UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

"ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN ADMINISTRADOR DE DOCUMENTOS DE PROYECTOS VÍA INTERNET/INTRANET

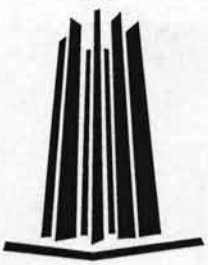
T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

J U Á R E Z Z A R C O R O C I O



DIRECTOR DE TESIS:

ING. GLADIS E. FUENTES CHÁVEZ

SANJUAN DE ARAGÓN, ESTADO DE MÉXICO **2004**

Quiero agradecer y dedicar esta Tesis, en primer lugar a **Dios**, por que siempre ha estado conmigo en todo momento y por permitirme estar aquí y vivir esta experiencia.

A mis Padres:
Ma. de Lourdes y José Luis, Por apoyarme en todo momento, por su amor, por que siempre están conmigo y me alientan a seguir adelante, son un ejemplo admirable. Gracias por todo.

A mis Abuelitas:
Margarita y Ana, Por mostrarme fortaleza, sabiduría, amor y comprensión.

A mis hermanos:
Karina, José Luis y German. Por ayudarme a no crecer sola, gracias por su apoyo incondicional, por sus consejos y opiniones, los quiero mucho.

A mis amigos:
Arturo, Emilio, Andrés, Jonatan, Abraham, Paty, Crystal.
Que siempre están cuando los necesito, gracias por su apoyo en verdad se los agradezco infinitamente, gracias por permitirme pasar los mejores momentos en la escuela y por ser como son.

A todos mis familiares y compañeros de la ENEP por compartir conmigo esta experiencia.

Quiero agradecer a la Universidad Nacional Autónoma de México, por la enorme oportunidad que me ha brindado.

También a la Escuela Nacional de Estudios Profesionales campus Aragón por darme la formación profesional.

Al Instituto Mexicano del Petróleo, por haberme apoyado con la realización del proyecto de Tesis, por brindarme el equipo y la capacitación necesarios.

A mis profesores que a lo largo de esta carrera me brindaron sus conocimientos.

A Gladis por ser mi asesor, por su paciencia, su calidad humana y su disponibilidad al compartir sus conocimientos.

ÍNDICE

Capítulo I

Introducción

1.1 Objetivos

1.1.1 Generales

Este proyecto tiene como finalidad investigar, diseñar, probar e implantar herramientas de Internet que apoyen la Administración de Documentos en un Proyecto, en un entorno colaborativo que permita intercambiar información a especialistas del Instituto Mexicano del Petróleo (IMP).

1.1.2 Específicos.

1) Diseñar e implantar una página Web, así como desarrollar las aplicaciones en entorno Intranet / Internet con el lenguaje de programación Java y sus componentes entre los cuales están: los Servlets y la tecnología JSP(JavaServer Pages); el lenguaje de hipertexto HTML y JavaScript.
2) Con la implementación del Administrador de Documentos se permitirá tener una comunicación y manejo de información a través de Internet sin importar que plataformas estén utilizando los usuarios.
3) Generar un entorno visual para un sistema multiusuario de tal forma, que el usuario final pueda acceder a toda aquella información requerida de acuerdo a su perfil y actuar directamente sobre ella, independiente de los demás usuarios.
4) Crear herramientas para el trabajo colaborativo que faciliten el intercambio de información en las diferentes actividades de los usuarios.

1.2 Justificación del tema.

Este proyecto se realiza con la finalidad de obtener un conocimiento más amplio acerca de lo que son las aplicaciones que existen en Internet, así como de las herramientas que facilitan la comunicación y de los servicios que puede ofrecer Internet mediante los programas de aplicación.

Son cada día más las empresas formadas para proveer servicios de conexión y también son cada vez más las que utilizan esta tecnología para anunciar sus productos o servicios a través del mundo. Comercialmente, la Web proporciona oportunidades que ningún otro medio puede ofrecer como: cobertura mundial (dentro de Internet); bajo costo; e interactividad con los clientes potenciales. Las instituciones de gobierno, las científicas y educativas, las no gubernamentales, las iglesias, etc., también aumentan cada día su participación y cada vez se coloca

más información en la Web como, por ejemplo: catálogos de bibliotecas; reportes de investigación, publicaciones periódicas electrónicas; bancos de información estadística, etc.

La anunciada integración de la informática y las telecomunicaciones está ocurriendo en los años actuales, e Internet, juntamente con las tecnologías y herramientas asociadas, está teniendo un papel importante en su aceleración. Una computadora personal puede, a través de Internet, acceder y transmitir información de forma relativamente sencilla. Esta capacidad de comunicación se está mejorando en gran medida, incorporando posibilidades de comunicación interpersonal, ya sea con un usuario o con varios.

Se estima que esta integración impactará de forma muy importante en el modo de trabajo de los profesionales de diversas áreas. Hasta ahora, esta colaboración estaba limitada por la tecnología, ocurriendo a menudo sólo dentro de un mismo departamento o grupo.

La transferencia de información y la administración de documentos son actividades básicas para cualquier grupo de trabajo. Por ejemplo, una aplicación que esta conectada a una base de datos compartida permite a sus miembros ingresar, almacenar, acceder y modificar la información, la que puede incluir texto, gráficos, audio y vídeo. La información contenida en la base de datos puede ser identificada como de uso privado o público según se requiera. Para ello, el sistema asigna a los usuarios diferentes permisos (lectura, escritura, eliminación) según determinados perfiles o roles dentro del grupo.

Es por ello que el Administrador de Documentos a través de Internet/Intranet es de gran beneficio para el grupo de trabajo que participa en un proyecto, en el cual podrán interactuar con sus documentos de manera clara y sin problemas en la plataforma o navegador que utilicen, con la ayuda de esta tecnología es posible acceder fácil a la información, así como de enviar y recibir la documentación de manera que se pueda acceder a esta de acuerdo con los perfiles de usuario correspondientes.

Además, en la creación del Administrador de Documentos se utilizarán lenguajes de programación como Java que es un lenguaje Orientado a Objetos, Java es un lenguaje seguro y portable que soporta las construcciones de la programación orientada a objeto, la tecnología JSP está basada en el lenguaje de programación Java encaminada al desarrollo de sitios web para incorporar contenido dinámico a las páginas, lo cual hace al proyecto más atractivo al construirse mediante estas plataformas.

1.3 Instituto Mexicano del Petróleo

El Instituto Mexicano del Petróleo (IMP) —organismo público descentralizado del Gobierno Federal, sectorizado en la Secretaría de Energía — es una importante plataforma para la investigación científica y el desarrollo tecnológico al servicio de las industrias petrolera, petroquímica básica, petroquímica derivada y química.

El IMP se ha transformado en una institución moderna y competitiva que se propone asegurar el fortalecimiento de la investigación y el desarrollo tecnológico, con programas y proyectos de investigación .Orientar sus esfuerzos hacia soluciones con servicios integrados a plena satisfacción de Petróleos Mexicanos, su cliente principal, y fortalecer sus competencias institucionales.[1]

Para el IMP, el intercambio de información actualizada y la administración de documentos son actividades básicas para cualquier grupo de trabajo. Grupos de personas trabajando en conjunto, necesitan medios para compartir y almacenar la información relevante para sus objetivos y las tareas que deben realizar.

La Estructura Organizacional del Instituto Mexicano del Petróleo se muestra en la siguiente figura:

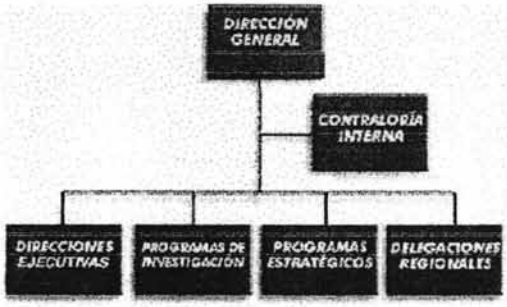


Figura 1. Estructura organizacional el IMP

La estructura organizacional del Instituto Mexicano del Petróleo está conformada por Competencias, que a su vez se dividen en Coordinaciones, éstas son las encargadas de llevar a cabo diferentes proyectos en el campo de la investigación y desarrollo.

[1] Artículo tomado del URL: http://www.imp.mx/imp/historia/

Un Administrador de Documentos de Proyectos, a través de Internet es de gran beneficio para grupos de trabajo que participan en proyectos, en el cual podrán interactuar con sus documentos de manera clara y sin problemas en la plataforma o navegador que utilicen, con la ayuda de esta tecnología es posible acceder a la información de su proyecto, así como de enviar y recibir la documentación de manera que se pueda acceder a esta dentro de una aplicación web.

1.4 Pasos que se realizan en un proyecto en el IMP

La Administración de Programas y Proyectos debe permitir la planeación, programación, presupuestación, ejecución, control, evaluación y cierre de los programas y proyectos, para lograr la satisfacción del cliente y cumplir con los objetivos estratégicos de la institución.[2]

En el IMP los pasos que se requieren para realizar un proyecto se muestran en la siguiente figura:

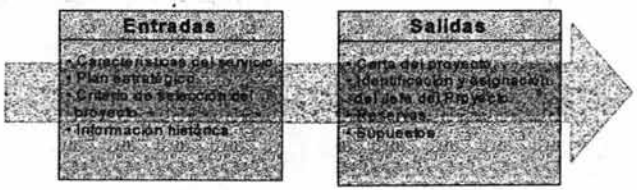


Figura 2. Fases de un proyecto

Supuestos, Restricciones y Factores Críticos de Éxito del Proyecto

La identificación y documentación de supuestos, restricciones y factores de éxito del proyecto es parte del proceso de planeación. El plan del proyecto está fundamentado en éstos y por lo tanto deben comunicarse por escrito como un anexo del plan. Los supuestos, restricciones y factores críticos de éxito son básicos para la administración del riesgo, programación y desarrollo del presupuesto.

[2]Artículo tomado del URL: http://intranet.imp.mx/apoyo/calidad/sic/ap/

Entre los supuestos, restricciones y factores críticos de éxito más comunes están:

- Disponibilidad y capacidad de recursos
- Dependencias de equipo, servicios y otros entregables que serán suministrados por grupos externos
- Disponibilidad de usuarios y otros participantes para definición de requerimientos y revisión y aprobación de entregables
- Factibilidad técnica
- Confiabilidad en herramientas y métodos
- Curvas de aprendizaje

La definición del proyecto es una conexión obligatoria con todos los objetos creados dentro de un proyecto. Contiene datos que se utilizan para todo el proyecto, datos organizacionales y parámetros de planeación.

1.4.1 Tareas de un proyecto

Un proyecto es un esfuerzo temporal llevado a cabo para crear un producto o servicio único para alcanzar un objetivo bajo restricciones de costo y tiempo.

Un proyecto tiene:

- Un principio y un final
- Un conjunto específico de objetivos
- Criterios de calidad medibles
- Muchas actividades interrelacionadas
- Recursos limitados
- Costo y tiempo definido

La jerarquía del plan de trabajo, puede dividir el proyecto en los siguientes niveles:

- Plan Maestro (arriba del nivel del proyecto)
- Plan de Liberación (producto interno)
- Plan de Fase
- Plan de Equipo o Grupo
- Plan de Trabajo Individual

Estos niveles dividen las tareas del proyecto jerárquica y organizacionalmente.[3]

[3]Artículo tomado del URL: http://intranet.imp.mx/siiimp/AdmnProy.htm

Además se deben desarrollar los siguientes planes:

- Plan de Desarrollo o Desempeño
- Plan de pruebas
- Plan de Conversión o Entregas

Para llevar a cabo estos procesos es necesario contar con una buena comunicación con todos los integrantes de un proyecto, es por eso necesario contar con una herramienta que permita tener contacto con todos los miembros de un equipo de trabajo para permitir un flujo de información y siempre estar al corriente con los procesos de desarrollo del proyecto.

1.5 Herramienta para la Administración de Documentos

Una herramienta útil para un proyecto es un Administrador de Documentos, en primera instancia definiremos lo que es un documento.

Documento.

Existen cientos de definiciones diferentes. Por ejemplo:"Un documento es cualquier depósito de información coherente que ha sido ensamblado para la comprensión humana.
Un documento es simplemente una taquigrafía: una anotación que nos hemos inventado para guardar información que no queremos olvidar".[4]

Ahora veremos lo que es un Administración de Documentos.

Gestión de Documentos es el proceso de manejar los documentos en forma sistemática de acuerdo a su ciclo de vida: desde su creación inicial, el proceso de revisiones, el almacenamiento, difusión y utilización, hasta su destrucción.

La Gestión de Documentos es una tarea clave en todas las organizaciones y afecta a todas sus partes. Con una inversión relativamente baja puede edificarse rápidamente un sistema de Gestión de Documentos que sea funcional, dependiendo de las características que requiera el sistema puede construirse con aplicaciones que tienen componentes que se adaptan a las necesidades de la empresa, y que están en el mercado a bajo costo.

En el proceso de introducir un nuevo sistema de Gestión de Documentos, frecuentemente se descubren las deficiencias en cuanto a las prácticas vigentes en el manejo de la información en la organización. Es un reto gerencial propiciar un efectivo ambiente de cambio orientado a la sustitución de los procesos

[4]Artículo tomado del URL:
http://www.cyt.net/wwwroot/knowledgemanagement/gestiondocumentos.htm

anteriores por los nuevos, y modificar los aspectos "culturales" implicados. Esto se debe a que cuando se hace un proyecto se tienen que revisar varios informes y los usuarios tienen que consultarlos, hay veces en que los usuarios no se encuentran en la misma ubicación que los demás o están de comisión que es cuando realizan un viaje a otras instalaciones y les hay veces en que tardan en llegar los documentos.

La gestión de documentos proporciona a los usuarios la capacidad de sacar el máximo provecho de sus aptitudes para controlar, estructurar, acceder y compartir la información de forma rápida, sencilla y segura.

Para cualquier organización resulta extraordinariamente beneficioso:

- Perder menos tiempo buscando información
- Convertir el conocimiento en algo verdaderamente valioso
- Incentivar la capacidad de colaboración
- Conseguir que los procesos de distribución y revisión sean más eficientes

1.5.1 Ejemplos de aplicación que utilizan un Administrador de Documentos

Existe software comercial con este tipo de características de las cuales se explicarán a continuación.

INTRA.NET

Esta primera aplicación se llama Intra.net y permite a cualquier organización que agregue y distribuya información de manera uniforme y simple.

Intra.Net es un sistema para la Administración de información que, simplifica el proceso de creación y distribución de la información, permite que las organizaciones sean más eficientes.

El sistema cuenta con varios módulos:

- Administración de la Información

En el centro del sistema hay un potente motor para el **manejo de documentos** que permite a las empresas construir una base de conocimiento interno, categorizarlo y clasificarlo, y hacerlo accesible de forma muy simple. Permite que los usuarios publiquen información directamente al sistema (Incluye IntraEditor, editor HTML) o mediante la carga (upload) de documentos a ser compartidos con otros usuarios.

Maneja efectivamente el control de versiones de documentos, sin importar el formato de los mismos. La forma de organizar la información es muy flexible de forma de adaptarse a las necesidades de cualquier empresa. Puede organizarse por Departamentos, por Sucursales, por Proyecto, por Grupos, etc., o por una combinación de las anteriores.

- Fácil administración.

No son necesarios conocimientos técnicos o de programación. Permite que los líderes de grupo, departamento o proyecto administren y personalicen la información de su sector sin necesidad de soporte por parte del área de sistemas.

- Control de acceso de múltiples niveles.

A cada usuario se le pueden asignar niveles de acceso a los diferentes sectores del sistema, con derechos a "leer", "publicar" o "modificar". Una suite integrada de herramientas de Colaboración.

- Versiones de la aplicación.

El sistema está disponible en dos formas, como una aplicación LAN(Linux/Solaris/WinNT/Win2000/WinXP) o como una aplicación rentada(ASP) accesible por Internet.[5]

Ejemplo de Álbum de fotografías

El siguiente ejemplo es de un álbum de fotografías realizada en Servlets/JSP por OOP Reserch.

Para este ejemplo, se necesita el nombre y la contraseña del usuario, solamente los usuarios registrados pueden ingresar a este sistema. Y el directorio será asignado uno por usuario, los archivos subidos serán almacenados allí. La carga de archivos o upload, acepta archivos solamente de extensiones JPG y GIF. Además, si el tamaño del archivo que suba el usuario excede los 30 KB, mandará un mensaje de error.
Los usuarios pueden ver la lista de los archivos que han subido.

Este ejemplo guarda el archivo subido con el nombre del archivo igual que el nombre del archivo original en el lado del cliente.

Como es una aplicación comercial solo se puede bajar el código fuente y probarlo por 30 días.[6]

[5] Artículo tomado del URL: http://www.bcdasociados.com

[6] Artículo tomado del URL: http://www.oop-reserch.com/

1.5.2 Aplicaciones gratuitas para la implementación de carga de documentos.

Para la construcción del Administrador de Documentos, se necesita que suban documentos los usuarios, por lo que existen en Internet aplicaciones para la subida o carga de archivos (upload) que constan de diferentes módulos para permitir el envío de un archivo desde la página *web* al servidor. En ésta tesis se utilizará alguna de estas aplicaciones para llevar acabo la construcción de una herramienta para la administración de documentos, estas aplicaciones son gratuitas de manera que se verá cual de las aplicaciones siguientes es más conveniente en el desarrollo del Administrador de Documentos.

Dentro de los cuales están:

- JspSmartupload la aplicación se encuentra en la siguiente dirección: www.jspsmart.com
- UploadServlet v. 1.7 URL: http://coldjava.hypermart.net/servlets/upload.htm
- File upload using server side Java su dirección en Internet es: http://developer.netscape.com/docs/examples/java/file_uploading.html
- Extra uploader URL: http://www.javer.narod.ru/fuplinst.htm
- Java Guru upload a file url: http://www.jguru.com/faq/view.jsp?EID=160

Algunas de estas aplicaciones tienen ciertas restricciones en cuanto a su forma de uso por lo que se deberá analizar que características tienen y verificar cual es la más favorable para esta aplicación.

Como se vio en este capítulo uno de los objetivos primordiales es la construcción de una aplicación para la Administración de Documentos vía Internet/Intranet para ello, en el siguiente apartado se presentará una descripción de las herramientas y técnicas de programación, que existen para la construcción de sitios que utilizan alguna tecnología para el desarrollo de aplicaciones Web, esto se tomará como guía para el análisis y construcción del Administrador de Documentos.

Capítulo II

Tecnología para el desarrollo de aplicaciones Web

2.1 Internet y World Wide Web

Internet es la denominada "red de redes" y se llama así a la infraestructura física de comunicación entre computadoras implementada por medio de las redes telefónicas públicas y privadas, y de una serie de protocolos y programas (software) que manejan la comunicación entre máquinas al nivel más básico.
Sobre esta infraestructura se operan diversos servicios mediante programas de aplicación con sus respectivos protocolos de comunicación, como por ejemplo, el correo electrónico (EMAIL), la transferencia de archivos (FTP), los círculos de conversación en línea (IRC) y también la transferencia de documentos de hipertexto (HTTP).

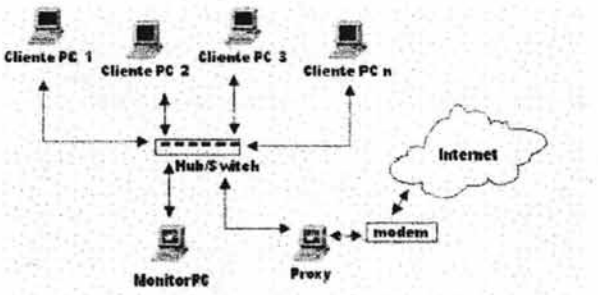


Figura 1. Esquema de Internet

El World Wide Web (W W W) es una aplicación creada por un organismo científico suizo hacia 1990, que no tiene aún un nombre propio en español, siendo usado el inglés "Web" que significa telaraña. Consiste en un protocolo de comunicación entre computadoras que hace uso de la infraestructura básica de la Internet para intercambiar documentos en formato multimedia e hipertexto desde y hacia cualquier lugar del mundo que se encuentre conectado a través de Internet.

Características principales del Web mundial

Es un protocolo de comunicación de documentos multimedia, es decir, que pueden manejarse textos, imágenes, sonidos, tablas, etc.

Tiene una Interfase de usuario gráfica, de tipo Windows. Las funciones del programa se seleccionan con el ratón haciendo clic en los iconos correspondientes.

Los documentos están organizados por "páginas" independientes, que son archivos con un nombre y una dirección particular (la del servidor correspondiente), pudiendo estar localizados en cualquier parte del mundo.

En los documentos se integran funciones de hipertexto, que consisten en realizar instrucciones mediante el ratón haciendo clic en partes del texto o de las imágenes del documento. Generalmente la instrucción consiste en llamar otro documento que guarda alguna relación con el actual.

La estructura de las páginas es muy flexible ya que por medio del hipertexto se pueden realizar conexiones a otros documentos sin depender de una estructura jerárquica. Un documento llamado desde otro documento puede estar en cualquier nivel de directorio y en cualquier localización geográfica, siendo común que desde un documento se llame a otro que está en un país diferente.

Los servidores de páginas de Web funcionan 24 horas al día (transmiten bajo pedido), enviando a los usuarios (clientes) copias de las páginas solicitadas. A diferencia de otros medios, como el radio o la televisión, el cliente también puede transmitir información al servidor por medio de formas integradas a las páginas recibidas, de manera que el protocolo permite la comunicación en ambos sentidos. No existe un control central de la comunicación en el Web. Cualquier individuo u organización puede mantener un servidor de web y colocar páginas. Esto ha permitido un crecimiento muy rápido y un desarrollo de contenido que refleja las características de la sociedad en que está ubicado, sin limitaciones burocráticas o políticas.

Su operación tiene un costo relativamente bajo, ya que es una aplicación que hace uso de infraestructura previamente creada. Para tener acceso individual solamente hacen falta: una computadora personal; programas que son en muchos casos distribuidos gratuitamente; y el pago de una conexión a la red por un proveedor de servicio, a un costo que es accesible para un número importante de personas.

Estas características han hecho que el Web se convierta en la aplicación más importante de la Internet en nuestros días, inclusive absorbiendo a otras como el correo electrónico y la transferencia de archivos.

Uno de los servicios de Internet importante es el protocolo de transferencia de hipertexto (http). El http puede leer e interpretar ficheros de una máquina remota: no sólo texto sino imágenes, sonidos o secuencias de vídeo. El http es el protocolo de transferencia de información que forma la base de la colección de información distribuida denominada World Wide Web[1].

World Wide Web (también conocida como Web o WWW) es una colección de ficheros, denominados lugares de Web o páginas de Web, que incluyen información en forma de textos, gráficos, sonidos y vídeos, además de vínculos con otras páginas. Las páginas son identificadas por un localizador universal de recursos (URL[2], siglas en inglés) que especifica el protocolo de transferencia, la dirección de Internet de la máquina y el nombre de la página. Los programas informáticos denominados exploradores —como Navigator, de Netscape, o Internet Explorer, de Microsoft— utilizan el protocolo http para recuperar esas páginas Web. Continuamente se desarrollan nuevos tipos de ficheros para la WWW, que contienen por ejemplo animación o realidad virtual. Hasta hace poco había que programar especialmente los lectores para manejar cada nuevo tipo de archivo. Los nuevos lenguajes de programación (como Java[3], de Sun Microsystems) permiten que los exploradores puedan cargar programas de ayuda capaces de manipular esos nuevos tipos de información[4].

2.2 HTML

Las siglas corresponden con la definición "Lenguaje para marcado de hipertexto". Más claro aún, se trata de un lenguaje para estructurar documentos a partir de texto en World Wide Web. Este lenguaje se basa en etiquetas (instrucciones que le dicen al texto como deben mostrarse) y atributos (parámetros que dan valor a la etiqueta).

La mayoría de los documentos tienen estructuras comunes (títulos, párrafos, listas...) que van a ser definidas por este lenguaje mediante etiquetas. Cualquier cosa que no sea una etiqueta es parte del documento mismo.

Este lenguaje no describe la apariencia del diseño de un documento sino que ofrece a cada plataforma que le dé formato según su capacidad y la de su navegador (tamaño de la pantalla, fuentes que tiene instaladas...). Por ello y para no frustrarnos, no debemos diseñar los documentos basándonos en como lucen en nuestro navegador sino que debemos centrarnos en proporcionar un contenido claro y bien estructurado que resulte fácil de leer y entender.

---

[1] World Wide Web, mecanismo proveedor de información electrónica para usuarios conectados a Internet.

[2] URL, acrónimo de Universal Resource Locator, método de identificación de documentos o lugares en Internet

[3] JAVA, lenguaje de programación orientado a objetos desarrollado por la empresa Sun Microsystems en 1995 y que se ha extendido ampliamente en World Wide Web.

[4] Artículo tomado del URL: http://www.iteso.mx/biblio/formacion/internet.htm

HTML tiene dos ventajas que lo hacen prácticamente imprescindibles a la hora de diseñar una presentación web: Su compatibilidad y su facilidad de aprendizaje debido al reducido número de etiquetas que usa.
Básicamente, los documentos escritos en HTML constan del texto mismo del documento y las etiquetas que pueden llevar atributos.
Esto llevado a la práctica, vendría a ser:

<etiqueta>texto afectado**/etiqueta>**

La etiqueta del principio activa la orden y la última (que será la del principio precedida del signo /) la desactiva. No todas las etiquetas tienen principio y final.

Documento HTML

Estructura básica de un documento HTML: Cabecera y cuerpo del documento

Tres son las etiquetas que describen la estructura general de un documento y dan una información sencilla sobre él. Estas etiquetas no afectan a la apariencia del documento y solo interpretan y filtran los archivos HTML.

- <HTML>: Limitan el documento e indica que se encuentra escrito en este lenguaje.
- <HEAD>: Especifica el prólogo del resto del archivo. Son pocas las etiquetas que van dentro de ella, destacando la del titulo <TITLE> que será utilizado por los marcadores del navegador e identificará el contenido de la página. Sólo puede haber un título por documento, preferiblemente corto aunque significativo, y no caben otras etiquetas dentro de él. En *head* no hay que colocar nada del texto del documento.

- <BODY>: Encierra el resto del documento, el contenido.

- <H1>, <H2>, <H3>...: Titulares. Sirven para dividir el texto en secciones. Se pueden definir seis niveles de titulares, el texto que deseamos que sea un titular se pone entre las etiquetas <H1> Titular </H1>. Se definen mediante las etiquetas <H1>.....</H1> hasta <H6>.....</H6>

- <P>: Párrafos. En principio, sin entrar en detalles de alineación u otras características, digamos que se definen por las etiquetas <P>.....<P>. Esta etiqueta, en un principio, se diseñó para saltar de párrafo por lo que puede ir sola "<P>" al final de un texto indicando que a continuación se quiere una línea en blanco aunque le recomendamos que se acostumbre a utilizarla abriéndola y cerrándola.

-
: Saltos de línea. Esta etiqueta sirve para realizar un salto de línea, puede poner tantas como desee y realizará un salto de línea por cada una de ellas.

- <!-- -->: Comentarios. Son directivas que nunca se mostrarán a través del navegador y que le servirán para recordatorios en futuras revisiones del documento.

Ejemplo:

```
<HTML>
<HEAD>
<TITLE>Ejemplo </TITLE>
</HEAD>
<BODY>
<H1>Mi primera página</H1>

<!-- Aquí va un comentario que no es
interpretado por el navegador -->

<P>Hola mundo, esta es una página con titular,
que tiene también un párrafo y unos cuantos
saltos de línea. </P>

Uno<br>
Dos<br>
Tres<br>
</BODY>
</HTML>
```

Figura 2. Ejemplo de código HTML

Creación de enlaces

Lo característico del lenguaje HTML es el poder generar vínculos de hipertexto para enlazar con ellos todos sus documentos en web.

Para generar un enlace a otro documento necesitamos el nombre de un archivo (o su dirección URL) y el texto que servirá de punto de activación del otro documento. Este segundo elemento será el que se ve en pantalla y que se servirá del primero para saltar de documento.

Los enlaces se generan mediante la etiqueta <A>.....</A> y, a diferencia de los vistos anteriormente, llevará siempre dentro de la etiqueta un atributo ya sea <A HREF=""> o <A NAME="">.

- <A HREF="URL">.....</A>: Es el más habitual de los atributos y sirve para saltar entre diferentes URLs. De momento veremos:
  - Saltar en una presentación del archivo 1 al archivo 2: En el archivo 1 incluiremos la directiva <A HREF="archivo2.html">Siguiente página
  - Saltar de nuestra presentación a otra presentación web llamada www.bienvenidos.es: <A HREF="http://www.bienvenidos.es">Visita esta página</A>

- <A NAME="parte1">Primera parte</A>: Utilizamos el atributo *name* para dar nombre a una sección de nuestro documento. Posteriormente, cuando en nuestro documento queramos incluir un vínculo a dicha sección escribiremos: <A HREF="#parte1">Ir a la primera parte</A>

URL: Localizador Universal de Recursos

Los URL son las direcciones de las informaciones que buscamos en Internet. Los URL constan de tres partes:

- Protocolo: Es el programa que utilizará el navegador para obtener el archivo elegido. Por ejemplo: HTTP, FTP, Gopher (El antiguo y ya casi extinguido sistema gopher se usaba, antes de que existiera la Web, para localizar documentos en Internet[5]).
- Nombre del host: Se trata del sistema donde se encuentra almacenada la información que buscamos.
- Ruta del fichero: Se trata de la ubicación del archivo dentro del host. http://www.bienvenidos.es/publico/saludos.html

Entre los principales tipos de URL destacan:

- **HTTP**: Son los más populares ya que son los utilizados por los servidores de WWW para mandar documentos a través de Internet.
- **FTP**: Se utilizan para apuntar hacia los archivos que estén en servidores que usan el protocolo FTP (File Transfer Protocol). Este protocolo es normalmente utilizado para enviar y recibir ficheros. Es el protocolo que se usa para enviar nuestras páginas al servidor de Internet. Como ya se puede imaginar en estos servidores se almacenan los archivos que forman parte de nuestra presentación web.

[5]Artículo tomado del URL: http://www.psicobyte.com/html/taller/url.html

- **File**: Apuntan hacia archivos contenidos en el mismo disco que se encuentra el navegador. No resulta muy interesante poner estos URL en nuestras presentaciones puesto que otra persona que desde otro sistema apunte hacia este URL, generalmente fallará en su intento y no podrá tener acceso a él.
- **Mailto**: Se usa para mandar correos electrónicos. Cuando seleccionamos este tipo de URL se abre la aplicación de correo electrónico de nuestro ordenador para enviar un correo a la dirección hacia la que apunta el URL. La forma estándar es: mailto:webmaster@bienvenidos.es
- **News**: Son URL de grupos de noticias, en estos servidores se almacenan mensajes en los que se discuten sobre diferentes temas. [6]

[6] Artículo Tomado del URL: http://www.webestilo.com/html/cap1a.phtml

2.3 Java

2.3.1 Historia de Java

Con Java, SunMicrosystems estableció el primer lenguaje de programación que no fue lanzado a cualquier sistema operativo o microprocesador. Las aplicaciones en Java pueden correr en cualquier plataforma, eliminando uno de los mayores dolores de cabeza de los usuarios de computadoras: incompatibilidad entre sistemas operativos y versiones de sistemas operativos.

Al principio el proyecto se enfocaba en construir software para electrónicos. Todo empezó en 1990 cuando el equipo de SunMicrosystems desarrolló algunos conceptos para la dirección de la nueva tecnología. Fue el enfoque principal aparatos del hogar: VCR, el horno de microondas, sistemas de seguridad y sistemas de sonido. Como fuera, cada producto necesitaba su propia interfaz. En otras palabras, para controlar los aparatos, los clientes necesitaban tener controles remotos y entender su programación. El prototipo inicial fue llamado StarSeven (*7), el cual fue un PDA (Personal Digital Assistant). El motivo del fracaso del *7 fue que el proyecto era demasiado caro en su elaboración, lo cual nos conduce a pensar que se adelanto a su época.

El equipo de trabajo fue llamado el "Green Team", el cual estaba formado por; Patrick Naughton, James Gosling y Mike Sheridan. Gosling fue el iniciador del desarrollo de lo que ahora es un nuevo lenguaje de programación. Esto debido a que C++ era muy bueno para la velocidad, pero no aseguraba la confiabilidad. En los aparatos electrónicos es más importante la confiabilidad que la velocidad. Esto sucedió en 1991. Un año después, el "Green Team" desarrolló un ayudante de mano (StarSeven); sin teclado, botones y con una amigable interfaz. Se podía tocar la interfaz del PDA y controlaba la acción de la interfaz. Pero como ya lo mencionamos el proyecto era mucho para la época y su elaboración implicaba muchos gastos. Al principio el nuevo lenguaje fue llamado Oak, pero su nombre fue cambiado debido a que ya existía un producto con ese nombre. La idea del nuevo nombre surgió en 1995, cuando los involucrados se encontraban en un café llamado Java, entonces ese fue el nombre adoptado y el que continúa hasta ahora y seguirá por mucho tiempo.
Como sea, Bill Joy, uno de los co-fundadores de SunMicrosystems, vio la oportunidad para Oak en el emergente mundo de Internet. Su idea fue lanzar a Java a Internet.

Hoy la visión de Bill Joy es una realidad. Pero aún SunMicrosystems se preguntaba como hacer del nuevo producto dinero y una rápida comercialización. Un gran paso fue tomado el 7 de Diciembre de 1995, cuando Microsoft demostró su interés por Java, para ser ocupado en Internet Explorer.[7]

[7] Artículo tomado del URL:
http://www.iespana.es/marcoescom/oldest/sexto/poo2/pooder00.htm

Características generales de Java

La sintaxis de Java deriva de la de C/C++, simplificándola y ampliándola.

Java se puede utilizar de varias formas:

- Creando aplicaciones independientes
- Creando applets (programas que viajan a través de la Internet y se ejecutan en un browser).
- Creando Servlets (programas que se ejecutan en un servidor).

Java es un lenguaje enteramente orientado a objetos y clases. Una clase es agrupación de variables (datos) y de los métodos (funciones) que manipulan esas variables.
La programación algorítmica tradicional está centrada en las funciones. La programación orientada a objetos está centrada en los datos, es decir en la información.

Este cambio de enfoque se traduce en programas más estructurados, más fáciles de desarrollar, de mantener y de re-utilizar.
Una clase es como un nuevo tipo de dato. Ejemplos: clase Rectángulo, Alumno, Cliente.

Un objeto es un ejemplar concreto de una clase. Ejemplos: todos los rectángulos que se quieran dibujar, todos los alumnos que se quieran matricular, los distintos clientes, etc.

En Java todos los datos y todos los métodos pertenecen siempre a una clase (con la excepción de las variables locales que pueden crear los métodos)

Java maneja dos tipos de datos: tipos primitivos y referencias.

Tipos primitivos de variables: bolean, char, byte, short, int, long, float y double (Por la frecuencia con que se utilizan tienen unas reglas un poco especiales).

Las referencias son nombres de objetos de una clase determinada.

Características de Java como lenguaje orientado a objetos:

- Encapsulación. Se puede regular el acceso a los miembros (variables y métodos) de una clase, declarándolos como public, private, protected y package.
- Herencia. Una clase puede derivar de otra, heredando sus variables y métodos, y añadiendo o redefiniendo algunas variables y métodos nuevos.

Java dispone de una jerarquía de clases en la que todo deriva de la clase Object. En Java no hay herencia múltiple.

- Polimorfismo. Se puede tratar a una colección de objetos de distintas clases, similares pero distintos, de una forma unificada. Esto simplifica la programación y el mantenimiento de los programas.

En Java no se parte de cero: El JDK[8] proporciona un gran número de clases preprogramadas, a partir de las cuales se construyen las aplicaciones mediante el mecanismo de la herencia.

Variable de entorno CLASS .Sirve para poder encontrar los ejecutables de Java (compilador, Java Virtual Machine, etc.)

Se puede establecer de modo permanente o para una sesión

Variable de entorno CLASSPATH .Sirve para encontrar clases o librerías de usuario, cuando están en un directorio distinto del directorio actual

Clases, interfaces, ficheros y packages

Las clases de Java pueden ser public y package (por defecto).

Las clases public pueden ser accedidas por cualquier otra clase. Las clases package pueden ser accedidas sólo por otras clases del package. Cada clase public se define en un fichero cuyo nombre es el mismo que el de la clase con la extensión *. java.

Cuando un fichero filename.java es compilado con javac se obtiene un fichero filename.class.

El package de una clase se define introduciendo al comienzo del fichero la sentencia: package NombrePackage.

Los packages se pueden agrupar de forma jerárquica. La jerarquía de packages se refleja en la jerarquía de directorios.

Una interfase es un conjunto de declaraciones de métodos (indicación del valor de retorno, nombre y número y tipo de los argumentos). Cuando una clase

[8]JDK es el acrónimo de "Java Development Kit", es decir Kit de desarrollo de Java. Se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java. Desarrollado por Sun Microsystems, los creadores de Java.
Url: http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/A_I.htm

implementa una interfase está obligada a definir (dar el código) todos los métodos de la interfase.

Las interfaces se definen también en ficheros con mismo su nombre y la extensión *. java.

Las interfaces permiten que clases muy separadas en la jerarquía de clases de Java se comporten de una forma similar. Realizan una función similar a la de la herencia múltiple de C++.

2.3.2 Programación Orientada a Objetos

Cada una de las "variables" (instancias) declaradas de una clase determinada recibe el nombre de objeto. Cada objeto tiene sus propias copias de las variables miembro de la clase. No tiene copia particular de las funciones miembro, sino el derecho a utilizarlas sobre sus variables miembro.

Un package es una agrupación de clases que comparten un mismo directorio.
Las variables y funciones miembro de una clase pueden ser public, private, protected y package, según se permita o no el acceso a ellas desde el exterior de la clase, se permita acceder a las clases derivadas o a las clases del mismo package.

La encapsulación de Java consiste en, mediante el carácter public, private, protected y package de sus miembros, controlar el acceso que los usuarios tienen a la información y métodos contenidos en la clase.
De ordinario, los usuarios sólo pueden acceder a las variables miembro de la clase por medio de las funciones públicas. En otras palabras, los aspectos internos de la clase están ocultos al usuario, que sólo conoce el aspecto externo (la declaración) de los mismos[9].

[9]Artículo tomado del URL:
http://www1.ceit.es/Asignaturas/Informat2/Clases/Clases9899/Clase01/JavaIntro/tsld002.htm

2.4 Servlets y JSP (JavaServer Pages)

2.4.1 Que es un Servlet

Introducción a los Servlets

Los Servlets son módulos Java que nos sirven para extender las capacidades de los servidores web. Aunque es una definición un poco ambigua los Servlets son programas para los servidores, mientras que los applets son programas para los clientes y los middlets los programas para microdispositivos.

Dentro de una evolución cronológica los Servlets son la siguiente etapa de los CGI[10]. En algunas bibliografías son referenciados como CGI de 2ª generación, la cual comparten con lenguajes como ASP, PHP y JSP (que al fin y al cabo son Servlets).

El uso de los Servlets viene a ser en un tanto por ciento, una ayuda en el desarrollo de páginas web dinámicas (en contenido y diseño) apoyándose además en la potencia que nos proporciona el lenguaje Java.

Podremos desarrollar desde un simple Servlet que nos muestre una página web simple, saludándonos hasta uno que se conecte a una base de datos utilizando un pool de conexiones, encriptando la información en su envío, accediendo a bases de datos distribuidas y manteniendo su información de forma persistente en un EJB. Todo ello para conseguir una información dinámica. A partir de aquí las posibilidades son "infinitas".

Describir un Servlet es como describir una maquina de estados. Desde el momento que inicializamos el Servlet hasta que el Servlet es destruido, éste, pasará por una serie de estados dependiendo de cada una de las situaciones ante las que se encuentre.

Muy por encima podríamos decir que la secuencia de acciones que se producen en un servlet son las siguientes. La primera vez que realicemos una petición sobre el servlet se ejecutará un método de inicio, denominado **init,** en el cual inicializaremos las variables generales. Una vez que nos hemos inicializado nos pondremos a la escucha en espera de peticiones. Cada una de las peticiones que recibamos serán atendidas en hilos de ejecución diferentes, a no ser que indiquemos lo contrario. Dependiendo de como lleguen los datos (mediante post o get) al servlet se ejecutará un método u otro **doPost o doGet.** Por último el servlet tendrá un estado de finalización en el cual eliminará las variables creadas en su inicialización, conexiones a bases de datos,... este el método **destroy.**

---

[10] CGI (Common Getaway Interface). Interfase Común de Pasarela. Interfase de intercambio de datos estándar en WWW a través del cual se organiza el envío de recepción de datos entre visualizadores y programas residentes en servidores WWW.

A la hora de codificar lo primero que debemos de saber es que nuestro servlet deberá de heredar de la clase HttpServlet la cual contendrá todos los métodos necesarios para generar un servlet. Dicha clase la podemos encontrar en el **paquete javax.servlet.**

```
import javax.servlet.*;
public class MiServlet extends HttpServlet {}
```

Solamente deberemos de sobrescribir aquellos métodos que consideremos oportunos a implementar en el servlet. Si por ejemplo no necesitaremos realizar ninguna inicialización, no haría falta el rescribir el método **init.**

Así un primer servlet que mostrase la frase, de "Hola Mundo" quedaría de la siguiente forma:

```
import javax.servlet.*;
import javax.servlet.http.*;
public class MiPrimerServlet extends HttpServlet {
public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
PrintWriter out; out = res.getWriter();
res.setContentType("text/html");
out.println("<html>"); out.println("<head><title>Mi Primer Servlet
</title></head>"); out.println("<body>);
out.println("<h1>Hola Mundo/h1");
out.println("</body></html>");
}
}
```

Figura 3. Código fuente de un servlet

Como se puede apreciar, la salida que se está generando es una página web (sus etiquetas). Es decir, recalcamos la idea de que generamos el contenido de una página web dinámicamente.

Características de los Servlets

Dentro de las características que presenta la plataforma de desarrollo de Servlets podemos numerar:

1. Es independiente de la plataforma en la que sé este ejecutando. Otras soluciones como ISAPI o NSAPI son dependientes de la plataforma y de los servidores donde se ejecuta haciendo muy costoso una migración en la plataforma de ejecución.

2. Ejecución multihilo. Cada una de las peticiones sobre el servlet creará una instancia que se ejecutará de manera independiente. A no ser de que le indiquemos lo contrario. El servlet permanece cargado en memoria por lo que atiende rápidamente las peticiones.

3. Un servlet puede ejecutarse en un sandbox. Que lo que hace es limitar los privilegios del servlet a un modelo controlado como el de los applets. Salvaguardando la integridad del host donde se ejecuta.

4. Un servlet puede llamar a otro servlet, incluso a métodos de otros Servlets. Esto nos permite que un servlet realice balanceado de carga entre diferentes Servlets. Además, desde un servlet, podemos redirigir una petición sobre otro servlet (en la misma máquina o en una máquina remota).

5. El servlet puede obtener información acerca de la maquina que ha realizado la petición (IP, puerto, tipo de método de envío: get o post,...).

6. Uno de los problemas del protocolo HTTP es que es un protocolo sin estado. No existe una relación entre las diferentes peticiones HTTP realizadas por un usuario sobre un servidor, sino que tiene que ser el propio servidor el que mantenga esta sesión. Por ejemplo, por si queremos mantener algún tipo de información del usuario (su identificación, los productos comprados en las diferentes pantallas,...). En los Servlets podemos utilizar las sesiones y cookies para poder llevar a acabo esto. La única diferencia es que en las sesiones la información del usuario se almacena en el servidor, mientras que con las cookies la información del usuario se almacena en su propia máquina.

7. Conexión a Bases de Datos. A través de los Servlets podemos establecer conexiones a diferentes tipos de bases de datos. Esta característica acopla perfecta a los Servlets dentro de una arquitectura cliente/servidor en 3 capas(cliente-servidor-datos).

8. Proxy para applets. Dentro del desarrollo de applets nos encontramos con un gran número de limitaciones, dentro de las cuales encontramos el acceso al sistema de ficheros. Para subsanar dicha carencia podemos interponer un servlet entre el applet y el sistema de ficheros, de tal manera que el applet se comunicaría con el servlet, que sería el encargado de acceder al sistema de ficheros.

9. Generación dinámica de código. Esta es una de las características más utilizadas en los Servlets, la generación dinámica de HTML. Esto nos permite que una misma página tenga múltiples salidas o representaciones en cuanto a estructura y contenido atendiendo a las evaluaciones que tome el servlet: id del usuario, información de una base de datos, fecha del sistema.

10. Recursos compartidos entre usuarios. Los Servlets pueden definir estructuras o información que va a ser compartida por diferentes usuarios que utilicen el servlet. A la hora de utilizar esta información compartida o global deberemos de tomar las precauciones oportunas para que siempre sea una información correcta, integra y fiable. [11]

2.4.2 JSP (JavaServer Pages)

La tecnología Java Server Pages, cuyo cometido es facilitar la labor del programador Java y la labor del programador Web, en la construcción de los cada vez más complicados y exigentes sitios Web que el mercado requiere. La tecnología Java Server Pages (JSP) está posicionada en la cresta de la ola en la evolución del desarrollo de aplicaciones web, y muy probablemente la próxima generación de herramientas online y servicios sean creados en base a Java Server Pages.

La tecnología JSP está basada en el lenguaje de programación Java y encaminada al desarrollo de sitios web. Mediante el uso de páginas JSP, los diseñadores web y los programadores Java pueden incorporar contenido dinámico en sitios web mediante código Java embebido en las páginas JSP a través de etiquetas. Estas etiquetas proporcionan al diseñador web el modo de acceso a los datos y lógica almacenada en objetos Java, sin necesidad de ser un experto en el desarrollo de aplicaciones Java. Es más, en la actualidad, proyectos como Jakarta-Taglibs proporcionan etiquetas JSP que permiten multitud de funciones, desde la conexión al vuelo del contenido de una página para visualizarla en diferentes dispositivos físicos: ordenador, teléfono móvil, PDA.

[11] Artículo tomado del URL: http://www.aulambra.com/ver.asp?id=111

La tecnología JSP es otro tipo de lenguaje script ejecutado en el servidor, aunque su funcionamiento es bastante diferente. Las páginas JSP son archivos de texto con extensión .jsp. Las páginas JSP contienen etiquetas HTML, familiares al diseñador web, junto con código Java embebido, que permite el acceso de la página a datos desde ese código Java ejecutado en el servidor. Cuando una página JSP es solicitada por un usuario, la parte HTML de la página es procesada tal cual; sin embargo, las porciones de código se ejecutan en el mismo momento de recibir la petición y el contenido dinámico generado por ese código es insertado en la página antes de devolverla de nuevo al usuario. Esto proporciona una separación entre la parte de presentación HTML de la página y la parte lógica de programación incluida en el código Java; es el gran aporte de la tecnología JSP.

Desde que Sun Microsystems lanzó Java al mundo de la programación, todo han sido parabienes para esta tecnología. Términos como "código independiente de plataforma" o "escribir una vez, ejecutar en cualquier lugar" han hecho que muchos desarrolladores se hayan acercado a Java y se ha hecho tan popular que ya son pocos los que no han oído hablar del tema.

En la primera época de la Red, casi todo el contenido de las páginas era libre y textual. El lenguaje HTML es muy fácil de aprender y cualquier persona está en condiciones de crear una página web sin apenas dificultades. No obstante, la infraestructura de la Red ha ido mejorando y el contenido de las páginas ya no se restringe al uso de texto, sino que ya está poblado de imágenes e incluso vídeos. Es más, muchos desarrolladores añaden interactividad a sus sitios web, utilizando alguno de los lenguajes que han ido surgiendo para satisfacer los complejos requerimientos de las páginas actuales.

Aquí surge Java para intentar paliar estos inconvenientes. Utilizando Servlets, que es código Java ejecutado en el servidor, no en el navegador cliente, es posible crear sitios interactivos.
Sin embargo, tampoco esta solución carece de problemas. El principal de ellos es que cualquier modificación en la interfaz de usuario, o cualquier mínima alteración de la lógica del programa, obliga a recompilar el Servlet. Por ello surge la tecnología Java Server Pages, para paliar este problema, permitiendo que la presentación de las páginas sea independiente de la lógica de aplicación web. De este modo, la programación Java estará encomendada a los programadores, que proporcionarán a los diseñadores web etiquetas para que puedan utilizarlas en sus diseños sin temor a que una modificación implique una alteración de sus diseños.

Además, la tecnología JavaServer Pages da un gran paso a favor de la escalabilidad al soportar arquitecturas basadas en componentes como JavaBeans o Enterprise JavaBeans, que permiten al programador Java crear módulos de código reusable que acelerarán futuros desarrollos. Y como todo esto es Java, resultará muy sencillo conectar las aplicaciones web de sistemas propios de una empresa, a servidores web para lanzar esa empresa al ciberespacio.

En la actualidad, los desarrolladores de software buscan nuevas formas de distribución de sus aplicaciones que permitan el aprovechamiento de la arquitectura n-capas. En el caso del cliente tradicional, con la aplicación ejecutándose en su ordenador local, resulta muy difícil la distribución y, sobre todo la actualización. Los clientes basados en web proporcionan una excelente alternativa para aplicaciones Internet o Intranet, más allá incluso de las aplicaciones tradicionales de este entorno, como puede ser la distribución de contenido HTML o el comercio electrónico. La tecnología Java Server Pages proporciona a los desarrolladores Java y autores web un mecanismo muy potente para crear este tipo de aplicaciones.

La tecnología JSP proporciona la capacidad de acceso a datos remotos a través de mecanismos como Enterprise JavaBeans (EJB), Java Database Connectivity (JDBC) y Remote Method Invocation (RMI). Además, permite a los desarrolladores encapsular y separar la lógica de aplicaciones, código Java, de la presentación, código HTML, lo que redunda en una mayor flexibilidad a la hora de crear aplicaciones y reutilización de código.

Esta separación entre lógica y presentación es la mayor de las ventajas que ofrece la tecnología JSP sobre otras arquitecturas de aplicaciones web, como los servlets o scripts CGI.

Ejecución de páginas JSP

Una página JSP no es más que otra forma de ver un servlet. El concepto inherente en un archivo JSP es permitir ver un servlet Java como una página HTML. Este elimina la desagradable lista de sentencias print() que normalmente lleva todo el código Java del servlet. Es decir, una página JSP trata de permitir que se pueda incluir código Java dentro de una página HTML normal.

Una página JSP es preprocesada a un archivo.java, que luego es compilado para generar un archivo.class. Ésta es la innovación que proporciona la tecnología Java Server Pages y que la diferencia de otras semejantes, por ejemplo, una página Active Server Pages (ASP) de Microsoft, se compila en memoria, no en un archivo separado. La figura muestra el flujo de la conversión de una página JSP.

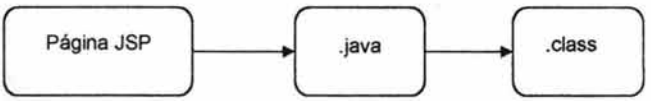


Figura 4. Conversión de una página JSP

Cuando en un servidor web se recibe la petición de una página JSP, éste lanza el motor JSP, que se ejecuta en el mismo proceso que ese servidor web. El motor JSP comprueba si la página es nueva o ha cambiado, en cuyo caso realiza el proceso de traslación de la página y luego compila el resultado. El proceso de traslación es la parte principal del funcionamiento de la tecnología JSP, y consiste en la conversión de la página JSP en un servlet de Java. Este servlet es compilado mediante el compilador Java estándar y ejecutado utilizando el API estándar Java.

El proceso de traslación es el que ralentiza la ejecución de las páginas JSP; sin embargo, una vez que la página JSP ha sido convertida a servlet y compilada, su ejecución es tan rápida como si su origen hubiese sido un servlet normal.

La siguiente figura muestra los componentes involucrados en el procesamiento de la petición HTTP realizada por el cliente de una página JSP.

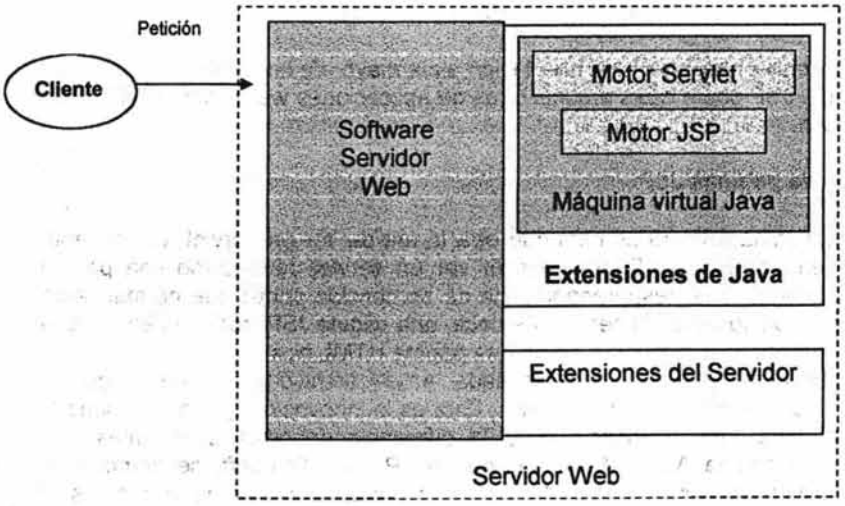


Figura 5. Petición de un cliente para una página JSP

Como se muestra en la figura anterior y se puede observar en la siguiente figura, que se muestra el proceso de la petición HTTP en más detalle, todo el tratamiento de la petición HTTP que se hace al servidor web hasta que devuelve la respuesta al cliente, se resume en cuatro pasos:

1. El motor de JSP analiza la página solicitada y crea un fichero de código fuente Java correspondiente al servlet.
2. El servlet generado se compila parea obtener el archivo .class, que pasa al control del motor servlet, que lo ejecuta del mismo modo que si se tratase de cualquier otro servlet.
3. El motor servlet carga la clase del servlet generado para ejecutarlo.
4. El servlet ejecuta y devuelve su respuesta al solicitante.

La figura muestra el flujo completo desde la petición de la página JSP hasta la obtención de la respuesta de forma detallada:

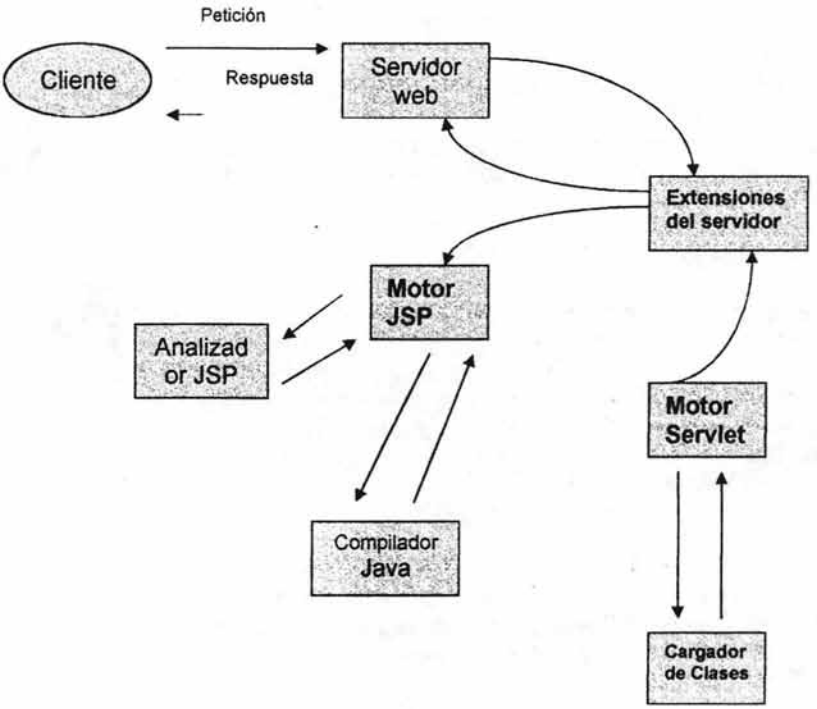


Figura 6. Petición de una JSP más detallada

El proceso es mucho más eficiente de lo que se puede deducirse observando la figura anterior. El análisis de la página y la traslación a servlet

solamente ocurren una vez, la primera ocasión en que se realiza la solicitud de la página, o cuando se modifica ésta.

La carga de la clase en el motor servlet también ocurre una sola vez desde la última vez que se haya arrancado el servlet. Después de esto, el servlet ya está disponible durante toda la vida de la máquina virtual Java. Incluso para mejorar la eficiencia del último paso, hay algunos servidores web que proporcionan un mecanismo de caché de páginas que reduce el coste de ejecutar la petición y mejora el rendimiento de la aplicación; es decir, mediante el mecanismo, la generación de la respuesta puede realizarse una única vez, o bien solamente cuando su contenido dinámico haya variado[12].

2.5 Servidor Jakarta-TOMCAT

El servidor Jakarta-Tomcat, uno de los proyectos de código abierto liderado por la Apache Software Foundation. El servidor Tomcat es una aplicación Web basada en Java creada para ejecutar servlets y páginas JSP, siendo la implementación oficial de referencia de las especificaciones Servlet 2.3 y Java Server Pages 1.2.

2.5.1 Aplicación Web

Antes de entrar en la instalación y configuración de Tomcat, es necesario tener un conocimiento básico del concepto de *Aplicación Web*, que fue introducido en la versión 2.2 de la especificación servlet. De acuerdo con esta especificación, una aplicación web es una colección de servlets, páginas JSP, clases Java, archivos de descripción de la aplicación, documentos estáticos: HTML, XHTML, imágenes, etc. y otros recursos que pueden ser empaquetados y ejecutados en distintos servidores de diferentes proveedores. Es decir, una aplicación web se podría definir como la capa web de cualquier aplicación.

Una de las características principales de una aplicación web es su relación con el **ServletContext**. Esta relación está controlada por el contenedor de servlets, que asocia un único **ServletContext** para cada aplicación, garantizando que las aplicaciones no van a colisionar a la hora de almacenar objetos en el **ServletContext**.

El contenedor que alberga una aplicación web no es más que la estructura de directorios en donde están colocados todos los archivos necesarios para la ejecución de la aplicación web. Por tanto, el primer paso en el desarrollo de cualquier aplicación web consiste en crear la estructura de directorios en donde colocar sus componentes. A continuación se indican los directorios necesarios para la aplicación **ejemplo Web**, que debe colgar del directorio raíz del contenedor de servlets, que puede diferir de unos servidores JSP a otros. En el caso de

[12] FROUFE, Quintas Agustín. Java Servlet Pages Manual de usuario y tutorial. Ed. Alfaomega. Madrid España, 2002

Tomcat, el directorio a partir del cual se instala cualquier aplicación web debe ser *TOMCAT-HOME/webapps*, en donde **TOMCAT_HOME** apunta al directorio de instalación de Tomcat, que se comentará posteriormente en este mismo apéndice. Los directorios de la aplicación **ejemploWeb** serán los siguientes:

/ejemploWeb

Directorio raíz de la aplicación web, en el cual se colocan todos los archivos HTML y JSP que utiliza la aplicación. Se pueden crear subdirectorios adicionales para mantener cualquier otro recurso de tipo estático que forme parte de la aplicación web y constituyan la parte de acceso público desde cualquier navegador.

/ejemploWeb/WEB-INT.

Directorio que contiene todos los recursos relacionados con la aplicación web que no se han colocado en el directorio raíz y que no deben servirse al cliente. Esto es importante, ya que este directorio no forma parte del documento público, por lo que ninguno de los ficheros que contenga va a poder ser enviado directamente a través del servidor web.

En este directorio se coloca el archivo web.xml, donde se establece la configuración de la aplicación web.

/ejemploWeb/WEB-INF/classes

Directorio que contiene todos los servlets y cualquier otra clase de utilidad o complementaria que se necesite para la ejecución de la aplicación web. Normalmente contiene solamente archivos .class.

/ejemploWeb/WEB-INF/lib

Directorio que contiene los archivos Java de los que depende la aplicación web. Por ejemplo, si la aplicación web necesita acceso a base de datos a través de JDBC, en este directorio es donde deben colocarse los ficheros JAR que contengan el driver JDBC que proporcione el acceso a la base de datos. Normalmente contiene solamente archivos.jar.

/ejemploWeb/WEB-INF/tlds

Directorio que contiene los archivos TLD, descriptor de la librería de etiquetas, en el caso de que la aplicación web utilice cualquier librería de etiquetas, o acciones personalizadas.

En esta estructura, el cargador de clases consulta en primer lugar el directorio classes y posteriormente el directorio *lib*, de forma que en el desarrollo se pueden colocar clases de usuario en el directorio *classes* y las clases ya probadas o

clases de terceros en el directorio *lib*; de este modo el cargador de clases resolverá en primer lugar las clases con las que se encuentran en desarrollo, y si no es capaz de encontrarlas en el directorio *classes* las buscará en los archivos del directorio *lib*.

Una segunda parte muy importante de toda aplicación web es su descriptor de configuración. Este descriptor no es más que un archivo XML de nombre web.xml, localizado en el directorio *WEB-INF*, que contiene la descripción de la configuración correspondiente a la aplicación web. En el caso anterior el archivo web.xml estará situado en el directorio */ejemploWeb/WEB-INF*. La información que contiene este descriptor puede incluir los siguientes elementos:

- Parámetros de inicialización del ServletContext
- Configuración de la sesión
- Definiciones de Servlets/JavaServer Pages
- Mapeado de Servlets/JavaServer Pages
- Mapeado de tipos MIME
- Configuración de seguridad
- Páginas de error
- Páginas de bienvenida

Finalmente, la tercera parte importante de una aplicación web es el archivo **WAR** (*Web ARchive*), que es el método estándar empleado para empaquetar una aplicación web y dejarla lista para su distribución y acceso a través de servidores web con soporte para servlets y páginas JSP. Utilizando el archivo WAR se puede distribuir una aplicación web completa, compuesta por cualquier número de recursos, en una única unidad de distribución, en un único archivo.

El archivo WAR se genera con la herramienta **jar**, desde el directorio inmediatamente anterior al directorio que contiene la aplicación web. En realidad, un archivo WAR es un archivo comprimido que utiliza tecnología *zip* que permite agrupar múltiples ficheros y directorios en un único fichero, manteniendo su estructura original y comprimiendo su contenido; por lo que cualquier herramienta que permita comprimir en este formado puede ser utilizada, renombrando el archivo de salida de dicha herramienta a un archivo con extensión WAR.

El archivo WAR es el que se proporciona al proveedor de Internet, que debe colocarlo en el directorio adecuado del servidor web que utilice, dejando la aplicación accesible al público a través de cualquier navegador.

Durante el desarrollo de una aplicación web, si se están construyendo constantemente archivos WAR, se debe considerar la utilización de alguna herramienta que automatice este proceso, por ejemplo **ANT**. Ant es una herramienta, también del grupo Apache y bajo licencia GNU, que permite hacer mucho más que generar automáticamente archivos WAR; en realidad se trata de una poderosa herramienta de tipo **makefile** basada en Java y configurada a través

de archivos XML. Con *Ant* es muy simple reconstruir un proyecto completo y generar el archivo WAR listo para su distribución, aunque necesita un mínimo de aprendizaje que se escapa al alcance de esta breve sugerencia.

El archivo WAR siempre incluye dos directorios especiales: *META-INF y WEB-INF*. Si el lector es recién llegado al mundo de las páginas JSP seguramente se sorprenderá ante la presencia del primer directorio; en él se almacena el archivo *manifest* e información útil para las herramientas Java y no es de especial interés para el desarrollador.

2.5.2 Instalación y Configuración de Tomcat

Una vez visto en qué consiste una Aplicación Web y la forma en que es posible su generación y distribución, los siguientes párrafos tratan la instalación y configuración del servidor web **Jakarta-Tomcat**, que como se ha dicho, es la implementación oficial de referencia de la especificación Servlet 2.3 y JavaServerPages 1.2.

La explicación que sigue trata de la instalación de Tomcat en modo aislado, es decir, que es el propio Tomcat quien servirá todas las peticiones web, ya sean éstas de contenido HTML estático, páginas JSP o servlets. Es necesario tener instalado el Java Development Kit (JDK) 1.1 o posterior, para la instalación de Tomcat. La versión de Tomcat que se ha utilizado para probar la instalación y configuración que aquí se explica es ***Jakarta-Tomcat 3.2.3***. Las indicaciones que se proporcionan van dirigidas al uso de Tomcat en Linux y Windows NT/2000.

Instalación del Java Development Kit

A continuación explicaremos los pasos para instalar y trabajar con *Java Development Kit* en *Windows* NT y *Windows* 98, tomando para ello como ejemplo la instalación del JDK 1.2.2.

- Acceder al fichero ejecutable.

Desde el sitio *Web* de *Java Software*: http://www.java.sun.com tenemos la posibilidad de ejecutar directamente el fichero *j2sdk-1_2_2-win.exe* o bien descargarlo a nuestro disco duro.

- Ejecutar el instalador Java 2 SDK.

El fichero *j2sdk-1_2_2-win.exe* es el instalador Java 2 SDK. Si hemos descargado el fichero en lugar de ejecutarlo directamente desde la página de Java, debemos hacer doble-clic sobre su icono. A partir de aquí seguir las instrucciones que va dando el instalador. Una vez finalizada la instalación podemos borrar el fichero descargado para liberar espacio del disco duro.

- Establecer la variable PATH.

Es posible ejecutar el Java 2 SDK sin establecer un valor para la variable PATH o bien, según conveniencia puede dársele un valor.

Trabajar con la variable PATH nos permite invocar a los ejecutables del Java 2 SDK (*javac.exe, java.exe, javadoc.exe,* etc.) desde cualquier directorio sin tener que teclear la ruta completa al comando. Es decir, si no establecemos la variable PATH, deberemos especificar la ruta completa al ejecutable cada vez que lo queramos invocar de la siguiente forma:

C:\jdk1.2.2\bin\javac MyClass.java

Es útil establecer la variable PATH de forma permanente para que persista después de reiniciar el ordenador. La forma de hacer esto depende de si trabajamos con *Windows* NT o con *Windows* 98.

Si el directorio de instalación ha sido *c:\jdk1.2.2*, para ambos casos la variable PATH tendrá el valor *c:\jdk1.2.2\bin*

- En *Windows* NT debemos establecer su valor accediendo a la ventana Sistema del Panel de Control, y una vez allí seleccionando la pestaña de entorno.
- Por el contrario, en *Windows* 98 debemos añadir en el archivo *AUTOEXEC.BAT* la siguiente línea: *PATH c:\jdk1.2.2\bin*

Linux

La instalación en Linux es muy sencilla. Antes de instalar Tomcat hay que instalar el JDK (Java Development Kit) a partir de la versión 1.1 o posterior para nuestro caso se instalará el JDK 1.2.2, si no estaba instalado previamente. A partir de ahora se supone que el JDK está instalado en el directorio *c:/jdk1.2.2* ; si el lector lo instala en otra localización, deberá sustituir esta cadena por la que corresponda a su configuración.
Es necesario comprobar que la variable de entorno **JAVA_HOME** está definida y apunta al directorio de instalación del JDK. Si no está definida, se define; en *bash* se establece su valor con los siguientes comandos:
JAVA_HOME=/opt/jdk1.2.2
export JAVA_HOME

También es necesario añadir el directorio en donde se encuentra el intérprete Java a la variable de entorno **PATH**; en caso de que el lector no lo haya hecho ya en la instalación del JDK.

Una vez configurado el JDK, se extrae Tomcat del fichero de distribución en un directorio, por ejemplo */opt/jakarta-tomcat*. Este directorio debe asignarse a la variable de entorno **TOMCAT_HOME**. En *bash* se asigna con los siguientes comandos:
TOMCAT_HOME=/opt/jakarta-tomcat
export TOMCAT_HOME

Con esta asignación se concluye la instalación de Tomcat en Linux. Para comprobar que todo está correcto, es necesario arrancar y parar el servidor web, para lo cual se utilizan, respectivamente, los comandos siguientes:

TOMCAT_HOME/bin/startup.sh
TOMCAT_HOME/bin/shutdown.sh

Windows NT/2000

La instalación en Windows es un poco más enredada, como ocurre en todo entorno de ventanas, en los cuales las acciones se ven facilitadas por la presencia del entorno gráfico, pero es más difícil encontrar la ventana que contiene la opción que es necesario configurar.

Al igual que ocurre en Linux, en Windows es necesario instalar el JDK 1.2.2, si no está instalado previamente, y fijar la variable de entorno **JAVA_HOME** apuntando al directorio de instalación del JDK, que en este caso se toma como *c:\jdk1.2.2*
La secuencia a seguir para llegar a la ventana de configuración de las variables de entorno en **Windows 2000** es la siguiente:
Inicio -> Configuración -> Panel de control -> Sistema ->
Ventana de "Propiedades del sistema"; pestaña "Avanzado"; botón "Variables de entorno" ->
Ventana de "Variables de entorno"
En la ventana *Variables de entorno* se encuentran las variables de usuario en la parte superior. Pulsando el botón "Nueva" aparece una ventana en la que se debe introducir el *Nombre de la variable* y el *Valor de la variable*. El lector debe comprobar que todas las variables de entorno que se han indicado en párrafos anteriores se encuentran definidas antes de cerrar las ventanas.

En **Windows NT** la secuencia que se debe seguir para poder fijar las variables de entorno es ligeramente diferente, tal como se puede comprobar en la secuencia de comandos que se indica a continuación:
Inicio -> Configuración -> Panel de control -> Sistema ->
Ventana de "Propiedades del sistema"; pestaña "Entorno"

En los cuadros de texto de la parte inferior de la ventana se introduce el nombre de la variable en el campo *Variable:* y el valor de la variable se introduce en el campo *Valor:* . Pulsando el botón "Establecer" la variable pasará a formar parte de las variables de usuario que se presentan en la ventana.

Siguiendo los pasos anteriores el lector debe comprobar que la variable de entorno JAVA_HOME apunta al directorio de instalación del JDK. Debe seguir los mismos pasos para fijar la variable de entorno **TOMCAT_HOME** apuntando al directorio donde se ha instalado Tomcat, en este caso se supone instalado en el directorio *c:\jakarta-tomcat*. Y como paso final, debe asegurarse el lector de que el directorio *c:\jdk1.2.2\bin* forma parte de la variable de entorno **PATH**, introduciendo ese camino en caso que no se encuentre todavía formando parte de ella.
Para lanzar y parar el servidor Tomcat, los comandos que se utilizan en Windows, desde una ventana de "Símbolo del sistema", son los siguientes, respectivamente:

TOMCAT_HOME\bin\startup.bat
TOMCAT_HOME\bin\shutdown.bat

Habiendo realizado la instalación de Tomcat siguiendo las indicaciones proporcionadas hasta ahora, el sistema estará en condiciones de arrancar el servidor web y atender peticiones a través de Tomcat. Para ello es necesario levantar el servidor con el comando correspondiente al sistema operativo sobre el que se haya realizado la instalación.

Una vez arrancado el servidor, ya es posible realizar la comprobación de la bondad de la instalación recibiendo en un navegador una página servida por Tomcat. Se abre un navegador, por ejemplo *Internet Explorer*, solicitando la siguiente dirección: http://localhost:8080/ y debería aparecer la página que muestra la figura siguiente, correspondiente a la bienvenida de Tomcat.

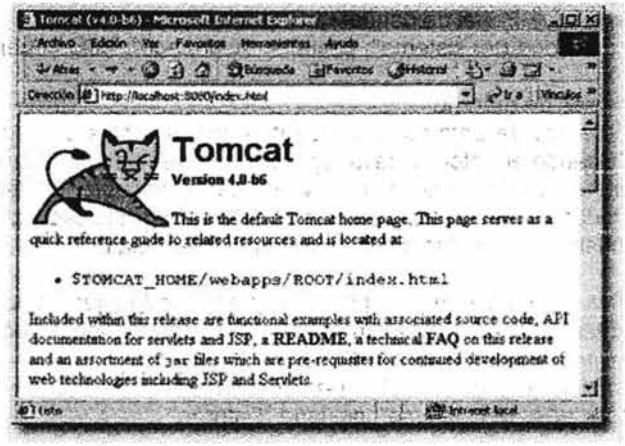


Figura 7. Página de bienvenida al servidor

Si se desea que Tomcat atienda las peticiones a través del puerto 80, que es el puerto de defecto *HTTP*, en lugar del puerto 8080, es necesario editar el fichero de configuración de Tomcat y rearrancar el servidor para que tome en cuenta el cambio. Tomcat lee este fichero de configuración, *TOMCAT_HOME/conf/server.xml*, cada vez que arranca. Además del puerto a través del cual Tomcat atiende las peticiones, se pueden configurar otros parámetros como el puerto de administración, el directorio base de los documentos, el directorio de trabajo en donde se crean ficheros temporales o el intervalo máximo de inactividad permitido para una sesión.
La comprobación final de la instalación y configuración del entorno de desarrollo consiste en la ejecución de alguno de los ejemplos de páginas JSP que se distribuyen con Tomcat. En la página de presentación que aparece en el navegador, se selecciona "JSP Examples", que hará aparecer en el navegador otra página en la que es posible elegir varios ejemplos de páginas JSP. Si se selecciona sobre *Execute* del ejemplo **Date**, debe aparecer una página como la que se muestra en la figura siguiente (por supuesto, con una fecha diferente).

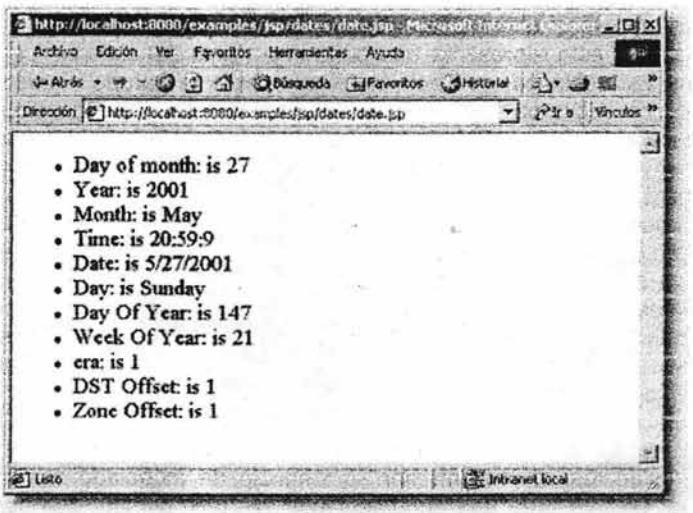


Figura 8. Página de ejemplo de una página JSP

Si esa página no aparece, se debe comprobar que la variable **JAVA_HOME** apunte de forma correcta al directorio de instalación del JDK, porque suele ser el problema más frecuente.[13]

[13] Artículo tomado del URL: http://usuarios.lycos.es/froufe/parteJ1/capa-1.html

2.6 Metodología de análisis y modelado

2.6.1 Lenguaje de Modelado Unificado (UML)

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

Con UML nos debemos olvidar del protagonismo excesivo que se le da al diagrama de clases, este representa una parte importante del sistema, pero solo representa una vista estática, es decir muestra al sistema parado. Sabemos su estructura pero no sabemos que le sucede a sus diferentes partes cuando el sistema empieza a funcionar. UML introduce nuevos diagramas que representa una visión dinámica del sistema. Es decir, gracias al diseño de la parte dinámica del sistema podemos darnos cuenta en la fase de diseño de problemas de la estructura al propagar errores o de las partes que necesitan ser sincronizadas, así como del estado de cada una de las instancias en cada momento. El diagrama de clases continua siendo muy importante, pero se debe tener en cuenta que su representación es limitada, y que ayuda a diseñar un sistema robusto con partes reutilizables, pero no a solucionar problemas de propagación de mensajes ni de sincronización o recuperación ante estados de error. En resumen, un sistema debe estar bien diseñado, pero también debe funcionar bien.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

UML es ahora un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.

UML permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir mediante la restricción estamos forzando el comportamiento que debe tener el objeto al que se le aplica.

Diagramas.

La explicación se basará en los diagramas, en lugar de en vistas o anotación, ya que son estos la esencia de UML. Cada diagrama usa la anotación pertinente y la suma de estos diagramas crean las diferentes vistas. Las vistas existentes en UML son:

- Vista casos de uso: Se forma con los diagramas de casos de uso, colaboración, estados y actividades.
- Vista de diseño: Se forma con los diagramas de clases, objetos, colaboración, estados y actividades.
- Vista de procesos: Se forma con los diagramas de la vista de diseño. Recalcando las clases y objetos referentes a procesos.
- Vista de implementación: Se forma con los diagramas de componentes, colaboración, estados y actividades.
- Vista de despliegue: Se forma con los diagramas de despliegue, interacción, estados y actividades.

Se Dispone de dos tipos diferentes de diagramas los que dan una vista estática del sistema y los que dan una visión dinámica. Los diagramas estáticos son:

- Diagrama de clases: muestra las clases, interfaces, colaboraciones y sus relaciones. Son los más comunes y dan una vista estática del proyecto.
- Diagrama de objetos: Es un diagrama de instancias de las clases mostradas en el diagrama de clases. Muestra las instancias y como se relacionan entre ellas. Se da una visión de casos reales.
- Diagrama de componentes: Muestran la organización de los componentes del sistema. Un componente se corresponde con una o varias clases, interfaces o colaboraciones.
- Diagrama de despliegue. : Muestra los nodos y sus relaciones. Un nodo es un conjunto de componentes. Se utiliza para reducir la complejidad de los diagramas de clases y componentes de un gran sistema. Sirve como resumen e índice.
- Diagrama de casos de uso: Muestran los casos de uso, actores y sus relaciones. Muestra quien puede hacer que y relaciones existen entre acciones(casos de uso). Son muy importantes para modelar y organizar el comportamiento del sistema.

Los diagramas dinámicos son:

- Diagrama de secuencia, Diagrama de colaboración: Muestran a los diferentes objetos y las relaciones que pueden tener entre ellos, los mensajes que se envían entre ellos. Son dos diagramas diferentes, que se puede pasar de uno a otro sin perdida de información, pero que nos dan puntos de vista diferentes del sistema. En resumen, cualquiera de los dos es un Diagrama de Interacción.
- Diagrama de estados: muestra los estados, eventos, transiciones y actividades de los diferentes objetos. Son útiles en sistemas que reaccionen a eventos.
- Diagrama de actividades: Es un caso especial del diagrama de estados. Muestra el flujo entre los objetos. Se utilizan para modelar el funcionamiento del sistema y el flujo de control entre objetos.

Diagrama de casos de uso.

Se emplean para visualizar el comportamiento del sistema, una parte de él o de una sola clase. De forma que se pueda conocer como responde esa parte del sistema. El diagrama de uso es muy útil para definir como debería ser el comportamiento de una parte del sistema, ya que solo especifica como deben comportarse y no como están implementadas las partes que define. Por ello es un buen sistema de documentar partes del código que deban ser reutilizables por otros desarrolladores. El diagrama también puede ser utilizado para que los expertos de dominio se comuniquen con los informáticos sin llegar a niveles de complejidad. Un caso de uso especifica un requerimiento funcional, es decir indica esta parte debe hacer esto cuando pase esto.

En el diagrama nos encontramos con diferentes figuras que pueden mantener diversas relaciones entre ellas:

- Casos de uso: representado por una elipse, cada caso de uso contiene un nombre, que indique su funcionalidad. Los casos de uso pueden tener relaciones con otros casos de uso. Sus relaciones son:
  - Include: Representado por una flecha, en el diagrama de ejemplo podemos ver como un caso de uso, el de totalizar el coste incluye a dos casos de uso.
  - Extends: Una relación de una caso de Uso A hacia un caso de uso B indica que el caso de uso B implementa la funcionalidad del caso de uso A.
  - Generalization: Es la típica relación de herencia.

- Actores: se representan por un muñeco. Sus relaciones son:
  - Communicates: Comunica un actor con un caso de uso, o con otro actor.
- Parte del sistema (System boundary): Representado por un cuadro, identifica las diferentes partes del sistema y contiene los casos de uso que la forman.

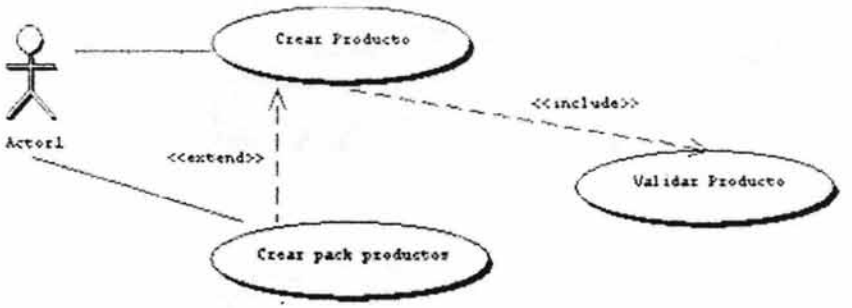


Figura 9. Ejemplo de casos de uso

En este gráfico encontramos tres casos de usos Crear producto utiliza Validar producto, y Crear pack productos es una especialización de Crear productos.
Podemos emplear el diagrama de dos formas diferentes, para modelar el contexto de un sistema, y para modelar los requisitos del sistema[14].

Modelado del contexto
Se debe modelar la relación del sistema con los elementos externos, ya que son estos elementos los que forman el contexto del sistema.
Los pasos a seguir son:

- Identificar los actores que interactúan con el sistema.
- Organizar a los actores.
- Especificar sus vías de comunicación con el sistema.

[14] Introducción a UML. Autor: Pere Martra. URL: http://www.programacion.com/tutorial/uml/

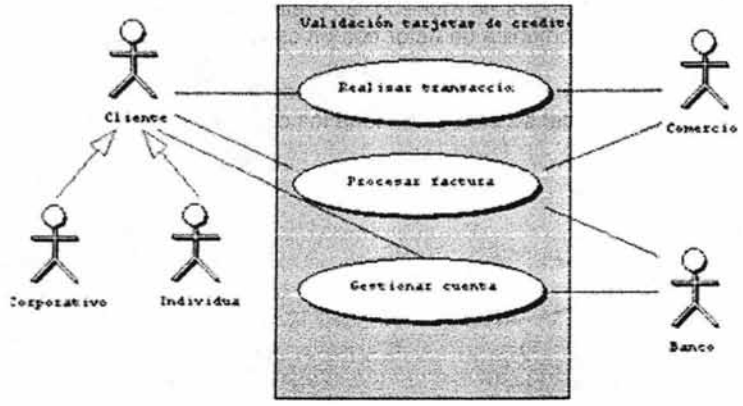


Figura 10. Modelado del contexto

Modelado de requisitos

La función principal, o la más conocida del diagrama de casos de uso es documentar los requisitos del sistema, o de una parte de él.

Los requisitos establecen un contrato entre el sistema y su exterior, definen lo que se espera que realice el sistema, sin definir su funcionamiento interno. Es el paso siguiente al modelado del contexto, no indica relaciones entre autores, tan solo indica cuales deben ser las funcionalidades (requisitos) del sistema. Se incorporan los casos de uso necesarios que no son visibles desde los usuarios del sistema.

Diagrama de clases

Forma parte de la vista estática del sistema. En el diagrama de clases como ya hemos comentado será donde definiremos las características de cada una de las clases, interfaces, colaboraciones y relaciones de dependencia y generalización.

Es decir, es donde daremos rienda suelta a nuestros conocimientos de diseño orientado a objetos, definiendo las clases e implementando las ya típicas relaciones de herencia y agregación.
En el diagrama de clases debemos definir a éstas y a sus relaciones.

La Clase
Una clase está representada por un rectángulo que dispone de tres apartados, el primero para indicar el nombre, el segundo para los atributos y el tercero para los métodos.

Cada clase debe tener un nombre único, que las diferencie de las otras. Un atributo representa alguna propiedad de la clase que se encuentra en todas las instancias de la clase. Los atributos pueden representarse solo mostrando su nombre, mostrando su nombre y su tipo, e incluso su valor por defecto.

Un método o operación es la implementación de un servicio de la clase, que muestra un comportamiento común a todos los objetos. En resumen es una función que le indica a las instancias de la clase que hagan algo.

Para separar las grandes listas de atributos y de métodos se pueden utilizar estereotipos.

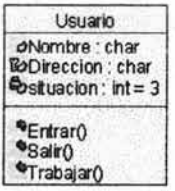


Figura 11. Diagrama de clases

Aquí vemos un ejemplo. La clase usuario contiene tres atributos. Nombre que es public, dirección que es protected y situación que es private. Situación empieza con el valor 3. También dispone de tres métodos Entrar, Salir y Trabajar.

Relaciones entre clases

Existen tres relaciones diferentes entre clases, Dependencias, Generalización y Asociación. En las relaciones se habla de una clase destino y de una clase origen. La origen es desde la que se realiza la acción de relacionar. Es decir desde la que parte la flecha, la destino es la que recibe la flecha. Las relaciones se pueden modificar con estereotipos o con restricciones.

Dependencias

Es una relación de uso, es decir una clase usa a otra, que la necesita para su cometido. Se representa con una flecha discontinua va desde la clase utilizadora a la clase utilizada. Con la dependencia mostramos que un cambio en la clase utilizada puede afectar al funcionamiento de la clase utilizadora, pero no al contrario. Aunque las dependencias se pueden crear tal cual, es decir sin ningún estereotipo (palabra que aparece al lado de la línea que representa la dependencia) UML permite dar mas significado a las dependencias, es decir concretar mas, mediante el uso de estereotipos.

- Estereotipos de relación Clase-objeto.
  - Bind: La clase utilizada es una plantilla, y necesita de parámetros para ser utilizada, con Bind se indica que la clase se instancia con los parámetros pasándole datos reales para sus parámetros.
  - Derive: Se utiliza al indicar relaciones entre dos atributos, indica que el valor de un atributo depende directamente del valor de otro. Es decir el atributo edad depende directamente del atributo Fecha nacimiento.
  - Friend: Especifica una visibilidad especial sobre la clase relacionada. Es decir podrá ver las interioridades de la clase destino.
  - InstanceOF: Indica que el objeto origen es una instancia del destino.
  - Instantiate: indica que el origen crea instancias del destino.
  - Powertype: indica que el destino es un contenedor de objetos del origen, o de sus hijos.
  - Refine: se utiliza para indicar que una clase es la misma que otra, pero mas refinada, es decir dos vistas de la misma clase, la destino con mayor detalle.

Generalización
Pues es la herencia, donde tenemos una o varias clases padre o superclase o madre, y una clase hija o subclase. UML soporta tanto herencia simple como herencia múltiple. Aunque la representación común es suficiente en el 99.73% de los casos UML nos permite modificar la relación de Generalización con un estereotipo y dos restricciones.

- Estereotipo de generalización.
  - Implementation: El hijo hereda la implementación del padre, sin publicar ni soportar sus interfaces.
- Restricciones de generalización.
  - Complete: La generalización ya no permite mas hijos.
  - Incomplete: Podemos incorporar mas hijos a la generalización.
  - Disjoint: solo puede tener un tipo en tiempo de ejecución, una instancia del padre solo podrá ser de un tipo de hijo.
  - Overlapping: puede cambiar de tipo durante su vida, una instancia del padre puede ir cambiando de tipo entre los de sus hijos.

Asociación
Especifica que los objetos de una clase están relacionados con los elementos de otra clase. Se representa mediante una línea continua, que une las dos clases. Podemos indicar el nombre, multiplicidad en los extremos, su rol, y agregación.

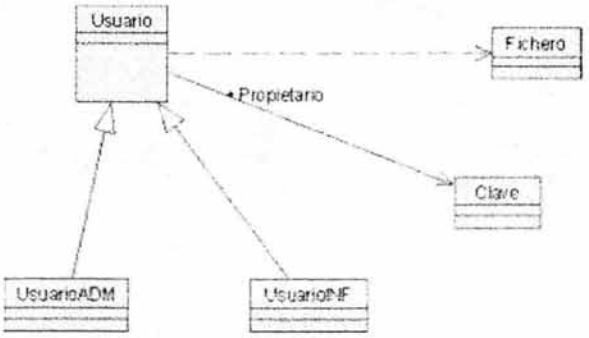


Figura 12. Relaciones entre clases

Diagrama de Interacción

El diagrama de interacción, representa la forma en como un Cliente (Actor) u Objetos (Clases) se comunican entre si en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente.
Dicho diagrama puede ser obtenido de dos partes, desde el Diagrama Estático de Clases o el de Casos de Uso (son diferentes).

Los componentes de un diagrama de interacción son:

- Un objeto o actor.
- Mensaje de un objeto a otro objeto.
- Mensaje de un objeto a sí mismo

Elementos

- **Objeto/Actor**:

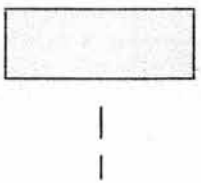


Figura 13. Objeto

El rectángulo representa una instancia de un Objeto en particular, y la línea punteada representa las llamadas a métodos del objeto.

- **Mensaje a Otro Objeto**:

Se representa por una flecha entre un objeto y otro, representa la llamada de un método (operación) de un objeto en particular.

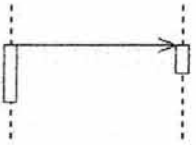


Figura 14. Mensaje a otro objeto

- **Mensaje al Mismo Objeto**:

No solo llamadas a métodos de objetos externos pueden realizarse, también es posible visualizar llamadas a métodos desde el mismo objeto en estudio.

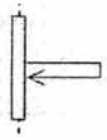


Figura 15. Mensaje al mismo objeto

Ejemplo

En el presente ejemplo, tenemos el diagrama de interacción proveniente del siguiente modelo estático:

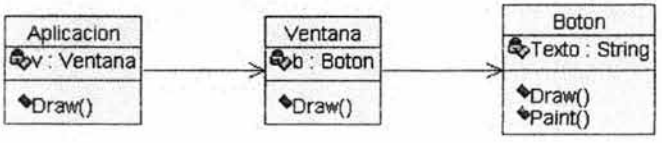


Figura 16. Diagrama de interacción

Aquí se representa una aplicación que posee una Ventana gráfica, y ésta a su vez posee internamente un botón.
Entonces el diagrama de interacción para dicho modelo es:

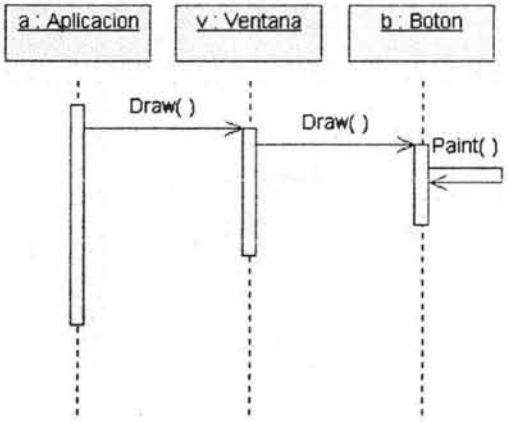


Figura 17. Otro ejemplo de diagrama de interacción

En donde se hacen notar las sucesivas llamadas a Draw() (entre objetos) y la llamada a Paint() por el objeto Botón[15].

El objetivo principal de esta Tesis es investigar, plantear y desarrollar los principales pasos para la construcción de un sistema cuya finalidad es la puesta en marcha del Administrador de Documentos. Una vez que se ha investigado sobre la tecnología que se utilizará para la aplicación, lo siguiente es definir los requerimientos correspondientes al análisis y desarrollo del sistema.

En el siguiente capitulo se explicarán los aspectos del análisis del sistema, incluyendo la exposición de la metodología utilizada, para así poder comprender la relevancia del por qué de cada fase del proyecto.

[15]Artículo tomado del URL:
http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html

Capítulo III

Análisis del Administrador de Documentos

3.1 Proceso de desarrollo

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema.

La metodología que se manejará para el desarrollo de la aplicación del Administrador de Documentos, es una metodología orientada a objetos que utiliza UML para diseñar un sistema estable, reutilizable y tolerante al cambio. Se utilizará para esta metodología el Proceso Unificado de *Rational* (RUP, *Rational Unified Process*) el cual se caracteriza porque proporciona una serie de modelos que se basan en los conceptos de objeto y clase y en las relaciones entre ellos.

Un proceso de desarrollo de software es un método para organizar las actividades referentes a la creación, desarrollo y mantenimiento de sistemas de software.

Un sistema por naturaleza es complejo, por lo que es necesario descomponerlo en partes comprensibles de tal forma de poder manejar su complejidad.

La primera etapa para la construcción de un sistema es la creación de **modelos**, mediante los cuales sea posible organizar su construcción y también comunicar los detalles importantes desde el mundo real hacia el mundo del software a través del ciclo de vida del sistema.

Etapas del Proceso a macro Nivel

- Plan de elaboración. Conceptualización y definición de requerimientos.
- Edificación. La construcción del sistema.
- Distribución. La implantación y uso del sistema.

3.1.1 Etapa de requerimientos

Los requerimientos son las necesidades que se tienen para un producto.

Las siguientes son recomendaciones (artefactos) para construir la etapa de requerimientos.

- Objetivos — Son el propósito del proyecto.
- Clientes: — Quienes serán los usuarios del proyecto.
- Metas: — Especificar los alcances que debe tener el sistema.

- Funciones del sistema: Describir lo que debe de hacer el sistema, estas actividades deben ser listadas de manera lógica, si es necesario agruparlas, hacerlo en grupos cohesivos.

Las funciones deben ser clasificadas dentro de la siguiente categoría:

FUNCIONES DEL SISTEMA	
CATEGORÍA DE FUNCIONES	SIGNIFICADO
EVIDENTES	Son claramente definidas, el usuario es consiente de lo que realiza
OCULTAS	Realizan un trabajo determinante pero no son evidentes al usuario; tales como guardar información, usando un mecanismo persistente
OPCIONALES	No son relevantes en cuanto a la funcionalidad de la aplicación

- Atributos del sistema: Definir las características o dimensiones del sistema. Cuando se especifican los atributos también son descritos los detalles y restricciones de los mismos.
- Casos de uso: Es un documento narrativo que describe la secuencia de eventos de un actor el cual usa el sistema para llevar a cabo un proceso. Los casos de uso no son exactamente requerimientos o especificaciones funcionales. Los casos de uso ilustran y se refieren a requerimientos en la historia que cuentan.

Una vez que se ha planteado el problema, se ha llevado a cabo una investigación preliminar, se han definido funciones y atributos del sistema bajo desarrollo, se continúa con el planteamiento de los **casos de uso**, los cuales deben cubrir las funciones planteadas, para lo cual se cuenta con los siguientes formatos:

- **Formato de nivel superior**

Describe el proceso de una forma global.

- **Formato Expandido**

Un caso de uso en formato expandido describe un proceso a mayor detalle que uno de nivel superior. La principal diferencia entre ambos formatos es que el caso de uso de nivel superior no se describen todos los eventos, mientras que en el expandido si.

Referencias cruzadas: Relación entre los casos de uso y las funciones del sistema especificadas para el procedimiento.

Cursos de eventos típico	
Acciones del actor	Respuesta del sistema
Acciones enumeradas ejecutadas por el actor	Descripción de las respuestas del sistema

Curso alternativo: Se emplea cuando en un caso de uso se plantean cursos alternos u otros procesos, se indican con letras y números. Por ejemplo; E-1, E-2, S-1 ó S-2.

Casos de uso primario, secundario u opcional

- Primario: Representan un proceso común.
- Secundario: Representa un proceso que ocurre raramente o en menor número de ocasiones.
- Opcional: Representa procesos que pueden o no ocurrir.

Establecer jerarquías para los casos de uso.

Una vez que han sido encontrados y escritos los casos de uso, el siguiente paso es distribuirlos para que sean tratados de acuerdo a un orden de importancia o de peso para el sistema.

Se pueden clasificar en una jerarquía de la siguiente forma:

JERARQUÍA	JUSTIFICACIÓN
ALTO	Aquellos casos de uso que tienen un alto impacto en el sistema.
MEDIO	Aquellos que afectan la seguridad del sistema y aquellos que representan procesos importantes.
BAJO	Tienen efectos mínimos en la arquitectura y su definición es dependiente de otros caos de uso

- **Riesgos:** Son los eventos que puede hacer que el desarrollo del sistema no culmine con éxito.

3.1.2 Planteamiento de la problemática del Administrador de Documentos

En la actualidad Internet es una herramienta que es utilizada para conectar a redes de computadoras de Organismos oficiales, Educativos y Empresariales en todo el mundo, con el fin de enviar y recibir información a través de los servicios que este ofrece.

El intercambio de información actualizada y la administración de documentos son actividades básicas para cualquier grupo de trabajo. Grupos de personas trabajando en conjunto, necesitan medios para compartir y almacenar la información relevante para sus objetivos y las tareas que deben realizar.

El administrador de documentos permitirá intercambiar información a especialistas del IMP que estén involucrados en un proyecto.

La aplicación asignará a los usuarios diferentes permisos (lectura, modificación, eliminación) según los perfiles o roles dentro del grupo. Podrá interactuar el usuario con sus documentos de manera que pueda acceder a esta de acuerdo con los perfiles de usuario correspondiente.

Perfiles de usuario. Dentro de la aplicación se tendrán diferentes perfiles de usuario de acuerdo a sus responsabilidades o actividades.

En primer lugar se tiene al Administrador del sistema (pueden ser el Líder de Proyecto o Coordinador), que es el encargado de mantener y gestionar la información de los usuarios.

El Administrador tiene todos los privilegios en cuanto al control y seguimiento de la aplicación.

Después se encuentran los usuarios que están trabajando dentro del proyecto y están registrados en el sistema, dependiendo de su función o cargo en el proyecto, serán agrupados.

Los usuarios tienen un nombre de usuario y contraseña, pueden acceder a la aplicación, en la cual podrán subir documentos a la aplicación y podrán restringir la visibilidad de sus documentos.

Y por último están los usuarios que solo son visitantes a la aplicación, y únicamente podrán ver los documentos autorizados.

Con la utilización de herramientas para trabajo en grupo y de servicios como: HTTP, FTP entre otros, y con las herramientas de programación utilizando Java con sus diversos componentes: los Servlets y la tecnología JSP (JavaServer Pages), así como el lenguaje de hipertexto HTML y JavaScript, se realizará la creación de un administrador de documentos a través de Internet/Intranet para el Instituto Mexicano del Petróleo.

La aplicación se hará bajo los estándares y protocolos que utiliza Internet, que son compatibles con cualquier Sistema operativo, plataforma o explorador que manejen los usuarios.

Investigación preliminar

Una investigación en el dominio del problema arroja los siguientes resultados:

Se quiere realizar un Administrador de Documentos por Vía Internet/Intranet. La aplicación se manejaría con cuentas de usuarios, cada usuario registrado tiene nombre y contraseña.

Los usuarios de esta aplicación son trabajadores del IMP que trabajan dentro de un proyecto en común.

En el cual hay diferentes perfiles de usuario, primero se encuentra el Líder de proyecto, que es el encargado de llevar el control del proyecto, después se encuentra un Coordinador que es el que realiza la tarea de verificar la colaboración de los usuarios(trabajadores) que están dentro de un proyecto, también se encuentran los usuarios o empleados que trabajan dentro de proyecto y por último se encuentran los usuarios finales o clientes que son los que revisan los avances del proyecto y a los que se les entrega el trabajo final del proyecto.

Dependiendo de su cargo en el proyecto los usuarios serán agrupados conforme a su perfil y el encargado de asignar los grupos será el Administrador o Líder de Proyecto.

Se pretende que los miembros del grupo de trabajo se comuniquen con esta aplicación sin necesidad de estar físicamente en el mismo lugar de trabajo, ya que solo se necesitará que estén conectados a Internet y usando un visualizador estándar de Web.

Tendrán la ventaja de que los documentos que suban a la aplicación los pueden ver los miembros del grupo de trabajo lo que es de gran importancia ya que podría reducir el tiempo en poder llevar a cabo cierta acción que sea determinante para el proyecto.

Características:

- Entra el usuario a la aplicación.
- Ve el contenido general de la página (índice o mapa de sitio, consulta de documentos.)
- Cuando entre a su cuenta se le pedirá nombre y contraseña y se verificará que este correcta.
- Si no esta registrado el usuario no podrá subir documentos a la aplicación
- El administrador o Líder de proyecto, será el encargado de llevar a cabo los registros de los usuarios que estén dentro del proyecto.

- Una vez registrado el usuario, puede subir documentos a la aplicación, modificar sus datos personales y podrá restringir la visibilidad de los documentos ya sea que los deje ver a todo público o sea privado.
- En cuanto a los documentos a subir se tendrá una restricción del tipo de archivo, así como al tamaño de estos.

Especificación de requerimientos

Los requerimientos se establecen mediante la descripción de las necesidades o deseos que se piensan para un producto de software. La meta primaria en esta fase es identificar y documentar que es realmente necesario, de una manera que sea claramente entendible por el cliente y el equipo de desarrollo.

Puntos a considerar para el análisis de la especificación de requerimientos.

a) Definir los objetivos más importantes.
b) Del planteamiento del problema y de la investigación preliminar, considerar las oraciones más relevantes.
c) Analizar las consideraciones del cliente.
d) Analizar las metas planteadas.
e) Escribir una lista de funciones del sistema.
f) Escribir una lista de atributos más importantes que debe tener el sistema.

Desarrollando los puntos anteriores basándose en el Caso de Estudio.

a) Objetivos del sistema.

- Permitir intercambiar información a personas involucradas en un proyecto.
- Interactuar con sus documentos sin problemas en la plataforma o navegador que utilicen.
- Enviar y recibir la documentación de manera que se pueda acceder a ésta de acuerdo con los perfiles de usuario correspondientes.
- Administrar la gestión de documentos.
- Administrar el control de usuarios.

b) Oraciones más relevantes.

- Registro de datos de usuarios.
- Validar entrada de acuerdo a un nombre y una contraseña.
- Subir archivos, verificando que sean válidos el tipo de archivo y el tamaño.
- Modificar los datos personales.
- Restringir la visibilidad de documentos.
- Agrupar a los miembros del proyecto de acuerdo a sus responsabilidades.

c) Consideraciones del cliente.

- Se desea tener un Administrador de Documentos para proyectos que se desarrollan en el IMP.
- El Administrador de Documentos debe contar con la posibilidad de subir archivos a la aplicación de manera que los integrantes de un proyecto tengan la posibilidad de contar con una página en la cual ingresen y vean información sobre su proyecto de manera que pueden estar comunicados y enterados de los informes de todos los colaboradores.

d) Metas planteadas.

- Tener un entorno colaborativo que permita intercambiar información a usuarios dentro de un mismo proyecto.
- Compartir y almacenar información
- El sistema sea compatible con cualquier Sistema Operativo o navegador.

e) Funciones del sistema.

Tomando en cuenta la investigación preliminar y la especificación de requerimientos, así como la categoría de las funciones en Evidentes, Ocultas y Opcionales, se puede escribir la siguiente lista:
Como primera tentativa se considerará un solo grupo de funciones, posteriormente se puede analizar su división en grupos lógicos.

FUNCIONES DEL SISTEMA		
No.	FUNCIÓN	CATEGORÍA
R1	Registro datos de usuario	Evidente
R2	Almacenar datos usuario	Oculta
R3	Validar entrada usuario	Oculta
R4	Modificar datos usuario	Evidente
R5	Subir documentos a la aplicación	Evidente
R6	Almacenar documentos	Oculta
R7	Desplegar descripción del documento	Oculta
R8	Permisos de acceso a documentos	Oculta
R9	Mostrar documentos	Evidente
R10	Borrar Documentos	Evidente
R11	Salir de sesión	Oculta
R12	Mostrar contenido general	Evidente
R13	Consulta de usuarios	Evidente
R14	Definir grupos de usuarios	Oculta
R15	Eliminar registro usuario	Oculta

f) Atributos del sistema.

En esta parte se deben considerar aquellas características o dimensiones del sistema, tales como tolerancia a fallas, tiempo de respuesta, tipo de interfase de usuario, plataforma sobre la que va a operar, etc.

ATRIBUTOS DEL SISTEMA	
ATRIBUTOS	DETALLES Y RESTRICCIONES
Presentación	Será mostrada en una página web de fácil acceso.
Interfase	La interfase de usuario se desarrollará en un ambiente visual, se podrá visualizar en cualquier navegador web.
Tolerancia a fallas	Se validarán errores de acceso al sistema para evitar que se sature el sistema y deje de funcionar adecuadamente.
Lenguajes de programación	Se realizará en el Lenguaje Orientado a Objetos Java junto con Servlets y la Tecnología JSP, para el contenido dinámico de la aplicación. Para la parte de Presentación se utilizará HTML.
Plataforma	Windows 95/98/2000

3.1.3 Casos de uso del sistema

En un caso de uso interesa expresar la funcionalidad mediante la interacción (pasos de comunicación) de los actores (es) del sistema.

Caso de uso para la aplicación **Administrador de Documentos de Proyectos Vía Internet/Intranet.**

Los conceptos enfocados al sistema bajo desarrollo, del Administrador de Documentos de Proyectos Vía Internet/Intranet, se ven reflejados en la siguiente figura.

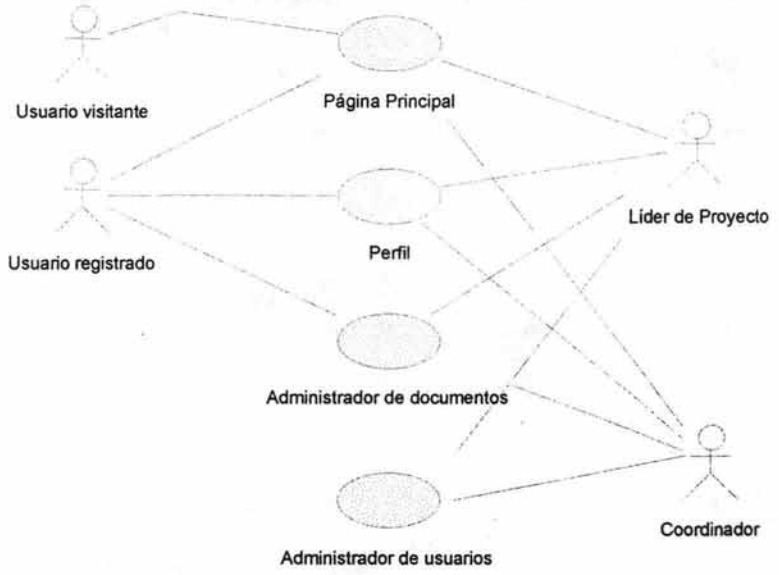


Figura 1. Casos de uso del sistema

Utilizando el formato expandido para la descripción de cada caso de uso mostrado en la figura anterior se tienen las siguientes características:

Caso de uso:	**Administrador de usuarios**
Nombre:	Administrador de usuarios
Actores:	Administradores (Líder de proyecto o Coordinador)
Propósito:	Dar de alta a un usuario
Tipo:	Primario
Descripción:	Cuando entra a la aplicación, el administrador o Líder de proyecto puede dar de alta a los usuarios, seleccionando la opción de registro. Se desplegará un formulario para el llenado de los datos personales.

Se asignará al usuario el grupo que pertenezca según su perfil de usuario.
Posteriormente podrá consultar la información de los usuarios registrados.
Podrá eliminar el registro de un usuario.

Referencias cruzadas: R1, R2, R13, R14, R15

Curso de Eventos: **Administrador de Usuarios**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el Administrador o líder de proyecto desea dar de alta a usuarios. Accede a la opción de Registro de usuarios	2	Muestra el Formulario
3	El Administrador llena la información correspondiente con los datos personales de los usuarios, así como la definición del perfil de usuario. Posteriormente el Coordinador definirá el grupo de trabajo al que pertenece.		
4	Envía la información	5	Guarda la información de los datos de usuario.
		6	Manda mensaje de envío de datos.
7	Solicita consulta de la información de los usuarios registrados	8	Despliega la información.
9	En el menú del administrador se encuentra la opción de Eliminar Registro de usuario. Sí selecciona la opción lo manda a la página de eliminar registro.	10	Despliega la página con los usuarios registrados.
11	Busca al usuario e indica que desea eliminar el registro del usuario	12	Mande mensaje de eliminación de registro. Actualiza la información del sistema.

Caso de uso: **Página principal**

Nombre: Página Principal

Actores: Usuarios (Visitantes y Registrados), Administradores (Líder de proyecto o Coordinador)

Propósito: Entrar a la aplicación y ver el contenido de la página.

Tipo: Primario

Descripción: Cada vez que entra el usuario accede a esta página, la cual contiene información general.

Referencias cruzadas: R12

Curso de Eventos: **Página Principal**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el usuario accede a la página principal del proyecto	2	Muestra la página
3	El usuario ve el contenido de la página		
4	Selecciona la opción requerida	5	Enlaza a la opción requerida

Caso de uso: **Perfil**

Nombre: **Perfil de usuario**

Actores: Usuario, Administradores (Líder de proyecto o Coordinador)

Propósito: Validar la entrada del usuario, con su nombre de usuario y contraseña.

Tipo: Primario

Descripción: Cada vez que el usuario accede a la aplicación se le pide un nombre de usuario y contraseña, para poder validar su sesión de perfil de usuario, así podrá ir a su menú de opciones y modificar sus datos personales.

Referencias cruzadas: R3, R4, R11

Curso de Eventos: **Perfil de usuario**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el usuario pide iniciar sesión.	2	Muestra el formulario donde se solicita nombre de usuario y contraseña
3	El usuario proporciona un nombre de usuario y contraseña para validar su entrada.	4	Verifica que el nombre de usuario y contraseña sean correctos.
		5	Dependiendo del resultado despliega la información, si fue aceptado se dirige a su Perfil de usuario, si no se le pide que intente de nuevo. Dependiendo de su Perfil, el usuario tendrá diferentes opciones.
6	Solicita ver sus datos personales	7	Muestra la información
8	Desea modificar sus datos personales	9	Actualiza su información y despliega sus datos.

Caso de uso: **Administrador de Documentos**

Nombre: **Administrador de Documentos .**

Actores: Usuario registrado, Administradores (Líder de proyecto o Coordinador)

Propósito: Administrar los documentos que se carguen a la aplicación

Tipo: Primario

Descripción: Cuando entra a su Perfil de usuario, tiene la opción de poder subir documentos a la aplicación y almacenarlos, se tomará en cuenta el tipo de archivo y tamaño. Cada vez que el usuario sube un documento tiene la opción de restringir la visibilidad de los documentos a los demás usuarios, de manera que un documento lo puede poner como público o como privado. Así como de poder eliminar el documento. Los administradores también podrán restringir la gestión de cualquier documento dependiendo de su requerimiento.

Referencias cruzadas: R5, R6, R7, R8, R9, R10

Curso de Eventos: **Administrador de documentos**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el usuario entra a su perfil (nombre y contraseña) y desea subir algún documento a la aplicación. Dependiendo de su Perfil, el usuario tendrá diferentes opciones.	2	Verifica que su sesión de usuario este activa Dependiendo de la respuesta, muestra la información
3	El usuario selecciona el documento a subir	4	Muestra una ventana de exploración para buscar el documento
5	El usuario envía el documento	6	Examina el archivo para verificar que sea de un tipo y tamaño específicos
		7	Envía un mensaje dependiendo del resultado de la transferencia de documentos
		8	Almacena el documento en una carpeta
9	Solicita la descripción y visualización del documento	10	Despliega la información correspondiente en la página
11	Dentro de su perfil se encuentran opciones dentro de las cuales están: la de eliminar y mostrar documentos, al momento de subir un documento puede restringir la visibilidad dejando el documento como público o como privado, el usuario selecciona la opción requerida	12	Manda a la opción requerida
13	Modifica los atributos	14	Actualiza la información
		15	Muestra los cambios

Estableciendo la Jerarquía y calendarización de los Casos de Uso.

Una vez que se ha definido la Etapa de requerimientos y los Casos de uso y Diagramas de Casos de uso, es necesario preparar la transición de la siguiente etapa, "El ciclo de Desarrollo Iterativo del Sistema", para ello es conveniente establecer una jerarquía en los casos de uso y definir como se van a ir analizando cada una de ellos.

Los ciclos de desarrollo son organizados basándose en los requerimientos de los casos de uso, lo que nos dice que en un ciclo de desarrollo se debe implementar uno o más casos de uso, o versiones simplificadas de uno o varios casos de uso. Por lo tanto, un caso de uso complejo se podrá dividir de tal forma de completarlo en más de un ciclo de desarrollo.

Algunas de las ventajas al jerarquizar casos de uso complejos y desarrollarlos en varios ciclos, pueden ser las siguientes:

- Impacto significativo en el diseño de la arquitectura.
- Se obtiene información significativa en el dominio del problema sin mucho esfuerzo.

Se pueden jerarquizar los casos de uso como sigue:

JERARQUÍA	CASO DE USO	JUSTIFICACIÓN
Alto	Administrador de documentos	Es el caso de uso de mayor impacto en el sistema
Medio	Administrador de usuarios	Este caso de uso es menos importante que el administrador de documentos
Medio	Perfil	Su funcionamiento no es de mayor impacto al sistema
Bajo	Pagina principal	Su funcionamiento es de menor impacto al sistema

3.2 Edificación

3.2.1 Construcción de un Modelo Conceptual

Un **Modelo conceptual**: es una representación de conceptos en el dominio del problema.

La primera tarea en la etapa de análisis es identificar conceptos en el dominio del problema y documentarlos en un modelo conceptual. En esta etapa, se recomienda construir el Modelo conceptual de todo el sistema.

Concepto: Es una idea, cosa u objeto y puede ser considerado en términos de:

- Símbolos: Palabras o imágenes que representan un concepto.
- Intención: Definición de un concepto.
- Extensión: Conjunto de ejemplos que se aplican al concepto.

El modelo muestra conceptos, asociaciones entre conceptos y actividades de conceptos.
Los conceptos pueden ser identificados mediante 2 estrategias:

- Usar una lista de categoría de conceptos.
- Identificar los conceptos mediante frases sustantivas.

Errores comunes en la identificación de conceptos.

Cuando se construye un modelo conceptual es común presentar algunas cosas como si fueran atributos, cuando en realidad deben ser conceptos. Para prevenir este error se puede seguir la siguiente regla de dedo.

Si el concepto **X** no es algo que se puede representar con un número o texto en el mundo real, **X** probablemente es un concepto, no un atributo.

Se puede iniciar la construcción del modelo conceptual escribiendo una lista de conceptos candidatos utilizando una lista, la cual contiene categorías comunes que pueden ser consideradas en cualquier orden de acuerdo a su importancia.

Modelo Conceptual para el sistema "Administrador de Documentos de Proyectos Vía Internet/Intranet"

Lista de categorías de conceptos

CATEGORÍA DE CONCEPTO	EJEMPLO
Objetos físicos o tangibles	Computadora
Especificaciones, diseño o descripción de cosas	Descripción documento, Datos de usuario, Visibilidad de documentos, Consulta de usuarios
Lugares	Página principal
Transacciones	Registro usuarios, subir archivos
Línea o renglón de un elemento de transacciones	Usuario. Subir documentos Administrador. Administración de usuarios
Rol de las personas	Usuario registrado y visitante, Líder de Proyecto, Coordinador
Contenedores de otras cosas	Aplicación (almacena en una base de datos)
Cosas dentro de un contenedor	Documentos, datos usuario
Otros sistemas de computo o electromecánicos externos al sistema	Internet
Organizaciones	Instituto Mexicano del Petróleo
Eventos	Registro usuarios, consulta de información, subir documentos, permisos de acceso, mostrar/borrar documentos, Consulta de usuarios registrados
Procesos(A menudo no están representados como conceptos, pero pueden estarlo)	Transferir documentos, registro de usuarios
Reglas y políticas	Restricciones de visibilidad de documentos y de tipo y tamaño Los usuarios solo podrán subir documentos si están registrados
Catálogos	Catálogo de Usuarios, Catálogo de Documentos
Manuales y libros	Manual de Usuario(impreso), ayuda en pantalla

Identificar los conceptos mediante frases sustantivas.

Ésta es una técnica muy útil, se basa en la identificación de sustantivos y frases sustantivas de la descripción textual en el dominio del problema y tomarlos en cuenta ya sea como conceptos candidatos o como atributos. Vea a continuación los conceptos y/o atributos en negrita.

Curso de Eventos: **Administrador de Usuarios**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **Administrador** (Líder de proyecto o Coordinador) desea dar de alta a usuarios. Accede a la opción de **Registro de usuarios**	2	Muestra el **contenido** de la **página** para ser llenado con los **datos** correspondientes
3	El **Administrador** llena la información correspondiente con los **datos personales** de los usuarios, así como la definición de su **perfil** de usuario. Posteriormente el Coordinador definirá el **grupo de trabajo** al que pertenece.		
4	Envía la información	5	Guarda la información de los datos de usuario.
		6	Manda mensaje de envío de datos.
7	Solicita la **consulta** de los **usuarios** registrados	8	Despliega la información.
9	En el menú del administrador se encuentra la opción de **Eliminar Registro de usuario**. Sí selecciona la opción lo manda a la página de eliminar registro.	10	Despliega la página con los usuarios registrados.
11	Busca al usuario e indica que desea eliminar el registro del usuario	12	Mande mensaje de eliminación de registro. Actualiza la información del sistema.

Curso de Eventos: **Página Principal**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **usuario** accede a la **página** principal del proyecto	2	Muestra la **página**
3	El **usuario** ve el **contenido** de la **página**		
4	Selecciona la opción requerida	5	Enlaza a la opción requerida

Curso de Eventos: **Perfil de usuario**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **usuario** pide **iniciar sesión** para entrar a su **perfil** de usuario.	2	Muestra el formulario donde se solicita **nombre de usuario** y **contraseña**
3	El **usuario** proporciona un **nombre de usuario y contraseña** para **validar su entrada.**	4	Verifica que el nombre de usuario y contraseña sean correctos. Dependiendo de su **Perfil de usuario** tendrá diferentes opciones.
		5	Dependiendo del resultado despliega la información, si fue aceptado se dirige a su **Perfil** de usuario, si no se le pide que intente de nuevo
6	Solicita ver sus **datos personales**	7	Muestra la información
8	Desea modificar sus datos personales	9	Actualiza su información y despliega sus datos.

Curso de Eventos: **Administrador de documentos**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **usuario** entra a su **perfil** (nombre y contraseña) y desea subir algún **documento** a la aplicación. Dependiendo de su Perfil, el usuario tendrá diferentes opciones.	2	Verifica que su **sesión** de **usuario** este activa Dependiendo de la respuesta, muestra la información
3	El **usuario** selecciona el **documento** a subir	4	Muestra una ventana de exploración para buscar el **documento**
5	El **usuario** envía el **documento**	6	Examina el **archivo** para verificar que sea de un **tipo** y **tamaño** específicos
		7	Envía un mensaje dependiendo del resultado de la transferencia de **documentos**
		8	Almacena el **documento** en una **carpeta**
9	Solicita la **descripción** y **visualización del documento**	10	Despliega la **información** correspondiente en la página
11	Dentro de su **perfil** se encuentran opciones dentro de las cuales están: la de eliminar y mostrar documentos, al momento de subir un documento puede restringir la visibilidad dejando el **documento** como **público** o como **privado**, el usuario selecciona la opción requerida	12	Manda a la opción requerida
13	Modifica los **atributos**	14	Actualiza la información
		15	Muestra los cambios

Nombrando y modelando cosas.

También se puede emplear la estrategia que se sigue en la construcción de mapas:

- Usar los nombres que existen en el territorio. Se debe utilizar el vocabulario del dominio.
- Excluir características irrelevantes. En el modelo conceptual se deben excluir conceptos en el dominio del problema que no participan en los requerimientos.
- No agregar cosas que no existen. Se deberán excluir cosas que no existen en el dominio del problema bajo consideración.

Tomando como guía los lineamientos expuestos y considerando el sistema completo, se descubren los siguientes conceptos:

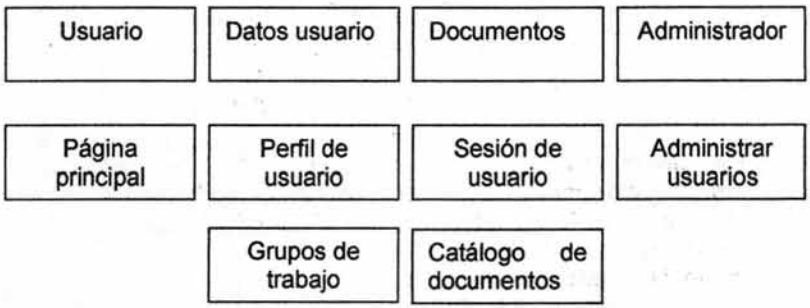


Figura 2. Conceptos del modelo del sistema

3.2.2 Agregando Asociaciones

Una vez que han sido descubiertos los conceptos se procede a encontrar las asociaciones que hay entre ellos.

A continuación se presenta una Línea guía para descubrir las asociaciones mediante una lista de categorías.

CATEGORÍA	EJEMPLO
A es una parte física de B	Computadora ; Aplicación
A es una parte lógica de B	Datos usuario ; Registro Documentos; Administrador de documentos Usuarios; grupo de trabajo
A está contenida lógicamente en B	Datos usuario ; Catálogo usuario Documentos; Catálogo documentos Contenido(inf.general) ; Página Principal
A es una descripción de B	Descripción de Documentos; Documentos Datos usuario; Usuario
A es un elemento de línea en una transacción o reporte de B	Documentos ; subir documentos
A se conoce / introduce /registra /presenta/ captura en B	Datos usuario; Registro usuarios Sesión de usuario; Página Principal Subir documentos; Administrador de Documentos Permisos acceso; Perfil
A es miembro de B	Datos usuario/Registro Documento/Catálogo de documentos
A usa o dirige a B	Perfil de usuario ; Datos usuario Registro usuarios; datos usuarios Perfil ; documentos Sesión de usuario ; Perfil Administrador ; Consulta de usuarios
A se comunica con B	Pagina principal ; Sesión de usuario
A se relaciona con una transacción de B	Administrador ; Registro usuario
A es una transacción relacionada con otra transacción B	Permisos de acceso ; Subir documentos
A es un artículo de una transacción o reporte de B	Documento ; Subir Documentos Consulta de usuarios ; Registro Usuarios

Las asociaciones más importantes son las siguientes:

- A es una parte física o lógica de B
- A está física o lógicamente contenido en B
- A está registrado en B

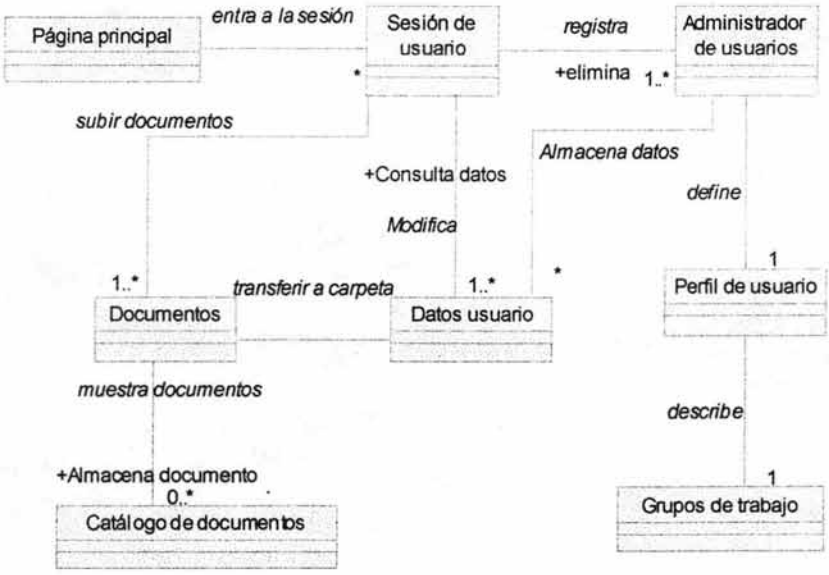


Figura 3. Modelo conceptual

Es necesario hacer notar que **Usuario y Administrador** no se incluye en el modelo conceptual ya que se manejó como actor en los casos de uso.

3.2.3 Definiendo Multiplicidad

La *multiplicidad* define cuántas instancias de un tipo A pueden asociarse a una instancia del tipo B en determinado momento. Las expresiones de multiplicidad son las siguientes:

*	cero o más, muchos
1..*	uno o más
1..40	de uno a cuarenta
5	exactamente cinco
2,4,6	exactamente dos, cuatro o seis

- La aplicación tiene una Página Principal
- El Administrador hace el Registro de los Datos de Usuario
- La Página Principal tiene Información General (Contenido)
- La Sesión de usuario permite acceder a los usuarios a la aplicación
- El Administrador de documentos Sube Documentos a la aplicación
- Los Documentos se guardan en el Catalogo de Documentos
- Los Documentos se muestran en el Contenido de la página

3.2.4 Agregando Atributos

Un **atributo** es un dato que representa un valor lógico de un objeto.

El agregar atributos consiste en identificar la información necesaria para satisfacer los requerimientos del caso de uso bajo desarrollo.

Es necesario tener presente que el Modelo Conceptual es una representación de cosas del mundo real, no representa componentes de software, cualquier atributo deberá interpretarse en el contexto de entidad del mundo real.

En el Modelo Conceptual se deben incluir aquellos atributos para los cuales se requiera recordar cierta información de acuerdo a los requerimientos.

Hay algunas sugerencias que se deben tomar en cuenta cuando se definan atributos.

1. Deberá ser simple, un valor. Por ejemplo: datos boléanos, números, cadenas, tiempo.
2. Algunos tipos de atributos incluyen: direcciones, color, números telefónicos, números del seguro social, códigos postales, tipos enumerados.
3. El tipo de un atributo puede expresarse también como un tipo no primitivo:

- Sí el valor esta compuesto de secciones separadas.

  Si al número de teléfono se le agrega el nombre de la persona.
- Hay operaciones asociadas con análisis de cadenas o validación.

  Un número de seguro social
- Si se tienen otros atributos

  Un precio promocional, deberá tener también fecha de inicio y fin.
- Cuando se usan cantidades y unidades

  Una cantidad a pagar, tiene además una unidad secuencial.

Habiendo definido los conceptos con sus asociaciones se pasa a agregar los atributos que son necesarios para entender el comportamiento del sistema.

Considerando los conceptos mostrados en el diagrama del Modelo Conceptual.
Los atributos encontrados para el Administrador de Documentos son:

- Página Principal
- Sesión de usuario
  - Nombre de usuario
  - Contraseña
- Datos usuario
  - Nombre
  - Apellidos
  - Dirección
  - Correo electrónico
  - Descripción de perfil
  - Descripción de grupo de trabajo
- Grupos de trabajo
  - Número de grupo
  - Nombre de usuario
  - Nombre del grupo de trabajo
- Perfiles de usuario
  - Nombre de usuario
  - Nombre del perfil del usuario
- Administrador de usuarios
  - Datos de usuario
- Documentos
  - Número de documento
  - Nombre del archivo
  - Ruta destino
- Catálogo de Documentos
  - Nombre
  - Tipo
  - Tamaño
  - Nombre Carpeta

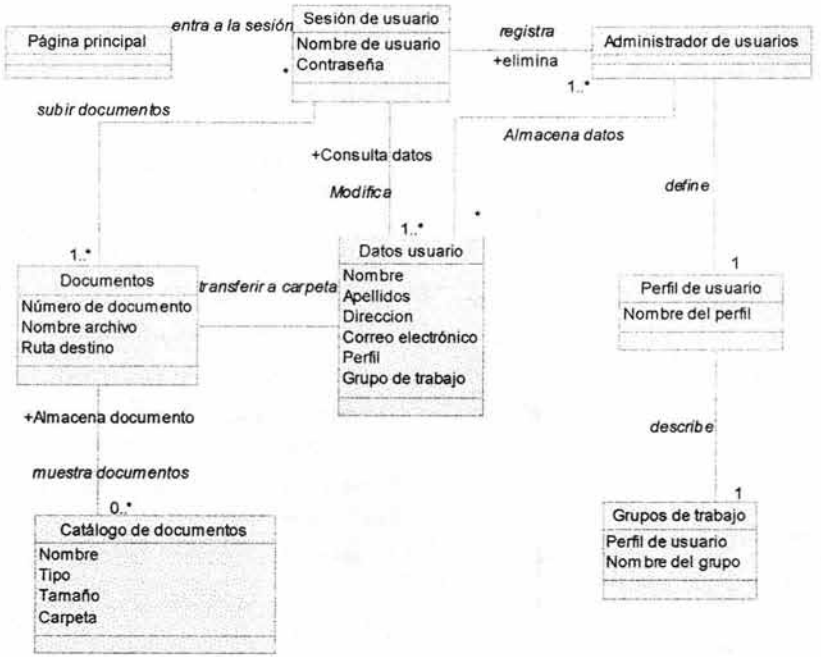


Figura 4. Modelo conceptual con atributos

3.2.5 Definición de términos usados para la aplicación

Se crea un glosario durante la fase de Planificación y Elaboración conforme se van generando términos, siendo continuamente refinado durante cada ciclo de desarrollo, conforme se van generando nuevos términos. El mantenimiento del glosario, es una actividad continua durante el desarrollo del proyecto.

TÉRMINO	CATEGORÍA	COMENTARIOS
Administrador(Líder de Proyecto o Coordinador)	Actor	Actor de la aplicación
Usuario	Actor	Actor Principal de la Aplicación
Datos Usuario	Concepto	Información del usuario sobre sus datos personales
Documentos	Concepto	Archivos que el usuario carga a la aplicación para ser vistos
Grupos de Trabajo	Concepto	Información del usuario respeto al grupo de trabajo que pertenece
Página principal	Concepto	Lugar donde inicia la aplicación
Perfil de usuario	Concepto	Información del usuario respecto al perfil que pertenece
Sesión de usuario	Concepto	Información necesaria del usuario para su identificación
Perfiles	Atributo	Consideración del tipo de usuario.
Nombre	Atributo	Dato para identificar el nombre del usuario
Dirección	Atributo	Datos para identificar el Domicilio del Usuario
Correo electrónico	Atributo	Dato sobre su dirección de correo electrónico del usuario
Nombre Usuario	Atributo	Identifica el nombre de usuario
Contraseña	Atributo	Identifica un valor que el usuario tiene para entrar a la aplicación
Tipo Documento	Atributo	Valor del Documento(público/privado)
Número de documento	Atributo	Número de identificación
Nombre documento	Atributo	Nombre del documento cargado en la página
Ruta	Atributo	Dirección a donde se guardará el documento
Tipo	Atributo	Información del documento de su extensión
Tamaño	Atributo	Información del valor del documento de su tamaño

3.3 Comportamiento del Sistema

3.3.1 Diagramas de Secuencia del Sistema

Los Diagramas de Secuencia del Sistema muestran los eventos que realizan los actores sobre el sistema. Este tipo de diagramas ayudan a descubrir las acciones sobre el sistema, ya que desde la perspectiva del actor, el sistema se ve como una **caja negra.**

Es conveniente la elaboración de este tipo de diagramas en la fase de análisis de cada ciclo de desarrollo, para descubrir acciones no visualizadas en las etapas anteriores.

Diagramas de secuencia.

Los casos de uso sugieren la forma de interactuar de los actores con el sistema, en este proceso un actor genera eventos sobre el sistema, siendo éstos una parte importante en la comprensión del comportamiento del sistema.

Un **Diagrama de secuencia del sistema,** es una representación que muestra, para un escenario particular de caso de uso, los eventos que actores externos generan, su orden y los eventos intermedios.

Un **Diagrama de Secuencia del Sistema** deberá hacerse para el curso de eventos típico del caso de uso, y tal vez elaborar otros diagramas de secuencia para el curso de eventos de otras alternativas interesantes.

Curso de eventos típico del caso de uso

Curso de Eventos: **Administrador de Usuarios**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **Administrador** (Líder de proyecto o Coordinador) desea dar de alta a usuarios. Accede a la opción de **Registro de usuarios**	2	Muestra el **contenido** de la **página** para ser llenado con los **datos** correspondientes
3	El **Administrador** llena la información correspondiente con los **datos personales** de los usuarios, así como la definición de su **perfil** de usuario. Posteriormente el Coordinador definirá el **grupo de trabajo** al que pertenece.		
4	Envía la información	5	Guarda la información de los datos de usuario.
		6	Manda mensaje de envío de datos.
7	Solicita la **consulta** de los **usuarios** registrados	8	Despliega la información.
9	En el menú del administrador se encuentra la opción de **Eliminar Registro de usuario**. Sí selecciona la opción lo manda a la página de eliminar registro.	10	Despliega la página con los usuarios registrados.
11	Busca al usuario e indica que desea eliminar el registro del usuario	12	Mande mensaje de eliminación de registro. Actualiza la información del sistema.

Curso de Eventos: **Página Principal**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **usuario** accede a la **página** principal del proyecto	2	Muestra la **página**
3	El **usuario** ve el **contenido** de la **página**		
4	Selecciona la opción requerida	5	Enlaza a la opción requerida

Curso de Eventos: **Perfil de usuario**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **usuario** pide **iniciar sesión** para entrar a su **perfil** de usuario.	2	Muestra el formulario donde se solicita **nombre de usuario** y **contraseña**
3	El **usuario** proporciona un **nombre de usuario y contraseña** para validar su entrada.	4	Verifica que el nombre de usuario y contraseña sean correctos. Dependiendo de su Perfil, el usuario tendrá diferentes opciones.
		5	Dependiendo del resultado despliega la información, si fue aceptado se dirige a su **Perfil** de usuario, si no se le pide que intente de nuevo
6	Solicita ver sus **datos personales**	7	Muestra la información
8	Desea modificar sus datos personales	9	Actualiza su información y despliega sus datos.

Curso de Eventos: **Administrador de documentos**

No	Acción del Actor	No	Respuesta del sistema
1	El caso de uso inicia cuando el **usuario** entra a su **perfil** (nombre y contraseña) y desea subir algún **documento** a la aplicación. Dependiendo de su Perfil, el usuario tendrá diferentes opciones.	2	Verifica que su **sesión** de **usuario** este activa Dependiendo de la respuesta, muestra la información
3	El **usuario** selecciona el **documento** a subir	4	Muestra una ventana de exploración para buscar el **documento**
5	El **usuario** envía el **documento**	6	Examina el **archivo** para verificar que sea de un **tipo** y **tamaño** específicos
		7	Envía un mensaje dependiendo del resultado de la transferencia de **documentos**
		8	Almacena el **documento** en una **carpeta**
9	Solicita la **descripción** y **visualización del documento**	10	Despliega la **información** correspondiente en la página
11	Dentro de su **perfil** se encuentran opciones dentro de las cuales están: la de eliminar y mostrar documentos, al momento de subir un documento puede restringir la visibilidad dejando el **documento** como **público** o como **privado**, el usuario selecciona la opción requerida	12	Manda a la opción requerida
13	Modifica los **atributos**	14	Actualiza la información
		15	Muestra los cambios

Diagramas de secuencia del sistema.

Diagrama de secuencia del sistema con todos los actores.

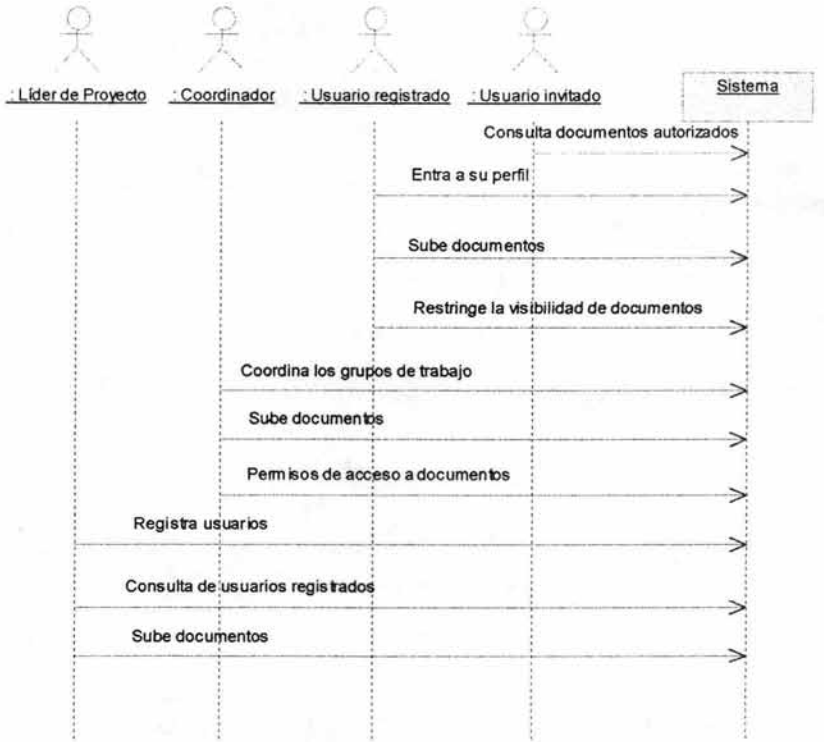


Figura 5. Diagrama de secuencia del sistema

Diagrama de secuencia del Líder de proyecto.

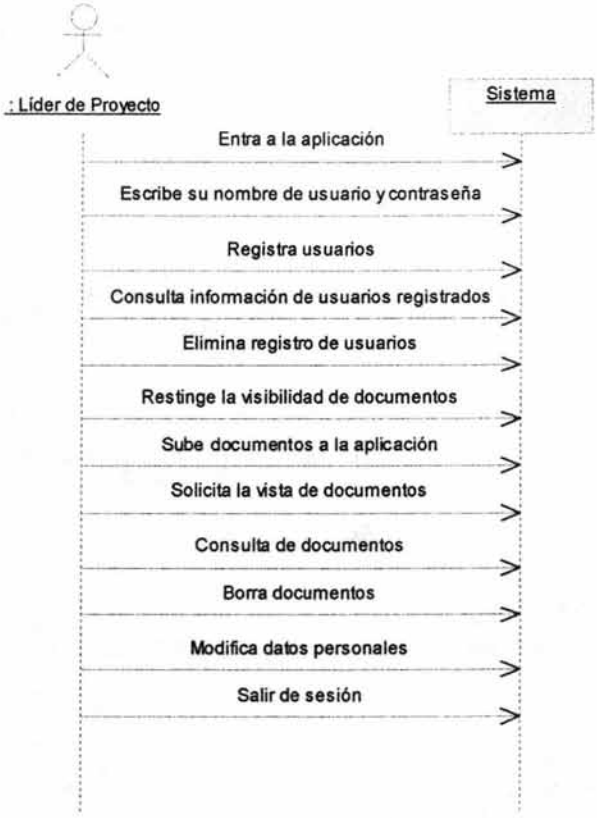


Figura 6. Diagrama de secuencia Líder de Proyecto

Diagrama de secuencia del Coordinador.

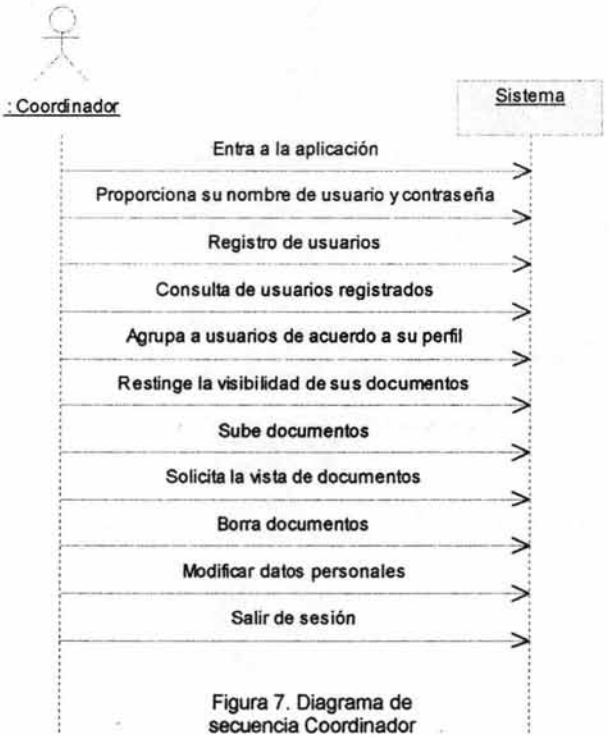


Figura 7. Diagrama de secuencia Coordinador

Diagrama de secuencia de usuarios.

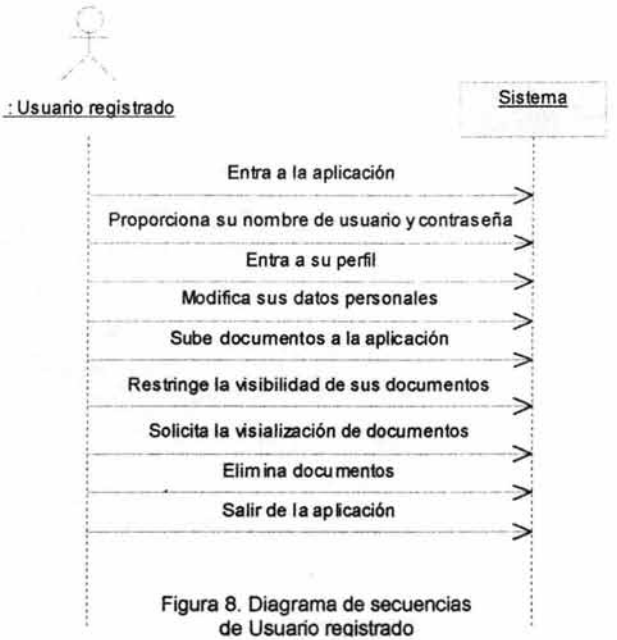


Figura 8. Diagrama de secuencias de Usuario registrado

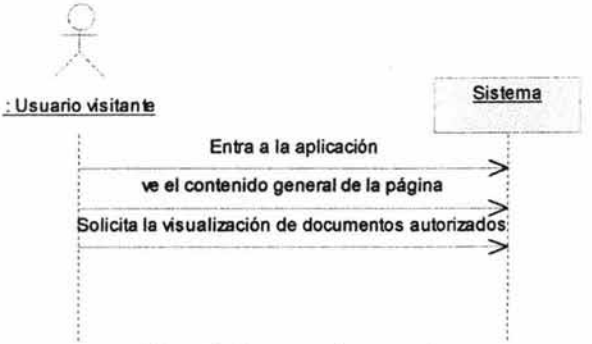


Figura 9. Diagrama de secuencias de Usuario visitante

3.3.2 Comportamiento del Sistema, Contratos

Los **Contratos** ayudan a definir el comportamiento del sistema, describen el efecto de las operaciones sobre el sistema. La creación de contratos de operación del sistema se lleva a cabo durante la fase de análisis dentro de un ciclo de desarrollo correspondiente.

Comportamiento del Sistema.

Antes de iniciar el diseño lógico de la forma en que el sistema deberá trabajar, es necesario investigar y definir su comportamiento como una **caja negra.** El **Comportamiento del sistema** es una descripción de lo que el sistema realiza, sin importar cómo lo hace. Los contratos son un documento útil para describir el comportamiento del sistema en función del cambio de estado que experimenta el sistema cuando se invoca una operación.

3.3.3 Secciones de un Contrato

Se muestra la descripción dé cada una de las secciones de un contrato. No todas las secciones son necesarias, sin embargo, se recomienda no omitir las secciones de *Responsabilidades y Post-condiciones.*

	SECCIONES DE UN CONTRATO
Nombre	Nombre de la operación y sus parámetros
Responsabilidades	Descripción informal de las responsabilidades que deben cumplir las operaciones
Tipo	Nombre del tipo. (concepto, clase, interfase)
Referencias cruzadas	Número de referencia de las funciones del sistema, casos de uso, etc.
Notas	Notas del diseño, algoritmos, etc.
Excepciones	Casos excepcionales.
Salida	Salidas de la Interfase de Usuario, tales como mensajes o registros enviados fuera el sistema.
Pre-condiciones	Asume el estado del sistema antes de la ejecución de la operación
Post-condiciones	Asume el estado del sistema después de que la operación ha concluido.

Como hacer un contrato

1. Identificar las operaciones del sistema desde el diagrama de secuencia.
2. Para cada operación del sistema construir contrato.
3. Iniciar escribiendo la sección de responsabilidades, describiendo informalmente el propósito de la operación.
4. Ahora se completa la sección de Post-condiciones, declarativamente describiendo los cambios de estado que le ocurren a los objetos en el modelo conceptual.
5. Para escribir las Post-condiciones, se recomienda usar las siguientes categorías.
   - Creación y eliminación de instancias.
   - Modificación de atributos
   - Formación y ruptura de asociaciones

Pre-condiciones

Las Pre-condiciones definen suposiciones acerca del estado del sistema al iniciarse la operación. Existen muchas Pre-condiciones que se pueden declarar para una operación, sin embargo, la experiencia sugiere que las siguientes son las más notorias.

- Cosas importantes de probar en el software durante algún punto durante la ejecución de la operación.
- Cosas que no van a ser aprobadas, pero que son importantes para la ejecución de la operación y que hay que tener en cuenta en el futuro.

Post-condiciones

Las Post-condiciones se expresan en el contexto del Modelo Conceptual. Es común que durante la creación de contratos descubrir la necesidad de agregar nuevos conceptos, atributos o asociaciones al modelo conceptual.

Una ventaja al usar las Post-condiciones expresadas como declaraciones de cambio de estado, se ve que el contrato es una excelente herramienta de investigación para describir los cambios de estado requeridos por una operación del sistema sin tener que describir como son realizados dichos cambios. Dicho en otras palabras el Diseño del software puede diferirse y enfocarse analíticamente sobre lo que deberá hacer, en vez de cómo deberá ser realizado.

En la etapa del análisis, el desarrollo se enfatiza en el entendimiento de los requerimientos, conceptos y operaciones relacionadas con el sistema.

3.3.4 Contratos para los casos de uso

Contratos para el sistema **Administrador de Documentos de Proyectos Vía Internet/Intranet.**

Contratos para el caso de uso Administrador de usuarios

Contrato: Registro_datos_usuario()	
Nombre	Registra_datos
Responsabilidades	Dar de alta a un usuario para crear su perfil.
Tipo	Sistema
Referencias cruzadas	R1, R2
Notas	Se almacenan los datos en el Catálogo de usuarios
Excepciones	Envía un error si no fueron llenados los datos correctamente
Salida	Manda mensaje de envío correcto
Pre-condiciones	El Administrador debe acceder a la aplicación para dar de alta a los usuarios.
Post-condiciones	Al introducir los datos personales para dar de alta se almacenan en el Catálogo de Usuarios. Se crea una instancia de Datos Usuario Se crea una instancia de Perfil

Contrato: Eliminar_registro_usuario()	
Nombre	Elimina_datos
Responsabilidades	Eliminar registros de usuarios del sistema.
Tipo	Sistema
Referencias cruzadas	R2, R15
Notas	Se actualizan los registros en el Catálogo de usuarios
Excepciones	Envía un error si no eliminó correctamente
Salida	Manda mensaje de eliminación de registro
Pre-condiciones	El Administrador debe acceder a la aplicación para eliminar el registro del usuario.
Post-condiciones	Busca el usuario ha eliminar y se actualiza los datos del sistema.

Contrato: Grupos_trabajo(nombre,perfil)	
Nombre	Grupos_trabajo
Responsabilidades	Asignar al usuario de acuerdo a su Perfil en un grupo de trabajo.
Tipo	Sistema
Referencias cruzadas	R13, R14
Notas	Si el grupo no existe se crea. Se almacenan los datos en el Catálogo de usuarios
Excepciones	Envía un error si no fueron llenados los datos correctamente
Salida	Manda mensaje de envío correcto
Pre-condiciones	El usuario debe estar registrado.
Post-condiciones	El Administrador designa en que grupo de trabajo el usuario pertenece. Puede hacer una consulta de los usuarios registrados. Se crea una instancia de Datos Usuario Se crea una instancia de Perfil Al introducir los datos del Perfil, se modifican los atributos del usuario.

Contrato: Consulta_datos_usuarios()	
Nombre	Lista_usuarios()
Responsabilidades	Consulta lista de usuarios registrados
Tipo	Sistema
Referencias cruzadas	R13
Notas	Los datos de usuario están guardados en una base de datos
Salida	Muestra la lista de usuarios
Pre-condiciones	El usuario debe estar registrado y debe ser un usuario autorizado.
Post-condiciones	Hace consulta de los usuarios registrados. Se crea una instancia de Datos Usuario

Contratos para el caso de uso Perfil

Contrato: Perfil()	
Nombre	Perfil()
Responsabilidades	Validar el nombre de usuario y contraseña para entrar a su perfil. Asigna un Perfil de usuario de acuerdo a sus actividades dentro del Proyecto.
Tipo	Sistema
Referencias cruzadas	R3
Notas	Dependiendo de los roles del usuario dentro del proyecto será su Perfil de usuario.
Excepciones	Envía un error si los datos de nombre y contraseña son incorrectos
Salida	Valida el Perfil de usuario.
Pre-condiciones	Debe estar registrado el usuario para acceder al Perfil y debe de conocer su nombre de usuario y contraseña.
Post-condiciones	Se crea una instancia de Documentos Se crea una instancia de Permisos_acceso Se crea una asociación con Datos Usuario

Contrato: Fin_sesion()	
Nombre	Fin_sesion()
Responsabilidades	Cerrar sesión del usuario
Tipo	Sistema
Referencias cruzadas	R11
Nota	El usuario debe estar en sesión activa
Salida	Manda mensaje de cierre de sesión
Pre-condiciones	No estar ejecutando ninguna acción
Post-condiciones	Cierra sesión de usuario Eliminación de instancias

Contrato: Consulta_datos_personales()	
Nombre	Consulta_datos ()
Responsabilidades	Manda los datos del usuario
Tipo	Sistema
Referencias cruzadas	R13
Notas	Los datos de usuario están guardados en una base de datos
Salida	Muestra los datos del usuario
Pre-condiciones	El usuario debe estar registrado.
Post-condiciones	Se crea una instancia de Datos Usuario. Hace consulta de los datos personales.

Contrato: Modifica_datos()	
Nombre	Modifica_datos()
Responsabilidades	Modifica los datos de un usuario.
Tipo	Sistema
Referencias cruzadas	R4
Notas	Se almacenan los datos en el Catálogo de usuarios
Excepciones	Envía un error si no fueron llenados los datos correctamente
Salida	Manda mensaje de envío de datos correctos
Pre-condiciones	Se debe entrar al Perfil y una vez ahí seleccionar la opción de modificar datos.
Post-condiciones	Se hace la asociación con Datos usuario. Se modifican los atributos de los datos de usuario. Al modificar sus datos personales se actualizan en el Catálogo de Usuarios.

Contratos para el caso de uso Administrador de documentos.

Contrato: Subir_documentos(documentos)	
Nombre	Subir_documentos(documentos)
Responsabilidades	Carga un documento a la aplicación
Tipo	Sistema
Referencias cruzadas	R5, R6, R7
Notas	Sólo los usuarios registrados podrán subir documentos a la aplicación. Se almacenan los documentos en el Catálogo de Documentos
Excepciones	Envía un error si el documento no tiene nombre, tipo y tamaño correctos.
Salida	Manda mensaje de envío correcto
Pre-condiciones	Se debe iniciar sesión con su Perfil
Post-condiciones	Se crea una instancia con Documentos. Se crea una asociación con Perfil Al cargar sus documentos se transfieren a Carpetas. Se actualiza el Catálogo de documentos.

Contrato: Permisos_acceso()	
Nombre	Permisos_acceso()
Responsabilidades	Dar permisos de visibilidad a los documentos
Tipo	Sistema
Referencias cruzadas	R7, R8, R9,
Nota	Los usuarios que estén permitidos pueden restringir la visibilidad del documento
Salida	Manda mensaje actualización de permisos
Pre-condiciones	El usuario debe de estar en sesión activa de su Perfil.
Post-condiciones	Se hace una asociación con Perfil Se crea una asociación de Documentos Se modifican los atributos de Permisos_acceso Se actualiza el Catálogo de Documentos

Contrato: Mostrar_documento	
Nombre	Lista_doc
Responsabilidades	Muestra los documentos que el usuario solicita
Tipo	Sistema
Referencias cruzadas	R7, R9
Notas	Se mostrarán los documentos del Catálogo de Documentos que estén permitidos.
Salida	Muestra los documentos
Pre-condiciones	El documento debe estar en el Catálogo de Documentos.
Post-condiciones	Se crea una asociación de Documentos

Contrato: Borrar_doc	
Nombre	Borrar_doc
Responsabilidades	Eliminar los documentos que el usuario desee
Tipo	Sistema
Referencias cruzadas	R10
Notas	Sólo los usuarios permitidos podrán borrar documentos. Se eliminarán los documentos del Catálogo de Documentos.
Salida	Manda mensaje de eliminación de documentos
Pre-condiciones	El documento debe estar en el Catálogo de Documentos
Post-condiciones	Se modifica el catálogo de documentos

Contratos para el caso de uso Página Principal

Contrato: Mostrar_contenido	
Nombre	Mostrar_contenido
Responsabilidades	Muestra el contenido, la información general de la Página Principal, así como los documentos autorizados.
Tipo	Sistema
Referencias cruzadas	R9, R12
Salida	Muestra la página
Notas	Cualquier usuario puede acceder a esta parte de la aplicación.
Pre-condiciones	Acceder a la aplicación
Post-condiciones	Despliega el contenido de la página

El análisis de requisitos ha sido modelado utilizando varios diagramas de UML, el software que se utilizó para el diseño de los diagramas para esta aplicación es el Rational Rose en el cual se pretende capturar los requerimientos y modelar las clases básicas y sus relaciones.

En esta fase de análisis, se describe lo que el futuro sistema debe hacer, pero no cómo debe hacerlo. Para eso se requiere de la fase de desarrollo y diseño del sistema en la cual se captura lo recabado del análisis, el siguiente capítulo hace referencia a estas fases de sistema.

Capítulo IV

Diseño y Desarrollo de la aplicación para el Administrador de Documentos

4.1 Diseño: Diagramas de Interacción

Durante un Ciclo de Desarrollo iterativo es posible pasar a la Fase de Diseño una vez completada la documentación de la fase de análisis. Durante esta etapa se desarrolla una solución lógica basada en el paradigma orientado a objetos. La parte central de esta solución se basa en la creación de **Diagramas de interacción.** Estos diagramas muestran la forma en que los objetos se comunican y de esta forma cumplen con los requerimientos establecidos.

Una vez terminados los Diagramas de Interacción propuestas para el ciclo de desarrollo actual, se procede al diseño de los **Diagramas de Clase,** los cuales contienen las clases (interfaces) que serán implementadas mediante software.

Diseño de diagramas de interacción

Durante la etapa del diseño hay dos pasos importantes a dar: la asignación de responsabilidades y la creación de diagramas de interacción.

Los **Diagramas de Interacción** ilustran la manera de cómo los objetos interactúan vía mensajes para llevar a cabo una tarea específica. La creación de Diagramas de Interacción ocurre dentro de la fase de diseño durante el ciclo de desarrollo en cuestión.

Uno de los problemas más comunes en la creación de proyectos utilizando tecnología de objetos es valorar la creación de los diagramas de interacción, al cuidar el asignar responsabilidades y hacer que cada una de ellas y todas en conjunto cumplan con los requerimientos establecidos. Por lo que al asignar responsabilidades y diseñar objetos colaborativos es muy importante, entonces, un porcentaje considerable del trabajo en el proyecto deberá aplicarse a la fase del diseño de diagramas de interacción.

4.1.1 Responsabilidades y métodos

La **Responsabilidad** se define como un contrato u obligación de un tipo o clase. Las responsabilidades están relacionadas a las obligaciones de un objeto en términos de su comportamiento, siendo básicamente de 2 tipos:

1. Responsabilidad de **conocer**.
2. Responsabilidad de **hacer**.

La responsabilidad de **conocimiento** de un objeto incluye:

- Conocimiento acerca de encapsulamiento de datos privados.
- Conocimiento acerca de objetos relacionados.
- Conocimiento de cosas que puede dirigir o calcular.

La responsabilidad de **hacer** de un objeto incluye:

- Hacer alguna cosa por él mismo.
- Inicializar acciones sobre otros objetos.

Las responsabilidades se asignan a objetos durante la fase de diseño. Las responsabilidades relacionadas con el conocimiento son frecuentemente identificadas desde el modelo conceptual ya que en él se muestran atributos y asociaciones.

La traslación de responsabilidades en clases y métodos se ve influenciada por la granularidad de la responsabilidad.

Una responsabilidad no es lo mismo que un método, los métodos son implementados para cumplir responsabilidades. Las responsabilidades se implementan usando métodos, los que actúan tanto sobre otros métodos como sobre objetos.

La manera de implementar las responsabilidades (implementadas como métodos) es mediante la creación de los **Diagramas de interacción**, los cuales muestran las interacciones de los mensajes entre instancias (y clases) en el Modelo de Clase. El punto de partida para estas interacciones son las descripciones de las Post-condiciones de los contratos. El UML define dos conjuntos de diagramas de interacción, donde cualquiera de los dos expresan los mismos conceptos en cuanto a la interacción de los mensajes:

- Diagramas de secuencia. Ilustran la interacción entre objetos, en una secuencia de tiempo, expresada en el eje vertical.
- Diagramas de Colaboración. Ilustran la interacción entre objetos, utilizando un formato de grafo o de red.

Patrones

En la tecnología de objeto, un **Patrón** es la descripción de un problema y su solución, la cual puede ser aplicada a un nuevo contexto de categorías especificas de problemas.

Patrones: Principios Generales de Asignación de Responsabilidades (del inglés GRASP)

Existen 5 categorías de patrones **GRASP.**

- *Experto*
- *Creador*
- *Bajo acoplamiento*
- *Alta cohesión*
- *Controlador.*

4.1.2 Diagramas de Colaboración

Diagramas de interacción del caso de uso Administrador de usuarios.

Tomando como base el Diagrama de Secuencia del Sistema, y las operaciones encontradas, se deberán construir los siguientes diagramas de iteración:

1. Registro datos usuarios
2. Agrupar usuarios.
3. Eliminar registro de usuarios
4. Consulta datos usuarios

1. Contrato: Registro datos.

Este contrato ocurre cuando el Líder de Proyecto desea dar de alta a usuarios.

Contrato: Registro_datos_usuario()	
Nombre	Registra_datos
Responsabilidades	Dar de alta a un usuario para crear su perfil.
Tipo	Sistema
Referencias cruzadas	R1, R2
Notas	Se almacenan los datos en el Catálogo de usuarios
Excepciones	Envía un error si no fueron llenados los datos correctamente
Salida	Manda mensaje de envío correcto
Pre-condiciones	El Administrador debe acceder a la aplicación para dar de alta a los usuarios.
Post-condiciones	Al introducir los datos personales para dar de alta se almacenan en el Catálogo de Usuarios. Se crea una instancia de Datos Usuario Se crea una instancia de Perfil

Definiendo un controlador.

De acuerdo a los patrones es necesario definir el controlador. Una opción es Sesión de usuario.

Figura 1. Definiendo controlador

Registrar un usuario nuevo.

El líder de proyecto, una vez que ha iniciado sesión puede dar de alta a usuarios.

Analizando el modelo conceptual y reflexionando sobre los objetos del dominio, Sesión de usuario es la candidata para crear una nueva instancia de Datos de Usuario.

- *Entra a la página principal.*
- *Valida la entrada como Líder de proyecto.*
- *Se crea una instancia de Datos de usuario.*
- *El mensaje registra() registra elementos a Datos usuario*
- *El mensaje guarda() guarda elementos en Catálogo de usuarios.*
- *El mensaje genera() genera una instancia de Perfil*
- *Se actualiza el Catálogo de Usuarios*

Diagrama de Colaboración, Registro datos:

Figura 2. Diagrama de colaboración Registro datos

2. Agrupar usuarios.

Este contrato ocurre cuando el Líder del proyecto o Coordinador desean agrupar usuarios de acuerdo a su perfil.

Contrato: Grupos_trabajo(nombre,perfil)	
Nombre	Grupos_trabajo
Responsabilidades	Asignar al usuario de acuerdo a su Perfil en un grupo de trabajo.
Tipo	Sistema
Referencias cruzadas	R13, R14
Notas	Si el grupo no existe se crea. Se almacenan los datos en el Catálogo de usuarios
Excepciones	Envía un error si no fueron llenados los datos correctamente
Salida	Manda mensaje de envío correcto
Pre-condiciones	El usuario debe estar registrado.
Post-condiciones	El Administrador designa en que grupo de trabajo el usuario pertenece. Puede hacer una consulta de los usuarios registrados. Se crea una instancia de Datos Usuario Se crea una instancia de Perfil Al introducir los datos del Grupo, se modifican los atributos del usuario.

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *El Coordinador entra a su Perfil.*
- *Se crea la asociación con Datos usuario.*
- *Se valida el perfil.*
- *Se genera una instancia de Grupos de trabajo.*
- *Se crea un Grupo de Trabajo*
- *Se asigna el grupo de trabajo al que pertenece el usuario.*
- *Se actualiza los datos de usuario.*

El Diagrama de colaboración queda como sigue:

Figura 3. Diagrama de colaboración
Grupo de Trabajo

3. Eliminar registro de usuarios.

Este contrato ocurre cuando el Líder del proyecto desea eliminar algún registro de usuario.

Contrato: Eliminar_registro_usuario()	
Nombre	Elimina_datos
Responsabilidades	Eliminar registros de usuarios del sistema.
Tipo	Sistema
Referencias cruzadas	R2, R15
Notas	Se actualizan los registros en el Catálogo de usuarios
Excepciones	Envía un error si no eliminó correctamente
Salida	Manda mensaje de eliminación de registro
Pre-condiciones	El Administrador debe acceder a la aplicación para eliminar el registro del usuario.
Post-condiciones	Busca el usuario ha eliminar y se actualiza los datos del sistema.

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación*
- *El Líder de Proyecto entra a su Perfil.*
- *Se valida el perfil.*
- *Se crea una instancia de Datos de usuario.*
- *El mensaje Consulta() busca los atributos de datos de usuario.*
- *Se elimina el registro de usuario*
- *Se actualiza el Catálogo de Usuarios*

El Diagrama de colaboración queda como sigue:

Figura 4. Diagrama de colaboración Elimina registro

4. Consulta datos usuarios.

Este contrato ocurre cuando el Líder de proyecto o Coordinador desean ver la lista de usuarios registrados.

Contrato: Consulta_datos_usuarios()	
Nombre	Lista_usuarios ()
Responsabilidades	Consulta lista de usuarios registrados
Tipo	Sistema
Referencias cruzadas	R13
Notas	Los datos de usuario están guardados en una base de datos
Salida	Muestra la lista de usuarios
Pre-condiciones	El usuario debe estar registrado y debe ser un usuario autorizado.
Post-condiciones	Se crea una instancia de Datos Usuario Hace consulta de los usuarios registrados.

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación*
- *El Líder de Proyecto o Coordinador entra a su Perfil.*
- *Se valida el perfil.*
- *Se crea una instancia de Datos de usuario.*
- *El mensaje Consulta() busca los atributos de datos de usuario.*

El Diagrama de colaboración queda como sigue:

Figura 5. Diagrama de colaboración Consulta usuarios

Diagramas de interacción del caso de uso Perfil de usuario.

Tomando como base el Diagrama de Secuencia del Sistema, y las operaciones encontradas, se deberán construir los siguientes diagramas de iteración:

1. Perfil
2. Consulta datos personales
3. Modificar datos usuario.
4. Salir de Sesión

1. Contrato Perfil.

El contrato ocurre cuando los usuarios (Líder de Proyecto, Coordinador y usuarios registrados) entran a su Perfil con su nombre y contraseña, se les asigna un Perfil de usuario de acuerdo a sus características.

Contrato: Perfil()	
Nombre	Perfil()
Responsabilidades	Validar el nombre de usuario y contraseña para entrar a su perfil. Asigna un Perfil de usuario de acuerdo a sus actividades dentro del Proyecto.
Tipo	Sistema
Referencias cruzadas	R3
Notas	Dependiendo de los roles del usuario dentro del proyecto será su Perfil de usuario.
Excepciones	Envía un error si los datos de nombre y contraseña son incorrectos
Salida	Valida el Perfil de usuario.
Pre-condiciones	Debe estar registrado el usuario para acceder al Perfil y conocer su nombre de usuario y contraseña.
Post-condiciones	Se crea una asociación con Datos Usuario Se crea una instancia de Documentos

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *El usuario entra a su Perfil.*
- *Da nombre de usuario y contraseña*
- *Se valida el perfil.*
- *Se crea la asociación con Datos usuario.*

El Diagrama de colaboración queda como sigue:

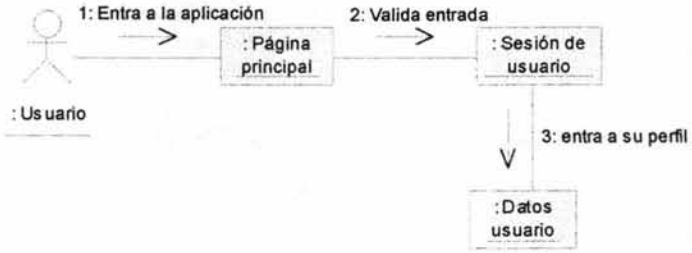


Figura 6. Diagrama de colaboración
Perfil

2. **Contrato Consulta datos personales.**

Este contrato ocurre cuando el usuario desea ver sus datos personales registrados.

Contrato: Consulta_datos_personales()	
Nombre	Consulta_datos ()
Responsabilidades	Manda los datos del usuario
Tipo	Sistema
Referencias cruzadas	R13
Notas	Los datos de usuario están guardados en una base de datos
Salida	Muestra los datos del usuario
Pre-condiciones	El usuario debe estar registrado.
Post-condiciones	Se crea una instancia de Datos Usuario. Hace consulta de los datos personales.

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación*
- *Se valida el perfil.*
- *Se crea una instancia de Datos de usuario.*
- *El mensaje Consulta() busca los datos de usuario.*

El Diagrama de colaboración queda como sigue:

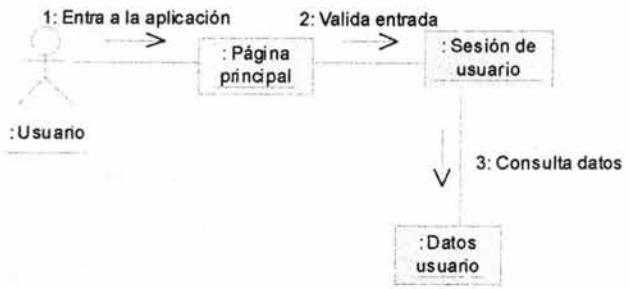


Figura 7. Diagrama de colaboración Consulta datos

3. **Contrato Modifica datos.**

Este contrato ocurre cuando el usuario entra a su perfil y desea modificar sus datos personales.

Contrato: Modifica_datos()	
Nombre	Modifica_datos()
Responsabilidades	Modifica los datos de un usuario y actualiza el catálogo.
Tipo	Sistema
Referencias cruzadas	R4
Notas	Se almacenan los datos actualizados en el Catálogo de usuarios
Excepciones	Envía un error si no fueron llenados los datos correctamente
Salida	Manda mensaje de envío de datos actualizados correctamente
Pre-condiciones	Se debe entrar al Perfil y una vez ahí seleccionar la opción de modificar datos.
Post-condiciones	Se hace la asociación con Datos usuario. Se modifican los atributos de los datos de usuario. Al modificar sus datos personales se actualizan en el Catálogo de Usuarios.

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, y al contrato anterior, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación*
- *Valida la entrada*
- *Se crea la asociación con Datos usuario.*
- *Se modifican los atributos de Datos de usuario.*
- *Se actualiza el Catálogo de usuario.*

El Diagrama de colaboración queda como sigue:

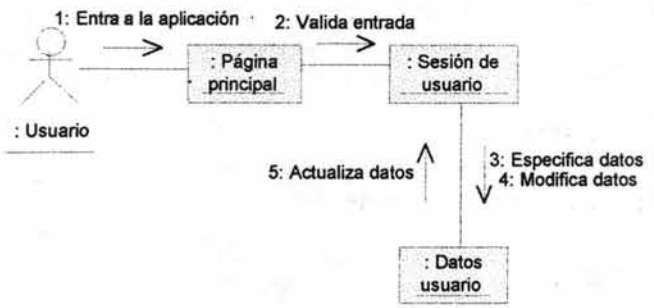


Figura 8. Diagrama de colaboración Modifica datos

4. **Contrato Salir de sesión.**

Este contrato ocurre cuando el usuario desea salir de la aplicación cerrando su sesión de usuario.

Contrato: Fin_sesion()	
Nombre	Fin_sesion()
Responsabilidades	Cerrar sesión del usuario
Tipo	Sistema
Referencias cruzadas	R11
Nota	El usuario debe estar en sesión activa
Salida	Manda mensaje de cierre de sesión
Pre-condiciones	No estar ejecutando ninguna acción
Post-condiciones	Cierra sesión de usuario Eliminación de instancias

Seleccionando el Controlador

De acuerdo al **Patrón GRASP**, y al contrato anterior, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación*
- *Valida la entrada*
- *Se crea la asociación con Datos usuario.*
- *Confirma cierre de sesión*
- *Se cierra sesión de usuario.*

El Diagrama de colaboración queda como sigue:

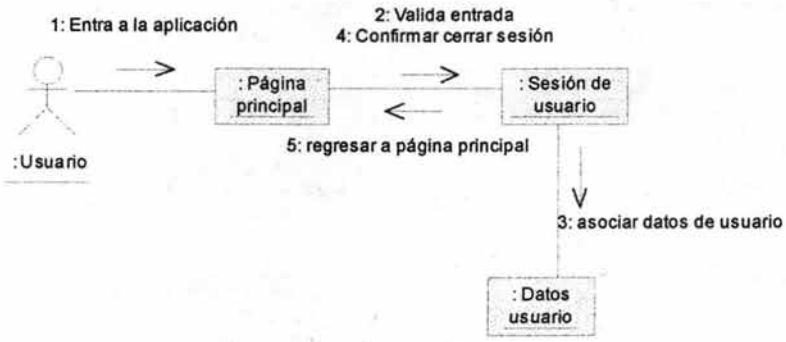


Figura 9. Diagrama de colaboración
Fin de sesión

Diagramas de interacción del caso de uso Administrador de Documentos.

Tomando como base el Diagrama de Secuencia del Sistema, y las operaciones encontradas, se deberán construir los siguientes diagramas de iteración:

1. Subir documentos
2. Permisos de acceso
3. Mostrar documento
4. Borrar documento

1. **Contrato Subir documentos.**

Este contrato ocurre cuando el usuario (Líder de proyecto, Coordinador o Usuario registrado) entra a su perfil y desea subir documentos a la aplicación.

Contrato: Subir_documentos(documentos)	
Nombre	Subir_documentos(documentos)
Responsabilidades	Carga un documento a la aplicación
Tipo	Sistema
Referencias cruzadas	R5, R6, R7
Notas	Sólo los usuarios registrados podrán subir documentos a la aplicación. Se almacenan los documentos en el Catálogo de Documentos
Excepciones	Envía un error si el documento no tiene nombre, tipo y tamaño correctos.
Salida	Manda mensaje de envío correcto
Pre-condiciones	Se debe iniciar sesión con su Perfil
Post-condiciones	Se crea una instancia con Documentos. Al cargar sus documentos se transfieren a Carpetas. Se actualiza el Catálogo de documentos.

Seleccionando el Controlador

De acuerdo con los **patrones**, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación*
- *Valida entrada*
- *Se crea una instancia de Documentos.*
- *Se cargan los documentos.*
- *Verifica que el documento sea válido.*
- *Se transfiere a Carpetas.*
- *Se actualiza el Catálogo de documentos.*

El Diagrama de colaboración queda como sigue:

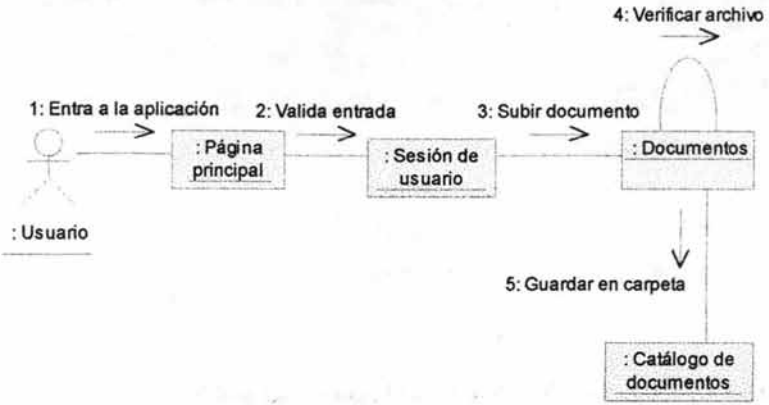


Figura 10. Diagrama de colaboración
subir documentos

2. **Contrato: Permisos de Acceso.**

Este contrato ocurre cuando se van a subir documentos a la aplicación y se desea, guardar en carpetas para que sean vistos por todo el público, por el grupo de trabajo al que pertenece el usuario o sean privados.

Contrato: Permisos_acceso()	
Nombre	Permisos_acceso()
Responsabilidades	Dar permisos de visibilidad a los documentos
Tipo	Sistema
Referencias cruzadas	R7, R8, R9,
Notas	Los usuarios que estén permitidos pueden restringir la visibilidad del documento
Salida	Manda mensaje de documento guardado
Pre-condiciones	El usuario debe ser validado
Post-condiciones	Se crea una asociación de Documentos Se actualiza el Catálogo de Documentos

Seleccionando el Controlador

De acuerdo a los **patrones**, y al contrato anterior, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación.*
- *Se valida la entrada.*
- *Se crea una asociación de Documentos.*
- *Se especifica la ruta de los documentos.*
- *Se actualiza el Catálogo de documentos.*

El diagrama de colaboración queda de la siguiente manera:

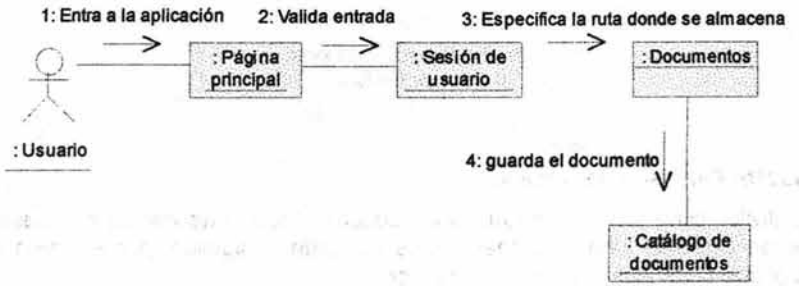


Figura 11. Diagrama de colaboración Permisos acceso

3. Contrato: Mostrar documentos.

Este contrato ocurre cuando el usuario requiere ver algún documento, ya sea usuario visitante o registrado.

Contrato: Mostrar_documento	
Nombre	Lista_doc
Responsabilidades	Muestra los documentos que el usuario solicita
Tipo	Sistema
Referencias cruzadas	R7, R9
Notas	Se mostrarán los documentos del Catálogo de Documentos que estén permitidos.
Salida	Muestra los documentos
Pre-condiciones	El documento debe estar en el Catálogo de Documentos.
Post-condiciones	Se crea una asociación de Documentos Se muestra el documento.

Seleccionando el Controlador

De acuerdo a los **patrones**, y al contrato anterior, se selecciona como ***controlador*** el concepto **Sesión de usuario**.

- *Entra a la aplicación.*
- *Se valida la entrada en caso de que sea usuario registrado.*
- *Se crea una asociación de Documentos.*
- *Se muestra la lista de documentos permitidos.*
- *Se selecciona el documento para ser visualizado.*

El diagrama de colaboración queda de la siguiente manera:

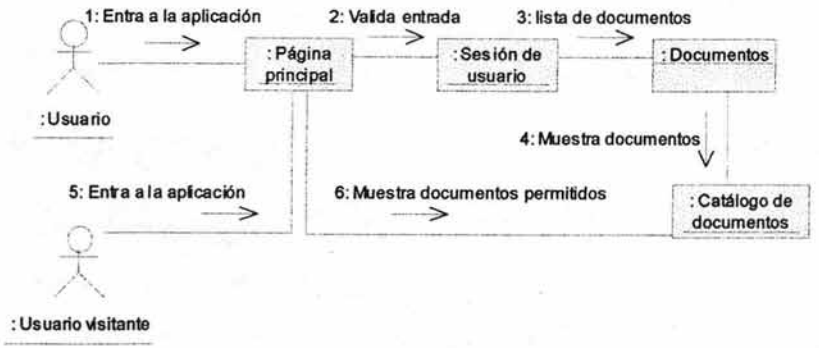


Figura 12. Diagrama de colaboración
Mostrar documentos

4. Contrato: Borrar Documento.

Este contrato ocurre cuando el usuario desea eliminar algún archivo de su perfil.

Contrato: Borrar_doc	
Nombre	Borrar_doc
Responsabilidades	Eliminar los documentos de la aplicación.
Tipo	Sistema
Referencias cruzadas	R10
Notas	Sólo los usuarios permitidos podrán borrar documentos. Se eliminarán los documentos del Catálogo de Documentos.
Salida	Manda mensaje de eliminación de documentos
Pre-condiciones	El documento debe estar en el Catálogo de Documentos
Post-condiciones	Se modifica el catálogo de documentos

Seleccionando el Controlador

De acuerdo al contrato el **controlador** será el concepto **Sesión de usuario**.

- *Se valida la entrada*
- *Se crea una asociación de Documentos.*
- *Se muestra la lista de los Documentos.*
- *Se busca el documento en el Catálogo de documentos.*
- *Se eliminan los documentos requeridos.*
- *Se actualiza el Catálogo de documentos.*

El diagrama de colaboración queda de la siguiente manera:

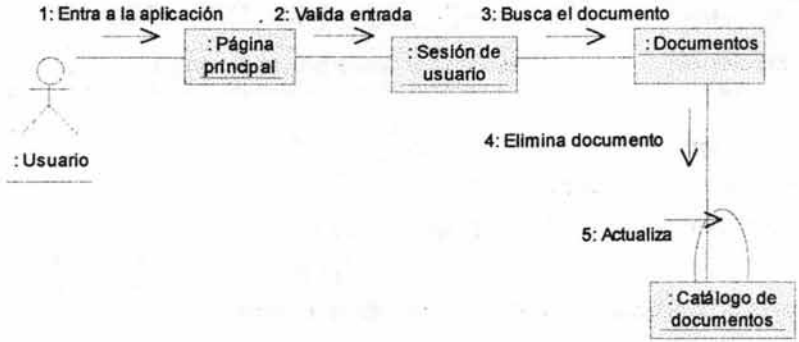


Figura 13. Diagrama de colaboración
Borrar documento

Diagramas de interacción del caso de uso Página Principal.

Tomando como base el Diagrama de Secuencia del Sistema, y las operaciones encontradas, se deberán construir los siguientes diagramas de iteración:

1. **Mostrar Contenido.**

Contrato: Mostrar_contenido	
Nombre	Mostrar_contenido
Responsabilidades	Muestra el contenido, la información general de la Página Principal, así como los documentos autorizados.
Tipo	Sistema
Referencias cruzadas	R9, R12
Salida	Muestra la página
Notas	Dependiendo del Perfil de usuario es lo que se le mostrará en el contenido.
Pre-condiciones	Acceder a la aplicación
Post-condiciones	Despliega el contenido de la página

- *El usuario entra a la aplicación.*
- *Ve el contenido general de la página.*
- *Se muestra la lista de los Documentos autorizados.*

El diagrama de colaboración queda de la siguiente manera:

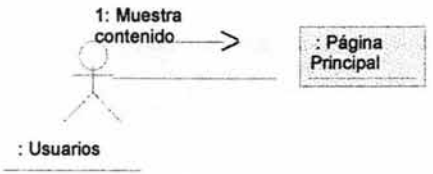


Figura 14. Diagrama de colaboración Mostrar contenido

4.1.3 Determinando Visibilidad

Como se mencionó anteriormente, la visibilidad es la habilidad que tiene un objeto de ver o tener referencia a otro objeto. Por lo tanto si un objeto emisor envía un mensaje a un objeto receptor, el emisor deber ser visible al receptor y el emisor debe tener algún tipo de referencia al receptor.

Existen 4 tipos de visibilidad:

- Visibilidad como atributo o asociación. C es un atributo de B
- Visibilidad como parámetro. C es un parámetro de un método de B
- Visibilidad declarada localmente. C es declarado como un objeto local en un método de B
- Visibilidad global. C es de alguna forma globalmente visible

Estos tipos son considerados cuando: un objeto de B envía un mensaje a un objeto C, C debe ser visible a B.

Cuando se terminó de desarrollar los diagramas de colaboración, se analiza el tipo de visibilidad que tendrá cada mensaje y una vez que se ha definido, el siguiente paso es el desarrollo del diagrama de clases.

4.2 Diseñando Diagramas de Clase

El diagrama de clases se construye cuando han sido desarrollados los diagramas de interacción (aunque pueden hacer paralelamente) y por supuesto con la ayuda del modelo conceptual.

Un diagrama de clases ilustra las especificaciones para las clases de software e Interfases, esta información se incluye como sigue:

- Clases, asociaciones y atributos.
- Interfases, con sus operaciones y constantes.
- Métodos.
- Atributos.
- Navegabilidad
- Dependencias

  En contraste con el modelo conceptual, un diseño de un diagrama de clases presenta definiciones para entidades de software en lugar de conceptos del mundo real.

4.2.1 Creación del Diagrama de Clases del Administrador de Documentos

Identificando clases de software.

Se pueden encontrar analizando cada diagrama de colaboración.

- *Sesión de usuario*
- *Credenciales*. Esta clase no fue identificada en el modelo conceptual, sin embargo analizando nuevamente se necesitaba esta clase para identificar el usuario y contraseña del usuario.
- *Administrador_usuarios*
- *Datos usuario*
- *Perfiles de usuario*
- *Grupos de usuarios*
- *Documentos*
- *Catálogo de documentos*
- *MultipartRequest* (esta clase es la que se utilizará para subir documentos a la aplicación, en el modelo conceptual habíamos tomado el nombre de Documentos, pero se explicará más adelante por que utilizamos esta clase para cargar archivos)

El siguiente punto es dibujar un diagrama de clases para estas clases incluyendo los atributos.

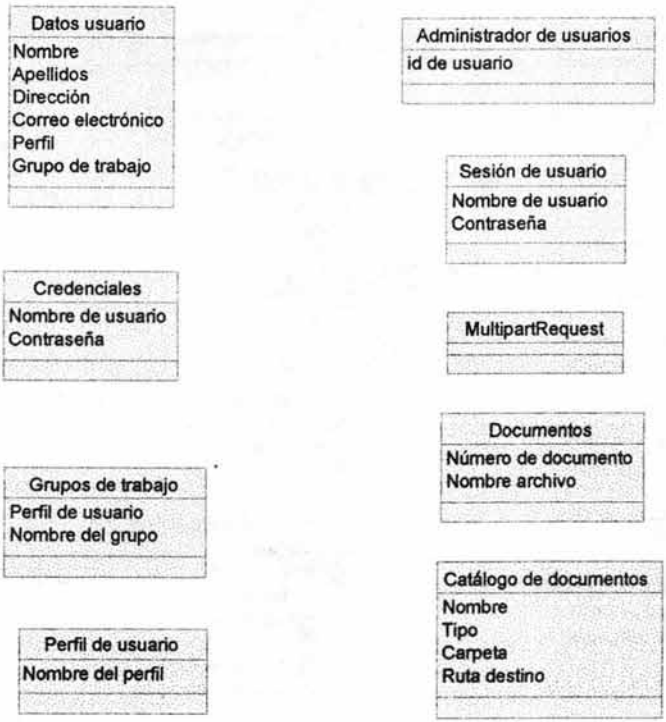


Figura 15. Diagrama de clases

4.2.2 Agregando nombres de métodos

Los métodos que correspondan a cada clase pueden ser identificados analizando los Diagramas de Colaboración.

En general, el conjunto de todos los mensajes enviados a una clase X a través de todos los Diagramas de Colaboración, se deberán incluir en la clase X.

Por inspección de todos los diagramas de colaboración de los casos de uso, las clases quedan conformadas como sigue:

Figura 16. Diagrama de clases con atributos y métodos

4.2.3 Asociaciones y navegabilidad

Cada terminación de una asociación es un **rol**, él cual puede ser adornado con una flecha para indicar navegabilidad. La **navegabilidad** es una propiedad del **rol** que indica el sentido de la navegación a través de la asociación desde el objeto de la clase fuente al objeto de la clase destino. La **navegabilidad** también indica visibilidad, usualmente visibilidad de atributos.

La interpretación más común de una asociación con una flecha de navegación es la visibilidad de atributos desde la clase fuente a la clase destino. Durante la implementación en un lenguaje de programación orientado a objetos se traslada indicando que clase fuente tiene un atributo que se refiere a una instancia de la clase destino.

En el diseño de un diagrama de clases, las asociaciones son seleccionadas de acuerdo a criterios de necesidad de conocer que asociaciones se requieren para satisfacer la visibilidad y necesidades de memoria indicadas por los diagramas de interacción. Esto es en contraste con las asociaciones en el modelo conceptual, las cuales se justifican por la intención de mejorar la comprensión del dominio del problema. Una vez más, se aprecia una distinción en las metas en el diseño de los diagramas de clase y el modelo conceptual, uno es analítico y el otro es una descripción de los componentes del software.

La visibilidad requerida y las asociaciones entre clases son indicadas por los diagramas de interacción

A continuación se presenta el diagrama de clases de la aplicación del Administrador de Documentos con las asociaciones que tienen cada clase.

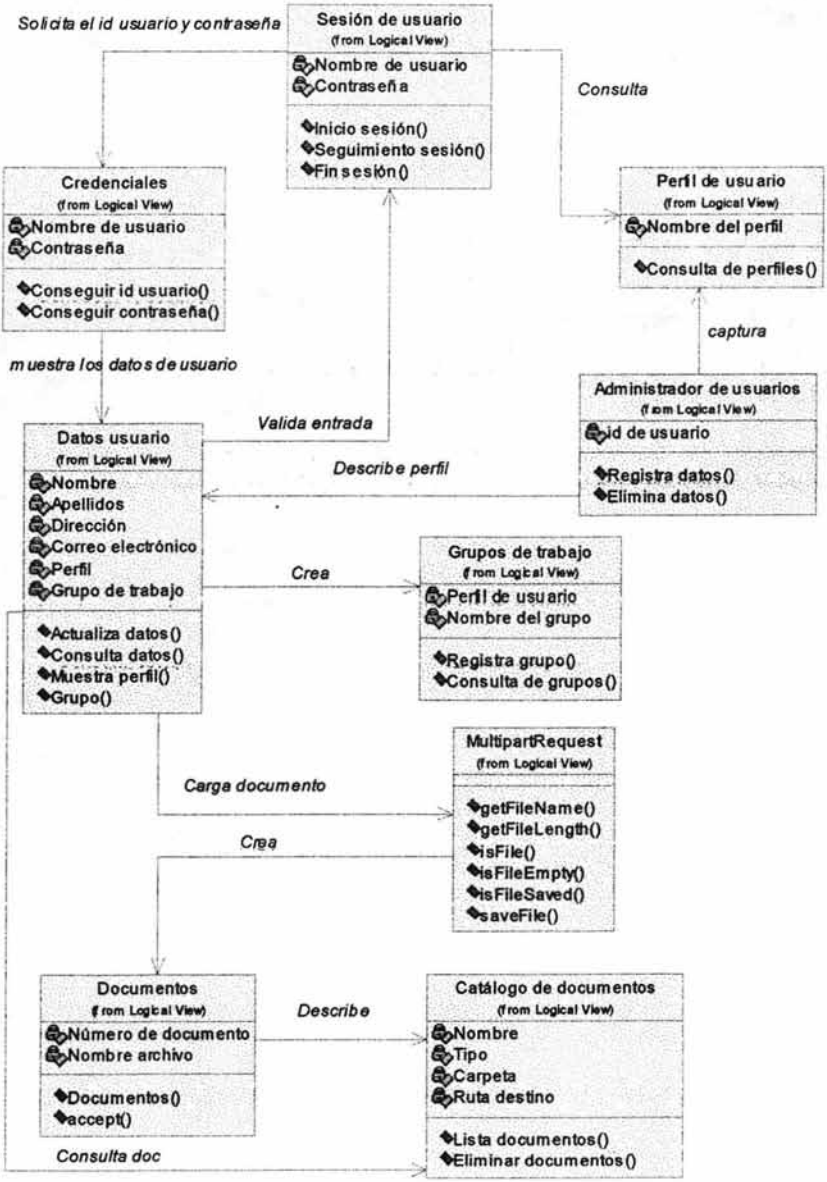


Figura 17. Diagrama de clases con asociaciones

4.3 Desarrollo de la aplicación para el Administrador de Documentos

Se detalla a continuación el desarrollo de los componentes utilizados para la aplicación del Administrador de Documentos de Proyectos.

Se presentarán las clases más importantes para el desarrollo de la aplicación así como la arquitectura utilizada y la Base de Datos.

Para diseñar el modelo de datos, como en el resto de los aspectos del proyecto, se ha seguido el paradigma de orientación a objetos, obteniéndose así el diagrama de clases del sistema (figura 17).

En el análisis, solamente las clases que están en el dominio del problema (conceptos del mundo real) fueron modeladas. En el diseño, se han detallado estas clases definidas en el análisis, además de añadirse otras nuevas relacionadas con las restricciones técnicas y soluciones en el sistema de *software.* Así, en la parte de desarrollo se explicará el funcionamiento del módulo para subir documentos vía web.

4.3.1 Módulo del sistema para subir documentos

Para el módulo de ***subir documentos*** se necesitaba una aplicación que nos ayudara a la carga de archivos.

Esta aplicación se encontró después de un análisis comparándola con otras que tienen similares características. Se requería para el modulo de subir documentos que la aplicación fuera gratuita y de libre uso sin restricciones, ya que se estuvo revisando y comparando con otras aplicaciones y se encontró que algunas aplicaciones sólo eran de prueba y otras tenían restricciones en cuanto a su uso, ya que solo podía ser a nivel personal, no para uso comercial.

La aplicación se llama *Extra uploader* y la página donde la pueden encontrar es URL: http://www.javer.narod.ru/fuplinst.htm

Como se había indicado en el diagrama de clases del sistema para la carga o subida de archivos se necesitaba la clase ***MultipartRequest***, esta clase es la que nos ayuda a subir los documentos a la aplicación.

Con la ayuda de esta clase, se utiliza para llamarla desde un ***Bean*** dentro de una página ***JSP***.
En primer lugar se utiliza código ***HTML*** que muestra el campo *Archivo* de un formulario para buscar el archivo a enviar.

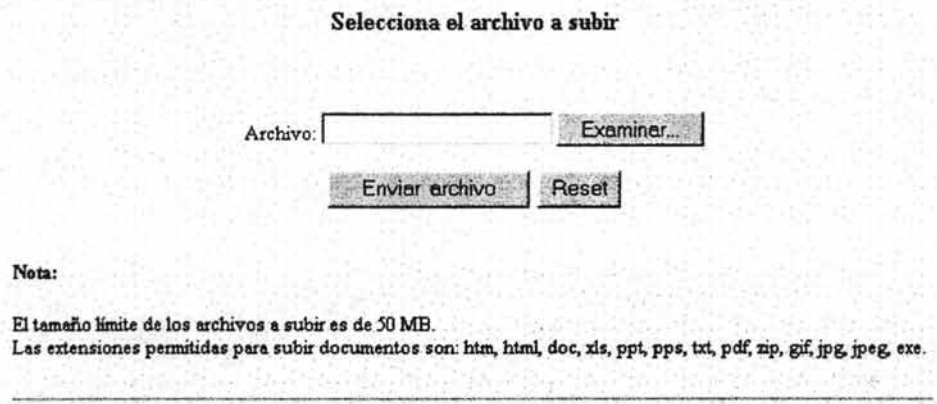


Figura 18. Formulario para subir archivos

El código JSP invoca a la clase mediante un *JavaBean* después de que el usuario ha pulsado el botón de *Enviar Archivo* y comprueba que el fichero elegido cumple las restricciones indicadas.

Las restricciones son en el tamaño del archivo que es de hasta 50 MB y en el tipo de extensión del archivo a subir, las extensiones permitidas para subir documentos son: htm, html, doc, xls, ppt, pps, txt, pdf, zip, gif, jpg, jpeg, exe.

4.3.2 Clases más importantes para la aplicación del Administrador de Documentos de Proyectos

A continuación se explicará brevemente, las clases que se desarrollaron en lenguaje ***Java*** para el Administrador de Documentos.

Clase ***admin_usuarios*** esta clase se utiliza para los registros de datos de usuarios.
El programa contiene funciones para registrar usuarios nuevos y eliminar datos de usuarios.

Clase ***AdministradorSesion*** contiene funciones para administrar y acceder a las sesiones de usuario.
Valida la entrada a un usuario registrado, controla el inicio y cierre de sesión.

Clase ***CredencialesUsuario*** contiene funciones para encontrar el identificador del nombre de usuario y la contraseña para validar la sesión de usuario.

Clase ***Datos_usuario*** contiene funciones para actualizar datos de usuario, así como funciones para la búsqueda de datos de usuarios.

Clase ***grupos_trabajo*** contiene funciones para registrar grupos de trabajo para usuarios registrados.

Clase ***perfil*** contiene funciones para encontrar los perfiles de usuario.

Clase ***Cat_documentos*** contiene funciones para mostrar y eliminar los archivos guardados por los usuarios.

Cada clase de acuerdo a su función esta organizada en *paquetes* o *packages* que es una manera de separarlas para tener un mayor control y administración de las clases.

4.3.3 La arquitectura del Sistema

En la aplicación del Administrador de Documentos, se ha hecho énfasis en los objetos del dominio del problema, ya que representan la esencia del sistema y definen su comportamiento. Sin embargo, un sistema se compone de varios subsistemas, de los cuales los objetos del dominio son uno de ellos. Un sistema de información típico además consta de; una interfase de usuario, y un mecanismo de almacenamiento persistente.

4.3.4 Arquitectura de 3 capas

La Arquitectura clásica: Arquitectura de 3 capas.

La arquitectura de 3 capas es una arquitectura común para los sistemas de información que incluyen interfase de usuario y almacenamiento persistente de datos. En esta arquitectura las capas son tratadas de forma vertical:

- **Presentación:** Ventanas, reportes, etc.
- **Lógica de la aplicación:** Tareas y reglas que guían los procesos
- **Almacenamiento:** Mecanismos de almacenamiento persistente

Una cualidad importante de este tipo de arquitectura es la separación en una capa de software intermedia. La capa de la presentación esta relativamente libre de procesos de la aplicación, también se ve que la capa intermedia con la capa de almacenamiento.

4.3.5 La arquitectura multi-capa; Arquitectura Orientada a Objetos.

En los sistemas de información diseñados bajo el paradigma orientado a objetos, se recomienda una arquitectura de multi-capa, en donde la Capa lógica de la Aplicación es descompuesta a su vez en:

Objetos del dominio:	Clases que representan conceptos del dominio
Servicios:	Objetos de servicio para realizar funciones como interacción de bases de datos, reportes, comunicaciones, seguridad, etc.

Ventajas de una arquitectura multi-capa.

- Aislamiento de la capa Lógica de la aplicación en componentes separados, los cuales pueden ser rehusados.
- Distribución de capas a diferentes nodos o procesos, lo cual incrementa la coordinación y separación de información en sistemas cliente/servidor.
- Permite a los desarrolladores construir niveles específicos, para trabajar en equipo en una sola capa o nivel.

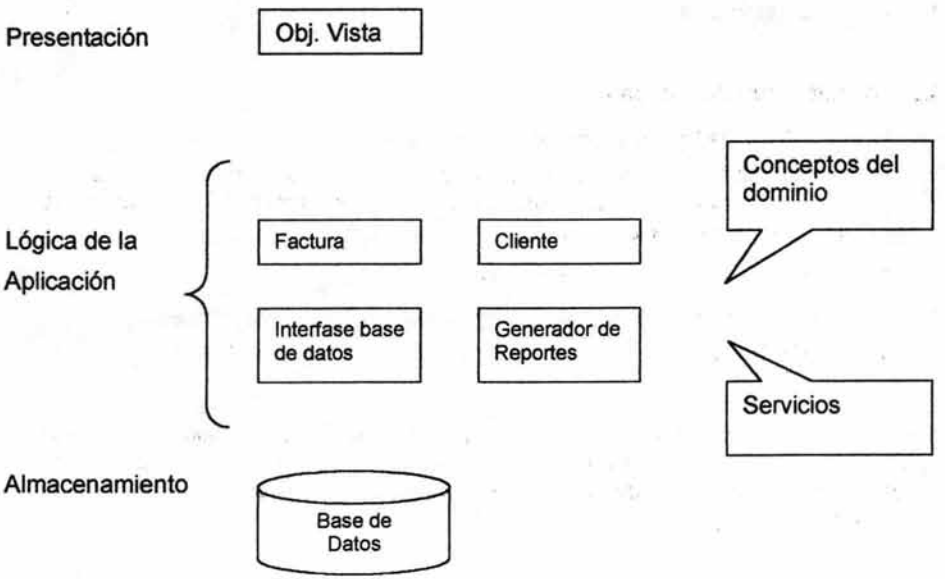


Figura 19. Ejemplo de la arquitectura multi-capa.

4.3.6 Arquitectura del Administrador de Documentos de Proyectos Vía Internet/Intranet

Como se había comentado en el análisis, la aplicación se realizará en el Lenguaje Orientado a Objetos Java y la Tecnología JSP, para el contenido dinámico de la aplicación, y para la parte de Presentación se utilizará HTML.

En el transcurso del desarrollo de la aplicación se vio que no sería necesario utilizar los servlets ya que era suficiente con las páginas JSP por que te permiten incluir código Java dentro de una página HTML normal.

Se vio que utilizando servlets se tenían varios inconvenientes por un lado se necesitaba una gran cantidad de código para realizar una tarea muy simple, además si se quería modificar el servlet se tenía que recompilar el servlet y volver a descargar los ficheros en el servidor.

Para el Administrador de Documentos utilizaremos la arquitectura multicapa ya que responde a las necesidades de organización y control de la aplicación. La arquitectura multicapa permite separar al servidor web que utilizará el contenido HTML estático, del servidor de base de datos, que permitirá acceder a la información que almacena.

Modelo n-capas, Modelo Vista Controlador (MVC)

En este modelo hay una página JSP que actúa como controlador de las peticiones, pasándolas a JavaBeans, o páginas JSP específicas. Al interrelacionarse los componentes que intervienen en la generación de la respuesta en el servidor, es necesario que ya en el diseño de la aplicación se identifiquen claramente los objetos y sus interacciones; es decir, hay que modelar los objetos. Pero también es necesario identificar las páginas JSP que se necesitarán. Estas páginas se dividen habitualmente en dos grupos, uno correspondiente a las que manejan el flujo de la aplicación y la lógica que ello engloba, sin responsabilizarse de ningún tipo de presentación; es decir, solamente contendrán indicaciones de presentación y lógica para presentar contenido dinámico.

Modelo
El modelo contiene el control de la funcionalidad de la aplicación. El modelo encapsula el estado de la aplicación sin saber nada de las otras categorías: Vista y Controlador.

Vista
La vista proporciona la presentación del Modelo, representando el look o la apariencia de la aplicación. La Vista puede acceder a los métodos get() del Modelo para obtener información, pero no tiene acceso a los métodos set() para proporcionar información al Modelo, sino que las actualizaciones en el Modelo

deben realizarse a través del Controlador. Además, la Vista será notificada cuando se produzca algún cambio en el Modelo.

Controlador

El controlador es quien reacciona a las acciones del usuario y el encargado de crear y asignar valores al Modelo para su funcionamiento.

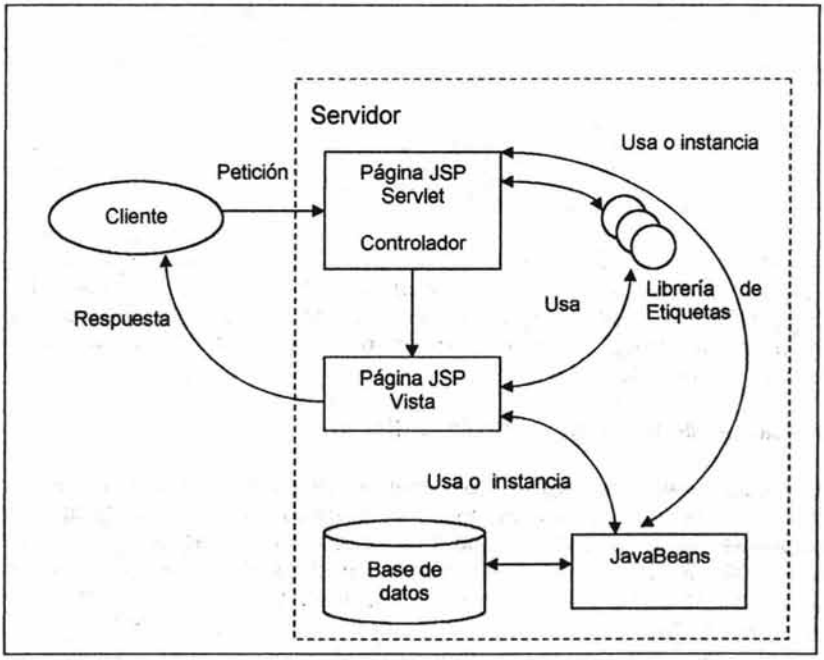


Figura 20. Modelo n-capas

La arquitectura que utiliza páginas *JSP*, *JavaBeans* y *JDBC* es la que se utilizará en la aplicación del Administrador de Documentos.

A continuación se exponen las tres partes de la arquitectura Modelo-Vista-Controlador, para la aplicación del Administrador de Documentos.

El Modelo

El modelo de la aplicación está constituido por los JavaBeans de las Clases *AdminSesion, Credenciales, Datos Usuarios y MultiPartRequest*.

Los *JavaBeans* encapsularían los métodos de acceso a la base de datos a través de *JDBC*, y las páginas *JSP* utilizarán estos *JavaBeans* a través de la acción *jsp:useBean*.

```
<%@ page errorPage="errorPage.jsp" import="java.sql.*,admon.*" %>
<%@ include file="db.jsp"%>
<jsp:useBean id="administradorSesion" class="com.instantjsp.AdministradorSesion"
  scope="application" />
<jsp:useBean id="credenciales" class="com.instantjsp.CredencialesUsuario"
  scope="session" />
<%
  if (administradorSesion.yaRegistrado(credenciales)) {
%>
      <jsp:forward page="PrimeroFinSes.jsp" />
<% } %>
<html>
<head>
  <title>Inicio de sesión</title>
</head>
<body bgcolor="#c0c0c0" link="#999999" vlink="#999999" alink="#999999"
onLoad="darEnfoque()">
<center>
<font size=+2><B>REGISTRO DE INICIO DE SESIÓN</B></font>
<FORM NAME="login" METHOD=POST ACTION="validar.jsp" >
<TABLE WIDTH="50%">
<TR>
<TD align=right>
<font size=+1><B>Nombre usuario:</B></font></TD>
<TD> <font size=+1><INPUT NAME="usuario " TYPE="TEXT" LENGTH="9"
 MAXLENGTH="9"></font></TD>
</TR>
    <TR>
      <TD align=center><font size=+1><B>Contraseña: </B> </font> </TD>
<TD>
<font size=+1><INPUT NAME="password" TYPE="PASSWORD" LENGTH="8"
MAXLENGTH="8"></font></TD>
</TR>
 </TABLE>
<font size=+1>
<b><INPUT TYPE="button" VALUE="INICIAR SESION" onClick="remitirForm()" >
<INPUT TYPE="button" VALUE="RESTABLECER" onClick="restablecerForm()"></b>
</font>
</FORM>
</center>
</body>
</html>
```

Figura 21.Código donde se utiliza JavaBean

En el código anterior se muestra el uso de *JavaBeans* al utilizarlo para validar la entrada a un usuario con su ***nombre de usuario y contraseña***, al momento de introducir los datos en los campos correspondientes los *JavaBeans* hacen una petición a la base de datos para buscar si el usuario está registrado para poder entrar a la aplicación.

La Vista

La vista en la arquitectura de la aplicación está constituida por el conjunto de páginas JSP y HTML que se utilizan para presentar información que genera el Modelo y enviar los comandos que ejecutan acciones en el Controlador.

El usuario verá varias pantallas, en función de la acción que esté realizando. La primera pantalla es la que corresponde a la Bienvenida de la aplicación que es la página index.html es la que presenta la aplicación al usuario y muestra un menú que permite seleccionar las diferentes opciones que se le presenta al usuario.

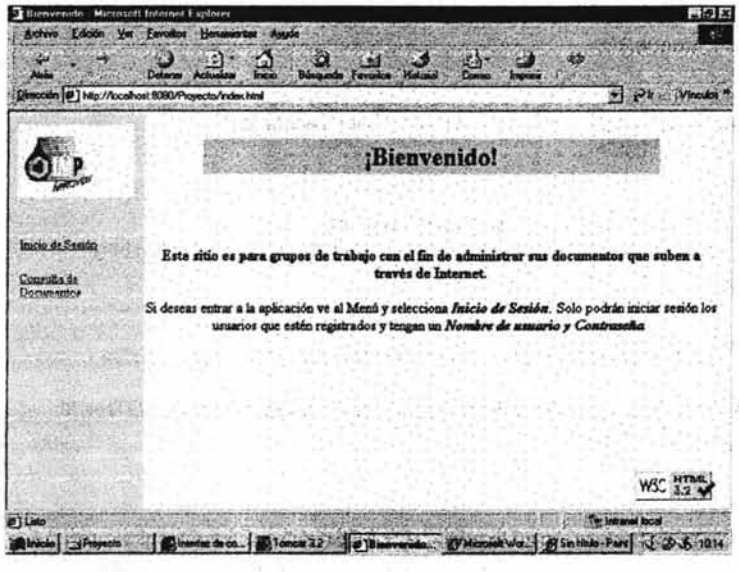


Figura 22. Pantalla de inicio
Index.html

Una parte importante corresponde al hiperenlace de cada elemento de la lista del menú construido en base a la etiqueta *href*, que realiza la invocación de páginas JSP para entrar de lleno en la aplicación.

El Controlador

El Controlador es el elemento del patrón MVC que constituye el corazón de la aplicación y al que se encomienda el ordenamiento de las acciones que van a realizar Modelo y Vista.

El Controlador carga el Modelo, crea y controla los JavaBeans del Modelo, los actualiza en respuesta de las acciones que se provocan desde la Vista y los prepara para generar las respuestas que se van a enviar al navegador, en función de la acción que haya sido solicitada.

En la figura se muestra como interactúan los componentes y se muestra como trabaja el controlador con los JavaBean de las clases.

La vista hace una petición, la recibe el controlador y dependiendo de la acción a ejecutar la dirige al modelo.

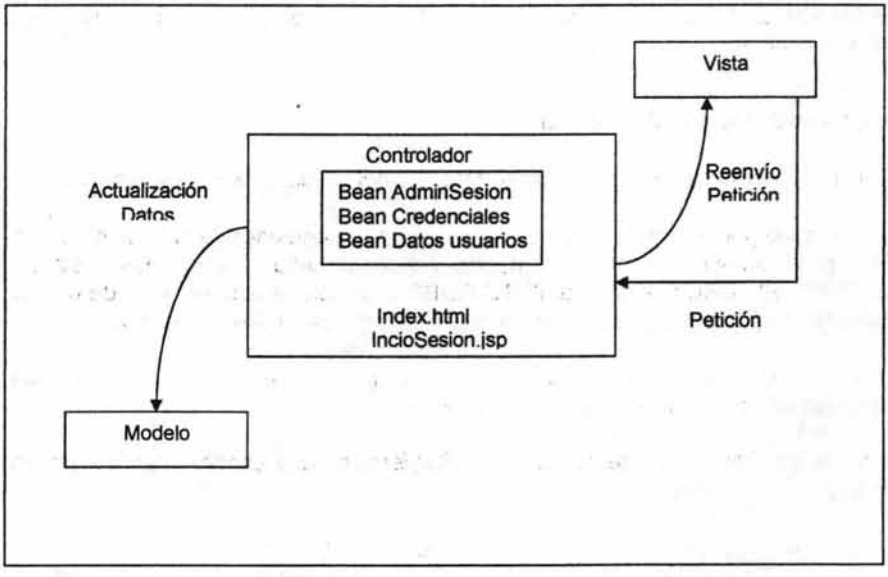


Figura 23. Esquema del Controlador

4.4 Base de Datos

El lenguaje de programación Java, por ser robusto, seguro, independiente de la plataforma, fácil de usar y de entender, y automáticamente se puede descargar de la red, se ha convertido en un excelente lenguaje para la creación de aplicaciones que acceden a bases de datos.

Por ello, es necesaria una manera de comunicar las aplicaciones Java con una gran variedad de fuentes de datos distintas.

Además, la naturaleza orientada a las redes, tanto Intranet como Internet, del lenguaje Java le hacen un candidato ideal para la filosofía cliente-servidor, resolviendo el tema de aplicaciones y paginas *web* que puedan acceder a bases de datos, lo cuál será básico para la aplicación del Administrador de Documentos.

El *API JDBC* es el mecanismo que permite hacer esto. Su valor reside en permitir que una aplicación acceda virtualmente a cualquier base de datos y se ejecute desde cualquier plataforma usando una Máquina Virtual de Java *(JVM, Java Virtual Machine)*.

4.4.1 Base de Datos del Sistema

Para la aplicación no se requería un sistema gestor de base de datos específico.

Se ha optado por el uso de *Microsoft Access* como manejador de la base de datos de la aplicación, ya que es de fácil manejo y acceso, como veremos más adelante, el uso del **API JDBC** y el puente **JDBC-ODBC** deja abierta la posibilidad de utilizar cualquier otro motor de base de datos que disponga de un *driver* apropiado.

En primer lugar se creó la base de datos en *Microsoft Access*, implantando en ella todas las tablas, sus relaciones y restricciones.

La base de datos del sistema se llama ***Registro.mdb*** y contiene las siguientes tablas:

- Idpassword. Contiene los datos de los usuarios registrados.
- Grupos. Contiene los nombres de los grupos registrados.
- Perfiles. Contiene los nombres de los perfiles de usuario.
- Estadpags. Contiene los datos de las páginas visitadas.

A continuación se detallará el diccionario de datos de las tablas utilizadas para la aplicación del Administrador de Documentos

Diccionario de Datos

Tabla Idpassword.

- Id_us. Es un identificador del número del usuario. Debe de ser único y obligatorio. Tipo de dato: Numérico.
- Nombre_usuario. Nombre del usuario registrado. Debe ser obligatorio. Tipo de dato: Texto.
- Apellido_pat. Apellido paterno del usuario. Debe ser obligatorio. Tipo de dato: Texto.
- Apellido_mat. Apellido materno del usuario. Debe ser obligatorio. Tipo de dato: Texto.
- Dirección. Domicilio del usuario. Debe ser obligatorio. Tipo de dato: Texto.
- Email. Correo electrónico del usuario. Debe ser obligatorio. Tipo de dato: Texto.
- Idusuario. Es un identificador de su nombre de usuario para entrar a la aplicación. Debe ser obligatorio. Tipo de dato: Texto.
- Password. Es la contraseña del usuario para entrar a la aplicación. Debe ser obligatorio. Tipo de dato: Texto.
- Perfil. Es el tipo de perfil que tiene el usuario. Debe ser obligatorio. Tipo de dato: Texto.
- Grupo. Es el grupo de trabajo al que pertenece el usuario. Debe ser obligatorio. Tipo de dato: Texto.

Tabla Grupos

- Id_num. Es un identificador para el número que tiene un grupo. Debe de ser único y obligatorio. Tipo de dato: Numérico.
- Nombre_grupo. Nombre de un grupo de trabajo de usuarios. Debe ser obligatorio. Tipo de dato: Texto.

Tabla Perfiles

- Num. Es un identificador para el número que tiene un perfil de usuario. Debe de ser único y obligatorio. Tipo de dato: Numérico.
- Perfil. Nombre del perfil de usuario. . Debe ser obligatorio. Tipo de dato: Texto.

Tabla Estadpags

- Idusuario. Nombre de usuario para entrar a la aplicación. Debe ser obligatorio. Tipo de dato: Texto.
- Idsesion. Sesión del usuario. Tipo de dato: Texto.
- Fe. Fecha en la que entra el usuario a la aplicación. Tipo de dato: Fecha.
- Ho. Hora en la que entra el usuario en sesión. Tipo de dato: Fecha.
- Descpags. Descripción de la página al que el usuario accede. Tipo de dato: Texto.

Diagrama de la base de datos

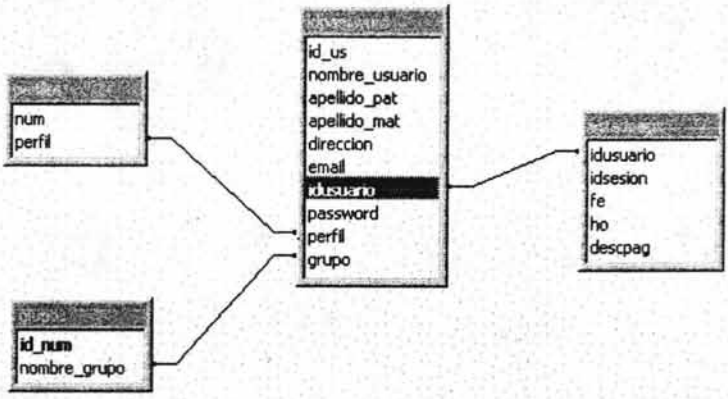


Figura 24. Tablas de la base de datos

Así se encuentran definidas las tablas para la aplicación del Administrador de Documentos en la que la tabla más importante es la de *idpassword* ya que contiene los datos del usuario registrado. Para poder entrar a la aplicación se le pide un nombre de usuario y contraseña y estos son buscados en esta tabla.

4.4.2 JDBC

JDBC es una *API (Application Programming Interface)* pura de Java que se relaciona muy a menudo con el acrónimo ODBC *(Open Database Connectivity)* por lo que se suele expresar como *Java Database Connectivity* aunque oficialmente, según *Javasoft*, no es acrónimo de nada.

El **API JDBC** es una parte integral de la plataforma Java, por lo tanto no es necesario descargar ningún paquete adicional para usarla.

La versión JDBC 1.0 se incluye con el *Kit* de Desarrollo de Java (JDK) en su versión 1.1. La versión JDBC 2.0 es parte del JDK 1.2 y conserva todas las características de la anterior versión, añadiendo otras nuevas, que incluyen los *ResultSets* navegables en ambos sentidos y soporte para los nuevos tipos de datos SQL3.

El API JDBC es básicamente un paquete de Java (*java.sql*) que contiene un conjunto de clases e *interfases* escritas en Java. Estas clases representan conexiones con bases de datos, sentencias SQL, conjuntos de datos y metadatos, etc.

Para trabajar con **JDBC** es necesario tener *controladores* (*drivers*) que son los encargados de implementar esas *interfases* genéricas proporcionadas por el API. El *driver*, que debe ser proporcionado por cada motor de bases de datos, se encargará de traducir las llamadas al lenguaje específico de la base de datos.

De esta manera, el programador puede abstraerse de la programación específica de la base de datos, creando código que funcionará para todos los orígenes de datos que cuenten con un *driver* **JDBC** con tan solo cambiar el *driver*.

Se realizaría la conexión para una base de datos *Microsoft Access* que requiere el puente **JDBC-ODBC**.

El puente **JDBC-ODBC** traduce las llamadas **JDBC** a llamadas **ODBC** y se las envía al *driver* ODBC, que actúa como una capa intermedia entre el *driver* JDBC y las librerías proporcionadas por el vendedor de la base de datos. Se implementa tanto en código binario como en Java. Por lo tanto, el *driver* ODBC, y en muchos casos las librerías de acceso a la base de datos, deben estar presentes en la máquina cliente.

Creación de un origen de datos ODBC

Antes de poder realizar la conexión se debe crear un *origen de datos ODBC*, esto se hace por que el manejador de bases de datos que utilizamos es *Microsoft Access*, si posteriormente se quiere cambiar a otro tipo de manejador de bases de datos se tendría que verificar que driver es el que utiliza dicho manejador. Los pasos a seguir se indican a continuación:

En el menú *Inicio* se va a *Configuración ->Panel de Control -> Orígenes de datos ODBC* ->Ventana de **"Administrador de orígenes de datos ODBC"**

En esta ventana se pulsa el botón **"Agregar..."**, apareciendo la ventana **"Crear nuevo origen de datos"** en donde se debe realizar la selección del controlador de base de datos que se va a utilizar, en este caso ha de ser el driver de **Microsoft Access(*.mdb)**. Se selecciona y se pulsa **"Finalizar"**. Aparecerá la ventana de **"Configuración de ODBC Microsoft Access"** en la que se introduce la información correspondiente al origen de datos:

Nombre del origen de datos: Registro
Descripción: Base de Datos del Administrador de Documentos

En el cuadro **"Base de Datos"** se pulsa el botón **"Seleccionar..."**, que abre la ventana de exploración de ficheros en donde se selecciona el nombre del fichero **.mdb** que contendrá la base de datos; en este caso la *base de datos* se llama ***Registro.mdb***.

Se pulsa **"Aceptar"**, para cerrar la ventana de configuración, apareciendo en la ventana de **"Configuración de ODBC Microsoft Access"** el camino completo del archivo ***Registro.mdb*** en el cuadro **"Base de datos"**. Pulsar **"Aceptar"** para cerrar la ventana de **"Configuración"** y **"Aceptar"** de nuevo para cerrar la ventana del **"Administrador de orígenes de Datos ODBC"**.

Una vez que se ha creado el origen de datos de **ODBC** se tiene que especificar el *driver* para la conexión con **JDBC**.

La conexión se realizará de la siguiente forma:

El *driver* para Microsoft Access es: ***JDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver***

La conexión para la base de datos. ***JDBCConnectionURL=jdbc:odbc:Registro***.
En donde ***Registro*** es el nombre de la base de datos de la aplicación.

Si la base de datos lo requiere puede poner un nombre de usuario y contraseña para autentificar la base de datos. Para esta aplicación no se requiere usuario y contraseña.

El uso del *API JDBC* deja abierta la posibilidad de utilizar cualquier motor de base de datos que disponga de un *driver* para **JDBC**. En caso de no disponerse de un *driver* de este tipo, como sucede con *Access*, la conexión se podría realizar usando el puente **JDBC-ODBC**, pero siempre teniendo en cuenta que el uso de **ODBC** exige la definición del origen de datos de forma local, con lo cual esta solución no permitiría el acceso remoto a la base de datos.

4.4.3 Pool de Conexiones

Es necesario tener un control del número máximo de conexiones concurrentes a la Base de Datos. Hay factores externos que determinan este número, tales como la cantidad de licencias disponibles en el motor y la memoria requerida del sistema para crear las conexiones.

Un **pool** o depósito de objetos es una estructura de datos que permite compartir los objetos que almacena entre varias aplicaciones o entre los diferentes módulos que las componen. En este caso particular, hablamos de objetos en Java. De esta manera, conseguimos eliminar el tiempo de carga, pues no necesitamos instanciar o construir esos objetos, y a la vez reducimos el trabajo del recolector de basura (garbage collector). En esta aplicación se implementa dicho pool de objetos y a partir de ahí, un pool de conexiones a la base de datos.

Con el objetivo de manejar las conexiones concurrentes a la Base de datos en forma eficiente, es recomendable definir un **pool o depósito de conexiones**.

Para crear un **pool de conexiones**, se crea y mantiene un pool de objetos **Connection** a la misma base de datos (igual host y base de datos). Este proceso se realiza una única vez.

Cada vez que desde una componente web (servlet,jsp) se requiere una conexión libre, esta se asigna desde el pool y la conexión deja de estar disponible para otro requerimiento. Una vez finalizada la operación sobre la conexión, el servlet o JSP la devuelve al pool y de esta forma vuelve a estar disponible.

El número máximo de conexiones configurado controla la cantidad de conexiones simultáneas a la base de datos. Todos los requerimientos que se realicen cuando estén en uso todas las conexiones creadas, serán rechazados.

Para la realización de un *pool* o depósito de conexiones se necesitó una clase llamada ***ConnectionPool.java*** y de un archivo de configuración del *driver* de la base de datos para el límite de conexiones.

```
JDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver
JDBCConnectionURL=jdbc:odbc:Registro
ConnectionPoolSize=10
ConnectionPoolMax=100
ConnectionUseCount=5
ConnectionTimeout = 2
User=""
Password=""
```

Figura 25. Archivo de configuración para el pool de conexiones

Para la aplicación se instaló el driver para el motor de base de datos Microsoft Access, con un tamaño inicial de 10 conexiones y máximo 100 conexiones, con respecto al usuario y contraseña la base de datos que se utiliza no los requiere para autentificarla es por eso que se deja en blanco.

Una vez terminado el desarrollo del sistema, en el apartado siguiente se tratarán los siguientes temas: las pruebas a realizar para comprobar que la aplicación instalada funcione correctamente , las operaciones que se deben realizar para llevar a cabo la puesta en marcha de la aplicación y por último, el recorrido del sistema.

Capítulo V

Realización de las pruebas para la implantación del Administrador de Documentos

5.1 Pruebas de la aplicación

Siguiendo la metodología orientada a objetos utilizando UML, las pruebas del sistema se han realizado durante todo el proceso de desarrollo, como fase final de cada iteración en la construcción del sistema, después del diseño y la codificación.

Así, se han realizado las pruebas que se obtuvieron en el diseño a partir de los contratos de los casos de uso, comprobando si el resultado obtenido coincide con el resultado esperado. Estas pruebas miran al sistema como una "caja negra", con el fin de validar que el sistema tenga la funcionalidad y comportamiento esperados por el usuario final.

A continuación se muestran las pruebas que se han realizado, junto con el resultado esperado y una marca indicando si el resultado obtenido coincide con el mismo.

Caso de uso Administrador de usuarios.

1. **Registro datos usuarios**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Dar de alta un usuario del proyecto.	Mensaje de registro insertado correctamente.	✓
Dar de alta un usuario que ya está resgistrado.	Mensaje de error.	✓
Dejar vacío un campo al llenar los datos	Mensaje de error no debe de haber campos vacios.	✓
Introducir datos repetidos como id de usuario y contraseña.	Mensaje de error datos repetidos.	✓

2. **Agrupar usuarios.**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Agregar un grupo de trabajo.	Mensaje de registro correcto	✓
Agregar un grupo de trabajo existente.	Mensaje de error	✓
Obtener una lista de los usuarios	Muestra el resultado de la búsqueda	✓
Dar de alta a usuario en un grupo de trabajo.	Mensaje registro insertado correctamente.	✓
Dar de alta a un usuario no registrado en grupo de trabajo	Mensaje de error.	✓

3. **Eliminar registro de usuarios**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Búsqueda del usuario.	Muestra el listado de usuarios	✓
Pregunta si desea eliminar el registro de usuario	Mensaje de confirmación	✓
Aceptar el mensaje de confirmación de eliminar registro de usuarios	Mensaje de eliminación correcta	✓
Eliminar registro de usuario no válido o inexistente.	Mensaje de error.	✓

4. **Consulta datos usuarios**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Búsqueda del usuario.	Mensaje de resultado de búsqueda	✓
Vista de la lista de usuarios registrados	Se muestra la lista de usuarios registrados	✓

Caso de uso Perfil de usuario.

1. **Perfil**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Introducir el id de usuario y contraseña correctos.	Valida la entrada del usuario y entra a su perfil, muestra el menú con sus opciones	✓
Inicia la sesión de usuario	Mantiene la sesión de usuario	✓
Introducir el id usuario y contraseña de un usuario que ya inicio sesión.	Mensaje de error no permite inicar sesión duplicada	✓
Introducir el id de usuario y contraseña incorrectos.	Mensaje de error de usuario no válido.	✓

2. **Consulta datos personales**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Búsqueda del usuario.	El usuario debe estar registrado e iniciado sesión	✓
Ver los datos personales del usuario registrado	Se muestran los datos del usuario registrado	✓

3. **Modificar datos usuario.**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Se solicitan los datos del usuario y la posibilidad de modificarlos.	Muestra los datos del usuario para actualizarlos.	✓
Actualiza los datos	Mensaje de datos actualizados correctamente.	✓
Dejar vacio un campo al actualizar los datos.	Mensaje de error no debe de haber campos vacios.	✓

4. **Salir de Sesión**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
El usuario debe tener una sesión activa.	Ventana activa de su menú	✓
Término de sesión	Manda mensaje de confirmación de cierre de sesión	✓
Término de sesión sin estar en sesión activa.	Manda mensaje de no ha iniciado sesión.	✓
Confirmar salir de la sesión	Mensaje de confirmación de salir de sesión.	✓
Salir de sesión.	Finaliza sesión. Muestra la página de Bienvenida para iniciar sesión.	✓

Caso de uso Administrador de Documentos.

1. **Subir documentos**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Se solicita la página con el formulario para buscar el archivo a subir.	Muestra la página donde se encuentra el explorador para cargar el archivo.	✓
Se selecciona el archivo a subir, con tipo y tamaño permitidos.	Mensaje de envio correcto.	✓
Se selecciona el archivo a subir, con tipo y tamaño no permitidos.	Mensaje de error de archivo incorrecto.	✓

2. **Permisos de acceso**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Antes de subir el documento se muestra la opción de donde guardar el documento.	Muestra la página donde se presenta las opciones.	✓
Se selecciona una de las siguientes opciones: para guardar en carpeta pública, personal o de grupo de trabajo.	Manda a la opción requerida.	✓
Se guarda el documento en donde se seleccionó.	Muestra que se guardo el documento.	✓

3. **Mostrar documento**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Se solicita ver la lista de documentos personales.	Muestra la página donde se presenta la lista de documentos.	✓
Se selecciona si se quiere ver la carpeta pública o la del grupo de trabajo.	Manda a la opción requerida.	✓
Se selecciona el archivo de la lista.	Muestra el archivo.	✓
Si el usuario es visitante solicita la vista de documentos públicos.	Muestra la página de lista pública de documentos.	✓

4. **Borrar documento**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Se solicita ver la lista de documentos personales.	Muestra la página donde se presenta la lista de documentos.	✓
Se selecciona el archivo que se quiere eliminar.	Manda mensaje de archivo eliminado.	✓
Elimina archivo no válido	Manda mensaje de error.	✓

Caso de uso Página Principal

1. **Mostrar Contenido**

DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO
Se solicita ver la página de Inicio.	Muestra la página de Bienvenida.	✓
Se selecciona del menú la opción deseada.	Manda a la opción requerida.	✓

Pruebas complementarias

Además de las pruebas del diseño se hicieron otras pruebas en cuanto al manejo de la aplicación en diferentes exploradores, se verificó que tanto en Internet Explorer como en Netscape, la aplicación se vio correctamente.

También se verificó las resoluciones de pantalla de manera que se viera la aplicación correctamente. La resolución ideal para verse es de 800 x 600 píxeles, pero también se ve con una resolución de 1024 x 768 píxeles. Esto es importante ya que se tenía que verificar para poder tener una presentación adecuada de la aplicación.

Para la elaboración de las páginas HTML y JSP se utilizó un editor llamado Dreamweaver MX, con este editor se podía ver una vista preliminar de las páginas, lo que fue de gran ayuda por si se tenía que modificar o ajustar alguna de las páginas.

5.2 Puesta en marcha de la aplicación del Administrador de Documentos

Requisitos Software.

El sistema se apoya en una base de datos para los registros de usuarios, será necesario en primer lugar tener instalado un Sistema Gestor. Por las características del *software* desarrollado no se impone ninguna restricción en cuanto al Sistema a utilizar, sino que este dependerá de las prestaciones que el cliente quiera obtener. En este caso se decidió que el gestor de Base de datos fuera *Microsoft Access* por ser de fácil manejo. Pero si se desea se podría cambiar a otro tipo de gestor de base de datos.

Para el funcionamiento de la aplicación de gestión únicamente se necesita tener instalada la máquina virtual de Java en la versión 1.2.1.

En cuanto a la aplicación *web* se requiere la instalación y configuración del servidor *Tomcat* tal y como se describe en el *Capítulo II* (Tecnología para el desarrollo de aplicaciones Web) junto con el JDK 1.2.2 o JDK 1.3.1. Para la visualización de las páginas desde un cliente *web* sólo es preciso tener instalado un navegador *web* (*Internet Explorer 5.0* y *Netscape 4.7* o superiores).

Requisitos Hardware.

Los requisitos *hardware* van a depender del *software* instalado, del número de usuarios potenciales de la aplicación y de las prestaciones que se quieran ofrecer.

Servidores: Se presupone que para ofrecer un buen servicio a todas las solicitudes, tanto el servidor *web* como el servidor de base de datos no deben estar muy sobrecargados, por lo que es recomendable que sean servidores dedicados.

El servidor *web Tomcat* requiere al menos 128 MB de memoria RAM aunque para un servidor que vaya a tener gran cantidad de accesos concurrentes es preferible disponer de más memoria ya que este es un factor primordial en cualquier servidor (más que la velocidad de CPU o el ancho de banda de red). En cuanto a capacidad de disco duro, señalar que el servidor *Tomcat* junto con la aplicación *web* necesitan aproximadamente 30 MB de espacio. Por último la velocidad de CPU recomendada es de 250 MHz.

El equipo recomendado para el servidor de base de datos dependerá de la base de datos que se vaya a instalar. Los requisitos del sistema para trabajar con *Access* son mínimos. Hay que tener en cuenta además el espacio que ocupa la propia base de datos, en la que se puede prever un crecimiento debido al mantenimiento de históricos de gran parte de los datos.

Esta aplicación, por estar escrita en el lenguaje de programación Java, no necesitará una cantidad de recursos considerable para ejecutarse de forma aceptable. La aplicación presenta un buen rendimiento en un equipo *Pentium II* con 128 MB de memoria RAM, aunque conviene tener en cuenta el crecimiento del espacio ocupado en disco a medida que se añadan registros de datos de usuarios.

Para la aplicación *web* por parte del usuario es suficiente con tener cualquier máquina que sea capaz de ejecutar un navegador *web*.

Procedimiento de instalación y configuración.

A continuación señalaremos los pasos que debe dar el usuario para instalar la aplicación en los ordenadores de explotación, suponiendo que se satisfacen los requerimientos *software* descritos anteriormente.

Instalación de la Base de Datos en el Servidor.

En cuanto a la base de datos se proporciona un archivo ***.mdb*** para *Access* con algunos datos iniciales que se necesitan para que esta pueda funcionar.

Base de Datos *Access*.

Se debe copiar en el disco duro el archivo *Registro.mdb* que se encuentra en el directorio */BaseDatos*. A continuación se debe crear un origen de datos **ODBC** que haga referencia al archivo. En el *Capítulo IV* (Diseño y Desarrollo de la aplicación para el Administrador de Documentos) se describen los pasos necesarios para la creación de un origen de datos **ODBC**.

Servidor web.

Una vez instalado el servidor web, en este caso es Jakarta Tomcat, se debe copiar en el directorio *JakartaTomcat/webapps* la estructura del directorio */Proyecto*.

Para ver la página funcionando, se debe arrancar el servidor *Tomcat* y a continuación escribir en el URL del navegador:

http://dirección_del_equipo:8080/Proyecto/index.html

En este momento, se debería ver la página de Bienvenida de la aplicación. Si no es así, se debe revisar que se han incluido todos los elementos en el directorio correcto.

Por último señalar que en el directorio */upload/* se encuentran las carpetas de los archivos de usuarios con los documentos que subieron desde la página web por Internet/Intranet.

5.3 Recorrido del Sistema

La aplicación web permite dependiendo del perfil de usuario, registrar usuarios, eliminar registros de usuario, agregar grupos de trabajo, permitir agregar usuarios a un grupo de trabajo, subir documentos, eliminar documentos, mostrar documentos, consulta de datos, actualizar datos personales y consulta de usuarios registrados.
Para acceder a todas estas funcionalidades será necesario introducir la dirección de la página en el navegador que se vaya a utilizar:

http://localhost:8080/Proyecto/index.html

Donde *localhost:8080* es la dirección IP donde esta el servidor web instalado.

De esta manera podremos ver la página principal que muestra la siguiente figura:

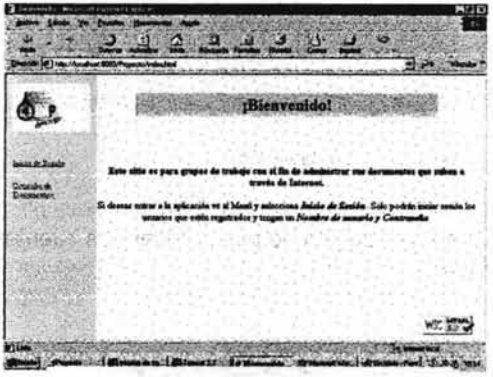


Figura 1. Página de inicio

Para permitir un fácil acceso a todas las opciones que ofrece la página, se tiene un *frame* con un Menú donde podremos acceder de forma rápida a todas ellas.

Inicio de Sesión.

Cuando el usuario ya esta registrado en la aplicación y desea iniciar sesión de usuario, puede acceder por medio de la liga del menú *Inicio de sesión.* La liga muestra la siguiente página:

Figura 2. Inicio de sesión

Si el usuario ya inicio sesión en otra página no puede tener dos cuentas de sesión activas manda mensaje de error.

Figura 3. Sesión ya iniciada

Si el usuario inserta el id de usuario y contraseña incorrectos para iniciar la sesión manda mensaje de error.

Figura 4. Falla en inicio de sesión

Ver documentos públicos

Si el usuario es visitante sólo podrá ver los documentos que los usuarios registrados han guardado en la carpeta pública.

Del menú se elige la liga que dice *Consulta de documentos* la siguiente figura presenta la página donde muestra la lista de documentos públicos.

Figura 5. Documentos Públicos

Perfiles de usuario

Una vez que ha iniciado sesión el usuario, dependiendo de su perfil se le presentará su página con el menú correspondiente, existen tres tipos de perfil:

- Líder de Proyecto
- Coordinador
- Usuario registrado

Figura 6. Página Líder de Proyecto

Figura 7. Página Coordinador

Figura 8. Página Usuario

Cada perfil tiene características diferentes, y describiremos cada uno de estos.

Perfil: Líder de Proyecto
Dentro del menú del Líder de Proyecto se encuentran las siguientes acciones:

- Registro de usuarios. Esta sección permite al líder de proyecto registrar un usuario a la aplicación. La página presenta campos para ser llenados como: Nombre, Apellidos, Dirección, Correo electrónico, El id de usuario, Contraseña y el Perfil de usuario. Todos los campos son obligatorios.

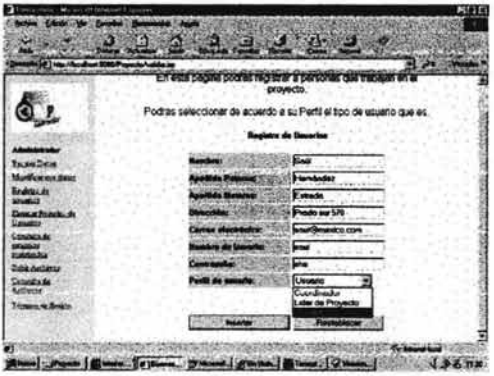


Figura 9. Registro de usuarios

Si deja en blanco alguno de los campos mencionados aparece mensaje de error, y si llena correctamente los datos aparece mensaje de Registro insertado correctamente.

Figura 10. Mensaje de error al llenar los campos

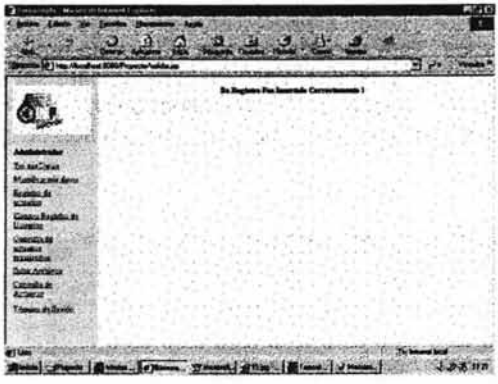


Figura 11. Mensaje de registro insertado

- Eliminar registro de usuario. En esta página selecciona el usuario a eliminar .Manda mensaje si desea realmente eliminar al usuario y posteriormente manda mensaje de registro eliminado.

Figura 12. Eliminar usuario

Figura 13. Registro eliminado

- Consultar usuarios registrados. Muestra una lista con los usuarios registrados.

Figura 14. Lista de usuarios registrados

Perfil : Coordinador

Tiene las siguientes características:

- Registro de grupo de trabajo. En esta sección se podrá registrar el nombre de un grupo de trabajo. Si agrega un grupo de trabajo existente manda mensaje de error.

Figura 15. Registro grupo de usuario

- Agregar un usuario a un grupo de trabajo. Se selecciona de la lista un usuario y se registra en un grupo de trabajo existente.

Figura 16. Agregar usuario a un grupo de trabajo

- Registro usuarios. Esta sección es igual que la que se ocupa en el perfil del Líder de Proyecto. Puede dar de alta a un usuario llenando los campos de registro.

Perfil : Usuario
Las funciones que tiene el perfil de Usuario también las tienen el perfil Líder de Proyecto y Coordinador.

Las funciones son las siguientes:

- Ver datos personales. En esta página se puede consultar los datos del usuario.

Figura 17. Consulta datos personales

- Modificar datos personales. En esta página el usuario puede actualizar los datos personales. Posteriormente se mostrará una página con los datos actualizados.

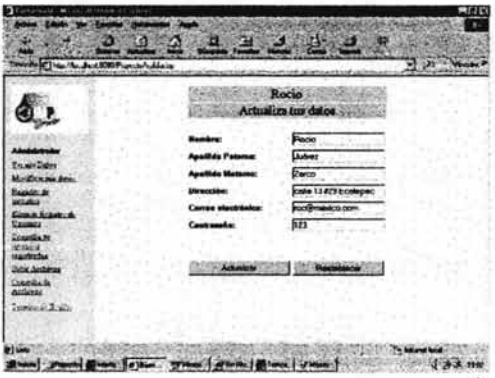


Figura 18. Modificar datos

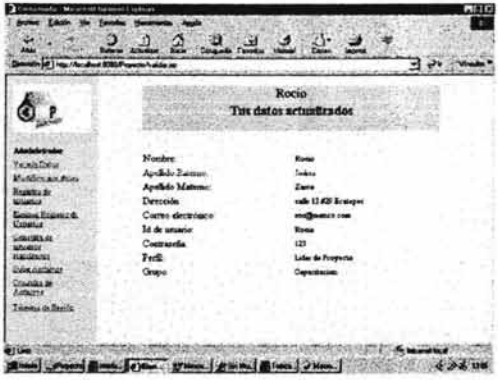


Figura 19. Datos actualizados

- Subir documentos .En esta sección se muestran las opciones que tiene el usuario para guardar los documentos que suba a la aplicación. Puede subir documentos para que se muestren en la carpeta pública, para que vean los usuarios que están dentro del mismo grupo de trabajo o para que sea de uso personal. Dependiendo de la elección es donde se guardará el archivo. Si el archivo se envío correctamente aparece una pantalla donde se muestra el nombre del archivo.

Figura 20. Opciones para
los documentos

Figura 21. Formulario para subir documentos

Figura 22. Mensaje de archivo enviado

- Consulta de archivos. En esta página se muestra la lista de los documentos que existen. Pueden consultar los documentos de la carpeta personal, la carpeta pública y la carpeta del grupo de trabajo a la que pertenece el usuario. Sólo los archivos de la carpeta personal pueden ser eliminados de la lista.

Figura 23. Lista de documentos públicos

Figura 24. Lista de documentos públicos

Figura 25. Lista de documentos
de grupo de trabajo

Salir de Sesión.

Cuando el usuario desea salir de la aplicación, se dirige hacia la liga de *Término de Sesión*, aparecerá una página de confirmación de cierre de sesión. Posteriormente regresa a la página de Bienvenida.

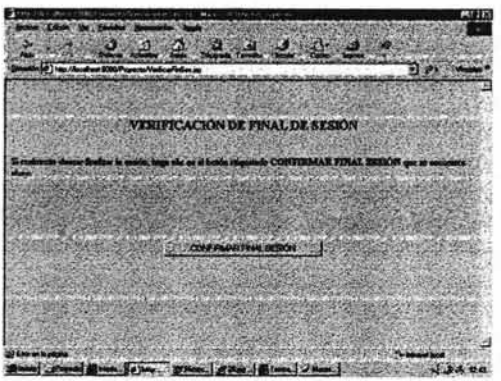


Figura 26. Salir de sesión de
usuario

Teniendo en cuenta las pruebas del sistema desarrollado se vio que cumplían con lo establecido en los contratos de los casos de uso realizados para el sistema.

Posteriormente se trataron los requisitos para la puesta en marcha del Administrador de Documentos a modo de que se sigan las indicaciones para el manejo de la aplicación. Una vez concluida la instalación del sistema se dio un recorrido por todas las páginas de la aplicación para ver su contenido y sus funciones.

Conclusiones

En este trabajo de Tesis se vio la manera de realizar una aplicación web acerca del uso de un ***Administrador de Documentos de Proyectos Vía Internet/Intranet***.

Uno de los objetivos era el investigar sobre diferentes herramientas y técnicas para la elaboración de sitios web para la construcción de la aplicación. Se consultaron varios libros y páginas en Internet para ver como se utilizaban dichas herramientas, se vieron varios ejemplos sobre el uso y manejo de aplicaciones que utilizaban Internet para manejar datos de forma dinámica con acceso a base de datos, y así tener una idea de cómo desarrollar la aplicación del Administrador de Documentos.

Se conoció las características que tiene Internet para el envío y consulta de datos, además de su estructura y de los diferentes servicios que proporciona esta red a todo el mundo. Uno de los servicios de Internet importante es el protocolo de transferencia de hipertexto (http). El **HTTP** puede leer no sólo texto, sino también imágenes, sonidos o secuencias de vídeo. Lo cual es importante para la aplicación del Administrador de Documentos ya que se consideró la manera de cómo trabaja Internet así como su funcionamiento.

Durante la creación del proyecto se decidió utilizar el paradigma orientado a objetos, con el fin de obtener un sistema estable, reutilizable y tolerante al cambio.

En este sentido se decidió utilizar el *Proceso Unificado de Rational* (RUP, *Rational Unified Process*) como metodología durante el desarrollo, que proporciona una serie de modelos que se basan en los conceptos de objeto y clase y en las relaciones entre ellos, y propone **UML** como la notación común para estos modelos.

El objetivo de esta metodología es permitir la producción de un *software* de la mayor calidad que satisfaga las necesidades de los usuarios finales, dentro de planificaciones y presupuestos predecibles.

Se trata de un proceso de desarrollo iterativo e incremental, en el sentido de que el *software* no es liberado de una sola vez al final del proyecto, sino que es desarrollado por partes.

La construcción del sistema consta de muchas iteraciones, en cada una de las cuales se construye *software* de calidad, probado e integrado que satisface un subconjunto de los requerimientos del proyecto. Cada iteración contiene todas las fases del ciclo de vida de análisis, diseño, implementación y prueba.

En la aplicación se utilizaron herramientas para la construcción de un sitio en el que se manejaran datos dinámicamente a través de Internet con lo cual era necesario tener un lenguaje que fuera lo suficientemente eficaz para la construcción del sistema.

El lenguaje de programación que se utilizó fue *Java* que es un lenguaje orientado a objetos y clases. Una clase es agrupación de variables (datos) y de los métodos (funciones) que manipulan esas variables. Con esto hace posible la reutilización de código para futuras aplicaciones o modificaciones al sistema. Java es un lenguaje que es portable lo que permite que se utilice en cualquier plataforma o sistema operativo, además es de libre distribución.

Los componentes de *Java* que se utilizaron fueron las páginas *JSP* (JavaServer Pages) que es una tecnología para crear contenido dinámico a través de la web. Además se utilizó el lenguaje de hipertexto *HTML*, que fue el encargado junto con las páginas JSP de dar la vista a las páginas realizadas para el sistema.

Para que la aplicación manejara información actualizada a través de Internet, el sistema requería conectarse a una base de datos. Se opto por utilizar Microsoft Access, por ser un gestor de base de datos fácil de usar, como el sistema esta escrito en Java, se puede cambiar de gestor de base de datos de una manera muy simple sin tener que volver a estructurar la aplicación, lo cual es de gran beneficio para el proyecto por que sólo se requiere cambiar dependiendo de la base de datos el controlador adecuado.

Para la conexión a la base de datos se utilizó una *API* (Interfaz para Programas de Aplicación) de Java llamado *JDBC* que se encarga de hacer las conexiones para la aplicación. Esta API junto con el origen de datos *ODBC* hacen un puente para establecer la conexión con la base de datos Microsoft Access que se utiliza en el sistema.

Para la arquitectura del sistema se utilizó una arquitectura multicapa llamada *Modelo-Vista-Controlador*. El Modelo contiene el control de la funcionalidad de la aplicación. La Vista proporciona la presentación del Modelo, representando la apariencia de la aplicación. El Controlador es quien reacciona a las acciones del usuario y el encargado de crear y asignar valores al Modelo para su funcionamiento.

La arquitectura multicapa permite separar el contenido *HTML* estático con el contenido de una página *JSP* y con el servidor de base de datos, para acceder a la información que almacena. Es de gran ayuda este tipo de arquitectura por que se tiene un mejor manejo y control de los datos a la hora de estar programando y de presentar la vista del sistema.

A nivel personal esta Tesis me dejó una experiencia muy grande ya que conocí herramientas que son de gran utilidad para la construcción de sitios con contenido dinámico como es *Java* y *JavaServer Pages*, también aprendí como se diseña un sistema con *UML* que es un lenguaje de modelado unificado, para el diseño de sistemas orientado a objetos.

Se realizó una ardua tarea en cuanto a la programación del sistema ya que no contaba con suficiente experiencia en los lenguajes utilizados para la Tesis, lo que me satisface por que aprendí mucho sobre las tecnologías que se utilizaron, y seguiré aprendiendo, ya que siempre hay actualizaciones de estos lenguajes.

El objetivo principal de la Tesis, es que se presenta como una herramienta para la simplificación de las actividades que tiene un grupo de trabajo, para el manejo e intercambio de información a través de Internet.

Bibliografía

- FROUFE, Quintas Agustín. Java Servlet Pages Manual de usuario y tutorial. Ed. Alfaomega. Madrid España, 2002.

- HALL, Marty. Servlets y JavaServer Pages. Guía Práctica. Ed. Prentice Hall. Madrid, 2000. pp. 580.

- LARMAN, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Ed. Prentice Hall. México, 1999. pp. 536.

- TREMBLETT, Paul. Superutilidades para JavaServer Pages. Ed. McGraw-Hill. México, 2002. pp. 525.

- WEISS, Mark Allen. Estructuras de datos en Java^{tm} .Compatible con Java^{tm} 2. Ed. Addison Wesley. Madrid, 2000. pp. 776.

Páginas de Internet

- Administrador de Documentos. URL: http://www.bcdasociados.com

- Análisis y diseño orientado a objetos URL: http://www.itlalaguna.edu.mx/academico/carreras/sistemas/Analisis%20y%20dise%F1o%20orientado%20a%20objetos/Indice.htm

- Apache Tomcat. URL: http://jakarta.apache.org/tomcat/

- Configuración de Tomcat. URL: http://usuarios.lycos.es/froufe/parteJ1/capa-1.html

- Cytec, Gestión de documentos URL: www.cyt.net/wwwroot/knowledgemanagement/gestiondocumentos.htm

- Estándar HTML URL: http://www.maestrosdelweb.com/editorial/estandar/

- Extra uploader URL: http://www.javer.narod.ru/fuplinst.htm

- Historia de Java URL: http://www.iespana.es/marcoescom/oldest/sexto/poo2/pooder00.htm

- HTML URL: http://www.webestilo.com/html/cap1a.phtml

Internet y World Wide Web. URL:
http://www.iteso.mx/biblio/formacion/internet.htm

Instituto Mexicano del Petróleo URL: http://www.imp.mx/imp/historia/

Instituto Mexicano del Petróleo URL:
http://intranet.imp.mx/apoyo/calidad/sic/ap/

Instituto Mexicano del Petróleo URL:
http://intranet.imp.mx/siiimp/AdmnProy.htm

Introducción a UML. URL: http://www.programación.com/tutorial/uml/

Java y conexión a Base de Datos URL:
http://www.linti.unlp.edu.ar/catedras/Laboratorio/Teorias/clase12.pdf

Localizador Universal de Recursos URL:
http://www.psicobyte.com/html/taller/url.html

Manejo de Documentos URL:
http://www.sistemasdeoficina.com/impdoc.htm

Pool de objetos en Java URL:
http://people.javahispano.org/vitxo/articles/howto-pool.html

Subir archivos URL: http://www.oop-reserch.com/

Servlets URL: http://www.aulambra.com/ver.asp?id=111

Tutorial de UML URL:
http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html

Tutorial Java URL:
http://www1.ceit.es/Asignaturas/Informat2/Clases/Clases9899/Clase01/JavaIntro/tsld002.htm

UML URL: http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html