello*.

Courier, Helvetica, Symbol, Times-Roman y ZapfDingbats son los diferentes tipos de letras que pueden ser creados a partir de PDF::WRITER, que a su vez también soporta caracteres especiales y Unicode, además permite la configuración de los márgenes, diferentes operaciones con un documento PDF, imágenes y tablas.

La extracción de textos en archivos como los PDF's, Word o TXT, se puede lograr mediante un código ruby e instalando XPDF, utilizado anteriormente en un trabajo especial de grado (BUSCONEST), en el que se encuentra la línea de comando que se observa en el Ejemplo de Código 2. (Sanchez, 2008).

```
"system("pdftotext #{nombre_archivo_binario} #{nombre_archivo_texto}")"
```

Ejemplo de Código 2. Transformar un archivo PDF a texto plano

2.9 Tipos de Base de Datos

- **Bases de datos con información factual**: Generalmente son muy concisas y concretas, almacenan resultados en su mayoría numéricos.

- **Directorios**: Recogen datos concretos, de personas, instituciones o una actividad específica. Se usan para directorios de investigación, profesionales, bibliotecas, revistas, etc.

- **Bases de Datos Documentales:** La principal característica de las bases de datos documentales es que cada registro corresponde a un archivo, por ejemplo, una publicación, un documento grafico o audiovisual, documento de texto, etc.

2.10 Indexación de una Base de Datos

La Indexación es una operación orientada a colocar la información de forma ordenada, haciendo énfasis en los elementos más representativos para su futura localización, buscando ocupar el mínimo espacio de almacenamiento y una maximización de respuesta. Está dividido en dos fases: (W3support, 2009)

1. Extracción de descriptores: Las palabras claves con mayor relevancia son extraídas (indexación).
    - Estadístico (análisis de frecuencias)

- Por asignación (listas normalizadas)
- Sintáctico (análisis morfológico y semántico)

2. Traducción a lenguaje documental: Es un lenguaje que incluye relaciones semánticas de tres tipos, relación de equivalencia o sinónimos, relaciones jerárquicas (término general y término específico), relaciones asociativas o términos relacionados.

2.9.10.1 Tipos de Indexación:

- **Por Palabra:** Cada palabra del documento es indexada (sin incluir la lista de palabras vacías, es decir, son las palabras que no se desean que formen parte del índice).
- **Por String:** Se incluyen unión de palabras o frases completas. Este permite índices menos extensos.

Además en la indexación se utilizan filtros que permitan igualar caracteres de manera de mejorar la búsqueda de coincidencias, por ejemplo, “a = A = **Á”**

2.11 Opciones de Búsqueda

En los sistemas de recuperación de información se permite realizar de diferentes maneras la búsqueda, ya sea por menús, formulario o lenguaje de comandos, además de permitir los operadores lógicos o booleanos (como la suma o unión, resta o negación o intersección).

2.11.1 Búsqueda directa: Se introduce una o más palabras en el cuadro de búsqueda, está a su vez se puede ser:

2.11.1.2 Interrogación de texto libre: El usuario realiza su consulta sin importarle la estructura de la base de datos. Normalmente el sistema busca la coincidencia de los términos en todos los campos de la Base de Datos

2.11.1.3 Interrogación en campos individuales: El usuario además de escribir la palabra puede definir sobre el campo que desee hacer la buscar, ejemplo, el autor de un documento, titulo, etc.

2.11.4 Búsqueda a través de Índices: En esta búsqueda el usuario no necesita introducir ninguna palabra, el sistema proporciona un diccionario o índice alfabético de los campos o palabras, esto hace un proceso más seguro de búsqueda aunque menos amigable.

2.11.5 Búsqueda a través de Códigos: En algunos casos, se realizan búsquedas no textuales, es decir por código número o alfanumérico, tales como los códigos de barra.

2.12 Motor de Búsqueda:

Es un sistema software que se encarga de buscar archivos almacenados en directorios Web, su principal objetivo es encontrar documentos que contengan las palabras claves o los filtros introducidos, y por medio de un puntaje (clasificación) se establece el orden de los resultados. (Sanchez, 2008)

Los motores de búsqueda están compuestos por tres partes.

- Los robots que recorren la red examinándola detalladamente.
- La base de datos que es construida por los robots
- El motor de búsqueda que facilita la consulta en la base de datos.

Como funciona:

Todo comienza con una interfaz de una aplicación que contiene un formulario con el fin de definir la palabra clave a buscar y en algunos casos opciones avanzadas de búsqueda (filtros) que permitan restringir más el criterio de búsqueda. El buscador retorna una lista de resultados según la semejanza de la palabra introducida.

2.12.1 Características de un Motor de Búsqueda:

- Un motor de búsqueda puede trabajar con operadores lógicos (por ejemplo: AND, OR, NOT) con el fin de ampliar, reducir o dirigir la búsqueda.
- El orden de los resultados a mostrar esta predefinido, puede ser por cantidad de repeticiones de la palabra, importancia, fecha, etc. Y en algunos casos puede ser cambiado el orden a mostrar.

- La cantidad de resultado a mostrar son truncados por cada página, es decir, se muestra una cantidad finita de resultados por página.

- Las mayúsculas y minúsculas son tomadas de igual manera, es decir, el resultado de una búsqueda con letras mayúsculas o minúsculas es el mismo.

2.13 BUSCONEST:

Es un motor de búsqueda de documentos académicos electrónicos en formato PDF desarrollado en la Facultad de Ciencias de la UCV, bajo el framework Ruby on Rails 1.2.5 con un sistema manejador de base de datos MySQL 5, usando como servidor de aplicaciones Mongrel y Apache como servidor Web. Con la base de datos, técnicas de indexación y un algoritmo de búsqueda hace posible obtener un listado de resultados de documentos electrónicos ordenados mediante

un criterio de posicionamiento en el que toman en cuenta, título, resumen, contenido, la eficiencia académica de los autores. (Sanchez, 2008)

BUSCONEST obvia algunas preposiciones, tales como a, con, en, por, sin, de manera de no sobrecargar la base de datos y de no mostrar resultados que no tengan que ver con la búsqueda deseada.

2.14 Servicios Web

La W3C define "Servicio Web" como un sistema de software diseñado para permitir interoperatibilidad computadora-computador en una red. En general, los servicios Web son sólo APIs Web[5] que pueden ser accedidas en una red, como Internet, y ejecutadas en un sistema remoto.

En otras palabras, servicio Web permite el intercambio de datos entre diferentes computadores con plataformas distintas por medio de mecanismos de comunicación estándares (ejemplo: SOAP[6]). Las características del servicio Web están descritas por un archivo que lleva como nombre WSDL. (Ver Figura 6)

[5] APIs Web Son un conjunto de rutinas que ofrecen acceso a funciones de un determinado software. (W3C, 2008 c).

[6] SOAP Protocolo para el intercambio de mensajes entre redes de computadores. (W3C, 2008 c).

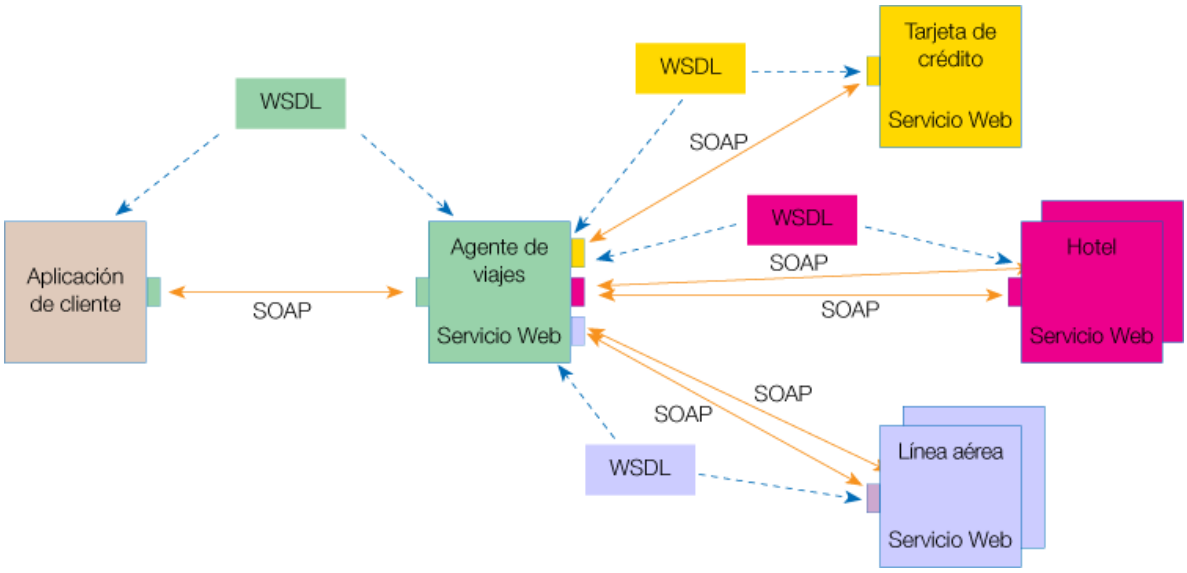

Figura 6. Ejemplo de Servicio Web[7]

2.15 Patrones de Interacción

Los patrones de interacción nos ofrecen soluciones efectivas a problemas recurrentes de diseño, ayudan notablemente a la aceptación del producto final ya que mejoran la usabilidad de la aplicación Web. Por otro parte, su desventaja radica en que no existen estándares ya definidos actualmente sino "mejores prácticas" al momento de diseñar.

La estructura de los patrones se define como:

- Problema: desafío relacionado al uso del sistema.
- Contexto: centrado en el usuario y el uso que le da al sistema.
- Solución: describe el patrón de diseño a implementar, y depende enteramente del problema y el contexto.

Entre los patrones más utilizados para el diseño de las aplicaciones Web tenemos: *Búsqueda Avanzada, Autocompletar, Ícono de Menú*, etc. (Welie, 2010)

[7] Imagen tomada del sitio Web
http://www.w3c.es/divulgacion/guiasbreves/images/serviciosWeb1.png&imgrefurl

2.16 Usabilidad Web:

Se define como la facilidad que tienen los distintos usuarios de lograr sus objetivos cuando visitan un sitio Web. Un dato de gran relevancia es que los usuarios de la Web son en extremo impacientes, esto nos conlleva a que si no consigue rápidamente lo que está buscando simplemente cierra el sitio Web y se dirige a otro sin pensarlo.

Jakob Nielsen[8], nos proporciona diez (10) reglas que debemos tener en cuenta a la hora de hacer una evaluación de usabilidad a un sitio Web. Todo sitio Web que desee ser accesible, comprensible y fácil de usar debe buscar la forma de no violar las reglas de Jakob Nielsen, las cuales se presentan a continuación:

1. Diálogo natural y simple.
2. Hablar el lenguaje del usuario.
3. Minimizar la carga cognitiva.
4. Consistencia.
5. Retroalimentación.
6. Proveer claramente las salidas.
7. Proveer atajos.
8. Mensajes de error descriptivos.
9. Prevención de errores.
10. Asistencia al usuario.

[8] Experto mundial en usabilidad Web (EE.UU. News & World Report).

2.17 Algoritmo MD5

El algoritmo MD5 (Algoritmo de Resumen del Mensaje 5) fue creado por el profesor Ronald Rivest del MIT[9] y es un algoritmo de reducción criptográfica de 128 bits. Es ampliamente usado para verificar que un archivo descargado de Internet no haya sido alterado, por lo tanto nos ayuda a evitar descargas de virus colocados por usuarios maliciosos. Otra utilidad del MD5 es la de calcular el hash[10] de las claves (password) de los usuarios y guardarlo en la base de datos para futuras autenticaciones en el sistema del usuario.

2.18 Antecedentes consultados

- **ARXIV**: Cornell University Library, es un portal de publicaciones electrónicas de estudios de física, matemáticas, ciencias de la computación, biología cuántica, finanza cuántica y estadísticas. Posee un buscador interno de documentos y si se desean búsquedas más específicas también cuenta con una búsqueda avanzada.

- **Google**: es el principal buscador Web de páginas, documentos electrónicos, sistemas Webs, entre otros, además de poseer múltiples opciones de búsqueda avanzada entre las cuales se tienen: tamaño de archivos, idioma, formato de archivos, fechas de visitas, etc.

- **ACM**: Association for Computing Machinery (A.C.M), es un portal de publicación de documentos electrónicos educacionales y científicos, que tiene como objetivo principal publicar revistas y periódicos científicos relacionados con computación. (Web de la ACM, 2010)

[9] Instituto Tecnológico de Massachusetts
[10] Función para generar claves que identifican inequívocamente a un archivo, registro, etc.

Capítulo III

Marco Aplicativo

En éste capítulo se describe como se realiza la adaptación del proceso de desarrollo de software Programación Extrema (XP) para el desarrollo de la aplicación Web. Además, se exponen detalladamente cada una de las iteraciones realizadas con sus respectivas fases asociadas.

3.1 Proceso de desarrollo XP

Enfoque desarrollado por Kent Beck que se caracteriza por poner más esfuerzo en la adaptabilidad del proyecto, en la comunicación con el cliente y en el trabajo en equipo. Como es sabido los requerimientos cambian eventualmente con el transcurso del tiempo y poner esfuerzos en la fácil adaptación a los cambios de requerimientos de los clientes pregona una mejor práctica que forzar los resultados y el camino del proyecto a unos requerimientos iniciales rígidos e invariables.

Las claves adoptadas por XP para el éxito de un desarrollo son: comunicación, simplicidad, retroalimentación (Feedback) y coraje / valor. Por otro lado surge la aseveración de que se debe usar XP cuando los proyectos poseen requerimientos cambiantes, el riesgo del proyecto es alto y el equipo de desarrollo es en parejas.

3.1.1 Características de XP (Programación Extrema):

- **Desarrollo iterativo e incremental.** Pequeñas mejoras al código en intervalos cortos de tiempo.

- **Programación en parejas.** Se deben llevar a cabo las tareas del proyecto con dos (2) personas sentadas en un (1) sólo puesto.

- **Retroalimentación frecuente con el cliente.**

- **Refactorización del código.** Reescribir ciertas partes del código para aumentar su legibilidad y corregir los errores arrojados por las pruebas.

- **Propiedad del código compartida.** En vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, XP propone que todo el personal pueda corregir y extender cualquier parte del proyecto.

- **Simplicidad en el código.** Las cosas se deben hacer siempre simples, y agregar un poco mas de trabajo si y sólo si es necesario.

3.2 Adaptación de XP

A continuación se describen las actividades que se llevaron a cabo en la adaptación de XP para el desarrollo de la aplicación.

3.2.1 Iteraciones

Las iteraciones presentes en el desarrollo estuvieron basadas en objetivos, es decir, se cumplían con los requerimientos presentes en las Historias de Usuarios[11] antes de comenzar la siguiente iteración.

3.2.2 Planificación

Según XP en la fase de planificación el cliente debe explicar sus necesidades definiendo las actividades que va a realizar el sistema. A éste documento se le

[11] Las historias de usuarios es una representación de los requerimientos del software, es la vía mediante la cual se realiza la comunicación y son escritas por los clientes.

conoce como historias de usuarios, son necesarias aproximadamente de 20 a 80 Historias de Usuarios y se recomienda que duren aproximadamente de una (1) a tres (3) semanas cada una. Se definen los tiempos de entrega de la aplicación para recibir Feedback del cliente. Se deben realizar reuniones constantemente con el equipo de desarrollo para así identificar problemas, proponer soluciones y obtener aquellos puntos a los que se le va a dar mayor importancia.

En este trabajo, las Historias de Usuarios se definen en la Tabla 1, la cual posee un campo ID que hace referencia al número de Historia de Usuario, también tiene una fecha que indica el inicio de la iteración, un campo para los requerimientos y la última columna llamada Tipo.

Cabe destacar que se tienen tres (3) variantes de Historias de Usuarios que describen el estatus de los requerimientos: nueva, corrección y mejora.

Id	**Fecha**	**Requerimiento**	**Tipo**

Tabla 1. Formato de Historias de Usuario

3.2.3 Diseño

XP propone que el diseño de soluciones debe estar basado siempre en la simplicidad. La refactorización resulta bastante útil para lograr alcanzar los objetivos. Se deben utilizar metáforas del sistema, así se ayuda al equipo de desarrollo a comprenderlo mejor y se facilita el trabajo de los mismos. Las tarjetas CRC (Clases, Responsabilidades, Colaboración) ayudan al equipo de desarrollo a definir las actividades durante el desarrollo del sistema. Se utilizan diagramas del

Lenguaje de Modelado Unificado (UML): Casos de Uso y Modelo de Objetos del Dominio.

En este trabajo, para la fase de diseño se explica con palabras y/o realizan los diagramas o prototipos necesarios para la fácil comprensión y solución del problema asociado a cada iteración.

3.2.4 Codificación

XP propone que el grupo de desarrollo debe ser pequeño, dos (2) programadores y el cliente. Éste último deberá estar siempre disponible para formar parte del equipo de desarrollo. Los programadores deben trabajar en pareja para así fomentar el intercambio constante de ideas a lo largo del desarrollo. Las pruebas unitarias deberán ser creadas antes del código fuente, así se asegura una programación ágil con bajas probabilidades de errores. Se deben realizar las integraciones constantemente para evitar obtener un software fragmentado y difícil de unir. Además también se evitan las confusiones de los programadores. Se recomienda dejar para el final la optimización del código y también utilizar estándares de programación.

En éste trabajo, la codificación se realiza en parejas pero no se trabaja en conjunto con la misma Historia de Usuario sino que cada programador se encarga de realizar y alcanzar el objetivo deseado con cada requerimiento dependiendo de la complejidad del mismo, es decir, mientras más complejo el requerimiento se torna se trabaja en parejas en el mismo lugar de trabajo, mientras que si la complejidad no es tan alta cada programador realiza la programación por separado.

3.2.5 Pruebas

XP establece que todos los módulos del sistema deben poseer pruebas asociadas. No se debe entregar ningún módulo del sistema sin antes haber sido sometido a prueba. Finalmente, en el denominado período de pruebas de aceptación o de caja negra se definen las entradas del sistema y sus salidas. Y además se recomienda automatizar las pruebas para que el sistema pueda ser probado varias veces.

Para éste trabajo, se llevaron a cabo las pruebas de aceptación recomendadas por XP pero con la diferencia de que dichas pruebas son realizadas después de que ya se tienen varios requerimientos completos. Por otra parte, se obvia la automatización de las pruebas.

3.3 Requerimientos generales del sistema

El desarrollo que se va a llevar a cabo es una aplicación Web con un motor embebido para realizar búsquedas de los T.E.G. de pregrado de los integrantes de la Facultad de Ciencias. No se necesita tener conocimientos avanzados de Internet para el manejo pertinente de la aplicación.

3.4 Requerimientos Funcionales de la aplicación

- Almacenar los T.E.G. de pregrado.
- Publicar los T.E.G. preparados por los estudiantes de pregrado.
- Hacer búsquedas básicas y avanzadas.
- Visualizar el texto completo de los T.E.G. encontrados.
- Ordenar los resultados de las búsquedas según un criterio determinado.

3.5 Requerimientos No Funcionales de la aplicación

- Plataforma robusta.
- Debe ser extensible.
- Alto grado de disponibilidad.

3.6 Desarrollo de la Aplicación

En ésta sección se explica cómo se realizó la implementación de la adaptación de XP en el proceso de desarrollo de BUBAG.

3.6.1 Iteración 0

En esta iteración se define el modelo de datos de la aplicación y el diseño de la aplicación en general.

- **Planificación:**

En la Tabla 2 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
1.00	12/08/2010	Definir la estructura y las tablas del sistema	Nueva

Tabla 2. Historias de usuario. Iteración 0

Diseño:

En la Figura 7 se observa el modelo de datos del sistema.

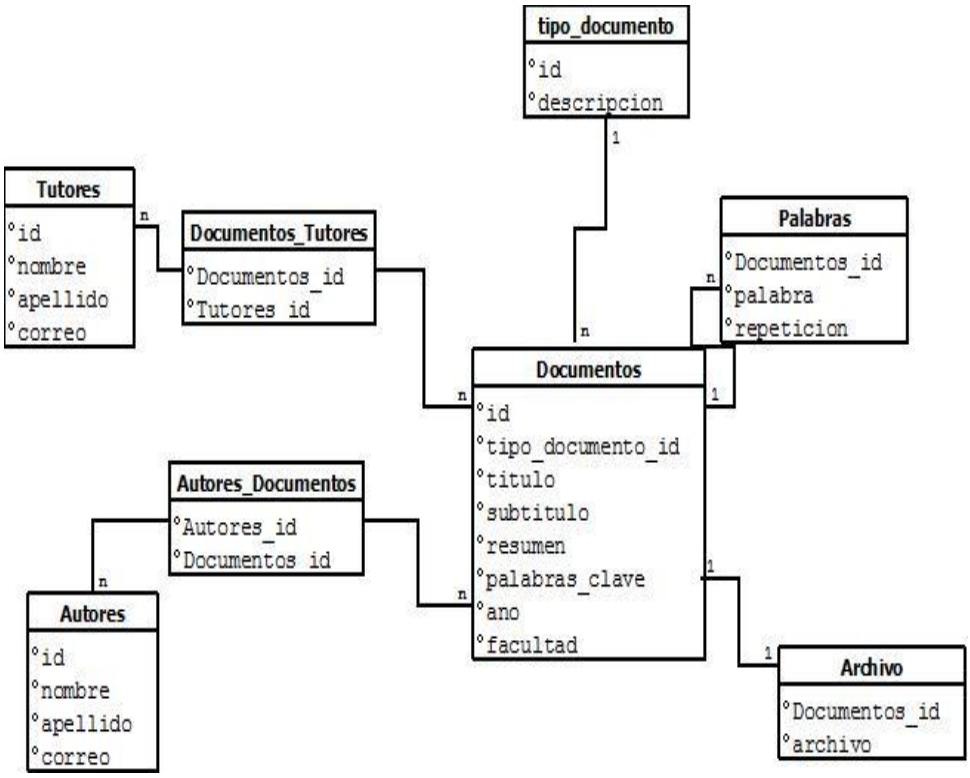

Figura 7. Modelo de datos del sistema

En el modelo de datos de la Figura 7 se muestra la base del sistema, compuesto principalmente de 8 tablas para el almacenamiento de la información de BUBAG, permitiendo independencia entre las tablas para futuras adiciones al sistema que permitan agregar otros módulos de producto intelectual de la Facultad de Ciencias, como por ejemplo, Trabajos Especiales de Grado de Postgrado, revistas, artículos, etc. sin necesidad de realizar algún cambio en la base de datos.

3.6.2 Iteración 1:

En esta iteración se crearon las interfaces para la búsqueda básica (index), avanzada y el ingreso de los datos del documento y se les aplican la hoja de estilo en cascada (CSS).

- **Planificación:**

En la Tabla 3 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
2.00	15/08/2010	Crear las interfaces para búsqueda sencilla, avanzada e ingreso de datos	Nueva
3.00	18/08/2010	Aplicar el CSS a las interfaces	Nueva

Tabla 3. Historias de usuario. Iteración 1

- **Diseño:**

Para la creación de los formularios se utilizan los siguientes patrones de interacción recomendados por Martijn van Welie[12]:

  - Para la interfaz de búsqueda básica se utiliza el patrón: *Área de búsqueda*. Así como también para la búsqueda avanzada se utiliza el patrón *Búsqueda Avanzada*.
  - Para el formulario de ingreso de datos se utiliza el patrón: *Registro*.

- **Codificación:**

Para cumplir con los objetivos de ésta iteración se crearon las vistas necesarias que requieren las Historias de Usuarios asociadas a ésta iteración así como también se desarrolló el CSS de la aplicación. (Ver Figura 8 y 9 respectivamente)

[12] Especialista en diseño de interfaces para lograr interacciones efectivas con aplicaciones Web.

Figura 8. Interfaz del formulario búsqueda

Figura 9. Interfaz del formulario búsqueda avanzada

3.6.3 Iteración 2:

En esta iteración se realiza el almacenamiento de la información de los T.E.G.

- **Planificación:**

En la Tabla 4 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
4.00	21/08/2010	Ingresar documentos	Nueva

Tabla 4. Historias de usuario. Iteración 2

- **Codificación:**

Cumpliendo con los objetivos de ésta iteración se tomaron los datos enviados por el usuario en el formulario de ingreso de documentos y se procedió a guardarlos en el modelo de datos en sus respectivas tablas asociadas, cabe destacar que el archivo es guardado en un campo LONGBLOB que permite hasta 4gb de almacenamiento y en el formulario HTML se utilizó MULTIPART para el manejo de archivos. (Ver Ejemplo de Código 3)

```
180     aux2 = Documento.new(:tipo_documento_id=>0,:titulo=> params[:titulo]
181     ,:subtitulo=> params[:subtitulo],:palabras_clave=> params[:palabras_clave],
182     :resumen=> params[:resumen],:escuela=> params[:facultad],:ano=>
183     params[:ano],:bytes=>md5_archivo,:visible=>params[:visible])
184 aux2.save
185  id_documento = Documento.find(:last,:select=>'id')
186    aux1 = Archivo.new(:documentos_id=> id_documento['id'],:archivo=>documento)
187   aux1.save
```

Ejemplo de Código 3. Almacenamiento de la información del documento

En las líneas 180 hasta la 186 del Ejemplo de Código 3, se puede observar cómo se almacenan los campos del formulario suministrado al modelo documento y al modelo archivo.

- **Pruebas:**

Para las pruebas de aceptación se introdujeron datos en el formulario y se verificó manualmente en la base de datos.

3.6.4 Iteración 3:

En esta iteración se convierte el archivo PDF a texto plano, a continuación procede a la extracción de las palabras y el almacenamiento de las mismas en el modelo palabras.

- **Planificación:**

En la Tabla 5 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
5.00	30/08/2010	Convertir PDF a texto plano	Nueva
6.00	03/09/2010	Extraer las palabras del documento	Nueva

Tabla 5. Historias de usuario. Iteración 3

- **Diseño:**

Para convertir el archivo PDF a texto plano se utiliza un programa llamado PDFtoText, el cual ayuda notablemente a alcanzar el objetivo de la iteración.

- **Codificación:**

Para la conversión del archivo PDF a texto plano se utiliza una aplicación llamada PDFtoText instalada en el sistema operativo Linux, y se le invoca desde un controlador tal como se ve en el Ejemplo de Código 4 línea 115. Seguido se recorre el texto plano para almacenar todas las palabras en un arreglo eliminando los signos de puntuación. Luego se eliminan del arreglo las preposiciones y conectores de oraciones para depurar las palabras que se van almacenar. Se crea otro arreglo que contenga las palabras únicas sin repetición y se hace una comparación entre los arreglos para obtener el número de repeticiones por palabra, para finalizar se almacena en la tabla palabras el id del documento, las palabras y el número de repeticiones de cada una.

```
113     if extensionArchivo == '.pdf'
114     aux_ext = true
115     system("pdftotext #{params[:documento].path} #{params[:documento]}.txt")
116     doc_binario = File.new("#{params[:documento].path}.txt")
117     aFile = File.new("#{params[:documento].path}.txt", "r")
118
119 #guarda las palabras del txt en un arreglo
120 if aFile
121 #     aFile.each {|c| puts c}
122 # guarda cada palabra en un arreglo
123    aFile.each('') {|ch| a = ch.gsub(/[!,:,?,(,),.,-,–,;,',",@,{,},*,&,^,%,$,<,>,+,-,_,”,“,‘,‘,¿,]+/,"").downcase.split(' ')
124      a.each { |i| arreglo_palabras.push(i)}
125    }
126    #eliminar los adjetivos...
127    arreglo_palabras.delete("/")
128    arreglo_palabras.delete("de")
129    arreglo_palabras.delete("el")
130    arreglo_palabras.delete("a")
```

Ejemplo de Código 4. PDFtoText

- **Pruebas:**

Para las pruebas de aceptación se agregaron documentos de 500kb a 34 MB y se verificó manualmente en la base de datos si se estaban guardando correctamente, al igual se recorrió el PDF tomando palabras al azar, se realizaron cuentas manuales y se comparo que fuesen iguales a lo que había almacenado en el modelo palabras.

3.6.5 Iteración 4:

En esta iteración se realizan las búsquedas de los documentos.

- **Planificación:**

En la Tabla 6 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
7.00	09/09/2010	Realizar búsquedas sencillas y avanzadas	Nuevo

Tabla 6. Historias de usuario. Iteración 4

- **Diseño:**

Para realizar la búsqueda sencilla se necesita que el usuario ingrese una palabra de su interés, por otro lado para la búsqueda avanzada se debe ingresar ciertos parámetros como: título, autor, tutor, palabras clave, resumen, año o licenciatura.

Para la recuperación de los resultados de las búsquedas se deben crear dos (2) vistas nuevas, las cuales nos ayudan a mostrar los resultados por pantalla a los usuarios tal como se aprecia en la Figura 10.

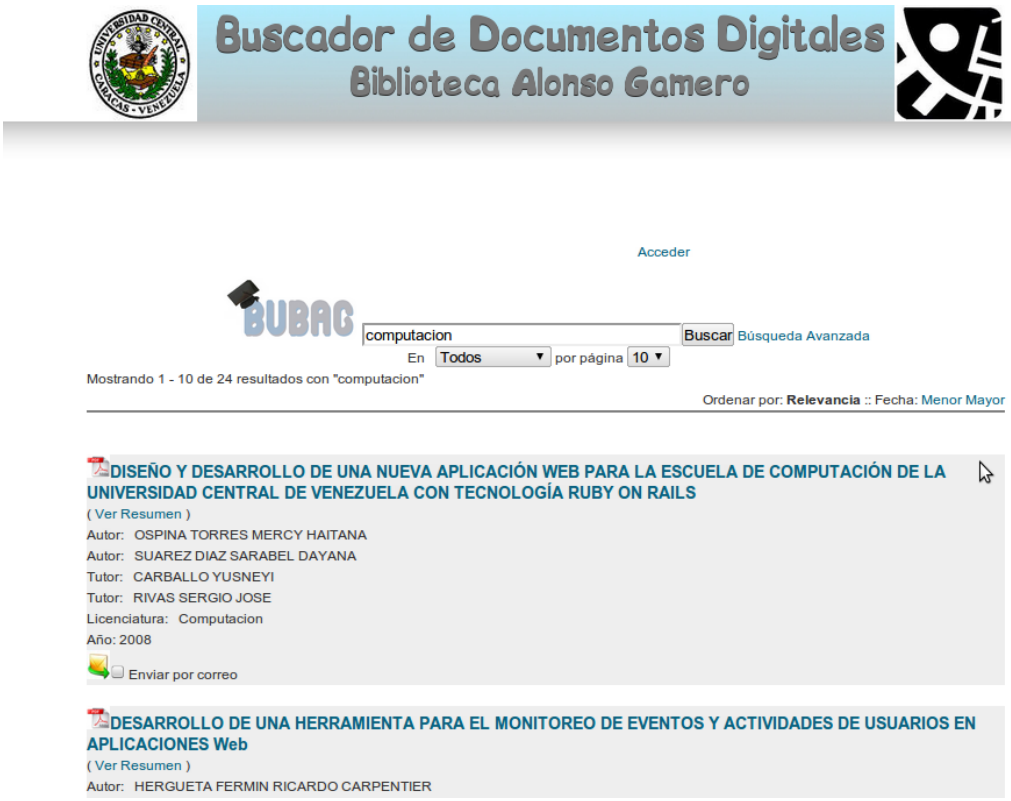

Figura 10. Resultado de búsqueda

- **Codificación:**

Al enviar el formulario de búsqueda con alguna palabra, se realizan comparaciones en los campos de las tablas *Autores*, *Tutores*, *Documentos* y *Palabras*, en busca de alguna coincidencia. De haberse conseguido se obtiene toda la información del documento y es enviada a una vista para ser mostrada al usuario. En el Ejemplo de Código 5 en la línea 42 se observa que el sistema hace la comparación de la palabra suministrada en modelo documentotutore en busca de alguna coincidencia, de haberla encontrado almacena el id del documento en un arreglo.

```
31
32      buscar = Tutore.find(:first, :select => 'id',:conditions =>[ 'id = ? or nombre = ? or correo = ?',h,h,h])
33      buscar2 = Autore.find(:all , :select => 'id',:conditions =>[ 'id = ? or correo = ? or nombre = ?',h,h,h])
34      buscar4= Documento.find(:all,:select => 'id', :conditions =>[ 'resumen like ? or id like ?
35      or titulo  like ? or subtitulo  like ? or palabras_clave  like ?',"%#{h}%","%#{h}%","%#{h}%","%#{h}%","%#{h}%"])
36      buscar5 = Palabra.find(:all,:select=> '*',:conditions=>['palabras = ?',h])
37
38
39
40  #busca en tutores
41      if !buscar.blank?
42     buscar_relacion = DocumentosTutore.find(:all,:select => 'documentos_id', :conditions => ['tutores_id = ? ', buscar['id']])
43     buscar_relacion.each do |i|
44       arreglo.push(i.documentos_id)
45       arreglo3.push([100,i.documentos_id])
46     end
47
48     end
```

Ejemplo de Código 5. Búsqueda sencilla

Para la búsqueda avanzada se realizan comparaciones puntuales en las tablas, es decir, se hacen sólo comparaciones de los campos que llenó el usuario y el campo de la tabla, por ejemplo, si solo se coloca la cédula del autor se realiza la búsqueda solo de la cédula. En el Ejemplo de Código 6, se muestra como en las líneas 89 y 97 se realizan búsquedas en el modelo documento, y se puede apreciar que en la 89 sólo se busca coincidencias con subtítulo y en la 97 sólo con palabras clave.

```
89          buscar4_1= Documento.find(:all,:select => 'id', :conditions =>[ 'subtitulo  like ?  and (ano  between ? and ?)
90          ',"%#{params[:subtitulo]}%", params[:ano_desde].to_i,params[:ano_hasta].to_i])
91      buscar4_1.each do |a|
92         documento1.push(a['id'])
93         arreglo.push(a.id)
94       arreglo3.push([100,a.id])
95  #       puts (a['cedula_autor1'])
96      end
97          buscar4_2= Documento.find(:all,:select => 'id', :conditions =>[ 'palabras_clave  like ? and (ano  between ? and ?)'
98          ,"%#{params[:palabras_clave]}%", params[:ano_desde].to_i,params[:ano_hasta].to_i])
99      buscar4_2.each do |a|
00         documento1.push(a['id'])
01         arreglo.push(a.id)
02       arreglo3.push([100,a.id])
03  #       puts (a['cedula_autor1'])
04      end
```

Ejemplo de Código 6. Búsqueda avanzada

- **Pruebas:**

Para las pruebas de aceptación se introdujeron palabras en el formulario de búsqueda y se constató que estuviese realizando la búsqueda correctamente, además de un seguimiento del output del sistema en la consola.

3.6.6 Iteración 5:

En esta iteración se ordenan los resultados mostrados por pantalla en el momento en el que se realizan las búsquedas de los documentos.

- **Planificación:**

En la Tabla 7 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
7.01	23/09/2010	Realizar búsquedas sencillas y avanzadas	Mejora

Tabla 7. Historias de usuario. Iteración 5

- **Diseño:**

Para el ordenamiento de los resultados mostrados por pantalla se deben actualizar las vistas asociadas a las búsquedas, es decir, se le debe mejorar las funcionalidades de búsqueda. Se agrega también el patrón: *Icon Menu* para lograr una interfaz más agradable.

- **Codificación:**

Al enviar una palabra a través del formulario de búsqueda, se realizan las comparaciones entre la palabra y cada campo de las tablas *autores*, *tutores*, *palabras* y *documentos*. Si la palabra no se encuentra en la tabla *palabras* se le asigna un puntaje y si se encuentra en la tabla *palabras* se toma el número de repeticiones como puntaje, por ejemplo, si la palabra coincide con el título del documento o el nombre del autor, al id de ese documento se le asignan 100 puntos, y por cada repetición de la palabra en la tabla *palabras* se le asigna 1 punto. Al final se realiza una suma del puntaje por cada documento, se ordena de mayor a menor, y es mostrada en una vista en el mismo orden que ha sido ordenada. En la Figura 11, se muestra la salida del sistema al realizar una búsqueda, en este ejemplo, se observa la palabra “puntos”, seguido por un numero, al igual q “id del documento” seguido por otro entero, el primer número son los puntos que le va asignando el sistema al documento(lo identifica por el id), y después de la línea que contiene los asteriscos(*), es la suma de lo acumulado por documento para otorgarle la relevancia y el orden de resultados a ser mostrados.

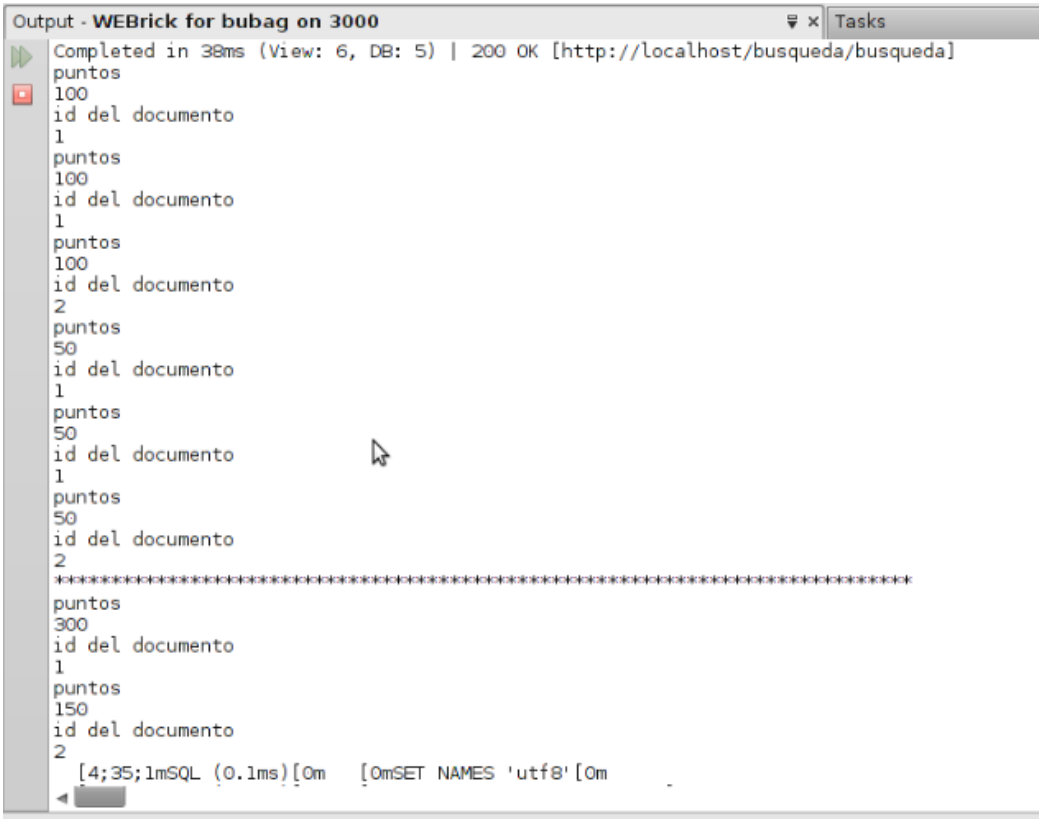

Figura 11. Búsqueda por relevancia

Al momento de mostrar la información del documento, se utilizaron dos gemas, will_paginate y pdf-writer, para facilitar la carga de la paginación y mostrar el archivo PDF. En el Ejemplo de Código 7 se muestra parte del archivo environment.rb el cual se configuró para añadir y utilizar las gemas mencionadas.

```
27
28      # Skip frameworks you're not going to use. To use Rails without
29      # you must remove the Active Record framework.
30      # config.frameworks -= [ :active_record, :active_resource, :act
31
32      # Activate observers that should always be running
33      # config.active_record.observers = :cacher, :garbage_collector,
34
35      # Set Time.zone default to the specified zone and make Active R
36      # Run "rake -D time" for a list of tasks for finding time zone
37      config.time_zone = 'UTC'
38
39      # The default locale is :en and all translations from config/lo
40      # config.i18n.load_path += Dir[Rails.root.join('my', 'locales',
41      # config.i18n.default_locale = :de
42    end
43    require 'pdf/writer'
44    Mime::Type.register 'application/pdf', :pdf
45    require 'will_paginate'
Ruby file                                                  2114 chars  2158
```

Ejemplo de Código 7. Archivo envioronment.rb

Para realizar la paginación de los resultados en la vista es necesario pasarle la variable que contiene los parámetros a buscar (Ver Ejemplo de Código 8)

```
181   <%= will_paginate (@buscar),:params => {:palabra => @palabra,:facultad => params[:facultad],
182   :items =>params[:items]},:next_label => 'Siguiente',:previous_label=>'Anterior' %>
183
```

Ejemplo de Código 8. Will_paginate en la vista

Para mostrar el documento PDF se uso un link a una nueva ventana con el identificador del documento tal como lo muestra el Ejemplo de Código 9.

```
139 <h3><%= link_to image_tag("../images/pdf.jpg", :border=>0,:size => "20x20"),
140 { :controller => 'ver_archivo', :action => 'ver_archivo',:id_archivo =>i.id },
141 {:popup => ['new_window','height=800,width=750']}%><%= link_to i.titulo,
142 { :controller => 'ver_archivo', :action => 'ver_archivo',:id_archivo =>i.id },
143 {:popup => ['new_window']}%> <br/></h3>
144
```

Ejemplo de Código 9. PDFWriter en la vista

- **Pruebas:**

Las pruebas de aceptación se llevaron a cabo realizando búsquedas, filtrando por autores, para después comparar el funcionamiento correcto del ordenamiento mediante la salida del sistema y la base de datos, además que mostrara el archivo PDF y también realizara la paginación con 10, 20 y 30 resultados respectivamente.

3.6.7 Iteración 6:

En esta iteración se crean y manejan las sesiones de los usuarios.

- **Planificación:**

En la Tabla 8 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
8.00	29/09/2010	Manejo de sesiones	Nueva

Tabla 8. Historias de usuario. Iteración 6

- **Diseño:**

Para manejar pertinentemente las sesiones de los usuarios se debe actualizar las tablas de la base de datos para agregar una nueva tabla llamada *sesiones* que almacene el nombre de usuario, correo y clave del administrador del sistema.

- **Codificación:**

Para logra el objetivo en ésta iteración se creó la vista para la sesión del administrador. Cuando el administrador desea entrar en el sistema e introduce su clave de acceso, ésta se encripta y se compara con la que se tiene guardada en la base de datos (la cual también está encriptada), si coinciden, se le otorga acceso a las funcionalidades que posee el administrador. (Ver Ejemplo de Código 10)

```
10  def iniciar_sesion
11  contrasena = params[:clave].crypt('salt')
12   usuario = Sesione.first(:all, :conditions => ['usuario = ?',params[:usuario]])
13   puts(contrasena)
14   puts(usuario.clave)
15   if !usuario.blank?
16
17     if contrasena==usuario.clave
18        $status="ok"
19        session[:usuario]=usuario.usuario
```

Ejemplo de Código 10. Inicio de sesión

También se tomó en cuenta que el administrador extraviara la clave y deseara obtener una nueva, y para resolver dicho caso se creó un enlace que se encarga de enviar una interfaz al administrador para que pueda recuperar su contraseña de nuevo. (Ver Figura 12)

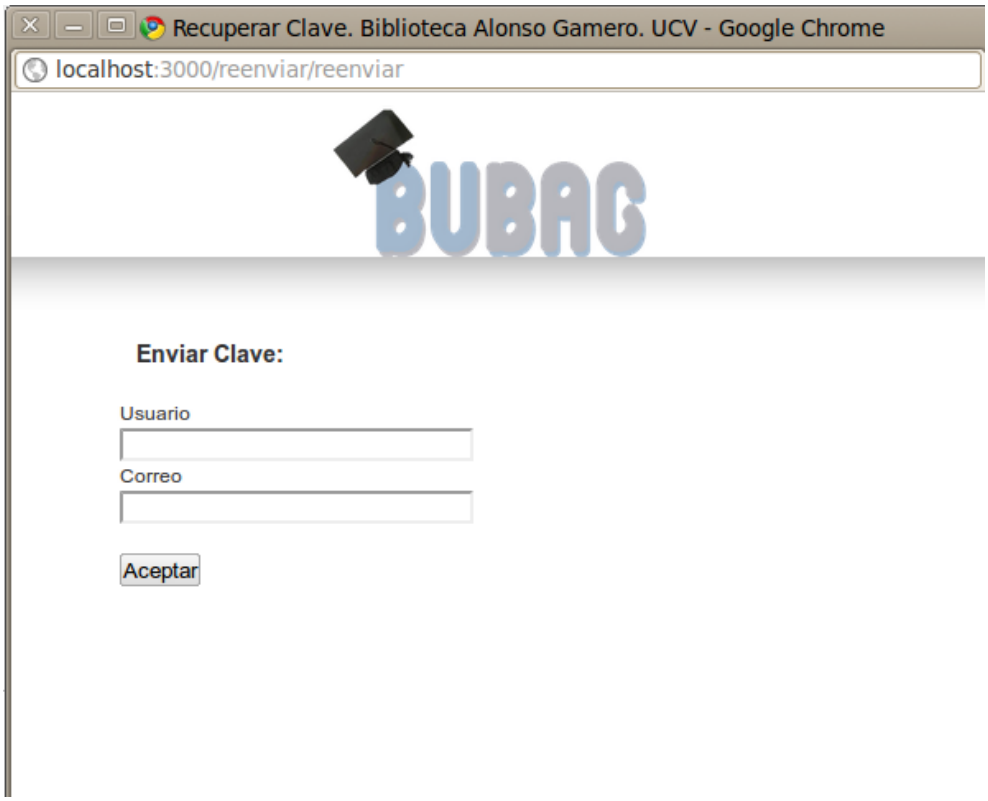

Figura 12. Recuperar clave

Al momento de solicitar la nueva contraseña el usuario debe llenar los dos campos (usuario y correo) y el sistema genera un numero al azar de 10 dígitos, lo encripta y cambia la clave en el modelo. Al administrador le llega su nueva contraseña numérica al correo (Ver Figura 13), y si desea cambiarla, se creó otro modulo que lo permite (Ver Figura 14).contraseña

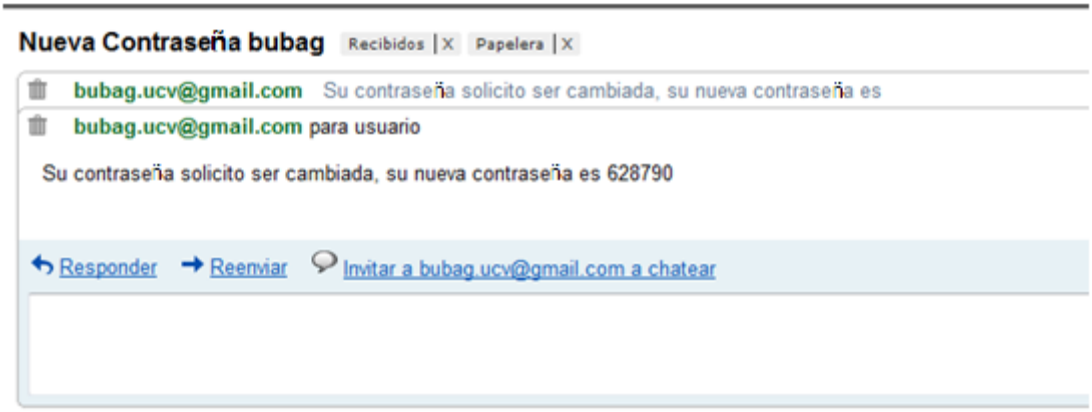

Figura 13. Correo que se envía para notificar cambio de clave

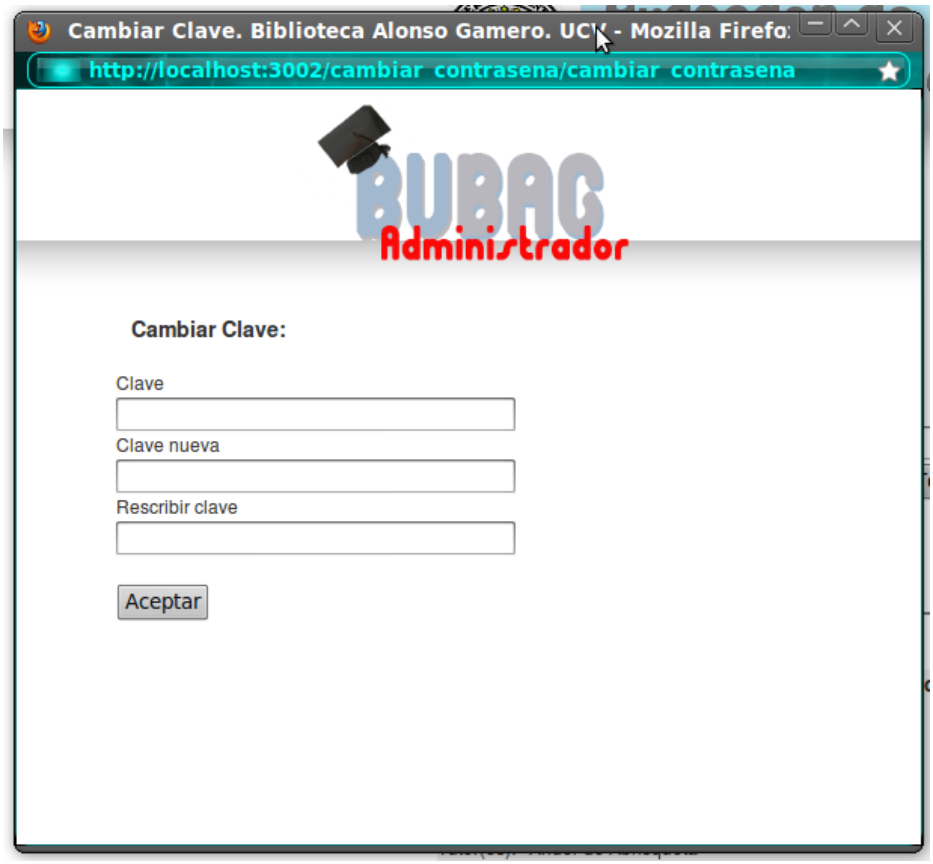

Figura 14. Cambiar clave

Para el envío de correo se creó un Mailer con la configuración del correo (SMTP, usuario, clave, puerto y dominio), al llamar a la función se le pasa el parámetro de la nueva clave sin encriptar. Al momento de cambiar la clave, el sistema la encripta y la compara con la que se encuentra en la tabla sesiones, de ser igual, actualiza la tabla y almacena la nueva clave encriptada.

- **Pruebas:**

Las pruebas de aceptación que se hicieron para verificar que el manejo de sesiones estuviese trabajando pertinentemente fue la siguiente: Se trato de acceder a las vistas que necesitan autenticación por medio del URL y la verificación se obtuvo al momento en el que la aplicación dirigió al usuario a la vista de inicio de sesión.

Para culminar con las pruebas de ésta iteración, se navego por el enlace que nos dirige a la recuperación de la clave y se colocaron los datos solicitados por el sistema. Al final se constató que le clave fuese enviada al correo del administrador de la aplicación.

3.6.8 Iteración 7:

En esta iteración se crean las interfaces que competen al administrador de la aplicación y se le dan las funcionalidades, entre las cuales tenemos: eliminar documento, agregar documentos, ver cantidad de documentos.

- **Planificación:**

En la Tabla 9 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
9.00	02/10/2010	Rol Administrador	Nueva

Tabla 9. Historias de usuario. Iteración 7

- **Diseño:**

En el siguiente diagrama de casos de uso de nivel 2 se detallan los roles que posee el usuario administrador. (Ver Figura 15)

Figura 15. Casos de uso de las funcionalidades del administrador

- **Codificación:**

Para lograr los objetivos propuestos en ésta iteración se dividió el trabajo en etapas, las cuales se explican a continuación:

  - Para eliminar los documentos se borra de la tabla *documentos* el archivo y su información, así como la información almacenada en las otras tablas de la base de datos que guarden relación de cualquier tipo con ese documento, para eliminar autor o tutor, primero se consulta si esa persona no tiene otro archivo en el sistema (ver línea 191 del Ejemplo de Código 11), de no tenerlo es eliminado de la base de datos.

```
190
191   autor = AutoresDocumento.find(:first,:select =>"autores_id", :conditions => ['autores_id = ?',autor_id['autores_id']])
192   if autor.blank? or autor == nil
193    Autore.delete_all(['id = ?', autor_id['autores_id']])
194   end
195
196   tutor = DocumentosTutore.find(:first,:select =>"tutores_id", :conditions => ['tutores_id = ?',tutor_id['tutores_id']])
197   if tutor.blank? or autor == nil
198    Tutore.delete_all(['id = ?', tutor_id['tutores_id']])
199   end
200
```

Ejemplo de Código 11. Eliminar Documento

- Para agregar documentos se procede igual como se realizó en la iteración 2 con la Historia de Usuario que posee Id=4.00.

- Como siguiente paso, para sabe cuántos documentos se encuentran almacenados en el sistema, se realiza una consulta a la tabla *documentos,* para conocer cuántos documentos posee cada licenciatura, se realizan varias consultas, utilizando de condición el campo licenciatura y a continuación se pasan las variables a la vista. (Ver Figura 16)

Figura 16. Ver total documentos en el sistema

- **Pruebas:**

Para la prueba de aceptación de ésta iteración se autenticó al sistema un usuario administrador y se procedió a realizar cambios en los documentos, eliminar y subir archivos y por último se verificaron los documentos montados en la base de datos.

3.6.9 Iteración 8:

En esta iteración se actualiza la forma de realizar las búsquedas, agregando un nuevo ordenamiento de los resultados.

- **Planificación:**

En la Tabla 10 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
7.01	5/08/2010	Realizar búsquedas sencillas y avanzadas	Mejora

Tabla 10. Historias de usuario. Iteración 8

- **Codificación:**

Para cumplir con el objetivo se realizó una nueva búsqueda como en la iteración número 5 con la diferencia de que se muestran primero los resultados dependiendo de la fecha en las cual se realizo la investigación. (Ver Figura 17)

Figura 17. Ordenar por fecha

Al seleccionar el link menor o mayor, se pasa por parámetro en el enlace la(s) palabra(s) con la(s) que se hizo la búsqueda, se llama a una función de un controlador y al consultar al modelo se ordena por año. (Ver Ejemplo de Código 12)

```
44      if params[:facultad] == "*" || params[:facultad]==nil
45  #           @buscar = Documento.find(:all, :select => "*", :conditions => ['id in (?)) order by FIELD(id,?', id_busqueda,id_busqueda])
46              @buscar = Documento.paginate :per_page => params[:items], :page => params[:page],
47              :conditions => ['id in (?)) order by (ano', id_busqueda]
48      else
49  #         @buscar = Documento.find(:all, :select => "*", :conditions => ['id in (?) and facultad_documento = ?', id_busqueda, params[:facultad]])
50          @buscar = Documento.paginate :per_page => params[:items], :page => params[:page],
51          :conditions => ['escuela = ? and id in (?)) order by (ano', params[:facultad],id_busqueda]
52      end
```

Ejemplo de Código 12. Controlador ordena por fecha

- **Pruebas:**

Para las pruebas de aceptación se realizaron búsquedas, y se ordenaron los resultados tanto de mayor a menor como en caso contrario, y se comparó con la base de datos que tuvieran el mismo orden.

3.6.10 Iteración 9:

En esta iteración se desarrolla el servicio Web que proporciona la comunicación con la aplicación CONEST.

- **Planificación:**

En la Tabla 11 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
10.00	07/10/2010	Desarrollar servicio Web	Nueva

Tabla 11. Historias de usuario. Iteración 9

- **Codificación:**

Se desarrolló un servicio Web (servidor) creando una función que recibe dos (2) parámetros, los parámetros que recibe son cédula de identidad y licenciatura, y devuelve un booleano, con la finalidad de conocer si un estudiante de una licenciatura ya posee un documento en la aplicación, tal como lo muestra el Ejemplo de Código 13.

```
class EstudianteController < ApplicationController
  wsdl_service_name 'Estudiante'
  web_service_scaffold :invoke
web_service_api EstudianteApi

  def subio_archivo(entero,escuela)
    a = 0
    autor_aux=[]
    autor = AutoresDocumento.find(:all,:conditions=>['autores_id = ?',entero])
    autor.each{|z|autor_aux.push(z['documentos_id'])}
    if !autor.blank?
      facultad =Documento.find(:all , :select => 'escuela',:conditions=>['id in (?)', autor_aux])
        facultad.each{|z|
          puts z['escuela']
puts z['escuela']
puts z['escuela']
puts (escuela)
puts (escuela)



      if (z['escuela'] == escuela)
        a = 1

      end
}
    end
return a
  end

end
```

Ejemplo de Código 13. Servicio Web

- **Pruebas:**

Al realizar el servicio Web, se generó otro controlador para emular el cliente del servicio Web, pasando por parámetro un entero y el nombre de una licenciatura tal como lo muestra la Figura 18 (los campos en rojo son el URL del cliente, los parámetros de cédula, licenciatura y el parámetro que retorna el servidor, en este ejemplo es el número 1) y verificando que el valor retornado es el correcto (que la cédula tenga un documento en el sistema en y corresponda a la licenciatura).

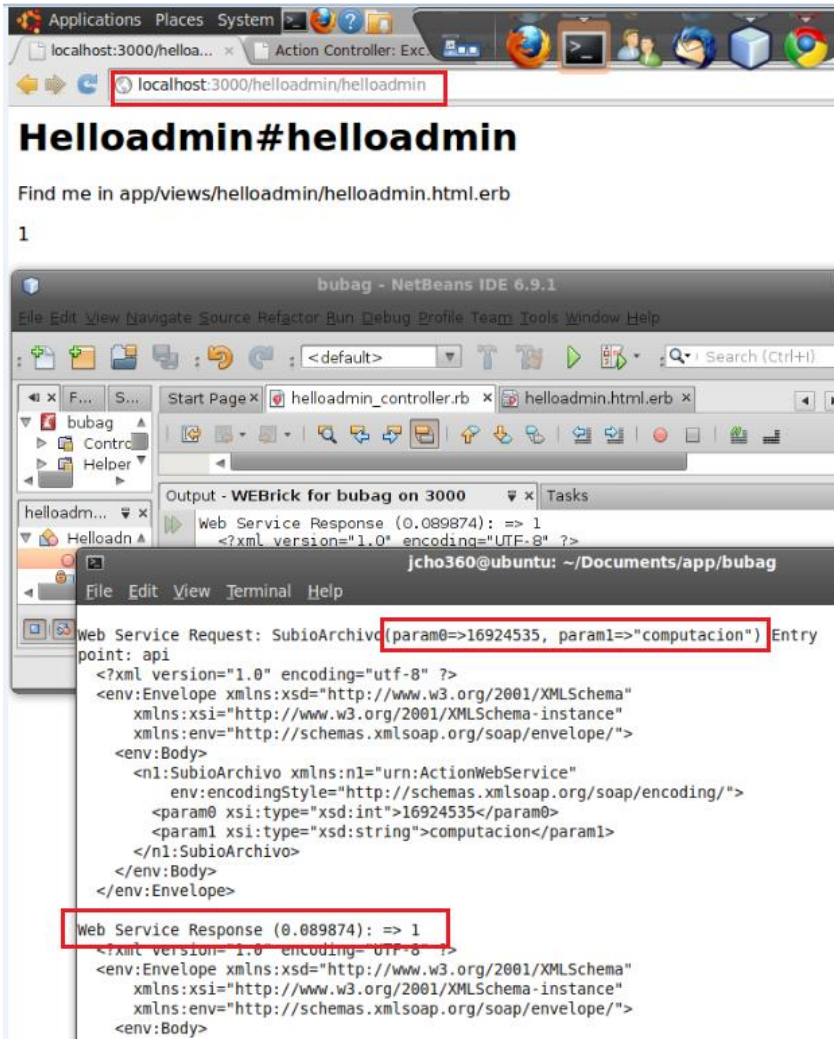

Figura 18. Funcionamiento del Servicio Web

3.6.11 Iteración 10:

En esta iteración se añade la funcionalidad de enviar los resultados de búsquedas por correo electrónico según desee el usuario. También, se agrega la posibilidad de acceder al resumen completo de cada documento si es requerido.

- **Planificación:**

En la Tabla 12 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
11.00	11/10/2010	Enviar por correo información de búsquedas	Nueva
12.00	11/10/2010	Ver resumen del documento	Nueva
13.00	11/10/2010	Restringir documento	Nueva

Tabla 12. Historias de usuario. Iteración 10

- **Diseño:**

Para resolver ésta iteración, se propone agregar una *casilla de verificación* a cada resultado de la búsqueda para así para identificarla y que cualquier usuario pueda enviar por correo el enlace de uno o más resumen (Ver Figura 19)

Figura 19. Casilla de verificación para mandar por correo

- **Codificación:**

Para poder realizar el envío de correo con cada resultado de búsqueda se creó un modelo *Mailer* con la configuración del correo que el sistema utiliza como remitente, también fue necesario crear una vista para solicitarle al usuario los campos del destinatario (Ver Figura 20) y se procedió a realizar consultas en la

base de datos para recuperar la información necesaria para enviar el correo (título y enlace del documento).

Figura 20. Formulario para enviar por correo

Por otra parte, se agregó un enlace al resultado de la búsqueda que contiene el id del documento y que abre una ventana emergente para mostrar la información del autor, tutor y el resumen de un documento seleccionado.

Para finalizar ésta iteración, en el momento cuando los usuarios agreguen su trabajo se le muestra una *casilla de verificación* que el usuario puede o no marcarlo para decidir si desea que su Producción Intelectual sea visible a otros usuarios, dicha *casilla de verificación* es una referencia a un campo booleano en la tabla *documentos*. Si la respuesta es negativa (con respecto a la interrogante de que su trabajo sea público o no), se desactiva el enlace al documento y no podrá ser visible el formato PDF del mismo, sin embargo, se deja público el resumen y

los datos del documento, autor y tutor. En la Figura 21 se muestra en rojo un documento que el usuario ha decidido ponerlo privado, de manera que no tiene enlace para ver el PDF.

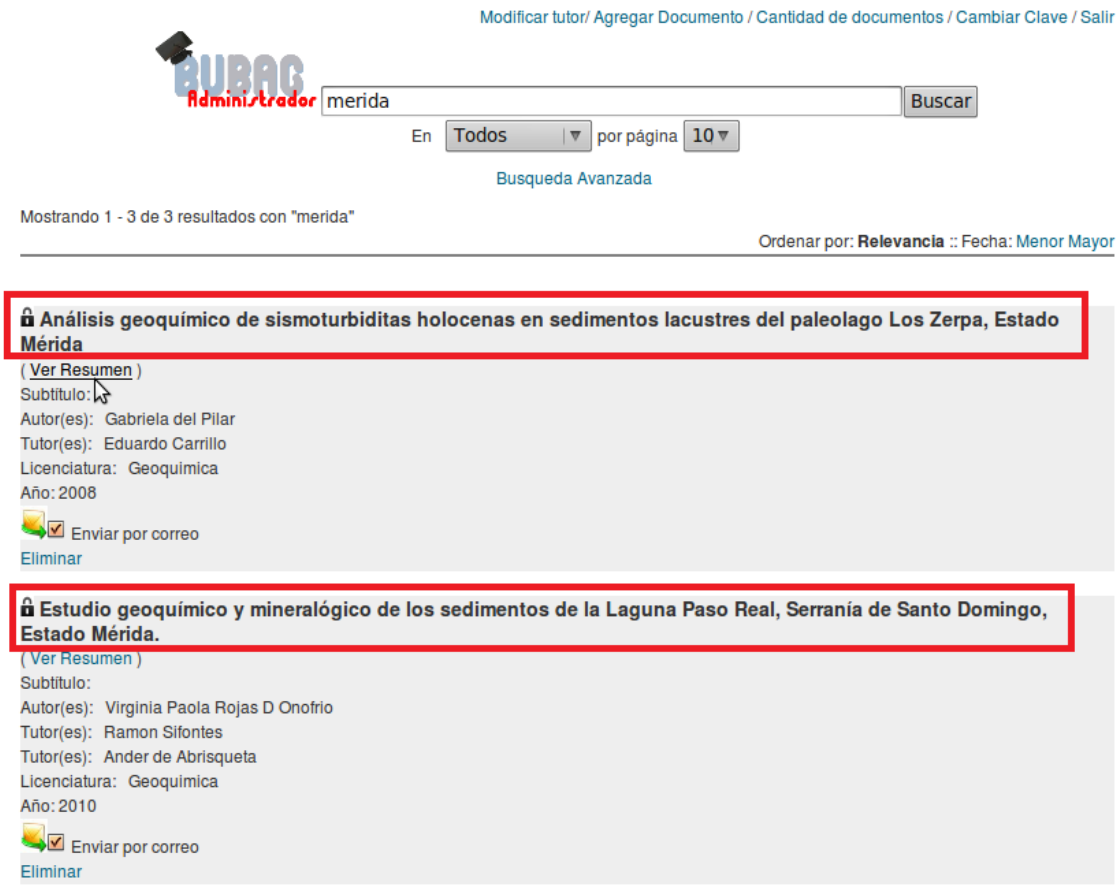

Figura 21. Bloqueo de PDF

- **Pruebas:**

Las pruebas de aceptación las funcionalidades agregadas en ésta iteración se componen por:

  - Ingresar documento restringido y verificar si se puede acceder al mismo.

- Acceder al enlace del resumen en un documento aleatorio y verificar que se muestre correctamente.

- Realizar una búsqueda y enviar por correo la información de documentos aleatorios y verificar en el servidor de correos que efectivamente se haya enviado.

3.6.12 Iteración 11:

En esta iteración se genera un enlace que posee una comunicación con CONEST para realizar la carga de los documentos en el sistema.

- **Planificación:**

En la Tabla 13 se observan las historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
11.00	12/10/2010	Modificar funcionalidad del usuario	Mejora
12.00	12/10/2010	Enlace de CONEST	Nueva

Tabla 13. Historias de usuario. Iteración 11

- **Codificación:**

Los parámetros del estudiante y su(s) tutor(es) son pasados por medio de un enlace que contiene un algoritmo de encriptación (BLOWFISH) y de codificación (base64) tal como lo muestra el Ejemplo de Código 14.

```
15  key = EzCrypto::Key.with_password('[redacted])
16  encrypted_text = key.decrypt(mensaje)
17
18
19  @mensaje = encrypted_text
20  # mensaje=  ezcrypto_key.decrypt(mensaje2)
21  #mensaje =Base64.b64encode(ezcrypto_ciphertext)
22  #mensaje=params[:mensaje]
```

Ejemplo de Código 14. Encriptación y Codificación

En la línea 15 se selecciona la clave de encriptación y desencriptación (por seguridad está cubierta), en la línea 21 se pasa a base64 para poder pasarlo por el enlace. Como se puede ver en el ejemplo que se encuentra en la Figura 22 se transfieren los datos por la barra de direcciones, encriptados y codificados.

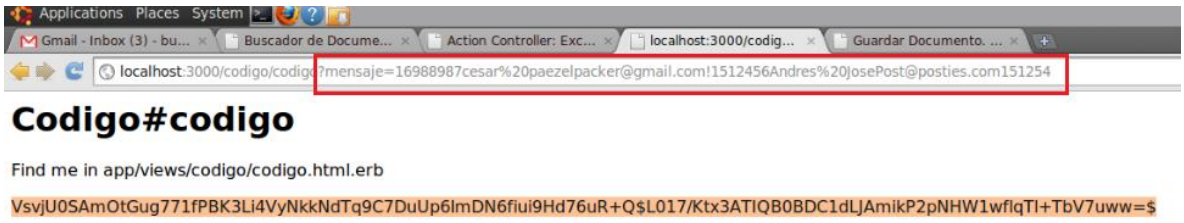

Figura 22. Encriptación y Codificación

BUBAG en el controlador “principal” recibe una cadena de caracteres similar a la que muestra la Figura 22 por la barra de direcciones, y con la misma clave y el mismo algoritmo de encriptación/desencriptación, hace una transformación de base64 a la cadena y luego la desencripta, obteniendo como resultado la cadena original que fue enviada. (Ver Figura 23)

Figura 23. Cadena desencriptada y decodificada

- **Pruebas:**

Se creó un controlador y una vista que simule el paso de la cadena de caracteres codificados y encriptados, utilizando las mismas claves para enviar y recibir se logró desencriptar y decodificar el mensaje, de tal manera de recibir los datos para proceder a guardar el trabajo en BUBAG y darle seguridad al sistema. Para los resultados de las pruebas de aceptación de las funcionalidades del sistema ver Formulario 1.

3.6.13 Iteración 12:

En esta iteración se modifica el modelo de datos y se agrega una nueva funcionalidad al sistema, que permita a cualquier usuario de BUBAG acceso a los documentos más referenciados.

- **Planificación:**

En la Tabla 14 se observan las Historias de usuarios desarrolladas en ésta iteración:

Id	Fecha	Requerimiento	Tipo
1.00		Modificar modelo de datos	Mejora
14.00		Interfaz de los 10 documentos más referenciados	Nueva

Tabla 14. Historias de usuario. Iteración 12

- **Diseño:**

Se modificó la base de datos, agregándole un nuevo campo al modelo *Documentos* (Ver Figura 24), que permite almacenar la cantidad de veces que ha sido consultado un Trabajo Especial de Grado. Además, se generó una nueva vista con un controlador para permitir dicha funcionalidad.

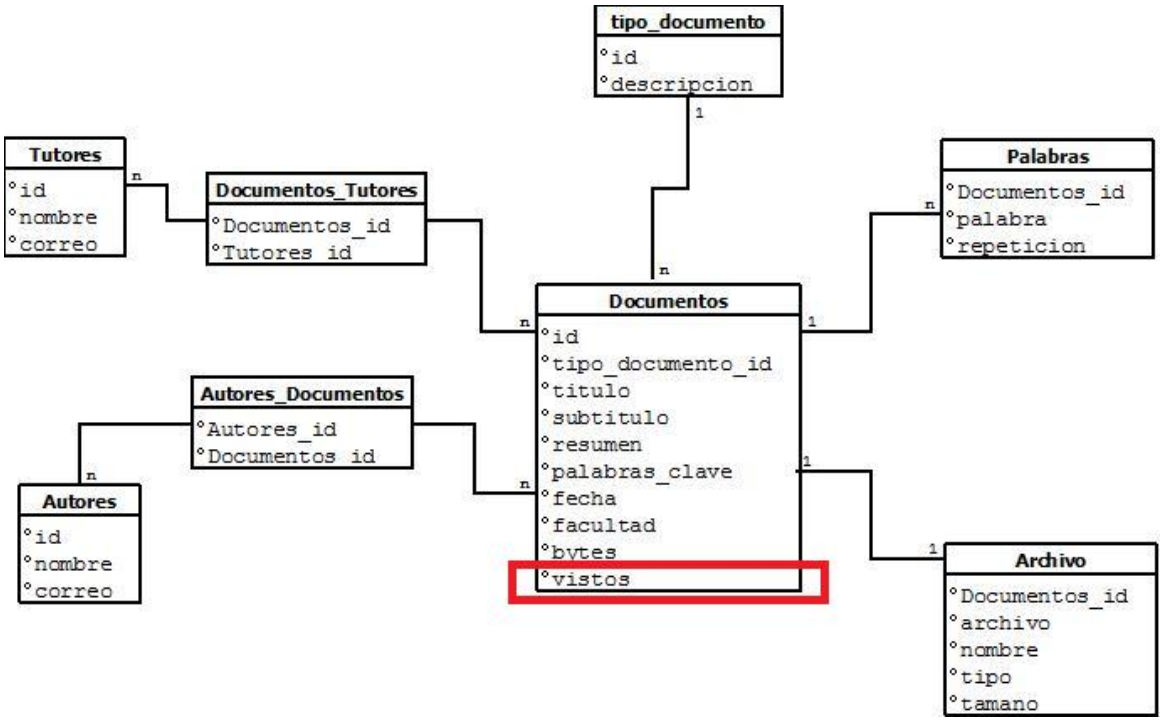

Figura 24. Mejora al Modelo de Datos

- **Codificación:**

Cada vez que un documento electrónico es abierto por los usuarios, el modelo de datos *Documentos* es actualizado, incrementando el valor del campo “vistos” (se le suma 1 al valor ya almacenado), como se ve en el Ejemplo de Código 15. También se crea un controlador que extraiga de la base de datos, la información de los diez (10) documentos más consultados (ya sea entre todos los documentos almacenados en BUBAG o por licenciatura), para luego ser presentado al usuario en una vista parecida a la de la iteración 4 (resultado de la búsqueda básica), donde se muestra la información del trabajo especial de grado (título, año y licenciatura), del autor y del tutor, además de ofrecer las posibilidades de abrir un documento directamente, ver el resumen e inclusive mandar por correo cualquier Trabajo Especial de Grado mostrado en “los 10 documentos más vistos”. (Ver Figura 25)

```
class VerArchivoController < ApplicationController

  def ver_archivo
    puts (params[:id_archivo])
    @archivo = Archivo.find(:first,:select=>'archivo',:conditions=>['documentos_id = ?',params[:id_a
    permite = Documento.find(:first,:select=>'visible',:conditions=>['id = ?',params[:id_archivo]])
    Documento.update_all(['visto = visto + 1'],['id = ?', params[:id_archivo]])
    if (permite[:visible] == false)
      redirect_to '/index'
    else
    pdf = @archivo.archivo
    send_data (pdf), :filename => 'products.pdf', :type => 'application/pdf', :disposition =>'inline
    end
    end
end

```

Ejemplo de Código 15. Incrementar el contador de los documentos más vistos

Figura 25. Documentos más vistos

- **Pruebas:**

Se realizaron búsquedas en BUBAG, y se abrieron documentos de diferentes licenciaturas seleccionados al azar, tomando en cuenta la cantidad de veces que

fueron abiertos y el titulo de dichos archivos, para luego ser comparado con el modelo de datos y con la vista de “los 10 documentos más vistos”, obteniendo que en ambas comparaciones el resultado de los documentos mostrados de mayor a menor fue el mismo para todos los casos (los más vistos en todo BUBAG o por licenciatura).

Formulario 1. Pruebas de aceptación del sistema BUBAG

Según su criterio evalúe las siguientes funcionalidades del sistema BUBAG, la puntuación para otorgar oscila entre 1 = Malo, 2 = Promedio, 3 = Bueno, 4 = Muy Bueno y 5 = Excelente.

1.- Búsqueda básica, puntuación: 5

Observaciones: *No se Obtienen Documentos pertinentes, ya que la búsqueda está dirigida a la obtención de los documentos que contengan cualquiera de las palabras y en cualquier lugar*.

2.- Búsqueda avanzada, puntuación: 5

Observaciones: *Sí es pertinente, localiza información con precisión.*

3.- Acceder al documento en formato PDF de los trabajos, puntuación: 5

Observaciones: *La imagen es nítida, se puede imprimir y grabar*.

4.- Ver resumen de los trabajos, puntuación: 5

Observaciones: *Muestra información valiosa sobre el documento y adicionalmente contiene los documentos constitucionales, los cuales permiten mayor difusión e intercambio de la Producción Intelectual de la Facultad*.

5.- Enviar por correo los resultados de las búsquedas que consideró más importantes, puntuación: 5

Observaciones: *Considero importante esta facilidad, ya que permite seleccionar las referencias que se deseen analizar*.

6.- Acceder a BUBAG como administrador y utilizar sus funcionalidades, puntuación: 5

Observaciones: *Permite con facilidad el acceso a: modificar el tutor, agregar documento, y conocer la cantidad de tesis de una determinada escuela*.

7.- Ordenar por fechas los resultados de las búsquedas, puntuación: 5

Observaciones:

8.- Eliminar documento del sistema, puntuación: 5

Observaciones: *Es importante que ésta función sólo la realice el administrador*.

9.- Navegabilidad de la aplicación, puntuación: 5

Observaciones: *Fácil y amena.*

10.- Agregar documento a BUBAG, puntuación: 5

Observaciones: *Es importante ésta función ya que permitiría eventualmente ingresar documentos.*

11.- Evaluación de la interfaz de BUBAG, puntuación: 5

Observaciones: *La aplicación es fácil de manejar, y se visualiza rápidamente los íconos de las búsquedas, los cuales son el objetivo principal*.

_____	_____
Nombre y Apellido	Firma

Conclusiones

Luego del desarrollo del presente trabajo se puede concluir que el objetivo general de este Trabajo Especial de Grado se cumplió satisfactoriamente, ya que se desarrolló la aplicación Web con el motor de búsqueda embebido BUBAG. Para ello se realizaron entrevistas con los representantes de CONEST y la B.A.G. y se estudió ampliamente el sistema BUSCONEST. Asimismo, se hizo énfasis en el desarrollo del modelo de datos de la aplicación, ya que se tiene previsto que la base de datos tendrá que manejar una alta cantidad de documentos, para cuidar que no se presenten bajos rendimientos por parte de BUBAG.

La adaptación del método de desarrollo Programación Extrema se ajustó a las condiciones de trabajo, para permitir obtener un producto usable en corto tiempo. La exigencia de XP de tener una retroalimentación por parte del cliente en cada una de las iteraciones hizo posible obtener una aplicación de calidad que cumple sus requerimientos. El sistema fue implementado en Ruby on Rails, marco de trabajo orientado al desarrollo ágil, con paquetes integrados de programación y convenciones que permitieron implementar y utilizar de inmediato la aplicación sin necesidad de configuraciones complejas.

Para la arquitectura lógica de la aplicación se implementó el patrón de diseño MVC y se aplicaron estándares de programación (como por ejemplo: nombres de variables descriptivos, código comentado, etc.) para así obtener un código ordenado y legible, que facilitará a futuros programadores que van a continuar con el proyecto, la tarea de entender la lógica del sistema.

Entre las funcionalidades principales del sistema se destaca el motor de búsqueda embebido donde se puede realizar las búsquedas por palabras en el campo autor, palabras en el campo tutor, palabras en el campo título, licenciatura, palabras en el área de palabras clave, palabras que se encuentren en el resumen e inclusive por las palabras que se encuentran en el documento PDF. Para la gestión de los

documentos se desarrolló un módulo de administración del sistema en el cual, además de las funcionalidades que posee el público en general, se podrán eliminar documentos, modificar información de tutores, recuperar o cambiar clave, agregar documentos al sistema y visualizar el estatus del repositorio (consultando la cantidad de documentos para consulta en el sistema). Para la agregación de documentos se emuló una integración con CONEST de manera que BUBAG reciba los datos de los usuarios que pueden almacenar documentos en el sistema, por lo tanto, se puede agregar los documentos electrónicos por dos (2) vías:

- El administrador del sistema agrega los documentos.
- Por medio de un enlace con CONEST se reciben los datos de los estudiantes y se les da acceso a BUBAG para que agreguen su documento.

El presente trabajo le otorga un valor agregado a la Biblioteca Alonso Gamero ya que se podrán publicar y gestionar la Producción Intelectual de la Facultad de Ciencias de la Universidad Central de Venezuela.

El sistema desarrollado provee adicionalmente la posibilidad de que cada persona decida si permite que su Producción Intelectual pueda ser consultada, de lo contrario el sistema sólo mostrará información básica de su trabajo. Cabe destacar que la interacción de los usuarios con el sistema es intuitiva y no se requieren conocimientos computacionales profundos. Se realizaron pruebas de aceptación con el cliente, el cual le otorgó la mayor puntuación posible en cada uno de las apartados incluidos en el formulario y sus resultados se encuentran reflejados en el marco aplicativo.

Recomendaciones

- Se recomienda continuar con ésta investigación para ampliar el alcance de la aplicación al resto de los productos intelectuales de la Facultad de Ciencias.

- Colocar en producción la aplicación en un servidor de altas prestaciones, por las exigencias de almacenamiento y procesamiento, además de requerir una alta velocidad de transferencia.

- Agregar la funcionalidad correspondiente en CONEST para la integración con BUBAG.

- Se recomienda al administrador de base de datos que realice un respaldo de los datos, si es posible, mensualmente.

Referencias Bibliográficas y Digitales

Adobe Systems (2009). [Página Web en línea]. Disponible en: http://www.adobe.com/ [Consultada: 25/02/2010].

ALEGSA (2008). [Página Web en línea]. Disponible en: http://www.alegsa.com.ar/Dic/aplicacion%20web.php/ [Consultada: 18/01/2010].

Amber, S. (2009). [Página Web en línea]. Disponible en: www.ambysoft.com/unifiedprocess/agileUP.html/ [Consultada: 23/03/2010].

Ambler, S. (2005). Agile Modeling. [Página Web en línea]. Disponible en: www.agilemodeling.com [Consultada: 25/01/2010].

Aplicacion Web. (2009). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Aplicación_web/ [Consultada: 18/04/2010].

Características de PHP. (2007). [Página Web en línea]. Disponible en: http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP [Consultada: 18/04/2010].

Ciclo de vida. (2001). [Página Web en línea]. Disponible en: http://www.dcc.uchile.cl/~jbarrios/servlets/vida.html [Consultada: 22/03/2010].

Cliente-Servidor. (2010). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Cliente-servidor [Consultada: 26/01/2010].

Definicion de CSS. (2009). [Página Web en línea]. Disponible en: http://www.masadelante.com/faqs/css [Consultada: 18/01/2010].

Duque, N. (s.f). [Página Web en línea]. Disponible en: http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap6-3.html [Consultada: 08/02/2010].

Java Servlets. (2009). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Java_Servlet [Consultada: 12/03/2010].

JavaServer Pages. (2008). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/JavaServer_Pages [Consultada: 25/01/2010].

Lopez, F. R. (2008). [Página Web en línea]. Disponible en: http://www.csae.map.es/csi/silice/Global71.html [Consultada: 27/02/2010].

Matsumoto, Y. (2003). [Página Web en línea]. Disponible en: http://www.informit.com/articles/article.aspx?p=18225 [Consultada: 22/04/2010].

Modelo Vista Controlador. (2009). [Página Web en línea]. Disponible en: http://www.webtutoriales.com/tutoriales/programacion/modelo-vista-controlador.54.html [Consultada: 22/04/2010].

Modelo Vista Controlador. (2010). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador [Consultada: 29/01/2010].

PHP: Nociones Básicas. (2005). [Página Web en línea]. Disponible en: http://www.wikilearning.com/curso_gratis/php_nociones_basicas-nociones_basicas/6092-2 [Consultada: 03/03/2010].

Proceso Unificado de Rational. (2010). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational [Consultada: 23/02/2010].

RDF/XML. (2004). [Página Web en línea]. Disponible en: http://www.w3.org/TR/REC-rdf-syntax [Consultada: 23/02/2010].

Ruby. (2009). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Ruby [Consultada: 25/03/2010].

Ruby on Rails. (2009). [Página Web en línea]. Disponible en: http://es.wikipedia.org/wiki/Rails [Consultada: 27/03/2010].

Ruby on Rails:download. (s.f.). [Página Web en línea]. Disponible en: http://rubyonrails.org/download [Consultada: 23/02/2010].

Sanchez, J. (2008). Tesis de Grado de Pregrado. Elaboración de un Prototipo de Buscador para Documentos Académicos de la Facultad de Ciencias. Universidad Central de Venezuela, octubre, 2008.

VALLE, J. G. (2005). [Página Web en línea]. Disponible en: http://temariotic.wikidot.com/la-arquitectura-cliente-servidor [Consultada: 28/04/2010].

W3C. (2010). [Página Web en línea]. Disponible en: http://www.w3.org/ [Consultada: 18/02/2010].

XHTML vs HTML. (2005). [Página Web en línea]. Disponible en: http://www.baluart.net/articulo/xhtml-vs-html-sus-principales-diferencias [Consultada: 13/03/2010].

Rodríguez, A. (2009). [Página Web en línea]. Disponible en: http://gestiondelconocimientoffyluanl.blogspot.com/ [Consultada: 26/03/2010].

W3support. (2009). [Página Web en línea]. Disponible en: http://es.w3support.net/index.php?menu=2 [Consultada: 28/04/2010].

Welie. (2010). [Página Web en línea]. Disponible en: http://www.welie.com/ [Consultada: 29/05/2010].

Web de la ACM. (2010). [Página Web en línea]. Disponible en: http://www.acm.org/ [Consultada: 15/05/2010].