UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE MONITOREO, APROBACIÓN Y REVISIÓN DE PROYECTOS DE SOFTWARE POR INTERNET.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

NANCY CANDELARIA CORTÉS GARCÍA

DIRECTORA DE TESIS:
DRA. ANA MARIA VÁZQUEZ VARGAS.

MÉXICO, D.F. SEPTIEMBRE 2003

Agradecimientos

Dra. Ana María Vázquez Vargas
Por su tiempo, su paciencia y su apoyo en la elaboración de esta tesis.

Ing. Jorge Alberto Rodríguez Campos.
Por su amistad, su cariño, su apoyo, su ayuda infinita y por todos los consejos que siempre me has dado. Gracias jorgito.

Marcelo, Mario, Angeles, Daniel, Javier, Ricardo, Juanito, Francely
Por ser mis hermanos, por crecer conmigo, por apoyarme, por aguantarme y por ser para mi el mayor tesoro que tengo en la vida. Gracias por ser mis hermanos

Karina, Gaby, Marcela, Juan Manuel, Héctor, Gerardo, Cesar, Jorge, Saúl, Alejandra, Juan Carlos, Samuel, Cinthya, Remedios, Manuel Saldaña, Marisol, Oliver, Aide, Agustín, Yolanda Rubio, Francisco Rosas, David Rodríguez.

Por todo los que aprendí de ustedes y con ustedes, por su cariño, por su paciencia, por compartir conmigo muchos buenos y malos momentos, sobre todo por ser mis amigos y mi familia.

Dedicatorias.

Este último esfuerzo por terminar un sueño es para ustedes, por luchar siempre a mi lado, por apoyarme, por su cariño, por sus desvelos, y por estar siempre conmigo.

Porque su amor ha hecho que logre lo que me propongo y porque al cumplir este sueño no solo cumplo un sueño mío, también cumplo un sueño de ustedes, los que siempre han buscado mi felicidad y le han dado mucha a mi vida. Gracias por su esfuerzo por eso este trabajo es por ustedes.

PAPÁ Y MAMÁ.

Por el inmenso cariño que me han dado, por su apoyo, por soñar para mi los más grandes sueños y por estar allí para poco a poco verlos hacerse realidad.

Por todo su amor, por sus desvelos tantas y tantas noches, por aprender a caminar conmigo, por su guía en todo momento y por hacer de mi la persona que soy. Gracias por ser mis padres.

PAPÁ JUAN Y TIA ESTHELA.

Por estar conmigo siempre que los he necesitado en los momentos felices y en los no tanto, por su cariño, por su apoyo, por sus consejos y por su tiempo y porque tengo la dicha de tener dos papás y dos mamás . Gracias por ser mis padres.

HUGO.

Por que al llegar a mi vida le diste la más bella ilusión.
Por ser tu la persona que tanto espere. la indicada para llenar mi vida de ternura y amor
Por luchar y lograr tus sueños y por apoyarme para lograr los míos
Por lo orgullosa que estoy por estar a tu lado. Gracias Pepe.

TEM !!!

Capítulo V. PRESENTACIÓN DEL SISTEMA.

INTRODUCCIÓN.

INTRODUCCIÓN.

En la vida cotidiana, siempre se tienen tareas que realizar, para ello se necesita tiempo y recursos ya sea económicos, tecnológicos o humanos.

Los seres humanos somos una sociedad y como tal se necesita trabajar en equipo para lograr que tareas muy complicadas sean más rápidas de resolver. Trabajar en equipo no es una tarea sencilla, se requiere de la disposición de las personas para trabajar en conjunto, pero además se necesita que el grupo de personas sea supervisado por alguien con conocimientos y que tenga la capacidad de asignar tareas a las personas más capacitadas, es decir, deberá ser un líder con la capacidad de identificar el potencial de cada una de las personas relacionadas con la actividad.

Planear la actividad y asignar responsabilidades no será toda su tarea, deberá gestionar también el tiempo que tardaran todas las actividades en conjunto para obtener un resultado, los recursos tecnológicos y económicos que se emplearan en dicha actividad.

Gestionar las actividades diarias, es una tarea difícil, pero gestionar a un equipo de trabajo lo es más. Gestionar es una necesidad para personas que buscan obtener buenos resultados en las actividades que desempeñan.

De igual manera que en la vida cotidiana, la gestión es una tarea indispensable en la creación de proyectos de software.

Crear sistemas de software complejos, como lo son hoy en dia los programas informáticos necesitados por las empresas, no es algo sencillo, se necesita personal capacitado para realizar las tareas de análisis de requerimientos, aplicación de métodos y medidas de calidad, programación, documentación, detección de errores e instalación de productos terminados, capacitación a usuarios, todas estas actividades deben de ser planeadas antes de comenzar cualquier desarrollo de software y está en manos de un gestor de proyectos las planeaciones de tiempo, recursos necesarios, asignación de tareas, costos entre muchas cosas más.

La tarea de gestión de proyectos de software como se ha mencionado no es una tarea sencilla, por ello los gestores de proyectos han buscado crear herramientas de software para hacer de esta tarea tan complicada algo más sencillo de llevar acabo.

La gestión de proyectos de software no es una tarea aislada, el gestor del proyecto no puede llevarla a cabo solo, necesita que el equipo de trabajo este comunicado, que sea informado de modificaciones de requerimientos, que cada responsable de alguna actividad entregue reportes de avances, que la experiencia de cada participante sea utilizada para beneficiar el desarrollo, es decir necesita la participación activa del equipo de trabajo.

La gestión consiste en coordinar la tareas asignadas a diferentes responsables, pero para coordinar a un equipo de trabajo, hay algo que es indispensable, que no puede faltar en

ningún momento y que gracias a esto se podrán obtener resultados de calidad, esto es la comunicación entre el equipo de trabajo.

Actualmente es una necesidad crear herramientas computarizadas que faciliten la gestión de proyectos y que coordinen las actividades de grupos de trabajo manteniendo la comunicación entre los miembros del equipo dando como resultado desarrollos de calidad y entregas en tiempos planeados.

Es por está razón que han nacido conceptos como el de Groupware o grupo de trabajo el cual es una respuesta a la búsqueda de espacios virtuales de colaboración y de comunicación entre miembros de un equipo de trabajo.

Esta tesis representa un esfuerzo por desarrollar un espacio virtual colaborativo en el cual sea posible coordinar a los grupos de trabajo, asignarles responsabilidades en los desarrollos de software manteniendo flujos de información entre los usuarios.

Se ha realizado este espacio colaborativo sobre Internet para librar la barrera de los espacios geográficos y permitir la coordinación de equipos de trabajo desde cualquier lugar y en cualquier momento. Los participantes podrán consultar y agregar avances cuando y en donde lo requieran.

El plan de la tesis es el siguiente:

Debido a que el sistema que se ha desarrollado se implementó en Internet y utilizando la programación orientada a objetos en el capítulo I se hace una breve descripción de estos dos conceptos. Además se explican los conceptos fundamentales de la gestión de proyectos como son: coordinación y comunicación, planificación y seguimiento de proyectos de software y el concepto de Groupware.

En el capítulo II se describe el problema a resolver y la solución propuesta, además de mencionar brevemente los recursos tecnológicos empleados para solucionar el problema.

En el capítulo III se describen las herramientas tecnológicas utilizadas para solucionar el problema justificando el uso de cada una de ellas.

En el capítulo IV se describe el diseño de la base de datos para almacenar proyectos, miembros de equipo, objetos, diagramas de ruta y se explica el proceso seguido para obtenerla.

En el capítulo V se presenta el funcionamiento del sistema Manejo de Actividades, así como la manera de cómo se crean los elementos del mismo como son los equipos, las rutas y los objetos.

En el capítulo VI se presentan las conclusiones.

Este trabajo busca satisfacer tres importantes necesidades de los actuales desarrolladores de software, busca fortalecer la comunicación entre los equipos, mejorar la coordinación de éstos y sobre todo busca ser un instrumento de colaboración.

Debido al cambio de Administración en la Dirección General de Cómputo Académico, muchos de los proyectos desarrollados en el Área de Aplicaciones Avanzadas se encuentran en estos momentos sin operar, entre ellos el desarrollo que corresponde a esta tesis.

Aunque la nueva administración sabe la utilidad e importancia del desarrollo y uso de este tipo de proyectos se esta tomando un tiempo para volverlos a poner a funcionar.

México, D.F, Septiembre 2003

CAPÍTULO I
ANTECEDENTES

PAGINACIÓN DISCONTINUA

Capítulo I ANTECEDENTES.

1.1 Internet: orígenes y servicios.

Historia de Internet

Desde los tiempos más remotos el ser humano ha buscado la mejor forma de comunicarse con otros de su misma especie, aún cuando éstos se encuentran en lugares lejanos. La historia de la comunicación está marcada por los adelantos tecnológicos de cada época y lugar.

Con el inicio de la era tecnológica, se dispuso de un medio con el cual fue posible establecer una comunicación a distancia y casi instantánea por medio de códigos y claves de sonido: el telégrafo; posteriormente la comunicación humana se vio beneficiada con la invención del teléfono permitiendo el uso de la voz, más adelante vino la radio, la televisión y con ellas las computadoras. Estos grandes inventos son la base de los adelantos tecnológicos de que disfrutamos hoy en día en cuanto a comunicación , desde el envío y recepción de un fax hasta la comunicación instantánea en cualquier lugar del mundo por medio de Internet.

Internet [1] es hoy en día una infraestructura informática extendida ampliamente, su influencia alcanza no solo al campo de la tecnología de las comunicaciones entre computadoras (redes) sino también a toda la sociedad en la medida en que su empleo se incrementa cada vez más para llevar acabo procesos como el comercio electrónico, la adquisición de información y la interacción entre comunidad o comunidades remotas.

Para llegar a los niveles de comunicación que hoy se logran gracias a Internet, se han dedicado años de investigación y perfeccionamiento del tipo de transmisión de datos.

En 1969 la Advanced Research Projects Agency (ARPA) del Pentágono creó la primera red llamada ARPANET, la cual constaba solo de cuatro computadoras conectadas, una en la Universidad de California en los Angeles (UCLA), otra en el Instituto de Investigaciones de Stanford (SRI), una mas en la Universidad de California en Santa Barbara (UCSB) y la última en Universidad de UTHA. Para el año de 1971, ya se contaba con 11 nodos más y en el año siguiente ya había un total de 40. En ese año se tiene registrado el primer mensaje enviado y recibido por correo electrónico de Ray Tomlinson, pero fue hasta el segundo mensaje de prueba cuando se estableció que todos los mensajes que se enviaran deberían emplear el signo @ .

En 1974 los investigadores Vint Cerf y Robert Kahn, redactaron un documento titulado A Protocol for Packet Network Internetworking, donde explicaban cómo podría resolverse el problema de comunicación entre los diferentes tipos de computadoras, dichos estudios fueron aplicados 8 años después, creándose de está forma la Transmisión Control Protocol- Internet Protocol (TCP-IP, protocolo de control de transmisión/protocolo de Internet), este protocolo fue adaptado de inmediato como estándar por el Departamento de Defensa de los Estados Unidos, quien en este mismo año se separó de ARPANET y creó una red propia llamada

MILNET. Asimismo, surgieron nuevos organismos que le dieron el término Internet, tal y como ahora se le conoce mundialmente.

El protocolo TCP/IP es un sistema de comunicación muy sólido y robusto bajo el cual se integran todas las redes que conforman Internet; durante su desarrollo se incrementó notablemente el número de redes locales de agencias gubernamentales y de universidades que participaban en el proyecto, dando de esta manera, origen a la red de redes más grande del mundo.

Las funciones militares de un principio se separaron y se permitió el acceso a la red a todo aquel que lo requiriera, sin importar de qué país proviniera, siempre y cuando fuera para fines académicos o de investigación, por tal razón Internet tuvo su etapa de desarrollo dentro de las Universidades. Hasta este momento la velocidad de transferencia entre nodos, era de 56 kilobits por segundo.

La red que dió origen a la red de redes, ARPANET dejó de funcionar en 1990, pero ya existían varios organismos encargados de Internet, en Europa existía CERN(European High-Energy Particles Physics Lab), dicho organismo dos años más tarde crearía al hoy conocido World Wide Web (WWW), para lo que empleó tres recursos: HTML (Hypertext Markup Language), HTTP (Hypertext Transfer Protocol) y un programa cliente llamado Web Browser.

Internet como ahora lo conocemos encierra una idea técnica clave, la arquitectura abierta en trabajo de red , así como múltiples redes interdependientes, de diseño casi arbitrario. En una red de arquitectura abierta, las redes individuales pueden ser diseñadas y desarrolladas separadamente, donde cada una puede tener su propia y única interfaz. Cada red puede ser diseñada de acuerdo con su entorno específico y los requerimientos de los usuarios, no existen restricciones en los tipos de red que pueden ser incorporadas ni tampoco en su ámbito geográfico.

En 1993 se funda Netscape, compañía que lanza al mercado un navegador con el cual Internet pasa de una fase escrita a una gráfica, lo que ayudó a popularizar está tecnología.
Más adelante surgieron otros navegadores en el mercado como Explorer de Microsoft. A partir de entonces el crecimiento de Internet ha sido impresionante, en enero de 1993 tan solo había 100 sitios WWW, para enero de 1996 ya existían 90 mil. Todo este crecimiento ha sido propiciado por los fines comerciales que persiguen la mayoría de las empresas que lo forman, de esta manera entramos a la nueva era comercial de Internet.

Internet en México

En lo que respecta a México, la historia de Internet comienza a finales de la década de los 80's. En el año de 1987, el Instituto Tecnológico y de Estudios Superiores de Monterrey, en el campus Monterrey (ITESEM) se conectó a BITNET a través de líneas conmutadas por medio de una línea privada analógica de 4 hilos a 9600 bits por segundo, en 1989 lo hizo a Internet al enlazarse por medio de la universidad de Texas en San Antonio (UTSA), por la misma línea privada. La Universidad Nacional Autónoma de México accedió a Internet por medio de una conexión vía satelital de 56 Kbps con el Centro Nacional de Investigación Atmosférica de Boulder, Colorado, siendo este el segundo nodo de Internet en México. Después se interconectaron ambas universidades mexicanas usando líneas privadas analógicas de 9600 bps, velocidad suficiente para proveer correo electrónico, transferencia de archivos y acceso remoto.

Poco a poco se fueron incorporando a Internet otras instituciones educativas mexicanas como son: Universidad de CHAPINGO en el Estado de México, el Centro de Investigaciones de Química Aplicada de Saltillo, el Laboratorio Nacional de Informática Avanzada de Jalapa, Veracruz, los cuales se conectaban al ITESEM para salir a Internet. Para este entonces, en México, ya existía un organismo civil, donde se discutían las políticas , estatutos y procedimientos que habrían de regir y dirigir el camino del control de la red de comunicación de datos de México. Tiempo más tarde, surgió otro organismo denominado MEXNET que reunía representantes legales de cada institución, el cual incluía a varias universidades de distintos lugares del país. Dicha organización, en 1992, establece una salida de 56 kbps al Backbone de Internet.

En 1993 el CONACYT se conecta a Internet mediante un enlace satelital al NCAR (Centro Nacional de Investigación Atmosférica) al igual que el ITAM , la UAM, en ese mismo año, se establece como el primer NAP (Network Access Point),al intercambiar tráfico entre dos diferentes redes. A finales de este año en México ya se contaba con distintas redes MEXNET, Red UNAM, Red ITESEM, RUTyC (desaparecida el mismo año). BAJANET, Red Total CONACyT y SIRACyT. Fue en 1994, con la fundación de la Red Tecnológica Nacional (RTN), integrada por MEXNET y CONACyT, que se generó un enlace a 2 MBps.

En el mismo año, Internet se abre en el ámbito comercial en México con lo cual se inicia una nueva era de desarrollo para nuestro país que beneficia a todas las personas, empresas o instituciones que deciden participar en el proyecto desde sus inicios, ya que hasta entonces solo instituciones educativas y de investigación tenían acceso a la super carretera de la información.

A finales de 1995 se crea el Centro de Información de Redes de México (NCI-México) el cual se encargó de la coordinación y administración de los recursos de Internet asignados al país, como son la administración y delegación de los nombres de dominio bajo ".mx". En 1996, se registran cerca de 17 enlaces contratados con TELMEX para uso privado, asimismo se consolidan los principales ISP (proveedores de servicios de Internet) en el país de los casi ya 100 ubicados a lo largo y ancho del territorio nacional. Para el año 1997 existen más de 150 ISP's , ubicados en los principales centros urbanos: Ciudad de México, Guadalajara, Monterrey, Chihuahua, Tijuana, Puebla, Laredo, Saltillo, Oaxaca, entre otros.

Actualmente, Internet es utilizado por instituciones educativas y gubernamentales, empresas privadas y personas de todo el mundo, entre quienes se llevan acabo intercambios constantes de información dando origen a la llamada globalización de la comunicación. Hasta el dia de hoy gracias a Internet, se puede recibir información al instante en cualquier parte del mundo, agilizando y facilitando de esta forma el proceso de comunicación a distancia.

World Wide Web.

La World Wide Web, o simplemente Web, es el sistema de información más completo y actual, que une tanto elementos multimedia como hipertexto. De hecho, tomando el todo por la parte, con mucha frecuencia la Web se utiliza como sinónimo de Internet.

El World Wide Web (WWW) es el resultado de cuatro ideas o factores:

1. La idea de Internet y los protocolos de transporte de información en que está basada.

2. La concepción de Ted Nelson de un sistema de hipertexto, extendida a la red.

3. La idea de programas cliente que interaccionan con programas servidores capaces de enviar la información en ellos almacenada. Para la Web, esto se hace mediante el protocolo HTTP(HyperText Transfer Protocol).

4. El concepto de lenguaje anotado (Markup language) y más en concreto del lenguaje HTML (HyperText Markup Language), herramienta fundamental de Internet. Gracias al hipertexto, desde una página Web se puede acceder a cualquier otra página Web almacenada en un servidor HTTP situado en cualquier parte del mundo. Todo este tipo de operaciones se hacen mediante un programa llamado browser o navegador, que básicamente es un programa que reconoce el lenguaje HTML, lo procesa y lo representa en pantalla con el formato más adecuado posible.

Protocolo HTTP

En 1991 se creó el protocolo llamado HTTP (HyperText Transport Protocol).

Una de las características del protocolo HTTP es que no es permanente, es decir, una vez que el servidor ha respondido a la petición del cliente la conexión se pierde y el servidor queda en espera, al contrario de lo que ocurre con los servicios de ftp o telnet, en los cuales la conexión es permanente hasta que el usuario o el servidor transmite la orden de desconexión. La conexión no permanente tiene la ventaja de que es más difícil que el servidor se colapse o sature, y el inconveniente de que no permite saber que es un mismo usuario el que está realizando diversas conexiones (esto complica la seguridad cuando los accesos se hacen con password, pues no se puede pedir el password cada vez que se realiza una conexión para pedir una nueva página, el password sólo se debería pedir la primera vez que un usuario se conecta).

La ventaja del protocolo HTTP es que se pueden crear recursos multimedia localmente,

transferirlos fácilmente a un servidor remoto y visionarlos desde donde se han enviado o desde cualquier otra computadora conectado a la red. El protocolo HTTP es una herramienta muy poderosa, que constituye la esencia del World Wide Web.

Lenguaje HTML

HTML es el lenguaje de presentación estándar utilizado en Internet. HTML son las siglas de Hypertext Markup Language, o lo que es lo mismo Lenguaje de Hipertextos a través de Marcas.

HTML fue el primer lenguaje de programación estándar utilizado para presentar páginas web en Internet. Se trata de un lenguaje interpretado, basado en la posición de etiquetas o "tags", que hacen referencia a un comando, como por ejemplo la escritura en negrita de una letra, palabra o frase. Como cualquier lenguaje de programación, HTML posee una serie de reglas que es necesario seguir con el objetivo de que un navegador intérprete los comandos que contienen las páginas web que se desea visualizar.

Los enlaces o links, es la posibilidad de enlazar un recurso o página web con otra y así sucesivamente, constituyendo una telaraña de dimensiones y posibilidades considerables, de ahí que el nombre con el que se le conoce a Internet sea WWW o World Wide Web, o lo que es lo mismo, telaraña extendida en el mundo, y es que es realidad, la red de redes, esta constituida por un sin fin de uniones entre servidores web en el mundo, que muchas veces nos obliga a perder el rumbo y no saber exactamente el lugar al que nos estamos conectando en un momento determinado.

1.2 Programación orientada a objetos

La programación orientada a objetos se hizo popular por ser capaz de dividir programas largos en unidades semi-autónomas. El lema de la programación orientada a objetos es: "divide y vencerás". En otras palabras un programa se puede dividir en partes fácilmente identificables.

La orientación a objetos promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

Un lenguaje orientado a objetos ataca estos problemas. Tiene tres características básicas: debe estar basado en objetos, basado en clases y capaz de tener herencia de clases. Muchos lenguajes cumplen uno o dos de estos puntos; muchos menos cumplen los tres. La barrera más difícil de sortear es usualmente la herencia.

El concepto de programación orientada a objetos (OOP) no es nuevo, lenguajes clásicos como SmallTalk se basan en ella. Dado que la OOP se basa en la idea natural de la existencia de un mundo lleno de objetos y que la resolución del problema se realiza en términos de objetos, un lenguaje se dice que está basado en objetos si soporta objetos como una característica fundamental del mismo.

El elemento fundamental de la OOP es, como su nombre lo indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización.

Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

1.2.1 Clases y objetos.

Clase

Una clase es un grupo de objetos con propiedades (atributos) similares, comportamiento común (operaciones), relaciones comunes entre objetos, y semántica común.

Objeto

Un objeto es algo real o abstracto acerca del cual almacenamos datos y métodos que manipulan dichos datos. Cada objeto es una instancia de la clase a la que pertenece.

Estructura de un objeto.

Un objeto puede considerarse como una especie de cápsula dividida en tres partes:

- Relaciones
- Propiedades
- Métodos

Cada uno de estos componentes desempeña un papel totalmente independiente:

Las **relaciones** permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.

Las **propiedades** distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.

Los **métodos** son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia.

Encapsulamiento y ocultación

Como hemos mencionado, cada objeto es una estructura compleja en cuyo interior hay datos y programas, todos ellos relacionados entre sí, como si estuvieran encerrados conjuntamente en una cápsula. Esta propiedad (encapsulamiento), es una de las características fundamentales en la OOP.

Los objetos son inaccesibles, e impiden que otros objetos, los usuarios, o incluso los programadores conozcan cómo está distribuida la información o qué información hay disponible. Esta propiedad de los objetos se denomina ocultación de la información.

Esto no quiere decir, sin embargo, que sea imposible conocer lo necesario respecto a un objeto y a lo que contiene. Si así fuera no se podría hacer gran cosa con él. Lo que sucede es que las peticiones de información a un objeto, deben realizarse a través de mensajes dirigidos a él, con la orden de realizar la operación pertinente. La respuesta a estas órdenes será la información

requerida, siempre que el objeto considere que quien envía el mensaje está autorizado para obtenerla.

El hecho de que cada objeto sea una cápsula facilita enormemente que un objeto determinado pueda ser transportado a otro punto de la organización, o incluso a otra organización totalmente diferente que precise de él. Si el objeto ha sido bien construido, sus métodos seguirán funcionando en el nuevo entorno sin problemas. Esta cualidad hace que la OOP sea muy apta para la reutilización de programas.

Beneficios que se obtienen del desarrollo con OOP

Día a día los costos del hardware decrecen. Así surgen nuevas áreas de aplicación cotidianamente: procesamiento de imágenes y sonido, bases de datos multimediales, automatización de oficinas, ambientes de ingeniería de software, etc. Aún en las aplicaciones tradicionales encontramos que definir interfaces hombre-máquina "a-la-Windows" suele ser bastante conveniente.

Lamentablemente, los costos de producción de software siguen aumentando; el mantenimiento y la modificación de sistemas complejos suele ser una ardua tarea; cada aplicación, (aunque tenga aspectos similares a otra) suele encararse como un proyecto nuevo, etc.

Todos estos problemas aún no han sido solucionados en forma completa. Pero como los objetos son portables (teóricamente) mientras que la herencia permite la reusabilidad del código orientado a objetos, es más sencillo modificar código existente porque los objetos no interaccionan excepto a través de mensajes; en consecuencia un cambio en la codificación de un objeto no afectará la operación con otro objeto siempre que los métodos respectivos permanezcan intactos. La introducción de tecnología de objetos como una herramienta conceptual para analizar, diseñar e implementar aplicaciones permite obtener aplicaciones más modificables, fácilmente extendibles y a partir de componentes reusables. Esta reusabilidad del código disminuye el tiempo que se utiliza en el desarrollo y hace que el desarrollo del software sea más intuitivo porque la gente piensa naturalmente en términos de objetos más que en términos de algoritmos de software.

Problemas derivados de la utilización de OOP en la actualidad

Un sistema orientado a objetos, por lo visto, puede parecer un paraíso virtual. El problema sin embargo surge en la implementación de tal sistema. Muchas compañías oyen acerca de los beneficios de un sistema orientado a objetos e invierten gran cantidad de recursos luego comienzan a darse cuenta que han impuesto una nueva cultura que es ajena a los programadores actuales.

Específicamente los siguientes temas suelen aparecer repetidamente:

Curvas de aprendizaje largas. Un sistema orientado a objetos ve al mundo en una forma única. Involucra la conceptualización de todos los elementos de un programa, desde subsistemas a los datos, en la forma de objetos. Toda la comunicación entre los objetos debe realizarse en la forma de mensajes. Esta no es la forma en que están escritos los programas orientados a objetos actualmente; al hacer la transición a un sistema orientado a objetos la mayoría de los programadores deben capacitarse nuevamente antes de poder usarlo.

Dependencia del lenguaje. A pesar de la portabilidad conceptual de los objetos en un sistema orientado a objetos, en la práctica existen muchas dependencias. Muchos lenguajes orientados a objetos están compitiendo actualmente para dominar el mercado. Cambiar el lenguaje de implementación de un sistema orientado a objetos no es una tarea sencilla; por ejemplo C++ soporta el concepto de herencia múltiple mientras que SmallTalk no lo soporta; en consecuencia la elección de un lenguaje tiene ramificaciones de diseño muy importantes.

Determinación de las clases. Una clase es un molde que se utiliza para crear nuevos objetos. En consecuencia es importante crear el conjunto de clases adecuado para un proyecto. Desafortunadamente la definición de las clases es más un arte que una ciencia. Si bien hay muchas jerarquías de clase predefinidas usualmente se deben crear clases específicas para la aplicación que se este desarrollando. Luego, en 6 meses ó 1 año se da cuenta que las clases que se establecieron no son posibles; en ese caso será necesario reestructurar la jerarquía de clases devastando totalmente la planificación original.

Performance (Rendimiento). En un sistema donde todo es un objeto y toda interacción es a través de mensajes, el tráfico de mensajes afecta el rendimiento de este. A medida que la tecnología avanza y la velocidad de microprocesamiento, potencia y tamaño de la memoria aumentan, la situación mejorará; pero en la situación actual, un diseño de una aplicación orientada a objetos que no tiene en cuenta el rendimiento no será viable comercialmente.

Idealmente, habría una forma de atacar estos problemas eficientemente al mismo tiempo que se obtienen los beneficios del desarrollo de una estrategia orientada a objetos. Debería existir una metodología fácil de aprender e independiente del lenguaje, y fácil de reestructurar que no drene el rendimiento del sistema.

1.2.2 Componentes reutilizables

Recursos de software reutilizables.

Cualquier estudio sobre recursos de software estaría incompleto sin estudiar la reutilización, esto es, la creación y la reutilización de bloques de construcción de software.

Tales bloques deben establecerse en catálogos para una consulta más fácil, estandarizarse para una fácil aplicación y validarse para la también fácil integración.

Cuatro categorías de recursos de software que se deberían tener en cuenta a medida que se avanza con la planificación, son las siguientes:

1) **Componentes ya desarrollados:** El software existente se puede adquirir de una tercera parte o provenir de uno desarrollado internamente para un proyecto anterior. Estos componentes están listos para utilizarse en el proyecto actual y se han validado totalmente.

2) **Componentes ya experimentados**: Las especificaciones, diseño, código o datos de prueba existentes y desarrollados para proyectos anteriores que son similares al software que se va a construir para el proyecto actual. Los miembros del equipo de software actual ya han tenido la experiencia completa en el área de la aplicación representada para estos componentes de total experiencia, tendrán un riesgo relativamente bajo.

3) **Componentes con experiencia parcial**: las especificaciones, los diseños, código o los datos de prueba existentes desarrollados para proyectos anteriores que se relacionan con el software que se va a construir para el proyecto actual, pero que requerirán una modificación sustancial. Los miembros del equipo de software actual han limitado su experiencia sólo al área de aplicación representada por estos componentes. Las modificaciones, por tanto, requeridas para componentes de experiencia parcial tendrán bastante grado de riesgo.

4) **Componentes nuevos**: Los componentes de software que el equipo debe construir específicamente para las necesidades del proyecto actual.

Deberían de considerarse las directrices siguientes por el planificador de software cuando los componentes reutilizables se especifiquen como recursos.

1. Si los componentes ya están desarrollados cumplen los requisitos del proyecto, adquiéralos. El costo de la adquisición y de la integración de los componentes ya desarrollados serán casi siempre menores que el coste para el desarrollo del software equivalente. Además el riesgo es relativamente bajo.
2. Si se dispone de componentes ya experimentados, los riesgos asociados a la modificación y a la integración generalmente se aceptan. En plan del proyecto debería reflejar la utilización de estos componentes.

3. Si se dispone de componentes de experiencia parcial para el proyecto actual, su uso se debe analizar con detalle. Si antes de que se integren adecuadamente los componentes con otros elementos del software se requiere una gran modificación, proceda cuidadosamente. El coste de modificar los componentes de experiencia parcial algunas veces pueden ser mayores que el coste de desarrollar componentes nuevos.

Estas directrices deben ser siempre consideradas, si el equipo d etrabajo tiene como tarea realizar un proyecto con características similares a un desarrollo anterior.

Esta actividad de análisis podría ahorrar muchas horas de desarrollo innecesario, si ya se cuenta con componentes reutilizables.

1.3 Gestión de proyectos de software.

La gestión de proyectos de software [2] es una actividad protectora dentro de la ingeniería de software. Empieza antes de iniciar cualquier actividad técnica y continua a lo largo de la definición, el desarrollo y el mantenimiento del software.

Hay tres 'pes' que tienen una influencia sustancial en la gestión de proyectos de software: **personal, problema y proceso.**

El personal debe organizarse en equipos eficaces, motivados para hacer un software de calidad y coordinados para alcanzar una comunicación efectiva.

El problema debe comunicarlo el cliente al desarrollador, dividirse (descomponerse) en las partes que lo constituyen y distribuirse para que trabaje el equipo de software.

El proceso debe adaptarse al personal y al problema. Se selecciona una estructura común de proceso, se aplica un paradigma de ingeniería de software apropiado y se de tareas para completar el trabajo.

La gestión de proyectos es una actividad intensamente humana, y por ello se debe de pensar que al elegir un jefe de proyecto de software que tenga la capacidad de motivar al personal técnico para que produzca conforme a sus mejores capacidades, que tenga la habilidad para moldear procesos existentes que permitan al concepto inicial transformarse en un proyecto final y además que tenga la habilidad para motivar al personal para crear y sentirse creativos incluso cuando deban de trabajar dentro de los límites establecidos para un producto o aplicación de software particular.

Existen casi tantas estructuras de organización de personal para el desarrollo de software como organizaciones que se dedican a ello. Para bien o para mal, el organigrama no puede cambiarse fácilmente. Las consecuencias prácticas y políticas de un cambio de organización no están dentro del alcance de las responsabilidades del gestor de un proyecto de software. Sin embargo, la organización del personal directamente involucrado en un nuevo proyecto de software está dentro del ámbito de gestor del proyecto.

Los ingenieros de software pueden organizarse en diferentes organigramas de equipo que van desde las tradicionales de control jerárquico a los equipos de paradigma abierto. Se pueden aplicar varias técnicas de coordinación y comunicación para apoyar el trabajo de equipo. En general las revisiones formales y las comunicaciones informales persona a persona son las más valiosas para los profesionales.

La mejor estructura de equipo depende del estilo de gestión de una organización, el número de personas que compondrán el equipo, sus niveles de preparación y la dificultad general del problema.

La actividad de gestión de proyectos comprende medición y métricas, estimación, análisis De riesgos, planificación de programa, seguimiento y control.

En la gestión de proyectos de software, hasta el desarrollador más agobiado estará de acuerdo con que el software de calidad es una meta importante. Pero ¿qué es la calidad del software?

La calidad del software se define como: Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente. La calidad de concordancia es el grado de cumplimiento de las especificaciones del diseño durante su realización. Una vez más, cuando sea mayor el grado de cumplimiento, más alta será el nivel de la calidad un aspecto centrado principalmente en la implementación. Si la implementación sigue el diseño, y el sistema resultante cumple los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta.

Los ingenieros de software afrontan la calidad aplicando métodos técnicos sólidos y medidas, realizando las revisiones técnicas formales y realizando pruebas de software bien planificado. Para obtener calidad en sus desarrollos de software realizan actividades que les permitan garantizarla. La garantía de calidad de software (SQA, de Software Quality Assurance) [2] es una actividad de protección que se aplica a lo largo de todo el proceso de ingeniería del software. La SQA engloba :

- Un enfoque de gestión de calidad
- Tecnología de ingeniería del software efectiva
- Revisiones técnicas formales que se aplican durante el proceso del software
- Una estrategia de prueba multiescalada
- El control de la documentación del software y de los cambios realizados
- Un procedimiento que asegure un ajuste a los estándares de desarrollo de software.
- Mecanismos de medición y de generación de informes.

El control de cambios puede equipararse al control de calidad . El control, de la calidad es una serie de inspecciones revisiones y pruebas utilizadas a lo largo del ciclo de desarrollo para asegurar que cada producto cumple con los requisitos que le haya asignado.
Las actividades de garantía de calidad realizadas por el equipo de ingeniería de software son gobernadas por un plan SQA.

El plan identifica:

- Evaluaciones a realizar
- Auditorías y revisiones a realizar
- Estándares que se puedan aplicar al proyecto
- Procedimientos para información y seguimiento de errores.
- Documentos producidos por el grupo.
- Realimentación de información proporcionada al equipo del proyecto de software.

El plan se desarrolla durante la planificación de proyecto y es revisado por todas las partes interesadas. Las actividades de control de calidad pueden ser manuales, completamente automáticas o una combinación de las herramientas automáticas o la interacción humana.

1.3.1 Coordinación y comunicación

Hay muchos motivos por los que los proyectos de software pueden tener problemas. La escala (tamaño) de muchos esfuerzos de desarrollo es grande, conduciendo a complejidades, confusión y significativas dificultades para coordinar a los miembros del equipo. La incertidumbre es corriente, dando como resultado un continuo torrente de cambios que impactan al equipo del proyecto. La interoperabilidad se ha convertido en una característica clave de muchos sistemas. El software nuevo debe comunicarse con el anterior y ajustarse a limitaciones predefinidas por el sistema o por el producto.

Estas características de software moderno son aspectos de la vida. Para enfrentarse a ellos eficazmente, un equipo de ingeniería de software debe establecer métodos efectivos para coordinar a la gente que realiza el trabajo. Para lograr esto se deben establecer mecanismos de comunicación formales e informales entre los miembros del equipo y entre múltiples equipos. La comunicación formal se lleva a cabo por escrito con reuniones organizadas y otros canales de comunicación relativamente no interactivos e impersonales. La comunicación informal es más personal. Los miembros del equipo de ingeniería de software comparten ideas de por sí, piden ayuda a medida que surgen los problemas e interactúan los unos con los otros.

Una colección de técnicas de coordinación de proyectos se dividen de la siguiente manera:

a) Formal, enfoque impersonal: Incluye documentos de ingeniería del software y entregas, memorandos técnicos, hitos del proyecto, planificaciones del programa y herramientas de control de proyecto, peticiones de cambio y documentación relativa, informes de seguimiento de errores e información almacenada.

b) Formal, procedimientos interpersonales: Se concentra en las actividades de garantía de calidad aplicada a productos de ingeniería de software. Esto incluye reuniones de revisión de estado e inspecciones de diseño y de código.

c) Informal, procedimientos interpersonales: Incluyen reuniones de grupo para la divulgación de información y resolución de problemas así como definición de requisitos y del personal de desarrollo.

d) Comunicación electrónica: Agrupa correo electrónico, boletines de noticias electrónicos, paginas Web y, por extensión, sistemas de videoconferencia.

e) Red interpersonal: Discusiones informales con personas que no están en el proyecto pero que pueden tener experiencia o una profunda visión que puede ayudar a los miembros del equipo.

1.3.2 Planificación de proyectos de software.

El proceso de gestión del proyecto de software comienza con un conjunto de actividades que, globalmente, se denominan planificación del proyecto. La primera de estas actividades es la estimación. Siempre que estimamos, echamos un vistazo al futuro y aceptamos resignados cierto grado de incertidumbre.

Aunque la estimación es más un arte que una ciencia, es una actividad importante que no debe llevarse a cabo de una forma descuidada. Existen técnicas útiles para la estimación de costos y de tiempos. Y, dado que la estimación es la base de todas las demás actividades de planificación del proyecto y sirve como guía para una buena ingeniería del software, no es en absoluto aconsejable embarcarse sin ella.

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permite al gestor hacer estimaciones razonables de recursos, costos, y planificación temporal, el objetivo de la planificación se logra mediante un proceso de descubrimiento de la información que lleve a estimaciones razonables.

El encargado de la planificación comienza elevando el ámbito y seleccionando las habilidades técnicas que se requieren para llevar acabo el desarrollo. Hay que especificar la posición dentro de la organización y la especialidad. Para proyectos relativamente pequeños una sola persona puede llevar acabo todos los pasos de ingeniería del software, consultando con especialistas siempre que se requiera.

El número de persona requerido para un proyecto de software solo puede ser determinado después de hacer una estimación.

El planificador de proyectos de software tiene que estimar tres cosas antes de que comience el proyecto: cuánto durará, cuánto esfuerzo requerirá y cuánta gente estará implicada. Además el planificador debe predecir los recursos (de hardware y software) que va a requerir, y el riesgo implicado.

La estimación del proyecto de software nunca será una ciencia exacta, pero la combinación de buenos datos históricos y de técnicas puede mejorar la precisión de la estimación.

1.3.3 Planificación temporal y seguimiento de proyectos.

La planificación temporal es la culminación de una actividad de planificación, componente primordial de la dirección de proyectos de software. Cuando se combinan métodos de estimación y análisis de riesgo, la planificación temporal se convierte en un mapa a seguir por el gestor del proyecto. La planificación temporal empieza con la descomposición del proceso. Las características del proyecto se emplean para adaptar un conjunto de tareas apropiado al trabajo a realizar.

Una red de tareas se usa para calcular el camino crítico de un proyecto, el gráfico de tiempo e información correspondiente a cada proyecto dependerá de características específicas (como lo es el tamaño del proyecto). El gestor de proyecto puede seguir y controlar todos los pasos del proceso de ingeniería del software usando la planificación temporal como directriz. El objetivo del gestor es definir todas las tareas del proyecto

La planificación temporal de un proyecto de software es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto, asignando el esfuerzo a las tareas específicas de la ingeniería del software. Es importante resaltar, sin embargo que la planificación temporal evoluciona con el tiempo. Durante las primeras etapas de la planificación del proyecto, se desarrolla una planificación macroscópica. Este tipo de planificación temporal identifica las principales actividades de la ingeniería del software y las funciones del producto a las que se aplican. A medida que el proyecto va progresando, cada entrada de la planificación temporal macroscópica se refina en una planificación temporal detallada. Aquí se identifican y programan las tareas del software específicas (requeridas para realizar una actividad)

La planificación temporal para proyectos de desarrollo de software puede verse desde dos perspectivas bastante diferentes. En la primera, se ha establecido ya (irrevocablemente) una fecha final de entrega de un sistema basado en computadora. La organización del software está limitada a distribuir el esfuerzo dentro del tiempo del marco previsto. El segundo tipo de vista de la planificación temporal asume que se han estudiado unos límites cronológicos aproximados pero que la fecha final estará establecida por la organización de la ingeniería de software.

El esfuerzo se distribuye para conseguir el mejor empleo de los recursos, y se define una fecha final después de un cuidadoso análisis del software. Desgraciadamente, la primera situación es más frecuente que la segunda.

Como todas las áreas de la ingeniería del software, la planificación temporal de proyectos de software se guía por unos principios básicos:

a) **Compartimentación**: El proyecto debe dividirse en un número de actividades y tareas manejables.

b) **Interdependencia:** Se deben determinar las interdependencias de cada actividad o tarea compartida. Algunas tareas deben ocurrir en una secuencia determinada; otras pueden darse en paralelo. Algunas actividades no pueden comenzar hasta que el resultado de otras no este disponible. Otras actividades pueden ocurrir interdependientemente.

c) **Asignación de tiempos:** A cada tarea que se vaya a programar se le deben asignar cierto número de unidades de trabajo. Además a cada tarea se le debe asignar una fecha de inicio y otra de finalización que son función de las interdependencias y de si el trabajo se hará a tiempo total o tiempo parcial.

d) **Validación de esfuerzo**: Todos los proyectos tienen un número definido de miembros de la plantilla. A medida que se hace la asignación de tiempo, el gestor del proyecto debe asegurarse de que no se haya asignado un número de personas mayor que el de la plantilla en ese momento.

e) **Responsabilidades definidas:** Cada tarea que se programe debe asignarse a un miembro del equipo específico.

f) **Resultados definidos:** Cada tarea programada debería tener un resultado definido. Para los proyectos de software, el resultado normalmente es un producto o una parte de un producto. Los productos se combinan frecuentemente en entregas.

g) **Hitos definidos:** Todas las tareas o grupos de tareas deberían asociarse con un hito del proyecto. Se consigue un hito cuando se ha revisado la calidad de uno o más productos y se han aceptado.

1.4 Groupware

El concepto "Groupware" o "Trabajo Colaborativo" es la convergencia de lo que en años anteriores se consideraban tecnologías independientes: como la mensajería, la conferencia y los flujos de información dentro de una organización o entre diferentes organizaciones. Poniendo el concepto en tres planos diferentes: la comunicación, la coordinación y la colaboración, podemos decir que Groupware es una herramienta que ayuda a los individuos a trabajar juntos en un modo cualitativamente mejor que el planteado por los esquemas de organización tradicionales, proporcionando:

- Comunicación con colegas a través de correo electrónico.
- Colaboración en grupos de trabajo a través de un espacio de trabajo virtual.
- Coordinación de procesos estratégicos rediseñando la estructura del proceso de negocios para comunicar y crear mecanismos de colaboración así como implementar políticas bien definidas en la empresa.

El Groupware responde a los principales retos que un negocio requiere hoy en día.

Es así como por medio del uso de está tecnología cada persona o empresa logra:

1. Hacer que la tecnología mantenga comunicados a todos los integrantes de una corporación, sin importar barreras geográficas o de tiempo.

2. Hacer que la tecnología provea una manera efectiva de diseñar y de evolucionar las prácticas de acción productivas para la corporación (Políticas y Procedimientos).

3. Estas prácticas constituyen la fuerza motriz de una organización y se pueden referir a la operación y administración o al proceso mismo del negocio.

4. Hacer que la tecnología permita a la organización capitalizar conocimientos, experiencias, creatividad e iniciativa de los individuos.

5. Hacer que está tecnología promueva una cultura de compromiso a través de los procesos de la corporación.

6. Permitir generar diagnósticos de los procesos para apoyar acciones efectivas que ofrezcan un valor agregado a la corporación.

De esta manera cada corporación logra capitalizar está evolución mediante diversos mecanismos:

- Recolectando la Información que generan.
- Almacenándola.
- Generando diagnósticos.
- Administrándola.
- Armando planes de acción alrededor de esta información.
- Implantando los planes de acción.

- Dando un seguimiento detallado de cada instancia.
- Recolectando de nuevo la información.
- Retroalimentando los procesos.

Las bases de datos y el Groupware

La tecnología de Groupware permite el manejo de Bases de Datos Documentales y su interrelación con Bases de Datos Relacionales, así como el desarrollo de páginas Web que permiten interrelacionar con las bases de datos relacionales, objetos ligados o incrustados, imágenes, voz// sonido y video. Además, permite también el control de versiones, el monitoreo de cambios a un documento por diferentes usuarios y la ligas a otros documentos.

Replicación

La tecnología de Groupware permite la replicación de información entre grupos de trabajo ayudando a mantenerla sincronizada a lo largo de sitios dispersos geográficamente. La replicación puede ser:

- **Bidireccional**

Sincroniza los cambios hechos en todos los sitios.

- **Eficiente**

Se maneja a nivel campo replicando sólo aquellos campos que se modificaron.

- **Selectiva**

Es posible replicar sólo un subconjunto de una bases de datos basándonos en políticas de acceso a las bases de datos.

- **Replicación a nivel cliente**

Permite el mismo nivel de acceso a la información a usuarios móviles o conectados remotamente.

- **Vía background**

Permite al usuario continuar con otras tareas mientras ésta se lleva a cabo.

Seguridad

La seguridad es un punto de suma importancia en el manejo de información, especialmente cuando se maneja información confidencial de la corporación. La tecnología de Groupware provee diferentes tipos y niveles de seguridad:

- Autenticación de usuarios.
- Control de Acceso por roles a sitios, aplicaciones e inclusive a documentos y campos.
- Encriptación de información.
- Manejo de firmas electrónicas.

Mensajería

La mensajería es otra de las bondades de Groupware y provee a los usuarios con las siguientes ventajas:

- Flujo de información entre diferentes clientes de correo.
- Flujo de correo sobre Internet.
- Consulta de correo usando browsers (Internet Explorer, Netscape, Spin, etc.).
- Colaboración entre Grupos de Trabajo.
- Las aplicaciones pueden habilitarse para envío electrónico de los documentos que estas

contienen.
- En los documentos enviados se pueden manejar ligas a otros documentos o a aplicaciones relacionadas.
- Notificación de eventos y recordatorios vía correo a las personas involucradas en los procesos.

No cabe duda, que Groupware es la tecnología que aunada a las bastas capacidades de la Internet, está destinada a ser el factor diferenciador para lograr la eficientización de los diferentes procesos críticos de manejo de información y la convivencia de los recursos dentro de toda compañía.

CAPÍTULO II

ANÁLISIS DE LOS REQUERIMIENTOS DEL PROBLEMA

Capítulo II ANÁLISIS DE LOS REQUERIMIENTOS DEL PROBLEMA.

2.1 Planteamiento del problema.

Debido a que los desarrollos de software son cada día más complejos, el tiempo y el tamaño de los desarrollos es cada día mayor, y los problemas cada vez son más especializados, es muy importante contar con grupos de trabajo que sean partícipes ya que las soluciones para los desarrollos requieren aptitudes diferentes y la experiencia de varias personas.

Para cumplir con los desarrollos, las personas necesitan trabajar como miembros de un equipo (cooperación), requieren intercambiar información más frecuentemente y necesitan estar comunicadas ya que el éxito de un equipo de trabajo dependerá no sólo de las aptitudes individuales, sino del nivel de cooperación.

Objetivo:

Construir un sistema para desarrolladores de software que permita monitorear, controlar y aprobar los cambios en el proyecto, crear una tecnología que mantenga comunicados a todos los integrantes de una corporación, sin importar barreras geográficas o de tiempo. Este sistema combinará los procedimientos humanos y las nuevas tecnologías para proporcionar un mecanismo que permita control de cambios y generará la documentación de los desarrollos de software de los equipos participantes en el sistema recolectando la información que se va generando en el sistema.

Definición del Problema:

La idea de realizar este sistema surgió en el Departamento de Aplicaciones Avanzadas de la Dirección General de Cómputo Académico de la UNAM. Este departamento tiene como objetivo entre muchos otros el desarrollo de nuevas tecnologías que beneficien a la comunidad universitaria.

Uno de los problemas que se tenía era que todos los desarrolladores que participaban en el departamento estaban involucrados en uno o mas desarrollos y para poder avanzar en el desarrollo se debía estar bien informados de los cambios en dichos desarrollos, pero al estar participando en actividades diferentes y a diferentes horarios, era muy difícil la reunión y comunicación física entre todos, además si los participantes no estaban en ese momento el equipo de trabajo tenía que estudiar el desarrollo para analizar qué cambios que había realizado la última persona que trabajo en el proyecto y revisar el correo electrónico para ver si había instrucciones para continuar. Por ello se pensó en desarrollar un sistema que cubriera la necesidad de mantener comunicado y enterado de los avances de los proyectos y los cambios que se están efectuando sobre el desarrollo al equipo de trabajo, saber en qué etapa se encontraba y quién era encargado de cada etapa en la que se dividió cada proyecto, además de

permitir ser partícipes en el proyecto haciendo propuestas para mejorar el desarrollo de cada uno de ellos.

Este sistema nos permite saber las decisiones tomadas sin necesidad de estar trabajando en el Departamento, y sin necesidad de que nadie nos explicara en qué etapa estaba el desarrollo, todo se puede revisar en el sistema.

Cada avance en un desarrollo de software debe ser informado y documentado no sólo al líder de proyecto sino a todos los que participan en él. De otra manera los desarrollos se vuelven más largos y no se ven los avances en ellos.

2.2 Análisis de los Requerimientos del Problema.

Método:

Para resolver este problema se pensó en construir un sistema basado en web, que pudiera ser consultado en cualquier momento y en cualquier lugar por cualquier participante del equipo, que en el se pudieran subir documentos (entre los cuales pueden encontrarse propuestas) los cuales indicaran qué decisiones se habían tomado y qué cambios eran necesarios en cada una de las etapas del desarrollo, se quería además que una vez revisados los documentos permitiera agregar comentarios y anexos a los documentos enviados, que se pudiera tomar decisiones sobre ellos (es decir aprobarlos o rechazarlos) y además que fuera una herramienta para comunicarnos con el equipo de desarrollo sin necesidad de enviar un correo electrónico, es decir crear un espacio virtual sobre la web para los grupos de trabajo.

Se realizó un estudio para detectar las características que debería tener el sistema y las herramientas que serían más factibles emplear en él.

Se tomaron las siguientes decisiones:

- Para el desarrollo del sistema se decidió usar herramientas de Open-Source (código libre).
- Estudiando las ventajas y beneficios de varios lenguajes de programación se decidió emplear el lenguaje de programación Java por ser un lenguaje orientado a objetos.
- Para poder llevar a web el sistema decidimos usar Apache-Tomcat servidor de aplicaciones que permite llevar a la web la tecnología conocida como JavaServer Pages (JSP).
- Se decidió emplear PostgresSQL como manejador de bases de datos por ser una gran opción y los beneficios que ofrece. Una vez elegido el manejador de bases de datos, se realizó el estudio de las necesidades y se creó el diagrama entidad-relación para generar la base de datos.

- Se decidió emplear el sistema operativo Unix, en una estación de trabajo Sun Microsystems Modelo E250 , que cuenta con la versión 8 de Solaris.

Una vez que se eligieron las herramientas se comenzó a definir la base de datos y se inició a realizar el diseño de las páginas en web y de los formularios . Con el diseño de la Base de Datos y las pantallas establecidas se dio inicio a la etapa de programación para obtener el sistema de colaboración, control y monitoreo de proyectos necesario para el departamento.

Lo que se logró fue crear un esquema organizacional, que pretende obtener una mejor planificación del proyecto con la asignación de etapas y tareas a las personas con mejores aptitudes para cada etapa y un control de cambios necesario para toda organización con la responsabilidad de crear software de calidad.

Por último, al ser un control de cambios se busca que genere la documentación de dichos cambios recolectando la información que genera y almacenándola, para poder ser expuesta cuando se realicen juntas para revisiones de proyectos, esta aplicación ahorra tiempo en la labor de realizar la documentación de los avances en el desarrollo del proyecto, ya que genera la información de cada documento (objeto) puesto a consideración en las etapas de desarrollo.

CAPÍTULO III

HERRAMIENTAS DE SOFTWARE

Capítulo III. HERRAMIENTAS DE SOFTWARE.

A continuación se describen las herramientas utilizadas para la realización de esta tesis: Código Libre, Sistema Operativo Unís, Base de Datos PostgresSQL, Lenguaje de programación Java, Apache-Tomcat, Java Database Connectivity (JDBC).

3.1 Código Libre (Open-Source)

A la hora de desarrollar un proyecto de alto grado tecnológico, siempre buscamos la receta que permita obtener la máxima funcionalidad posible por el menor precio, manteniendo además un elevado nivel de calidad. Pues bien, esa receta existe.

Se llama software libre (Open-Source) y permite reducir los costos porque elimna el gasto en licencias y, en muchas ocasiones, proporciona un grado de calidad mayor. La prueba es el servidor web Apache, que dispone de una cuota de mercado del 60% y ya ha sido integrado dentro de diversos paquetes comerciales.

Hacer las cuentas de lo que cuestan los equipos de cómputo, las licencias de los productos y los desarrollos a a medida, y encajar los cálculos con el presupuesto disponible puede convertirse en un problema de ingeniería financiera. El software libre puede ayudarnos a solucionar este problema: el ahorro en licencias permite que el margen de dinero destinado al desarrollo sea mayor, y abre la puerta a unos resultados mucho más satisfactorios, ya que podemos optar por el desarrollo de módulos que se descartaron en un principio. Las principales ventajas que el software libre ofrece a las empresas son las siguientes:

1) **Precio cero.** Estas aplicaciones se encuentran disponibles de forma gratuita para su descarga e instalación.

2) **Licencias Ilimitadas.** Frente a otros productos que obligan al pago de licencias por cada equipo de cómputo o por el número de usuarios (simultáneos o no) que lo utilizan.

3) **Actualizaciones Disponibles.** Tampoco se cobra por las actualizaciones de dichos productos . Si aparecen mejoras pueden ser instaladas inmediatamente. No es necesario pagar nuevas licencias por versiones más modernas de dicho producto cuando se quedan sin soporte las versiones anteriores.

4) **Soporte.** Existen empresas que se dedican a proporcionar soporte para dichos productos. Además, al ser código de abierto, se puede cambiar la empresa que proporciona el servicio de soporte, e incluso dedicar a empleados a realizar esta labor.

5) **Adaptación a las propias actividades.** No se esta atado a un producto cerrado. Cualquier modificación que se quiera realizar sobre los productos se puede contratar a una empresa de desarrollo o realizarlo a través de un equipo interno.

6) **Independencia del proveedor.** Al disponer del código del programa, cualquier empresa con conocimientos del lenguaje de programación en el que esté desarrollado puede adaptarlo.

7) **Calidad.** Equiparable o superior a los productos cerrados.

8) **Seguridad.** Al disponer del código fuente, no existen puertas traseras y los problemas de seguridad pueden detectarse y solucionarse de una forma rápida frente a la lentitud en otros tipos de software.

Esta reducción de costos se puede aplicar en el software base de diversos proyectos, sustituyendo sistemas operativos (por GNU/Linux, FreeBSD, NetBSD, GNU/Hurd), servidores web (Apache con sus módulos adicionales), servidores de aplicaciones (Tomcat, Jboss, Jonas), bases de datos (PostgresSQL, InterBase, mySQL), escritorios (GNOME, KDE), herramientas de productividad (OpenOffice, Koffice), entre otras muchas posibilidades. Esto permitiría, por ejemplo, disponer de una red completa basada en Software Libre, establecer aplicaciones web complejas desarrolladas sobre este tipo de programas y herramientas, o tener un conjunto de computadoras de trabajo con software de open-source.

3.2 Sistema operativo UNIX.

El sistema operativo es un conjunto de programas cuyas misiones son:

1. Gestionar los recursos del sistema informático (procesadores, memoria, discos, etc), entre los diferentes procesos que compiten por ellos.

2. Ofrecer al usuario una especie de "máquina virtual" o "máquina extendida", más fácil de usar que el hardware subyacente ("Principio de embellecimiento").

Historia de UNIX

A pesar de los sistemas abiertos, la historia de UNIX está dominada por el ascenso y caída de los sistemas hardware.

El **UNIX** es un sistema operativo multiusuario y multitarea que trabaja en el modo de tiempo compartido (time-sharing). Esto significa que el sistema operativo atiende y ejecuta varios programas simultáneamente de los distintos usuarios que estén trabajando en él, pero realmente sólo atiende a uno solo por vez durante una pequeña fracción de tiempo (del orden de los milisegundos). El sistema operativo ejecuta un trozo de un programa, luego lo interrumpe, toma otro programa, continúa su ejecución, y así sucesivamente. De esta manera le da a cada usuario la impresión de que él solo está utilizando la máquina.

El **UNIX** fue originalmente desarrollado por los laboratorios *Bell* de la *AT&T*. Estos laboratorios participaron alrededor del año 1969 en un proyecto conjunto con la *General Electric* en la elaboración de un sistema operativo multiusuario que pudiera aprovechar el hardware disponible y brindar unos servicios suficientemente interactivos a una gran cantidad de usuarios.

Este proyecto se denominó **Multics**. Más tarde, los laboratorios *Bell* consideraron que las metas propuestas no estaban en camino de ser alcanzadas y por lo tanto se retiraron del proyecto.

En ese entonces, uno de los investigadores de los Laboratorios *Bell*, Ken Thompson[5], quien había participado del proyecto Multics, disponía de una computadora PDP-7 e ideó un sistema operativo multiusuario según sus propias ideas sobre un manejo dinámico de la memoria.

Se Llama **UNIX** a este primer esbozo de sistema operativo, el cual fue puesto en marcha en 1970 en la PDP-7 y luego trascripto a una máquina un poco más grande, una PDP-11.

El **UNIX** estaba en sus orígenes escrito en lenguaje de máquina, es decir, estaba hecho mediante el ensamblador de la PDP-7.

Cuando se planteó el problema de transportar el **UNIX** de una máquina a otra se dieron cuenta de que era necesario reescribirlo en su totalidad, puesto que al hacerlo en ensamblador lo hacían fuertemente dependiente del hardware. Entonces vieron la posibilidad de reescribirlo en algún lenguaje de alto nivel, de tal manera que fuera portable de una máquina a otra y que solamente tuviera algunas partes dependientes estrictamente del hardware y configuración de la máquina.

De esta manera, Ken Thompson junto a Dennis Ritchie y Brian Kernighan desarrollaron un nuevo lenguaje de programación al cual denominaron **C** y escribieron el **UNIX** en este nuevo lenguaje, dejando lo mínimo necesario dependiente del hardware. Así se hacía más sencillo su transporte de una máquina a otra, aún entre máquinas con CPUs totalmente distintas.

Inicialmente, el **UNIX** fue utilizado en los Laboratorios *Bell* solamente para uso interno en la preparación de documentación relacionada con sus patentes y de programas de aplicación. No se comercializó, y se le distribuía entre aquellas universidades que lo solicitaban con propósitos educacionales. Recién en 1977 salió a la venta comercialmente cuando ya era popular debido a su uso en las universidades. En este mismo año fue portado por primera vez a una máquina distinta de las PDP, una Interdata 8/32. Los investigadores de varias universidades hicieron sus propias contribuciones al desarrollo del sistema, entre quienes se destaca la Universidad de Berkeley.

Juntando algunas de estas versiones y contribuciones entre 1977 y 1982, *AT&T* produjo el **UNIX** System III y en 1983 el actualmente conocido **UNIX** System V. Además la *AT&T* licenció a varias empresas los programas fuentes del sistema operativo escritos en **C** , con los fines de transportarlos a distintas máquinas. Estos sistemas operativos, derivados del **UNIX**, tuvieron nombres tales como BSD, XENIX, ONIX, CROMIX, etc.

Además esto produjo la diversificación del hardware que soporta un sistema **UNIX** o similar, y así actualmente se tienen versiones del **UNIX** para máquinas con un procesador tan chico como un Z80 hasta la máquina actualmente más grande del mundo, la CRAY-2, donde coexiste con otros sistemas operativos.

El resultado de esto es que además de la popularización y la diversificación, los usuarios pueden tomar ventajas de que los programas escritos en **C** para una máquina determinada son portables con mucha facilidad a otra máquina, y basta solamente compilarlos en la nueva máquina con el compilador correspondiente para que funcionen en la misma.

A comienzos de 1984, había sobre 100.000 instalaciones del sistema UNIX en el mundo, funcionando en máquinas con un amplio rango de computadoras, desde microprocesadores hasta mainframes. Ningún otro sistema operativo puede hacer esta declaración. Muchas han sido las razones que han hecho posible la popularidad y el éxito del sistema UNIX:

- El sistema está escrito en un lenguaje de alto nivel, haciéndolo fácil de leer, comprender, cambiar, y mover a otras máquinas. Ritchie estimó que el primer sistema en C era de un 20 a un 40 por ciento más grande y lento porque no estaba escrito en lenguaje ensamblador, pero las ventajas de usar un lenguaje de alto nivel superaban largamente a las desventajas.
- Posee una simple interfaz de usuario con el poder de dar los servicios que los usuarios quieren.
- Provee de primitivas que permiten construir programas complejos a través de programas simples.
- Usa un sistema de archivos jerárquico que permite un mantenimiento fácil y una implementación eficiente.
- Usa un formato consistente para los archivos, el flujo de bytes, haciendo a los programas de aplicación más fáciles de escribir.
- Provee una simple y consistente interfaz a los dispositivos periféricos.
- Es un sistema multiusuario y multitarea; cada usuario puede ejecutar varios procesos simultáneamente.
- Oculta la arquitectura de la máquina al usuario, haciendo fácil de escribir programas que se ejecutan en diferentes implementaciones hardware.

Sin embargo tiene algunos inconvenientes:

- Comandos poco claros y con demasiadas opciones.
- Escasa protección entre usuarios.
- Sistema de archivos lento.

A pesar de que el sistema operativo y muchos de los comandos están escritos en C, UNIX soporta otros lenguajes, incluyendo Fortran, Basic, Pascal, Ada, Cobol, Lisp y Prolog. El sistema UNIX puede soportar cualquier lenguaje que tenga un compilador o intérprete y una interfaz de sistema que defina las peticiones del usuario de los servicios del sistema operativo de la forma estándar de las peticiones usadas en los sistemas UNIX.

Estructura del sistema

El sistema operativo interactúa directamente con el hardware, suministrando servicios comunes a los programas y aislándolos de la particularización del hardware. Viendo el sistema como un conjunto de capas, el sistema operativo es comúnmente llamado como núcleo del sistema o kernel. Como los programas son independientes del hardware que hay por debajo, es fácil moverlos desde sistemas UNIX que corren en diferentes máquinas si los programas no hacen referencia al hardware subyacente. El sistema operativo UNIX se compone de bloques funcionales.

Control de procesos

Un programa es un archivo ejecutable, y un proceso es una instancia del programa en ejecución. En UNIX pueden ejecutarse varios procesos simultáneamente (esta característica es denominada algunas veces como multiprogramación o multitarea) sin un límite lógico en el número de ellos, y varias instancias del mismo programa pueden existir simultáneamente en el sistema. Algunas llamadas al sistema permiten a los procesos crear nuevos procesos, acabar procesos, sincronizar niveles de ejecución de procesos y controlar la reacción de algunos sucesos. Sujeto a sus propias llamadas al sistema, los procesos son independientes de los demás.

Generalmente, las llamadas al sistema permiten al usuario escribir programas que realicen sofisticadas operaciones, y como resultado, el kernel del sistema UNIX no contiene muchas funciones que son parte del "kernel" en otros sistemas. Estos programas, incluyendo compiladores y editores, son programas a nivel de usuario en el sistema UNIX. El principal ejemplo de estos programas es el shell, el intérprete de comandos que los usuarios ejecutan normalmente de entrar en el sistema.

El núcleo del sistema operativo UNIX conoce la existencia de un proceso a través de su bloque de control del proceso, donde se describe el proceso y su entorno, constituyendo un contexto consistente en:

- Espacio de direccionamiento y entorno de ejecución: Variables que utiliza el proceso.
- Contenido de los registros hardware: Contador de programa, registro de estado del procesador, puntero de la pila y registros de propósito general.
- Contenido de las estructuras del núcleo relacionadas con el proceso: Tabla de proceso, áreas, regiones, etc.

Cada proceso se reconoce dentro del sistema por un número que lo identifica unívocamente y que se conoce como Identificador del Proceso o PID.

Todos los procesos, excepto el proceso 0, son creados por otro proceso; es decir, el sistema de creación y gestión de procesos en UNIX es jerárquico.

Consideraciones sobre el hardware

La ejecución de procesos de usuario en sistemas UNIX se divide en dos niveles: usuario y kernel. Cuando un proceso ejecuta una llamada al sistema, el modo de ejecución cambia del modo usuario a modo kernel:

El sistema operativo ejecuta y atiende el servicio requerido por el usuario, devolviendo un código de error si falla. Incluso si el usuario no hace una petición explícita de los servicios del sistema operativo, el sistema operativo continúa realizando operaciones que relacionan a los procesos de usuario, manipulando interrupciones, planificando procesos, administrando la memoria, etc.

Muchas arquitecturas (y sus sistemas operativos) soportan más niveles que los dos descritos, pero estos dos modos, usuario y kernel, son suficientes para los sistemas UNIX.

3.3 Bases de datos relacionales.

Pocas herramientas de software son tan necesarias como las bases de datos. No hay empresa o institución que pueda prescindir de ellas; son el alma de los programas contables, de la nómina y de los inventarios en las empresas tradicionales, pero de igual manera alojan la información que se consumen en todo el mundo gracias a la red Internet.

Entre los diferentes modelos de bases de datos, el que se utiliza casi en la totalidad de las aplicaciones actualmente es el modelo relacional basado en el modelo relacional de Codd [6].

La importancia de las bases de datos relacionales ha generado una millonaria y gigantesca industria mundial, representada por empresas como Oracle, Informix, Sybase y muchas más, las cuales facturan anualmente varios millones de dólares por el licenciamiento de sus manejadores de bases de datos.

Afortunadamente la actual revolución de la tecnología de la información ha propiciado que se desarrolle el software de código abierto (open-source), nos ha permitido utilizar PostgresSQL, uno de los servidores de bases de datos eficientes y gratuitos.

3.3.1 PostgresSQL

Historia de PostgresSQL.

El más antiguo antecesor de PostgresSQL es Ingres [7], desarrollado en la Universidad de California en Berkeley de 1977 a 1985. Con el fin de mejorar Ingres, Michael Stonebraker generó en 1986 un nuevo servidor de bases de datos y lo llamó PostgresSQL, es decir posterior a Ingres. Para el año de 1994, Jolly Chen y Andrew Yu le agregaron la funcionalidad del lenguaje de consulta estructurado (SQL, por sus siglas en inglés), una norma mundial establecida varios lustros antes y lo llamaron Postgres95 (1994-1995).

Durante 1996 se dieron dos cambios importantes : se cambió el nombre a PostgresSQL y se formó un grupo especial de desarrollo, comandado por Marc. GG. Fournier en Toronto, Canadá.

Desarrollo de PostgresSQL.

Al igual que muchos proyectos de código abierto, PostgresSQL es desarrollado por un gran número de programadores que utilizaron la Internet como medio para discutir, acordar y enviar las mejoras que lo han convertido en el líder de su área. A diferencia del software comercial, cada 3 a 5 meses se libera una nueva versión con nuevas características, con menos errores (bugs) o más acorde a las normas ce SQL92.

El código fuente, fundamentalmente en lenguaje C, cuenta con más de 250,000 líneas;
y no cuesta un solo centavo.

PostgresSQL se encuentra disponible para los sistemas operativos tipo UNIX .

Una distribución clásica de PostgresSQL ofrece mucho más que el motor o servicio de la base de datos; siempre lo acompañan algunas herramientas que facilitan su configuración y administración: un cliente interactivo de modo texto, otro en modo grafico, utilerías para extraer la base de datos hacia un archivo de comandos SQL, una interfase para programadores (API) para hacer aplicaciones en lenguaje C, documentación para usuarios, programadores y administradores .

El servidor de la base de datos se puede acceder por medio de varios lenguajes: C, Perl, Visual Basic, Delphi, Pyton y otros más; lo cual le da una gran versatilidad para usarse como motor de bases de datos de prácticamente cualquier aplicación que requiere un manejador robusto, eficiente y que cumpla con las normas internacionales.

El sistema gestor de bases de datos PostgresSQL.

PostgresSQL es un sistema de gestión de bases de datos relacionales que une las estructuras clásicas de estos sistemas con los conceptos de programación orientada a objetos, lo que convierte a PostgresSQL en una base de datos objeto relacional. Utiliza el modelo cliente/servidor de un proceso por usuario. Este modelo consta de: un programa supervisor, uno o varios programas de aplicación, un servidor de base de datos por cada programa de aplicación. El programa supervisor se ejecuta en el sistema como un demonio y gestiona una colección de base de datos.

Los programas de aplicación realizan las peticiones de acceso a una base de datos por medio de la biblioteca de funciones libpq.

La conexión entre el programa de aplicación y el servidor de base de datos se realiza por medio de un socket TCP/IP (Transmission Control Protocol / Internet Protocol). El mecanismo de conexión es el siguiente: El demonio supervisor, que en adelante llamaremos postmaster, permanece a la escucha de un puerto TCP/IP. Cuando un cliente desea acceder a la base de datos lanza un connect dirigido al puerto del postmaster, el cual al recibir la petición realiza un fork de si mismo. El nuevo proceso creado con la función fork permanece en sesión con la aplicación cliente, mientras que el postmaster vuelve a la escucha para atender otras peticiones de conexión, este modelo de arquitectura al estar apoyado en TCP/IP, permite que los clientes puedan estar tanto en local como en remoto, utilizando en local la dirección loopback 127.0.0.1.

Con respecto a la arquitectura de los datos, las bases de datos se almacenan en directorios de tipo estándar, cuando se arranca el postmaster, se le indican en que directorio van a estar las bases de datos. La estructura de estos directorios parte de la variable PGDDATA que puede ser suministrada de varias formas: por medio de variables de entorno, en la secuencia de arranque con el parámetro -D.
A partir de este punto la estructura de directorios queda de la siguiente forma:

- Ficheros denominados con el nombre pg_xxxxxxx, conteniendo varias tablas que utiliza el propio sistema para gestionarse.
- Directorio base, que contiene las bases de datos, estando representada cada una de ellas por un directorio cuyo nombre coincide con el de la base de datos.

Dentro de cada directorio que almacena las diversas bases de datos, existen una serie de ficheros, una parte de los cuales tienen el nombre con la forma px_xxxxxx, que contiene las tablas de sistema que gestionan la base de datos, a la que pertenecen, y una serie de ficheros cada uno de los cuales coincide con las clases (tablas), que están definidas en dicha base de datos y que contienen los datos propiamente dichos.

Los conceptos clásicos de las bases de datos relacionales se unen a los conceptos de programación orientada a objetos (POO). En PostgresSQL las tablas se denominan clases, la filas se denominan instancias y las columnas se denominan atributos. Un concepto que aparece en PostgresSQL y los distingue de otros modelos es la herencia. Este nuevo concepto

viene claramente de la POO, y se ajusta a esta perfectamente. Así pues, cuando se crea una nueva clase heredada de otra, la clase creada adquiere toda las características de la clase de la que proviene, más las características que se definan en la nueva clase. De la misma forma cuando se hereda una clase, sé esta creando una tabla igual a la origen de la herencia.

PostgresSQL 7.1.3

Esta nueva versión contiene menos errores de programación (bugs) que la anterior, se acerca más a la norma SQL92 y presenta varios avances en la funcionalidad y el desempeño.

Los cambios entre las versiones anteriores se cuentan por decenas, algunos pequeños, otros muy técnicos. Entre los cambios se encuentran :

- Llaves foráneas (Foreign Keys): Necesarias para Asegurar la integridad relacional de las bases de datos.
- Revisión profunda de optimizador : Que ha permitido mejorar la ejecución de las consultas (queries) y un mejor desempeño con menos uso de memoria.

Así como estos, hay docenas de cambios que mejoran, hacen más confiable y eficiente esta nueva versión. Lo que hacen de PostgresSQL una buena opción totalmente gratuita.

3.3.2 Software ERWIN.

Erwin [8] es una herramienta para modelar, que ayuda a diseñar bases de datos de alto desempeño para cliente/servidor y *web/intranet*, así como aplicaciones de *data warehousing*.

La herramienta Erwin no solo ayuda a diseñar modelos de datos lógicos, también construye automáticamente estructuras de datos físicos con la información del diagrama entidad-relación.

Cuando el modelo de datos está listo para usarse, simplemente se selecciona el servidor donde se quiere construir la base de datos y se eligen las opciones de generación de esquema que se quieran incorporar. En minutos, Erwin automáticamente construye la base de datos física, incluyendo todas las tablas, índices, procedimientos almacenados, *triggers* de integridad referencial y otros componentes necesarios para manejar exitosamente los datos usados en la organización.

Cuando se crea un diagrama Erwin, el modelo de la información se representa por entidades (gente, lugares y cosas), atributos (hechos acerca de una entidad, tales como nombre de la persona, dirección, edad, etc.), y relaciones entre entidades.

Cada entidad corresponde a una tabla en la base de datos, con instancias de entidades que corresponden a los renglones de la tabla y atributos de entidades correspondientes a encabezados de columnas. Las relaciones, usadas por DBMS (*data base management system*) para ligar renglones de datos en tablas diferentes, están representadas como frases verbales en una línea conectando a dos entidades. Cuando se actualiza una base de datos física, Erwin automáticamente genera un *script* de definición de datos SQL, para crear tablas de bases de

datos, incluyendo llaves, constraints y códigos *trigger* SQL para reforzar la integridad referencial entre tablas relacionadas.

Cuando Erwin crea un esquema de bases de datos, genera un *script* de cliente DDL (*data definition language*) usando la sintaxis correcta de SQL para el servidor seleccionado. Se puede ver el código que genera Erwin y, si se desea, se puede modificar antes de que se cree la base de datos.

3.1 Lenguaje de programación Java: Orígenes del lenguaje de programación.

Java [9] fue concebido como James Gosling, Patrick Naughton, Cris Worth, Ed Frank y Mike Sheridan en Sun Microsystems, Inc en 1991. Se tardó 18 meses en desarrollar su primera versión. El lenguaje se llamó en un principio Oak pero después en 1995 fue renombrado como Java. Mientras se trabajaban distintos aspectos de Java surgió un factor definitivo para Java, este factor fue el Word Wide Web. El mundo de la Web se desarrollo al mismo tiempo que Java estaba siendo implementado. Aunque la motivación inicial para Java fue la de proporcionar un lenguaje de programación multiplataforma utilizado para programar dispositivos electrónicos como hornos de microondas y mandos a distancia no fue este su destino final, fue Internet quien le permitió situarse como líder de los lenguajes de programación.

Java es un lenguaje de programación seguro y portable, la clave para resolver estos problemas es que la salida del compilador Java no es un código ejecutable, sino un bytecode. El bytecode es un conjunto de instrucciones altamente optimizado diseñado para ser ejecutado por una máquina virtual que emula al intérprete Java (Java Virtual Machine JVM), es decir el intérprete de Java es un intérprete de bytecode. El hecho de que un programa Java sea interpretado permite resolver los problemas más importantes asociados a la transferencia de programas de Internet. Traducir un programa Java a bytecode hace que su ejecución en una gran variedad de entornos resulte mucho más sencilla y la razón es que para cada plataforma, solo es necesario implementar el intérprete de Java. Una vez que se dispone del programa de ejecución para un sistema determinado, cualquier programa Java puede ejecutarse en esa plataforma, aunque los detalles del intérprete Java difieran de una plataforma a otra, todos interpretan el mismo bytecode Java.

El hecho de que Java sea interpretado también ayuda a hacerlo seguro. Como la ejecución de cada programa Java esta bajo el control del intérprete Java, este puede contener al programa e impedir que se generen efectos no deseados en el resto del sistema. Las razones fundamentales de la invención de Java fueron la portabilidad y la seguridad, pero existen otros factores que también desempeñaron un papel importante en el modelado de la forma final del lenguaje. Las consideraciones clave fueron resumidas por el equipo de Java en la siguiente lista de términos:

- Simple
- Seguro
- Portable
- Orientado a objetos
- Robusto
- Multihilo
- Arquitectura Neutral
- Interpretado
- Alto rendimiento
- Distribuido
- Dinámico

3.4.1 JavaSever Pages (JSP).

La tecnología JavaServer Pages [10] fue introducida por la empresa Sun Microsystems en estrecha colaboración con Nestcape, buscando extender las funcionalidades del Lenguaje Java a las aplicaciones en Web. Desde su aparición su uso se ha ido incrementando, siendo actualmente una de las tecnologías más usadas y demandadas para aplicaciones web. JSP está basado en el lenguaje Java.

La tecnología JavaServer Pages permite mezclar HTML estándar y estático con un contenido dinámico generado por servlets. Los servlets son programas escritos en lenguaje de programación Java que se alojan en el servidor Web y que se encargan de gestionar adecuadamente el acceso a la base de datos y muchas otras funcionalidades. Se complementan con los Beans que son módulos escritos en Java que pueden ser llamados desde máquinas remotas, lo que transforma al Lenguaje Java en la Web en un lenguaje modular.
Separar el HTML estático del dinámico otorga diversos beneficios sobre los propios servlets. Las ventajas de los JSP se pueden describir en dos hechos: JSP es ampliamente utilizado y ello no lo asocia a un sistema operativo o servidor Web en particular y que JSP le da un total acceso a la tecnología de los servlets y Java en la parte dinámica.
El proceso para generar páginas JSP accesibles en Web es mucho más simple que en los servlets, no necesitan compilación, ni paquetes, ni configuración del classpath, aunque el servidor tendrá que ser configurado para acceder a los archivos de clase del servlet y de JSP, así como al compilador de Java.

Aunque lo que se escriba lucirá como un archivo HTML en lugar de un servlet tras bambalinas , la página JSP se convierte automáticamente en un servlet normal por el motor de JSP, donde el HTML estático tan solo se envía al flujo de salida. Esta traducción se realiza solo la primera vez que se solicita la página. Para asegurarse de que el primer usuario real no tenga ningún retraso momentáneo cuando se traduce la página JSP en un servlet y se compile, el desarrollador podría solicitar la página después de instalarla así es usuario no notaría una pausa generada por el tiempo de compilación del servlet.

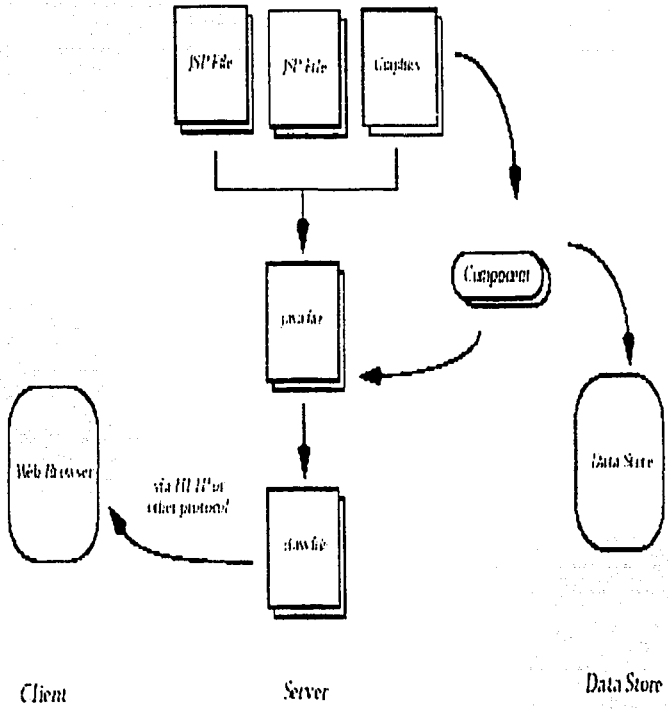


Fig 3.1. Como es compilado un JSP.

Apache-Tomcat.

Tomcat surgió de Sun Microsystems cuando desarrollaba un "Servidor de Páginas" que utilizará Java y posteriormente cedieron el código fuente a la fundación Apache. Tomcat es la implementación de referencia oficial de las especificaciones de Servlet 2.2 y JSP 1.1, puede ser usado como un pequeño servidor autónomo para probar los servlets y las páginas JSP, o puede estar integrado dentro del servidor Web Apache.
Tomcat es un contenedor de Servlets con un entorno de JSP. Un contenedor de Servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario.

Podemos dividir los contenedores de Servlets en:

1.- **Contenedores de Servlets Stand-Alone (Independientes)**
Estos son una parte integral del servidor Web. Este es el caso cuando se usa un servidor basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWebServer . Este es el modo por defecto usando tomcat.

2.-**Contenedor de Servlets dentro-de-Proceso.**
El contenedor de servlets es una combinación de un plugin para el servidor Web y una implementación de contenedor Java. El plugin del servidor Web abre una JVM (Máquina Virtual de Java) dentro del espacio de direcciones del servidor Web y permite que el contenedor Java se ejecute en él. Si una cierta petición debería ejecutar un servlet el plugin toma el control sobre la petición y lo pasa al contendor Java. Un contenedor de este tipo es adecuado para servidores multi-thread de un sólo proceso y proporciona un buen rendimiento pero esta limitado en estabilidad.

3.- **Contenedores de Servlets fuera-de-proceso.**
El contenedor de servlets es una combinación de la página plugin para el servidor Web y una implementación del contenedor Java que se ejecuta en una JVM fuera del servidor Web. El plugin del servidor Web y el JVM del contenedor Java se comunica usando algún mecanismo IPC (normalmente sockets TCP/IP. Si una cierta petición debería ejecutarse en un servlets, el plugin toma el control sobre la petición y lo pasa al contenedor Java. El tiempo de repuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad etc.).

Tomcat puede ser utilizado como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor Web existente (actualmente se soportan los servidores Apache, IIS y Nesteape). Esto significa que siempre que despleguemos Tomcat tendremos que decidor como usarlo y, si seleccionamos las opciones 2 o 3, tambien necesitaremos instalar un adaptador del servidor WEB.

Tomcat Es un programa Java y por lo tanto es posible ejecutarlo desde la línea de comandos, después de configurar varias variables de entorno. Sin embargo, configurar cada variable de entorno y seguir los parámetros de la línea de comandos usado por Tomcat es tedioso y propenso a errores . En su lugar, el equipo de desarrollo de Tomcat proporciona unos pocos scripts para arrancar y parar Tomcat fácilmente.

3.6 Principios Básicos de Java Database Connectivity (JDBC).

Java Database Connectivity (JDBC) [11] es una interfaz de acceso a bases de datos estándar SQL que proporciona un acceso uniforme a una gran variedad de bases de datos relacionales. JDBC también proporciona una base común para la construcción de herramientas y utilidades de alto nivel. Para usar JDBC con un sistema gestor de base de datos en particular, es necesario disponer del driver JDBC apropiado que haga de intermediario entre ésta y JDBC. Dependiendo de varios factores, este driver puede estar escrito en Java puro, o ser una mezcla de Java y métodos nativos JNI (Java Native Interface).

Qué es JDBC

JDBC es el API para la ejecución de sentencias SQL. (Como punto de interés JDBC es una marca registrada y no un acrónimo, no obstante a menudo es conocido como "Java Database Connectivity"). Consiste en un conjunto de clases e interfases escritas en el lenguaje de programación Java. JDBC suministra un API estándar para los desarrolladores y hace posible escribir aplicaciones de base de datos usando un API puro Java.

Usando JDBC es fácil enviar sentencias SQL virtualmente a cualquier sistema de base de datos. En otras palabras, con el API JDBC, no es necesario escribir un programa que acceda a una base de datos Sybase, otro para acceder a Oracle y otro para acceder a Informix. Un único programa escrito usando el API JDBC y el programa será capaz de enviar sentencias SQL a la base de datos apropiada. Y, con una aplicación escrita en el lenguaje de programación Java, tampoco es necesario escribir diferentes aplicaciones para ejecutar en diferentes plataformas. La combinación de Java y JDBC permite al programador escribir una sola vez y ejecutarlo en cualquier entorno.

Java, siendo robusto, seguro, fácil de usar, fácil de entender, y descargable automáticamente desde la red, es un lenguaje base excelente para aplicaciones de base de datos.

JDBC simplemente JDBC hace posible estas tres cosas

- Establece una conexión con la base de datos.
- Envía sentencias SQL
- Procesa los resultados.

JDBC es un API de bajo nivel y una base para API's de alto nivel.

JDBC es una interfaz de bajo nivel, lo que quiere decir que se usa para 'invocar' o llamar a comandos SQL directamente. En esta función trabaja muy bien y es más fácil de usar que otros API's de conexión a bases de datos, pero está diseñado de forma que también sea la base sobre la cual construir interfaces y herramientas de alto nivel. Una interfaz de alto nivel es 'amigable', usa un API más entendible o más conveniente que luego se traduce en la interfaz de bajo nivel tal como JDBC.

CAPÍTULO IV

DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

Capítulo IV DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA.

4.1 Diseño de la base de datos.

Considerando las necesidades que se tenían en los desarrollos de software que se realizan en el Departamento de Aplicaciones Avanzadas de la Dirección General de Cómputo Académico de la UNAM (DGSCA), se pensó en crear una herramienta de coordinación de proyectos que permitiera coordinar los avances en los desarrollos, mejorar la comunicación entre los equipos de trabajo, pero sobre todo que permitiera consultar los avances y los cambios generados en cada uno de los proyectos en desarrollo.

Una vez que se planteó la necesidad de conocer los avances por equipo, se planteó la necesidad de enviar resoluciones, comunicados, peticiones, sugerencias, avisos, reportes de avances, reportes de cambios y asignación de actividades se pensó en crear la base de datos que permitiera almacenar toda esta información.

Una vez determinado el objetivo del proyecto se inició la etapa de análisis de requerimientos, en ella detectamos que era necesario:

- El usuario debe ser identificado por el sistema en todo momento
- Todos los usuarios tendrán los mismos beneficios en creación de equipos, rutas y objetos, pero serán responsables de los elementos que creen.
- Un equipo debería tener un representante y varios integrantes
- Un equipo podrá contar con varias rutas en las cuales se podrían elegir diferentes responsables en cada bloque.
- La creación de bloques podría ser sencilla (una sola persona responsable del bloque) o paralela (con varios responsables en el bloque)
- Para cada ruta o camino se podrán generar objetos, que pasen sobre ella uno a uno a la vez.
- Cada objeto tendrá en el sistema una historia de revisión, que al término del objeto sobre la ruta se almacenará en una carpeta.
- El sistema permitirá la creación de mensajes
- El sistema tendrá como objetivo convertirse en un espacio virtual de trabajo.

El análisis de los requerimientos permitió encontrar todos los objetivos que debería cumplir el sistema, con esta información se inició el análisis conceptual, en el que se intento reflejar toda la información obtenida para la creación del diseño de la base de datos y se usó la herramienta conocida como ERWIN para crear nuestro modelo Entidad-Relación.

4.1.1 Creación de tablas

Además de permitir detectar los objetivos del proyecto, el análisis de requerimientos permitió realizar un diccionario de datos que indica cual es el uso de cada atributo en la base de datos.

A continuación se presenta la información del Diccionario de Datos obtenido para crear la base de datos.

Entidad
Descripción
Atributos
usuario

La tabla usuario se diseño para poder guardar la información de los usuarios en la base de datos.

id_usuario:
El atributo id_usuario es una llave primaria, generada por el sistema para identificar a cada uno de los usuarios que de una manera u otra se registran en el sistema.

nombre
El atributo nombre, guarda en la base de datos el nombre o nombres del usuario que esta creando su registro en el sistema.

apellido_pat:
Este atributo es responsable de guardar en la base de datos el apellido paterno del usuario.

apellido_mat:
Este atributo es responsable de guardar la base de datos del apellido materno del usuario que se registra en el sistema.

email:
Este atributo tiene la responsabilidad de guardar la dirección de correo electrónico del usuario que se esta registrando en el sistema.

telefono:
Este atributo guarda el teléfono que el usuario del sistema indica como el teléfono donde se le puede localizar.

ocupacion:
Este atributo guarda la información de la ocupación del usuario en la base de datos.

login:
Este atributo permite guardar en la base de datos el login con el que el usuario será identificado dentro del sistema.

password:
Este atributo permite identificar la palabra clave con la que el usuario entrara en el sistema.

status:
Este atributo pone un status al usuario al ingresar al sistema en este caso la letra R.

equipo:
Esta tabla contiene los datos de los equipos que genera un usuario, tiene su llave primaria para identificar a cada equipo y una llave foránea que es la del usuario que creó el equipo.

id_equipo:
Este atributo es la llave primaria de la tabla, esta formada por una serie de 10 caracteres. serie de 10 caracteres.

id_usuario:
Esta es una llave foránea, que permite relacionar el equipo, con el usuario que lo creó.

rol_usuario:
Este atributo guarda el rol de registro del usuario, es decir la letra R ya que será representante de los equipos que el cree y la que será representante de los equipos que el cree y la letra I si es agregado al equipo como integrante.

nombre_equipo:
Este atributo guarda el nombre del equipo que decidió el creador del equipo y que ingreso en la forma de creación de equipos.

status:
Este status, es el de equipo y se le coloca una letra A para el creador y una letra I para los integrantes.

integrantes:
Esta tabla es la responsable de guardar la información que relaciona a los bloques de cada ruta, con el equipo y con los usuarios de ese equipo. Esta tabla guarda la información que indica que usuario tiene el objeto en su posesión, que situación tiene este objeto es decir si ya ha sido revisado o no y guarda la información de la decisión que se toma sobre el.

id_equipo:
En esta tabla, este atributo es una llave foránea que permite relacionar los usuarios de un equipo con el bloque en el que tienen participación.

id_usuario:
Este atributo es una llave foránea, que permite identificar a los usuarios elegidos como responsables y relacionarlos con el bloque al que están siendo asignados.

id_bloque:
Esta es una llave foránea que identifica la relación de los integrantes, con los bloques que el representante elige para el envío de objetos.

rol_usuario:
Como lo que se van a incluir el los bloques son integrantes de un equipo, este atributo, guarda como identificar la letra I para los integrantes de los equipos en los bloques.

posesion:
Este atributo, indica a los usuarios si el objeto esta en su posesión o no, por default guarda el valor FALSE.

situacion:
Este atributo indica la situación del objeto y tiene por default la letra S.

calificacion:
Este atributo indica la calificación que se le da al objeto y por default tiene el valor 0.0.

decision:
Este atributo guarda la decisión sobre un objeto, guarda el valor de X.

historia_revision:
Esta tabla tiene como función , guardar la información de orden de revisión necesaria para el avance del objeto sobre la ruta, la fecha en que el participante recibe el objeto y el resultado tomado sobre el, esta tabla permite generar la información del avance del objeto sobre la ruta.

id_equipo:
Este atributo es una llave foránea que relaciona al equipo con la revisión.

id_usuario:
Este atributo es una llave foránea que relaciona al equipo con el usuario.

id_bloque:
Este atributo es una llave foránea que relaciona al equipo con el bloque.

orden:
Es un valor que identifica el avance de la revisión.

fecha_recibido:
Este atributo guarda la fecha en que el usuario recibe el objeto, esta fecha se obtiene del sistema.

resultado:
Este atributo guarda el resultado de la decisión del usuario sobre el objeto que le fue

enviado.

resultado:
Esta tabla guardará la información de las revisiones de los objetos en el sistema.

id_revision:
Este atributo es la llave primaria de la tabla revisión.

revision:
Este atributo indica si la revisión ya se realizó o aún no.

tiempo:
Este atributo es el que guarda el tiempo asignado para la revisión.

fecha_inicio:
Este atributo guarda la fecha de inicio de la revisión.

status:
Este atributo guarda el status de la revisión.

estado:
Este atributo guarda el estado del bloque en el sistema.

bloque:
Esta tabla guarda la información de cada uno de los bloques que se generan en las rutas o caminos.

id_bloque:
Es la llave primaria de la tabla bloque.

orden:
Este atributo, es un valor, que aumenta conforme avanza el objeto sobre la ruta.

anterior:
Este atributo sirve para identificar el tipo de revisión que se hace a los objetos, si es revisión tiene el valor R y el valor de anterior será 0.

tipo_revision:
Este atributo, indica el tipo de revisión del bloque.

mensaje:
Este atributo guarda el mensaje adicional, que el creador de la ruta agrega.

periodo:
Este atributo, guarda el periodo de desarrollo del bloque.

hora_recibido:
Guarda la hora en que el bloque recibe el objeto.

fecha_recibido.
Guarda la fecha en que el bloque recibe el objeto.

resultado_final:
Guarda el resultado final del bloque, por default es FALSE.

porcentaje_final:
Indica el porcentaje de avance del objeto después de salir del bloque.

prorroga:
Indica el tiempo prorroga que se le dará a los participantes del bloque para revisar el objeto.

status:
Este atributo indica el estatus del bloque y será por default S.

posesion:
Es el estatus del objeto en el bloque, por default este atributo es FALSE.

bloques_por_camino:
Esta tabla es una tabla auxiliar la cual nos permite relacionar los bloques generados, con las rutas o caminos a los que pertenecen.

id_camino:
Este atributo es una llave foránea que identifica el camino al que pertenece cada bloque.

id_bloque:
Este atributo es una llave foránea que identifica los bloques a que camino pertenecen.

camino:
Esta tabla guarda la información de las rutas de los equipos.

id_camino:
Es la llave primaria de la tabla camino.

nombre_camino:
Este atributo, es el nombre que se le pone a la ruta cuando se crea.

objetivo:
Este atributo es parte de la información que se solicita cuando se crea una ruta, sirve para establecer el objetivo de la ruta.

descripción:

Este atributo permite describir a grandes rasgos para que ha sido creada la ruta.

etapas:
Este atributo, es el que define la cantidad de etapas en las que se dividirá el proyecto.

bloques:
Este atributo, indica en cuantas bloques estará dividida la ruta.

fecha_creacion:
Este atributo es la fecha de creación de la ruta.

objeto_activo:
Este atributo indica si el camino tiene objetos activos, por default y cuando se crea la ruta, su valor será FALSE.

diagrama:
Este atributo, guarda el diagrama que se genera para cada ruta.

objetos_por_camino:
Esta tabla es una tabla auxiliar que relaciona la tabla camino con la tabla objeto.

id_camino:
Esta es una llave foránea que relaciona el camino con los objetos que se desplazan en el.

id_objeto:
Esta es una llave foránea que relaciona los objetos con el camino por el que avanzan para su revisión.

objeto:
Esta tabla guarda la información de la creación de objetos.

id_objeto:
Esta es la llave primaria de la tabla objeto.

nombre_objeto:
Este atributo guarda el nombre que se coloca al objeto cuando se crea.

descripcion:
Esta es la descripción del objeto.

fecha_creacion:
Este atributo guarda la fecha de creación del objeto.

tipo_objeto:
Este atributo guarda la información del tipo de objeto que se desea crear.

direccion_objeto:
Este atributo guarda la dirección de la máquina o el URL de donde proviene el objeto que se desea crear.

fecha_lanzamiento:
Este atributo guarda la fecha con la que se desea lanzar el objeto a la ruta.

dirección_historial:
Este atributo guarda una dirección temporal donde se guardan los documentos.

tiempo_espera:
Es el tiempo que podrá esperar el objeto , para realizar su revisión sin solicitar una prorroga.

status:
Es el estatus del objeto, en el momento de su creación.

historial_objeto:
Esta es una tabla auxiliar que permite relacionar las tablas objeto con la tabla avance_historial, objeto con la tabla avance_historial.

id_objeto:
Es una llave foránea que relaciona el avance del historial con el objeto.

id_avance:
Es una llave foránea que relaciona el objeto con su avance por la ruta y crea un historial.

status_historial:
Es el estatus con el que inicia el historial de cada uno de los objetos.

avance_historial:
Esta tabla permite controlar el avance de un objeto en una ruta, conocer el objetivo del avance y la dirección a la que va avanzar.

id_avance:
Este atributo, es la llave que identifica a cada objeto que ingresa a la tabla avance_historial.

fecha_avance
Esta es la fecha con la que será enviado un objeto sobre la ruta.

titulo_avance
Este campo contiene la información del bloque en el que se encuentra cada objeto.

descripcion:
Esta es la descripción del objeto

objetivo_avance:
Este atributo contiene el objetivo por el que fue creado el objeto y se les envía a cada participante que es responsable de revisar el objeto.

direccion_avance:
Este atributo contiene el orden con el que avanzará el objeto por la ruta.

Autor:
Este campo permite tener el control de quien agrega comentarios sobre cada objeto.

tipo material:
Este campo contiene el tipo de material, que se desea agregar a un objeto, que esta en proceso de revisión.

actividades:
Esta tabla tiene como fin asignar actividades a los participantes del equipo.

id_actividad:
Este atributo es la llave primaria de la tabla actividad, cada actividad tendrá una llave que la identifique.

id_usuario:
Esta campo se llenará con el id_usuario de la persona a la que se le esta asignando la actividad.

id_agenda:
Esta campo permitirá enviar a la tabla agenda un mensaje por cada actividad que se le asigne al participante, por cada actividad, existirá un mensaje.

titulo_actividad:
Este es el titulo con el cual se envía la actividad al participante.

horario:
Este será el tiempo con el que cuanta el participante para realizar la actividad.

actividad:
Este será el cuerpo de la actividad, en este campo se guardará la descripción que especifique en que consiste la actividad que se le esta asignando a cada participante.

Fecha:
Esta es la fecha con la que se le esta asignado la actividad a cada participante

tipo_mensaje:
Este campo guardará la información del tipo de mensaje que se le esta enviando al

participante.

mensajes:
Esta tabla contiene la información de los mensajes que envían por participantes del sistema de persona a persona o grupales.

autor:
Este campo contiene el id_usuario de la persona que esta enviando el mensaje.

destino:
Este campo contiene el id_usuario de la persona a la que se le va enviar el mensaje.

id_mensaje:
Este atributo es la llave primaria de los mensajes, cada mensaje tiene una llave primaria que lo identifica.

titulo_mensaje:
Este campo es el titulo del mensaje, que recibirá el usuario a quien se le envíe el mensaje.

mensaje:
Este campo corresponde al cuerpo del mensaje.

tipo_mensaje:
Este campo indica que tipo de mensaje, se le envío al participante, si es grupal, personal o del representante de un equipo.

fecha_mensaje:
Este campo contiene la fecha con la que se le envío el mensaje al participante.

status:
Este es el status del mensaje, cuando el mensaje ha sido revisado el estatus cambia.

agenda:
Esta tabla contiene la información del número de mensajes que tiene cada participante, esta tabla permite controlar los pendientes en revisión del participante.

id_usuario:
Este campo permite que cada participante, tenga una agenda, pues relaciona a la tabla usuario con la tabla agenda.

id_agenda:
Esta campo es el identificador que se genera para cada agenda.

mensajes:
Este atributo indica el número de mensajes que tiene relacionada cada uno de los

participantes de los equipos, los cuales son enviados por asignación de tareas.

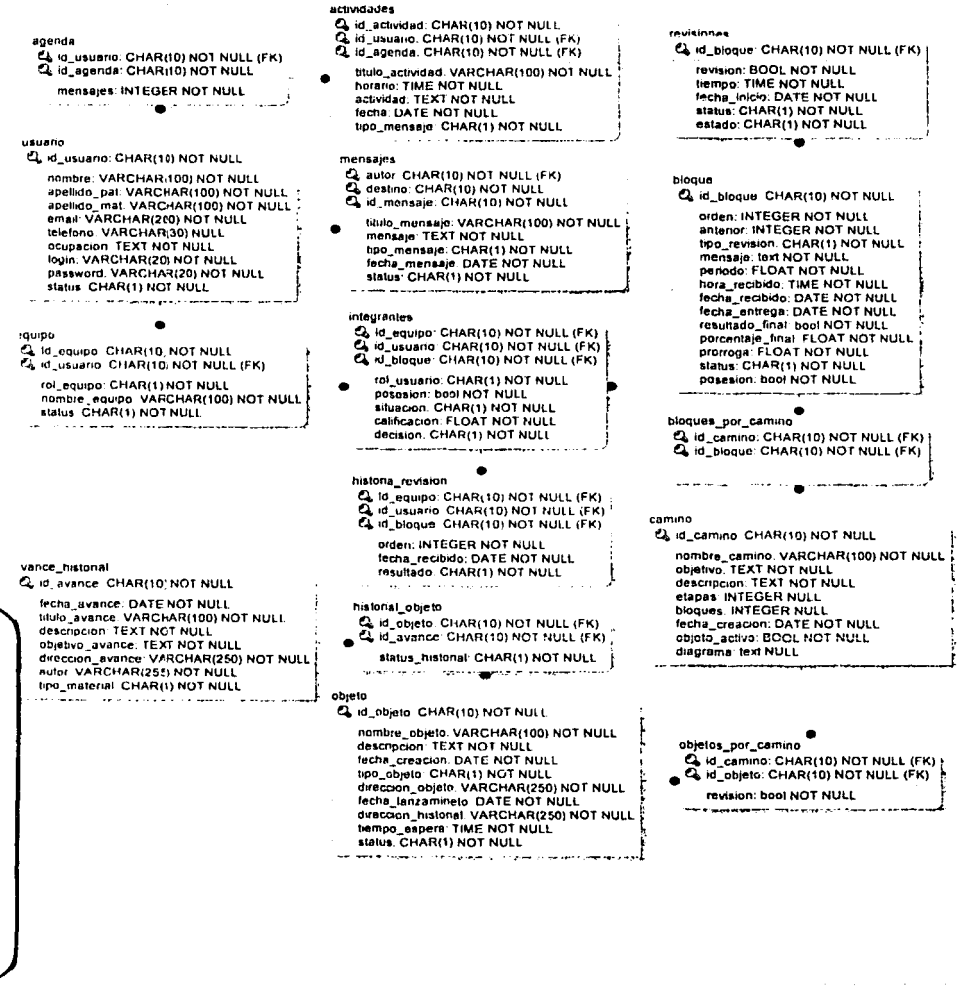
Con la ayuda de la herramienta CASE ERWIN y el Diccionario de Datos presentado anteriormente, se plantearon las entidades necesarias para el funcionamiento del sistema, las relaciones entre estas entidades, y se analizaron los atributos y el tipo de dato que sería asignado a cada uno.

El ambiente de este software permitió analizar las entidades, de una manera clara, ya que permite visualizar todos los aspectos necesarios para la base de datos, analizar las relaciones de las entidades, se buscó no guardar en la base de datos vectores, que los atributos no se repitieran en diversas tablas, es decir buscamos optimizar el funcionamiento en el modelo de la base de datos.

4.1.2 Modelo Entidad-Relación.

Dentro del Software ERWIN se creó el modelo de la base de datos que consta de 15 tablas que permiten al sistema mantener el seguimiento de las actividades de los participantes en el desarrollo.

El modelo obtenido con ayuda de ERWIN es el siguiente:



A cada atributo de la tabla se asignó un tipo de dato, se indicó qué atributos eran llaves primarias y cuales llaves foráneas, con esta información ya plasmada en el modelo puede decirse que termina la etapa de diseño lógico a continuación se seleccionó la opción de diseño físico dentro de ERWIN y se indica que tipo de manejador de bases de datos se va a utilizar (en nuestro caso PostgresSQL), se colocan las opciones para que la generación del script contemple que son necesarias llaves primarias, llaves foráneas y el uso de triggers si es necesario, además de otras opciones.

Terminadas estas actividades se solicita ver el script de código generado y con el se podrá crear la base de datos en el momento que sea necesario.

El script que se obtiene es el siguiente:

```
CREATE TABLE bloque (
    id_bloque          char(10) NOT NULL,
    orden              int NULL,
    anterior           int NULL,
    tipo_revision      char(1) NULL,
    mensaje            text NULL,
    periodo            float NULL,
    hora_recibido      TIME NULL,
    fecha_recibido     datetime NULL,
    fecha_entrega      datetime NULL,
    resultado_final    bool NULL,
    porcentaje_final   float NULL,
    prorroga           float NULL,
    status             char(1) NULL,
    posestor           bool NULL,
    PRIMARY KEY (id_bloque)
);

CREATE TABLE usuario (
    id_usuario         char(10) NOT NULL,
    apellido_pat       varchar(100) NULL,
    nombre             varchar(100) NULL,
    apellido_mat       varchar(100) NULL,
    email              varchar(200) NULL,
    telefono           varchar(30) NULL,
    ocupacion          TEXT NULL,
    login              varchar(20) NULL,
    password           varchar(20) NULL,
    status             char(1) NULL,
    PRIMARY KEY (id_usuario)
);


CREATE TABLE equipo (
    id_equipo          char(18) NOT NULL,
    id_usuario         char(10) NOT NULL,
    tol_equipo         char(1) NULL,
    nombre_equipo      varchar(100) NULL,
    status             char(1) NULL,
```

```
    PRIMARY KEY (Id_equipo, id_usuario),
    FOREIGN KEY (id_usuario)
                REFERENCES usuario
);


CREATE INDEX XIF58equipo ON equipo
(
    id_usuario
);



CREATE TABLE integrantes (
    Id_equipo       char(18) NOT NULL,
    id_usuario      char(10) NOT NULL,
    id_bloque       char(10) NOT NULL,
    rol_usuario     char(1) NULL,
    posesion        bool NULL,
    situacion       char(1) NULL,
    calificacion    float NULL,
    decision        char(1) NULL,
    PRIMARY KEY (Id_equipo, id_usuario, id_bloque),
    FOREIGN KEY (id_bloque)
                REFERENCES bloque,
    FOREIGN KEY (Id_equipo, id_usuario)
                REFERENCES equipo
);


CREATE INDEX XIF75integrantes ON integrantes
(
    Id_equipo,
    id_usuario
);


CREATE INDEX XIF78integrantes ON integrantes
(
    id_bloque
);



CREATE TABLE agenda (
    id_usuario      char(10) NOT NULL,
    id_agenda       char(18) NOT NULL,
    mensajes        int NULL,
    PRIMARY KEY (id_usuario, id_agenda),
    FOREIGN KEY (id_usuario)
                REFERENCES usuario
);
```

```
CREATE INDEX XIF54agenda ON agenda
(
    id_usuario
) ;


CREATE TABLE actividades (
    id_actividad        char(18) NOT NULL,
    titulo_actividad    varchar(100) NULL,
    id_usuario          char(10) NOT NULL,
    horario             TIME NULL,
    id_agenda           char(18) NOT NULL,
    actividad           TEXT NULL,
    fecha               datetime NULL,
    tipo_mensaje        char(1) NULL,
    PRIMARY KEY (id_actividad, id_usuario, id_agenda),
    FOREIGN KEY (id_usuario, id_agenda)
                    REFERENCES agenda
) ;


CREATE INDEX XIF72actividades ON actividades
(
    id_usuario,
    id_agenda
) ;


CREATE TABLE avance_historial (
    id_avance           char(18) NOT NULL,
    fecha_avance        datetime NULL,
    autor               varchar(255) NULL,
    titulo_avance       varchar(100) NULL,
    tipo_material       char(1) NULL,
    descripcion         TEXT NULL,
    objetivo_avance     TEXT NULL,
    direccion_avance    varchar(250) NULL,
    PRIMARY KEY (id_avance)
) ;


CREATE TABLE objeto (
    id_objeto           char(18) NOT NULL,
    nombre_objeto       varchar(100) NULL,
    descripcion         TEXT NULL,
    fecha_creacion      datetime NULL,
    tipo_objeto         char(1) NULL,
    fecha_lanzamineto   datetime NULL,
    direccion_objeto    varchar(250) NULL,
    status              char(1) NULL,
    direccion_historial varchar(250) NULL,
    tiempo_espera       TIME NULL,
    PRIMARY KEY (id_objeto)
).
```

```
CREATE TABLE historial_objeto (
    id_objeto          char(18) NOT NULL,
    id_avance          char(18) NOT NULL,
    status_historial   char(1) NULL,
    PRIMARY KEY (id_objeto, id_avance),
    FOREIGN KEY (id_avance)
                REFERENCES avance_historial,
    FOREIGN KEY (id_objeto)
                REFERENCES objeto
) ;

CREATE INDEX XIF81historial_objeto ON historial_objeto
(
    id_objeto
) ;

 CREATE INDEX XIF82historial_objeto ON historial_objeto
(
    id_avance
) ;

CREATE TABLE camino (
    id_camino          char(18) NOT NULL,
    nombre_camino      varchar(100) NULL,
    objetivo           TEXT NULL,
    descripcion        TEXT NULL,
    etapas             int NULL,
    bloques            int NULL,
    fecha_creacion     datetime NULL,
    objeto_activo      BOOL NULL,
    diagrama           text NULL,
    PRIMARY KEY (id_camino)
) ;

CREATE TABLE bloques_por_camino (
    id_camino          char(18) NOT NULL,
    id_bloque          char(10) NOT NULL,
    PRIMARY KEY (id_camino, id_bloque),
    FOREIGN KEY (id_bloque)
                REFERENCES bloque,
    FOREIGN KEY (id_camino)
                REFERENCES camino
) ;

CREATE INDEX XIF73bloques_por_camino ON bloques_por_camino
(
    id_camino
) ;


CREATE INDEX XIF74bloques_por_camino ON bloques_por_camino
(
    id_bloque
) ;
```

```
CREATE TABLE mensajes (
    autor           char(10) NOT NULL,
    destino         char(18) NOT NULL,
    id_mensaje      char(18) NOT NULL,
    titulo_mensaje  varchar(100) NULL,
    mensaje         TEXT NULL,
    tipo_mensaje    char(1) NULL,
    fecha_mensaje   datetime NULL,
    status          char(1) NULL,
    PRIMARY KEY (autor, destino, id_mensaje),
    FOREIGN KEY (autor)
                REFERENCES usuario
);

CREATE INDEX XIF77mensajes ON mensajes
(
    autor
);

CREATE TABLE objetos_por_camino (
    id_camino       char(18) NOT NULL,
    id_objeto       char(18) NOT NULL,
    PRIMARY KEY (id_camino, id_objeto),
    FOREIGN KEY (id_objeto)
                REFERENCES objeto,
    FOREIGN KEY (id_camino)
                REFERENCES camino
);


CREATE INDEX XIF79objetos_por_camino ON objetos_por_camino
(
    id_camino
);


CREATE INDEX XIF80objetos_por_camino ON objetos_por_camino
(
    id_objeto
);

CREATE TABLE revisiones (
    id_bloque       char(10) NOT NULL,
    revision        BOOL NULL,
    tiempo          TIME NULL,
    fecha_inicio    datetime NULL,
    status          char(1) NULL,
    estado          char(1) NULL,
    PRIMARY KEY (id_bloque),
    FOREIGN KEY (id_bloque)
                REFERENCES bloque
);
CREATE TABLE historia_revision (
    id_equipo       char(18) NOT NULL,
```

```
    id_usuario          char(10) NOT NULL,
    id_bloque           char(10) NOT NULL,
    orden               int NULL,
    fecha_recibido      datetime NULL,
    resultado           char(1) NULL,
    PRIMARY KEY (id_equipo, id_usuario, id_bloque),
    FOREIGN KEY (id_equipo, id_usuario, id_bloque)
                REFERENCES integrantes
);
```

Hasta aquí podemos decir que ha terminado el proceso de modelado de la base, y con la ayuda del script generado, realizamos la creación de las tablas necesarias después de haber analizado los requerimientos del sistema, el siguiente paso es la creación de la base de datos.

Como ya habíamos mencionado, para este proyecto se decidió el uso de PostgresSQL como manejador de bases de datos, su instalación en el servidor de bases de datos necesitará la creación de una cuenta de root con login y password y cada base que se cree dentro del sistema requerirá ser accesada por un usuario específico.

Para el manejo de la base de datos, se debe crear un usuario con login y password que tenga todos los permisos sobre la base, para que el sistema le permita a ese usuario interactuar con la base como le sea necesario. Con la ayuda del script obtenido en ERWIN se creó la base de datos y se inicio la programación del sistema.

4.2 Creación del sistema.

Con la base de datos terminada , se inicio el desarrollo del sistema, y lo primero fue crear las librerias.
Las librerias son aquellos programas que permiten conectarse a la base de datos, verificar sesiones, enviar mensajes, crear las llaves de identificación es decir la parte funcional que permite que la base de datos interactúe con el sistema.

Las librerías se programaron con el lenguaje de programación Java, del cual ya se ha explicado los beneficios que ofrece en el capítulo III .

Lo primero que se programó fue el programa de conexión a la base de datos, en el cual se especifico tanto el nombre de la base de datos, como el login y password del usuario con el cual hará las conexiones a la base.

Antes de especificar el usuario, se indico el tipo de base de datos se que usaría y para ello fue necesario utilizar un drive que coincidiera con la base de datos PostgresSQL. Hemos mencionado los beneficios del JDBC en el capítulo III.

Se crearon los métodos para generar las llaves, y para verificar la secuencia de la tablas, se programó también el script que permite identificar y mantener la sesión de los usuarios registrados en el sistema (se llama mantener la sesión a la capacidad de un servidor HTTP y de sus programas asociados para reconocer que una determinada solicitud de un servicio pertenece a un usuario que ya había sido identificado y autorizado)

Todas las librerías se colocaron dentro de un paquete, para de esta manera usar algunos de los beneficios que proporciona el lenguaje de programación Java. Una vez lograda la conexión a la base de datos y los metodos de creación de llaves se comenzó a realizar la programación y la creación de los jsp para hacer funcional cada una de las pantallas necesarias para hacer funcional el sistema.

CAPÍTULO V

PRESENTACIÓN DEL SISTEMA

Capítulo V. PRESENTACIÓN DEL SISTEMA.

En este capítulo explicaremos el funcionamiento del sistema. Para comenzar se presenta a continuación la terminología que se utilizará en el desarrollo de esta tesis.

5.1 Terminología:

1) Equipo de trabajo:

Un equipo de trabajo está integrado por diferentes participantes del sistema, elegidos por un representante para coordinar una o más etapas de la ruta, su elección se realiza por medio de un análisis de desempeño en proyectos anteriores o experiencia pasada.

2) Ruta:

Una ruta, es el camino que construye en el sistema el representante del equipo, la cual será la trayectoria que seguirá un objeto que se desplace sobre ésta.

Las rutas se dividen en bloques, el número de bloques lo designará el representante del equipo. Cada bloque podrá tener uno o más responsables.

La idea de generar etapas es representar cada una de ellas como una actividad que realizará una tarea, esto con el fin de llevar un control sobre los avances del desarrollo.

La creación de bloques, consiste en asignar actividades en una etapa, el bloque podrá tener varios responsables.

Las rutas pueden ser en de dos tipos:

Serie: Es aquella en la que los documentos pasarán uno a uno a cada responsable de cada una de las etapas, la revisión o decisión sobre el documento se realizará solo cuando haya terminado el turno de la persona asignada a revisión antes que él.

En este tipo de ruta, se considera que las etapas, requieren de la realización de la etapa anterior para comenzar a desarrollarse, es decir las actividades no se pueden realizar al mismo tiempo.

Paralela: Es aquella donde el documento llegará a varios responsables al mismo tiempo, esto, debido a que puedan existir dos etapas de revisión al mismo tiempo, es decir que las actividades se puedan realizar iniciando en un momento similar debido a que las revisiones son independientes una de la otra, pero la conclusión de todas proporcionará un solo resultado

El documento no saldrá del bloque, es decir, no pasará a la siguiente etapa si no ha sido revisada por todos los responsables del bloque.

3) Objeto:

Un objeto, es un material (documento, artículo, reporte, programa, proyecto, etc), el cual se desea compartir, desarrollar, criticar, aprobar, mediante el análisis que realizarán los integrantes del grupo de trabajo haciendo uso de una ruta que lo guiará para permitir que llegue a cada participante que realizará sobre él la tarea asignada por el representante del equipo.

5.2 Sistema Manejo de Actividades

El sistema Manejo de Actividades, es un sistema que busca ser un punto de participación entre los desarrolladores del software, su fin es controlar los procesos de comunicación entre los integrantes del equipo.

Para este sistema existen dos tipos de usuario, el representante del equipo y los integrantes del equipo, ambos estarán interactuando entre sí por medio de un sistema de colaboración que permite la asignación de tareas de revisión de documentos.

El representante del equipo será el encargado de crear, designar y vigilar el desarrollo, manejo y seguimiento de los documentos enviados a una ruta (objetos en curso de revisión). Durante todo el recorrido del objeto por la ruta, el representante será informado del avance del objeto y será generado por cada objeto un historial a lo largo del recorrido que el representante definió al crear la ruta.

Dentro del Sistema el orden de creación es muy importante, por ello es necesario que las participantes conozcan este orden.

Orden de Creación:

- Crear Equipo
  - Crear Rutas
    - Crear Objetos

Nótese que :

- Cada usuario podrá crear tantos equipos como sean necesarios, sobre un equipo podrán existir muchas rutas, cada ruta tendrá uno y solo un objeto sobre ella , es decir no podrá haber más de un objeto el ella, el orden de creación de elementos en el sistema no podrá ser alterado, y el creador de un equipo, se convertirá en representante de este y de todos los elementos que cree sobre el.

5.2.1 Tipos de usuarios para el sistema:

- El representante del Equipo
- El usuario

El representante es el responsable de crear equipos de trabajo, rutas y enviar objetos sobre éstas. Durante el recorrido del objeto, el representante será informado de sus avances y del historial que vaya recabando durante la ruta del objeto a lo largo del recorrido que designo el representante.

El usuario: Es el responsable de revisar los objetos, agregar comentarios, anexar documentos y de ésta manera realizar las tareas en las cuales fue incluido por algún representante de equipo.

Cuando un representante de equipo genera la ruta de un documento, puede elegir entre dos modos de operación para cada uno de los usuarios.

- **Modo condicional:** En esta modalidad, la siguiente etapa que recorrerá el objeto va a depender del resultado que emita el usuario asociado a la etapa, sus resultados son si o no, en ambos casos el representante de equipo debe definir a que etapa continuará el objeto, ya sea anterior o posterior a la etapa actual.

- **Modo opinión:** En esta modalidad no existe el condicional, es decir el objeto únicamente es revisado o modificado, el usuario puede anexar o corregir información, materiales, objetos en general, este modo de operación es útil cuando no se requiere de una decisión para continuar el recorrido, sino que el usuario solo "opina" sobre el objeto.

Durante el recorrido del objeto por la ruta, el representante del equipo podrá solicitar las veces que considere necesarias, un reporte del estado actual del objeto, de su información anexa y de las modificaciones que se han sugerido, también podrá solicitar información acerca del grupo de trabajo.

Por cada una de las etapas de desarrollo, el sistema registrará información acerca del objeto entre la que se encuentra:

- Periodos de Revisión
- Atrasos
- Información Anexa
- Resultados Obtenidos

Esto con la finalidad de generar el reporte que muestre todo el proceso de revisión por el cual pasó el objeto. A esto le llamaremos historial del objeto, el historial del objeto se construye al final de cada una de las etapas y se libera cuando el objeto ha terminado su recorrido, en el historial se muestran los resultados obtenidos.

Los mensajes que puede generar el sistema son los relacionados al envio de objetos, los relacionados a periodos de tiempo de revisión , los mensajes generados por los integrantes de un equipo para un compañero de desarrollo o para su responsable de equipo y destinados a uno o varios participantes de las rutas.

Cuando un objeto llega a un usuario, el usuario será notificado en su página personal .

Cuando esté por cumplirse el periodo de tiempo en el cual el usuario debe revisar su objeto y no ha sido revisado, el sistema deberá notificarle al usuario que está por terminar su periodo, también se tendrá la posibilidad de generar una prórroga, siempre y cuando el representante del equipo lo autorice, de lo contrario el objeto pasará a su etapa siguiente, la cual será asignada por el representante del equipo al generar la ruta.

Para poder acceder al sistema, el usuario debe estar registrado en el, para registarse encontrará un botón en la página principal que lo llevará a la forma de registro, la cual contiene los campos de información que requiere el sistema para poder realizar el alta de cada uno de los participantes del sistema.

A continuación se muestra la imagen de la pantalla de entrada al sistema o pantalla principal:

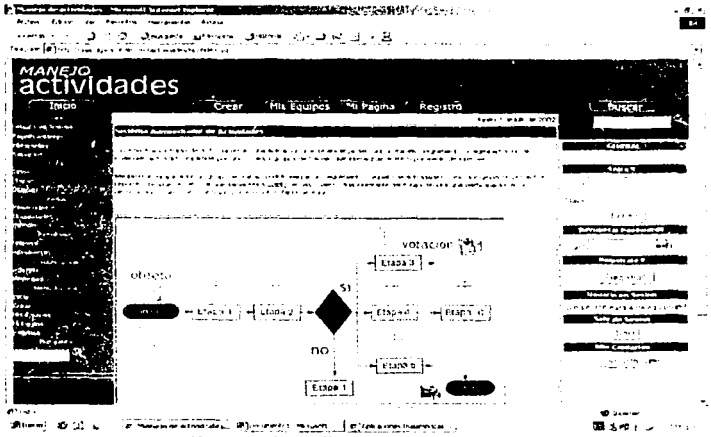


Fig. 5.1. Pantalla principal del sistema

Esta es la pantalla principal del sistema, esta página contiene información sobre los conceptos necesarios para entender a manejar esta herramienta de control de proyectos, al ser un punto de participación es necesario que todos los participantes sepan y logren obtener de él los mejores beneficios, por ello se ha colocado en esta página los conceptos básicos para que las personas que ingresen o solo lo visiten conozcan de una manera rápida como moverse dentro de esta herramienta.

La información del lado izquierdo está colocada por categorías y cada categoría tiene conceptos relacionados al sistema, además existe un menú auxiliar que permitirá acceder a los servicios si por alguna razón no funcionaran los botones del menú principal.

Para permitir el buen funcionamiento del sistema se programo dentro de una clase un método que permite generar llaves.

Cada equipo, ruta, objeto, mensaje y usuario en el sistema será identificado por una llave, el seguimiento de las actividades del usuario en este sistema se hará con ayuda de la llave.

El sistema tiene la capacidad de generar 10^{20} llaves que serán creadas con la combinación de 10 letras que cambiaran alfabéticamente conforme aumenten los elementos del sistema.

Las llaves comenzaran con la cadena AAAAAAAAAA y se modificarán sucesivamente (la siguiente llave será AAAAAAAAAB), el usuario estará siempre relacionado con la llave que lo identifica a él y a todo lo que genere dentro del sistema en la base de datos.

Por medio de la llave se identificará la sesión del usuario dentro del sistema.

A continuación se presenta la forma de inscripción que el usuario debe llenar para darse de alta en el sistema.

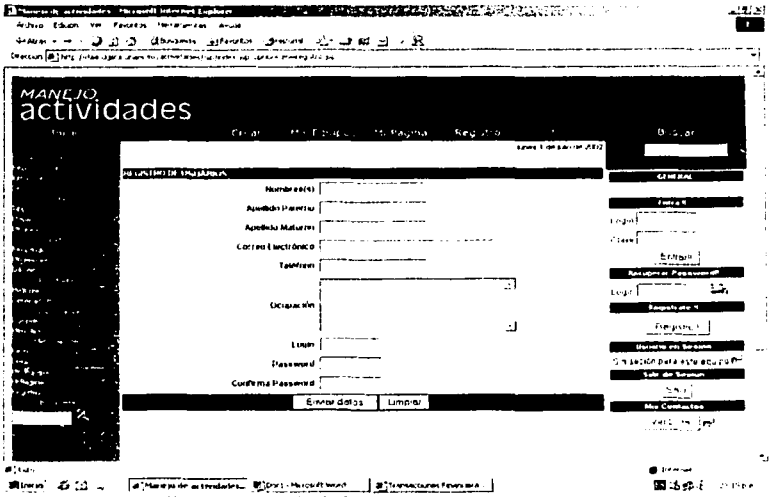


Fig. 5.2. Pantalla de la forma de inscripción al sistema

Este es el formulario que el usuario deberá llenar para poder participar en el sistema.

Cuando el usuario se ha registrado en el sistema e ingrese por primera vez se creará su página personal, a la cual podrá ingresar por medio del login y el password que indicó en la forma al realizar su registro.

Por medio una sesión el sistema mantendrá "vigilada" la actividad del participante dentro del sistema.

5.2.2 Página personal.

La página personal es un espacio personal para cada participante del sistema, en el cual encontrará, infomación relacionada con sus actividades en cada desarrollo.

Cuando el usuario entra a su página personal podrá observar que la página se encuentra dividida en tres secciones :

- Buzón
- Tareas
- Materiales

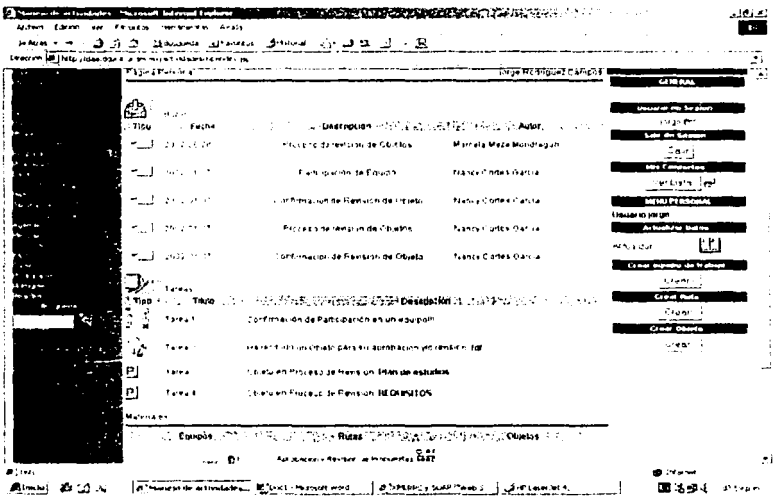


Fig. 5.3. Pantalla que presenta la página personal de un representante de equipo.

5.2.2.1 Sección buzón

El buzón es el encargado de presentar al usuario los mensajes generados por el sistema y los mensajes que le pueden enviar sus compañeros de desarrollo.

La sección de buzón esta organizada de la siguiente manera:

- Tipo: En donde se coloca un icono indicando que hay un mensaje para revisar.
- Fecha: Donde se obtiene la fecha en que le fue enviado el mensaje.
- Descripción: Que contiene una breve descripción del tipo de mensaje que fue recibido.
- Autor: Que indica el nombre de la persona que le envió el mensaje.

El icono de mensaje tiene dos funciones:

- Indicar al participante que tiene un mensaje nuevo y llevarlo a examinar el contenido completo de dicho mensaje.

- Borrar el mensaje de su página personal

Al dar clic en el mensaje, se desplegará la pantalla que tiene el mensaje completo que se le envió al participante.

La pantalla es la siguiente:

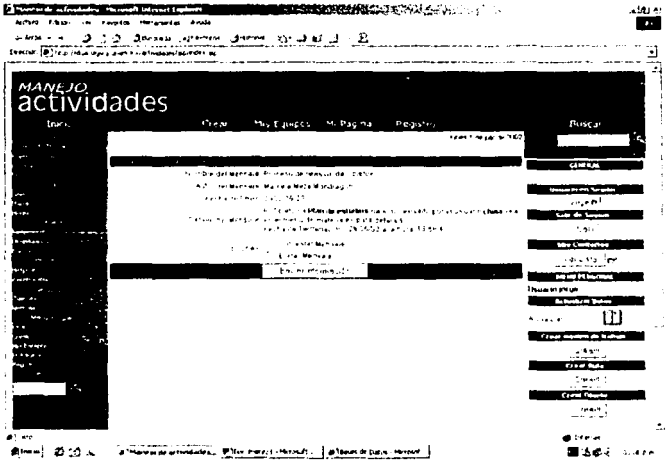


Fig. 5.4. Pantalla que presenta la forma para contestar o borrar mensajes.

Una vez que el participante ha revisado el mensaje puede tomar una de las dos decisiones antes mencionadas, contestarlo o borrarlo, al elegir la casilla de contestar se le presentará la siguiente pantalla en la que podrá crear un mensaje nuevo, que le llegará como contestación a la persona que le envió el mensaje.

Esta es la pantalla que se le presentará al usuario si su opción fue responder al mensaje enviado por un participante del sistema, la pantalla ya tiene identificado el nombre de la persona a la que se le va a responder el mensaje, por ello solo es necesario agregarle un nombre al mensaje que será el subject que le llegue en su página personal a la persona que lo reciba y llenar el cuerpo del mensaje, al cual podrá acceder con solo dar clic al icono en su buzon.

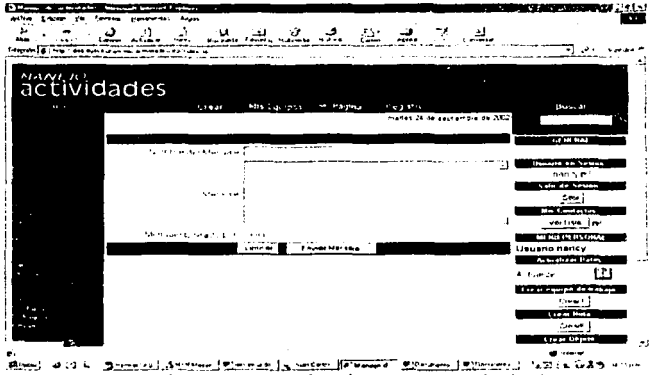


Fig. 5.5. Pantalla que permite el envío de mensajes desde el sistema.

Si su opción es borrar el mensaje entonces se le presentará un mensaje indicándole que su mensaje ha sido borrado satisfactoriamente. Una vez hecho esto el mensaje desaparecerá de su página personal.

5.2.2.2 Sección tareas

La sección de Tareas está organizada de la siguiente manera:

- Tipo: Indica el tipo de tarea que va a realizar el participante con iconos diferentes.
- Título: Enumera las tareas que tiene asignadas un usuario.
- Descripción: Presenta una breve descripción de la tarea.

Esta sección tiene como fin presentar al participante las tareas que le han sido asignadas por el representante de un equipo, esta sección enumera las tareas de cada participante en su página personal , por cada revisión de objeto que le sea asignada, contará con una tarea.

Conforma avanzan los objetos sobre las rutas, se va realizando la participación de los asignados a cada tarea . Esta sección le permitirá saber cuantas tareas tiene asignadas, en que momento de la revisión se encuentran, quien es el responsable de su revisión en el momento y si el ya ha realizado su trabajo sobre ese objeto. Conforme vaya terminando las tareas los iconos de esta sección se modificarán indicando cada uno el estado y turno de revisión.

Los cambios en los iconos se verán con mayor lujo de detalle en la página personal del representante del equipo ya que el será el que lleve el control del proceso de revisión del objeto hasta el fin de la trayectoria de la ruta.

Los usuarios participantes podrán ver un icono antes de la revisión y otro diferente después de ella.
Cuando la revisión se ha terminado por parte del usuario, le llegará un mensaje al responsable del equipo indicándole que el usuario en turno ha revisado o tomado una decisión sobre el objeto y que este a llegado al siguiente responsable.

Si el usuario no ha hecho la revisión el icono que le indica la llegada de un objeto, cambiará indicándole con un pequeño reloj que el tiempo para la revisión está acabando.

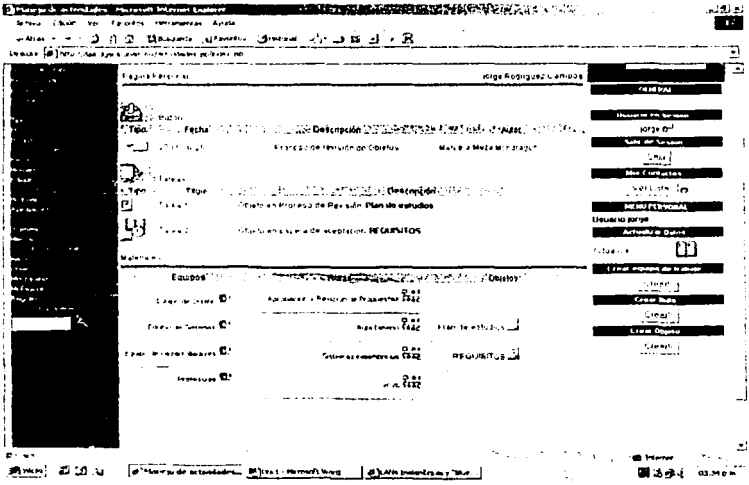


Fig. 5.6. Pantalla que presenta la página personal.

Todos estos cambios en las revisiones los podrá ver el participante para estar bien enterado de los avances del proyecto, pero sobre todo son de gran importancia para el representante del equipo ya que al ser el coordinador del proyecto y al haber realizado la planeación de este y haber asignado una cantidad de recursos humanos considerados, necesitará observar si se están logrando los objetivos de toda esta planeación, si hay avances, conocer quién está participando y de que forma, con qué problemas se están encontrando los participantes del proyecto y además observar si hay retrasos en el desarrollo. Esto le permitirá tomar decisiones sobre como debe avanzar el desarrollo.

En esta sección recibirá el representante del equipo respuestas a las tareas asignadas a los participantes, entre los que se pueden encontrar sus confirmaciones a participar dentro de uno de los equipos que el coordina.

El representante de un equipo también puede participar en otros equipos como responsable de alguna etapa por ello también podrán llegar a su página personal objetos para su revisión o aprobación.

5.2.2.3 Sección materiales

La sección de materiales es la responsable de presentar al participante en su página personal, los equipos en los que esta integrado, las rutas en las cuales tiene asignadas tareas y presentar los objetos relacionados a las rutas en las que participa.

La sección de Materiales esta organizada de la siguiente manera:

- Equipos: Estos equipos son en los que se encuentra participando el dueño de la página personal
- Rutas: Son aquellas en las que el usuario está asignado a ser responsable de una tarea o actividad.
- Objetos: Son los objetos que se encuentran en ese momento siguiendo su camino por la ruta a la cual esta asignado el participante. Cuando el objeto termina su camino se eliminará de la página personal.

Los iconos en esta sección son muy importantes ya que permiten que los usuarios revisen información sobre los integrantes del equipo en el que participan, en cualquier momento podrá revisar el estado de cada una de las rutas, podrá confirmar que esta incluido en un equipo para su participación o la de alguien más y podrá revisar el historial del objeto en su camino por una ruta.

El icono de equipo permite observar una lista de los participantes, pero además permite enviar mensajes grupales o personales a los participantes.

Los mensajes enviados de esta manera llegarán a la página personal de cada uno de los usuarios del sistema.

También permitirá agregar nuevos usuarios al equipo, buscándolos de dos maneras:

- Buscando usuarios registrados en el sistema
- Agregando nuevos usuarios al sistema.

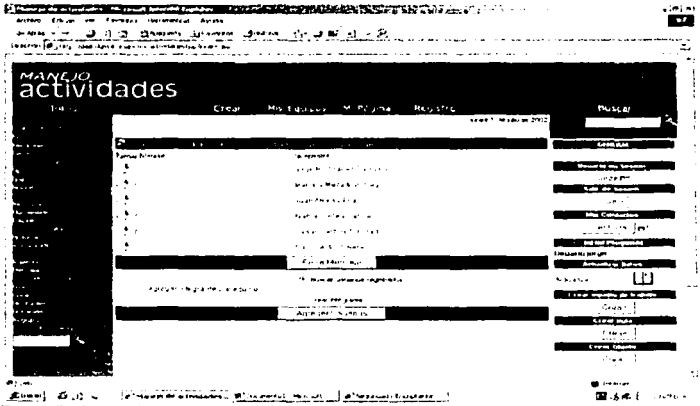


Fig. 5.7. Pantalla que presenta la forma de envío de mensajes grupales o privados.

El icono de ruta, permite observar un diagrama de cómo está organizada la ruta, en él, aparece el nombre de los responsables asociado a cada una de los bloques, se verá gráficamente como está planeada la ruta y a quién se ha delegado la responsabilidad de cada una de los bloques, es decir cómo se han asignado los bloques, como se realizará el proceso de revisión de los objetos.

Lo que se verá al dar clic a ese icono será un diagrama organizacional como el siguiente:

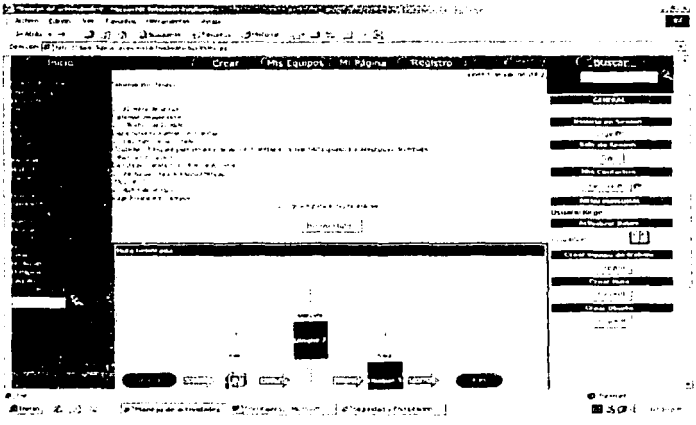


Fig. 5.8 Pantalla que presenta los datos generales de la ruta.

En la parte superior se observa la información general de la ruta creada en el sistema entre la que se encuentra el nombre de la ruta, el objetivo por el que se generó la ruta, una descripción de la ruta, si hay un objeto en la ruta se indicará en posesión de quien se encuentra o quien está realizando la revisión, la fecha en que fue creada la ruta y el nombre del autor de la ruta.

En caso de que el administrador sea el que esta analizando la ruta y no hay objetos sobre esta, tendrá una opción más solo la podrá usar él, la ruta podrá ser borrada.

La imagen de la ruta será la siguiente:

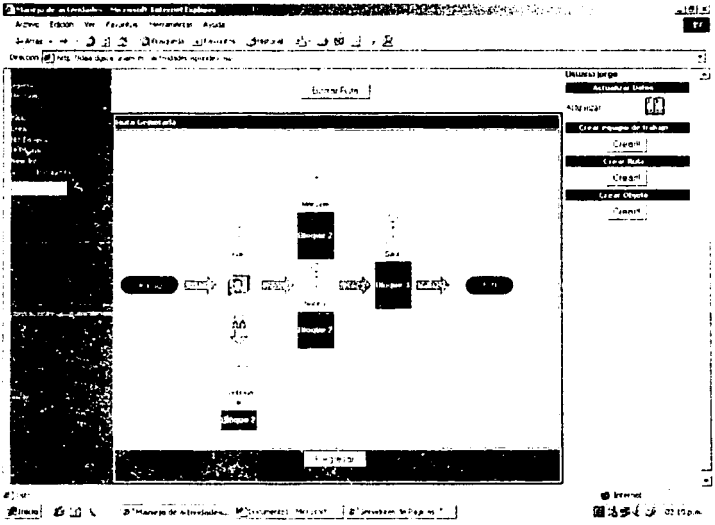


Fig. 5.9. Pantalla que presenta el diagrama de la ruta que puede ser revisado por cualquier participante del equipo.

El icono de los objetos, permite ver la información que se va "recopilando" conforme avanza el objeto por la ruta, es en esta sección que se puede ver la información recogida por el objeto en su camino por las etapas, esto es el seguimiento del objeto, el cual va creando el historial del objeto.

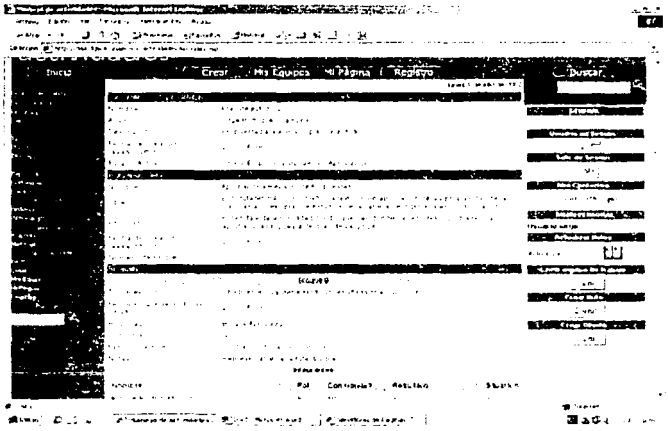


Fig. 5.10. Pantalla que presenta la información que puede consultar el participante de un equipo para observar el historial del objeto.

Fig. 5.11 Pantalla que presenta la información que puede consultar el participante de un equipo para observar el historial del objeto.

La información del objeto está organizada de la siguiente manera:

- Información del objeto.
- La ruta a la que esta asociada el objeto, indicando el número de bloques que lo componen
- La información de cada uno de los bloques
- El tipo de Revisión que se realizará en cada bloque
- El nombre de los integrantes Asociados a cada Bloque
- El Periodo de Revisión asignado a cada bloque
- La situación de cada etapa

En este ejemplo, el diagrama cuenta con 8 etapas y cada una de ellas debe ser documentada, por ello es que el historial del objeto se vuelve tan grande.

Todos los usuarios dentro del sistema tienen las mismas oportunidades de crear equipos, rutas y objetos, todos podrán enviar documentos a revisión, nadie en este sistema es observador, todos los usuarios deben ser partícipes de otra manera no se podría automatizar la comunicación de los usuarios por este sistema.

Además de buscar ser un punto de participación , el sistema busca convertirse en un sistema de planeación, un sistema de control que pueda usar el coordinador de proyectos no solo para observar los cambios de proyecto, sino también busca convertirse en la herramienta que proporcione los avances documentados del proyecto a quien los solicite, en el momento que los solicite, ya que al ir se generando la documentación de los objetos se genera el historial de cada uno lo que al final del sistema se convertirá en su documentación técnica.

El sistema también le ofrece al participante que ha olvidado su password la posibilidad de devolvérselo por medio de un correo electrónico, esta acción la podrá realizar desde la página principal ingresando su login.

Por medio del correo electrónico que proporciono a la hora que realizó su registro le será enviada la información olvidada, en este caso su password.

La pantalla para solicitar la recuperación del password es la siguiente:

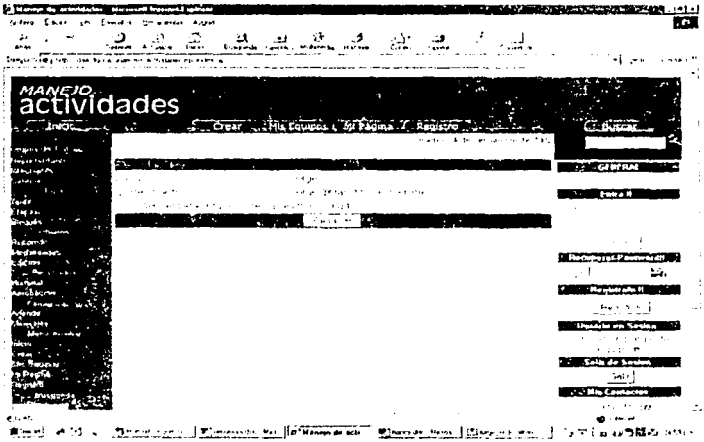


Fig. 5.12 Pantalla que permite la recuperación del password y lo envía por correo electrónico.

5.3 Creación de equipos de trabajo, rutas y objetos.

A continuación se presenta la manera como se construyen los equipos de trabajo, rutas y objetos.

5.3.1 Creación de equipos de trabajo.

Cuando un representante desea crear un equipo, puede hacerlo seleccionando la opción que aparece en su página personal denominada crear equipo.

Una vez seleccionada esta opción aparecerá una pantalla que le pedirá indique el nombre que desea tenga el equipo.

Una vez indicado el nombre, tendrá dos opciones para elegir a los integrantes del equipo:

- Eligiendo participantes ya registrados
- Inscribiendo Nuevos Participantes al Sistema.

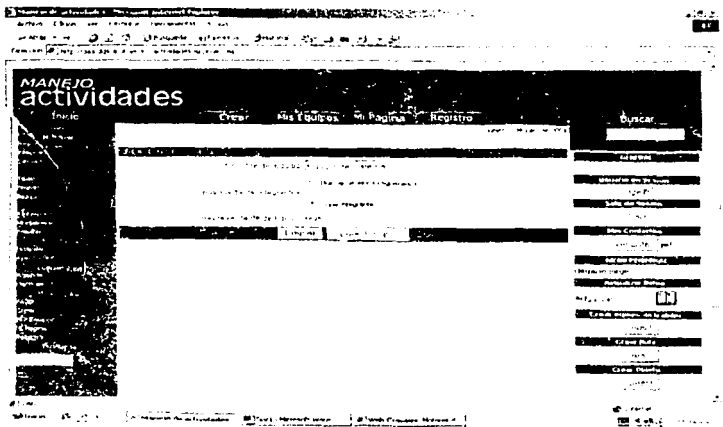


Fig. 5.13 Pantalla que permite crear un equipo de trabajo.

Una vez llenados los datos deberá presionar crear equipo y aparecerá la pantalla que indica la creación del nuevo equipo, que solicitará indicar quienes serán los participantes del equipo según se haya elegido la opción.

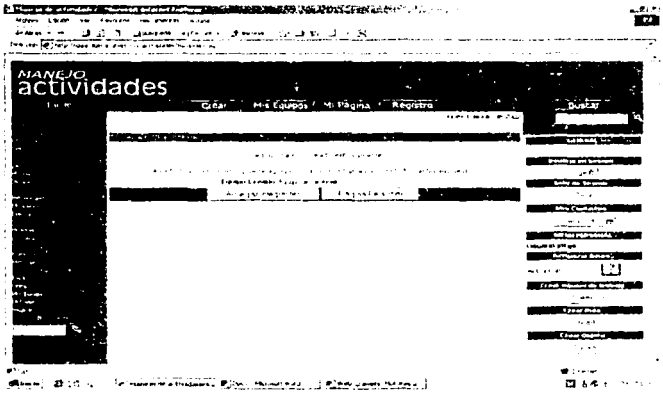


Fig. 5.14 Pantalla que permite agregar participantes a un equipo de trabajo

Las formas de agregar integrantes registrados en sistema se realiza con la siguiente pantalla de búsqueda, la cual permitirá buscar información con cualquiera de los parámetros que contiene:

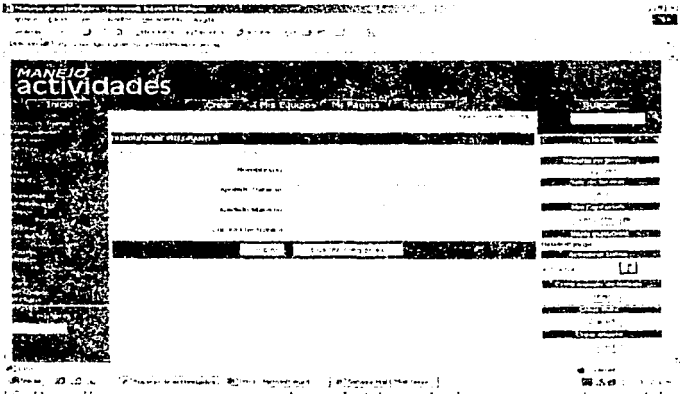


Fig5.15. Pantalla que se presenta para hacer la búsqueda de integrantes dentro del sistema.

Por medio de esta forma, se podrá buscar a los usuario registrados en el sistema, con solo indicar uno de los campos se realizara una búsqueda en la base de datos, lo que proporcionará a los usuarios registrados con esas características.

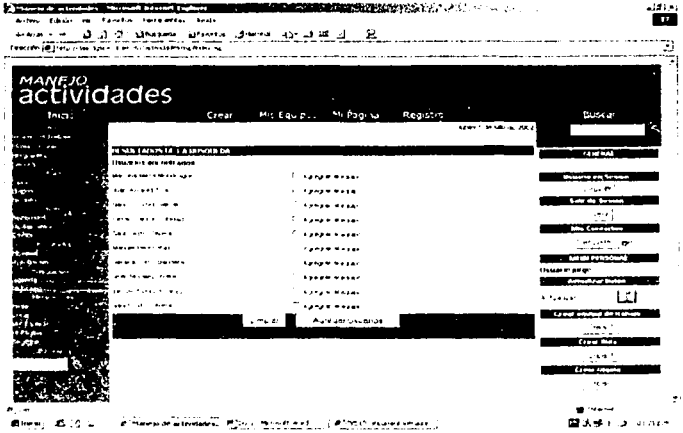


Fig. 5.16. Pantalla que regresa el resultado de la búsqueda de integrantes dentro del sistema.

El sistema regresa lo obtenido de la búsqueda y el usuario con solo seleccionar el nombre de los usuarios en la casilla los podrá inscribir en el equipo de trabajo.

Con esta elección de participantes, terminaría la etapa de creación de equipo de trabajo, pueden existir tantos equipos como el representante desee, siempre y cuando recuerde que será su responsabilidad crear un equipo con un análisis previo de los recursos humanos necesarios.

Si se desea agregar a los usuarios por medio de la segunda opción, el representante inscribirá a cada uno de los usuarios que desee incluir en el sistema.

Los participantes registrados al proyecto por medio de esta opción son usuarios que aún no participan en el portal, por ello la invitación a participar se realizará por medio del correo electrónico, por ello es indispensable que el responsable de la ruta proporcione los datos correctos de su correo electrónico, ya que si no será imposible enviar la invitación a participar.

Si el usuario es registrado de esta manera el representante estará obligado a llenar la siguiente forma:

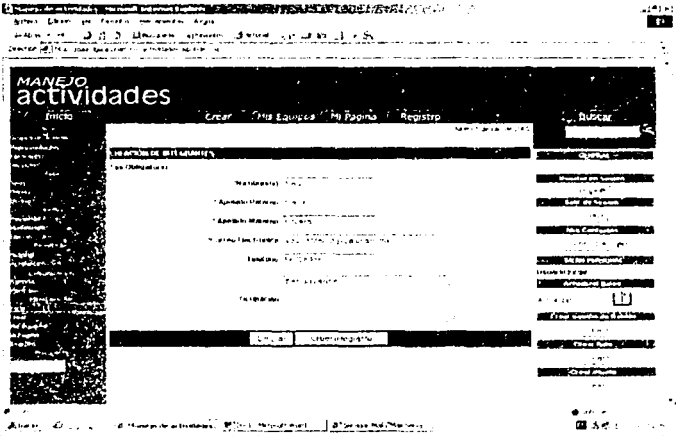


Fig. 5.1" Pantalla que presenta la forma de crear nuevos participantes no registrados en el sistema.

Como se puede ver los datos del nombre del futuro participante y el correo electrónico son datos indispensables.

Con estos datos, el sistema genera y envía el correo electrónico en el se le asigna al nuevo usuario un login y un password que consistirán en la llave que genera el sistema para su identificación dentro del sistema.

En este correo se le pide al usuario que ingrese al sistema, por medio de una liga al url donde se localiza la herramienta, si recibe el correo en un sistema Unix o Linux tendrá que teclear en su browser el url del sistema. Y se le solicita que cambie sus datos personales, ya que la persona que lo registro pudo haber equivocado algún dato, además el usuario, podrá cambiar su login y password.

Con la operación de ingreso de un nuevo usuario, el representante tendrá la opción de agregar tantos usuarios no registrados como el desee o podrá regresar a la página principal.

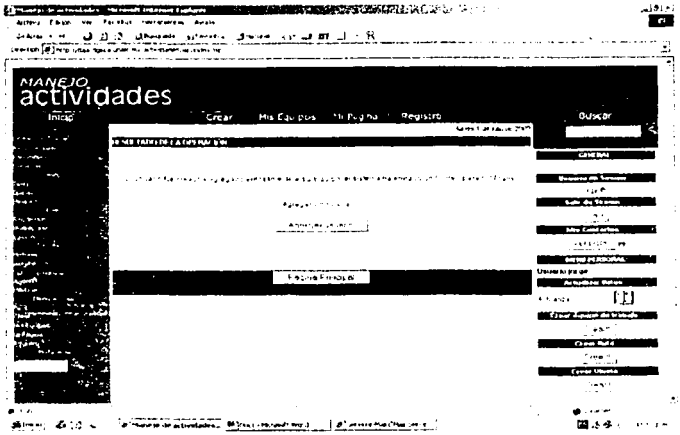


Fig.. 5.18. Pantalla que permite generar más nuevos integrantes dentro del sistema.

Una vez que el nuevo usuario ha decidido entrar al sistema, encontrará, la opción de actualizar datos en su página personal lo que le proporcionará una forma muy parecida a la de registro.

La opción de actualizar datos la tienen todos los participantes, los que se han registrado ellos mismos no podrán cambiar su nombre y apellidos, los que han sido registrados por el representante de un equipo lo podrán cambiar solo la primera vez que ingresan en el sistema.

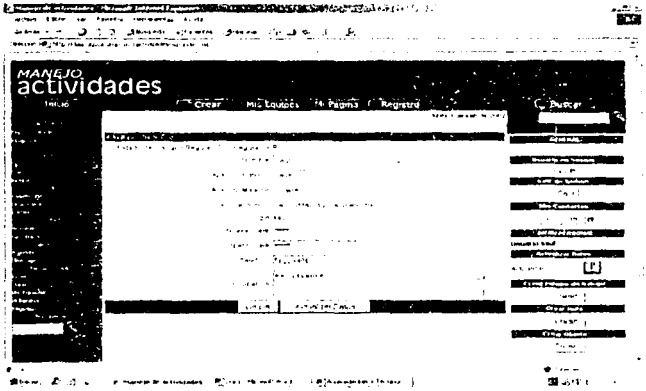


Fig..5.19. Pantalla que permite actualizar los datos de los participantes en el sistema.

A todos los participantes que hayan sido elegidos sin importar la manera de registro se les enviara su primera tarea de equipo que es confirmar su participación, este mensaje contiene el nombre de los demas participantes del equipo y sus roles de participacion en él.

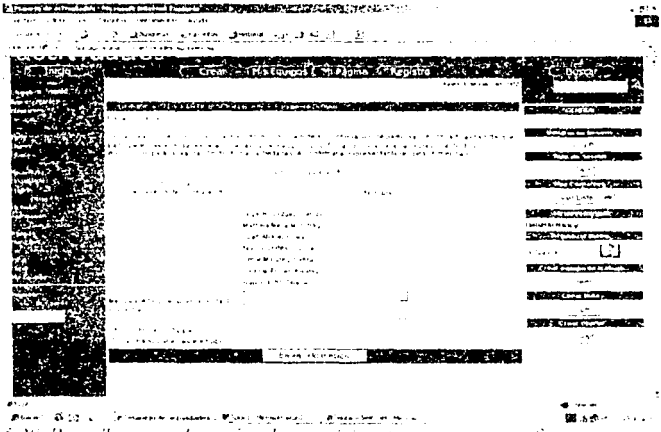


Fig. 5.20 Pantalla que se le envia a los participantes para que confirmen participación

En esta sección el usuario puede elegir dos opciones, decide confirmar su participación o decide no participar en este equipo de trabajo. Si la decisión del usuario es no participar en el equipo, será el representante del equipo el que deberá tomar una decisión y tomar como alternativa realizar la invitación a alguien más.

5.3.2 Creación de rutas.

Para comenzar a usar el sistema es necesario que existan rutas o caminos para enviar los objetos.

Cada participante en su página personal, puede elegir la opción de crear rutas, la creación de rutas comienza en la siguiente forma:

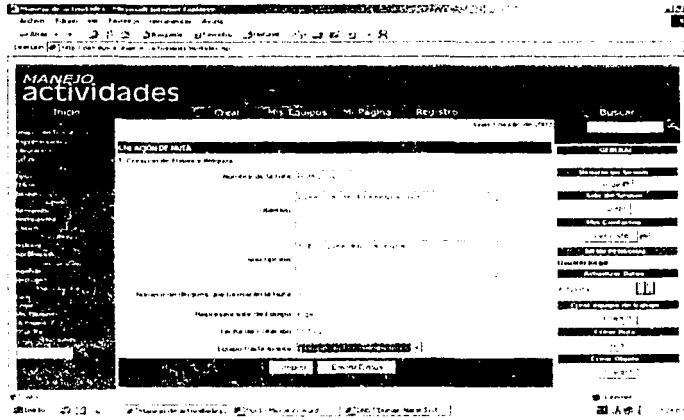


Fig. 5.23 Pantalla para la creación de una ruta.

Al crear una ruta es el momento en el que se plasma todo el trabajo que viene antes de comenzar con el desarrollo de un proyecto, cuando se crea una ruta ya se debió haber hecho toda una tarea de planeación de software, se debió de haber analizado los recursos necesarios tanto software, como hardware, se debieron haber estimado tiempos de desarrollo pero sobre todo y lo más importante los recursos humanos.

La ruta debe estar planeada para un equipo especial de participantes, cada participante cuenta con características especiales que deben ser usadas en beneficio del desarrollo sus conocimientos y habilidades son necesarias y no pueden ser desaprovechadas.

La planeación temporal, es otra parte muy importante, ya que la ruta necesitará ser generada indicando los tiempos para cada bloque y cada bloque será dividido en pequeñas actividades que permitan cuantificar los avances de una manera más eficiente o detectar las fallas con más claridad, en este momento es donde la estimación juega el papel más importante para la asignación de responsabilidades.

La ruta al igual que el equipo necesita que se le asigne un nombre, cada ruta se genera para lograr un objetivo, por ello es necesario identificar el objetivo antes de crearla, la descripción de la ruta también es de gran importancia, ya que en ella se puede plasmar información de cómo se planteó se desarrolló y que se espera del trabajo en equipo.

Otro campo de gran importancia es el de número de bloques, en este se deben señalar el número de bloques que se desean generar y el nombre del equipo elegido para esta ruta. El campo de número de bloques aunque solo parece un número tiene atrás de el todo el análisis de la planeación de proyectos, para llenar este dato ya se debe de tener pensado tanto el número de actividades en las que se va a dividir al proyecto, como el número de recursos humanos que serán necesarios.

Una vez asignados estos datos se envían para generar el diagrama organizacional.

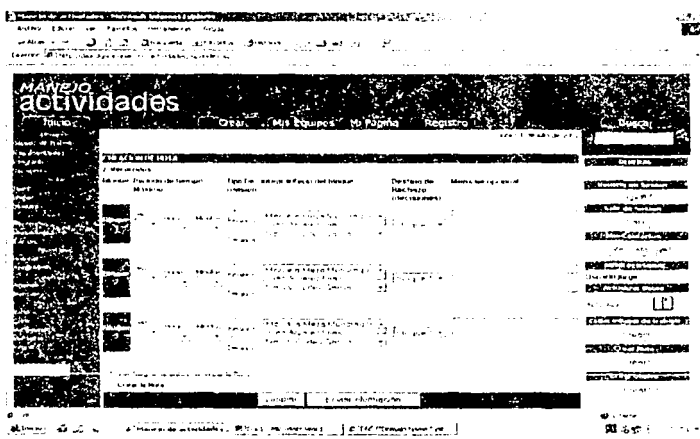


Fig. 5.22. Pantalla para la creación de una ruta.

A esta etapa se le llama recorrido, esta sección esta dividida en seis columnas que son las siguientes:

- Número de Bloque: En ela se coloca el número de bloques que designó el representante del equipo.
- Periodo de Tiempo Máximo: En ella se indica el tiempo máximo que puede durar la actividad.
- Tipo de Revisión: Es el modo de operación de los integrantes de la ruta.
- Integrantes del Bloque: En esta sección se asigna a los responsables de cada bloque.
- Destino de Rechazo: Aquí se indica la acción que tomaría el objeto si fuera rechazado por el responsable de la etapa.
- Mensaje Opcional: Aquí se pueden colocar comentarios opcionales.

Cuando se asigna a los responsables de la ruta, no se puede asignar a la misma persona como responsable en dos bloques seguidos, es decir, una sola persona no puede ser responsable de actividades dependientes. Si esto sucede el sistema enviará un mensaje indicando que existe un error al crear la ruta, al asignar como responsable a una sola persona de dos bloques seguidos, la ruta solo se podrá crear hasta solucionar el error.

Una vez que se han indicado, el tiempo de duración máxima , el tipo de revisión y la participación de los responsables, además del destino de rechazo de los objetos (en caso de ser decisión negativa sobre un objeto) el usuario tiene dos opciones más:

- Generar directamente la ruta
- Ver la ruta antes de generarla.

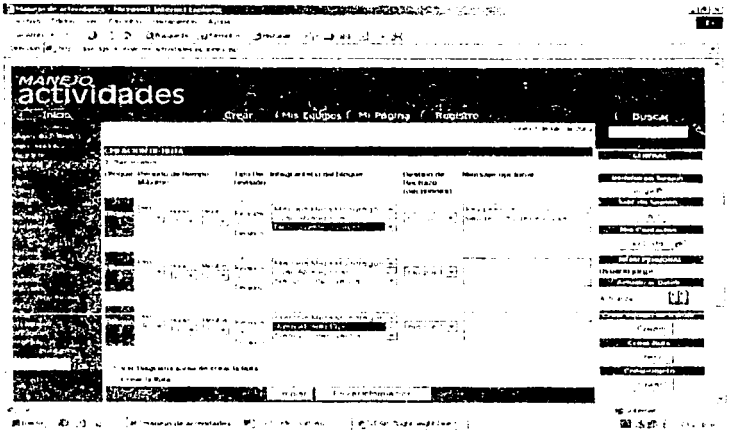


Fig. 5.23 Pantalla con los datos para la creación de una ruta.

Si su decisión es ver el diagrama de la ruta, cuando elija la opción verá algo muy similar a lo siguiente.

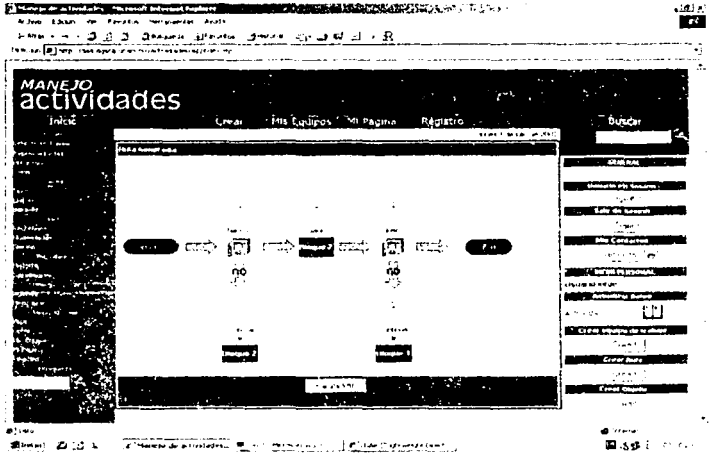


Fig. 5.24 Pantalla que presenta la como será la ruta en caso de generarse.

Si esta de acuerdo con esta ruta podrá regresar a la pantalla anterior y ahora si tomar la decisión de generar la ruta. Si no esta de acuerdo podrá replantear la trayectoria de los objetos y plantear una ruta que si satisfaga las necesidades del desarrollo.

Si la ruta ha sido de su agrado y ha decidido generarla entonces aparecerá un mensaje indicando que la ruta ha sido generada satisfactoriamente y que en el momento que se decida pueden ser enviados objetos sobre ella.

Una ruta puede ser creada para llevar todo el control del desarrollo del software, en general o para el control de pequeñas actividades, una buena planeación, permite proporcionar una base sólida para la toma de decisiones.

Cada ruta puede identificar exhaustivamente cada una de las actividades a realizar, se puede planear con los recursos necesarios ya sea humanos como de cualquier índole, por ello permite la posibilidad del diseño de una excelente planificación de tareas.

5.3.3 Creación de objetos

La creación de objetos es la parte más importante del sistema, es la que permite la colaboración de los diferentes participantes en este espacio virtual, en este punto de participación y de reunión de los integrantes del desarrollo.

Para comenzar a enviar objetos debe de elegirse una ruta y esta ruta debe estar libre de otros objetos, por una ruta no se podrá enviar más de un elemento a la vez.

La creación de objetos la puede realizar cualquier participante que sea representante de un equipo y haya generado una ruta. Un objeto se puede crear desde la página personal de un participante, simplemente se elegirá el botón "crear objeto".

Una vez elegida la opción de crear objeto se le presenta al participante una pantalla con cuatro tipos de iconos, estos indican que tipo de documentos puede generar el participante en el sistema.

A continuación se presenta la pantalla que podrá ver el participante del sistema

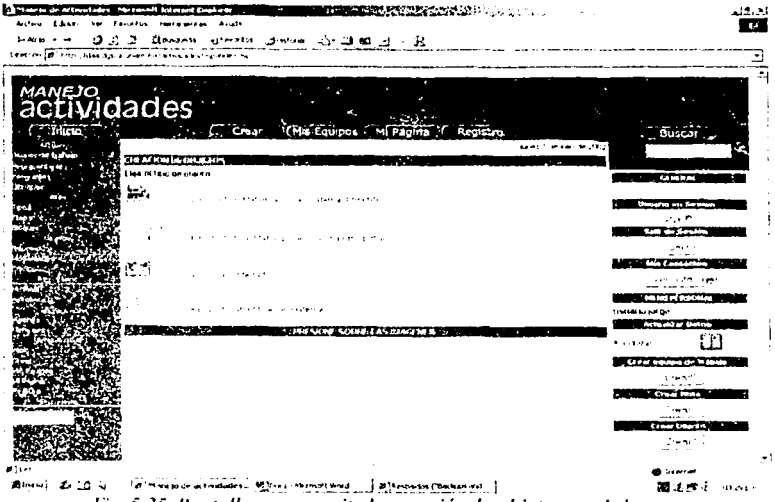


Fig. 5.25. Pantalla que permite la creación de objetos en el sistema.

Entre los documentos que puede subir al sistema se encuentran:

- Objetos Editables desde html o txt.
- Los Objetos no Editables como son doc, pdf
- Alguna Dirección de Internet
- Realizar Creación de un Documento en el Sistema

El usuario podrá elegir el tipo de documento que desee subir al sistema con solo dar clic en el icono elegido.

Una vez que haya elegido el tipo de documento se le presentará una forma parecida a la siguiente:

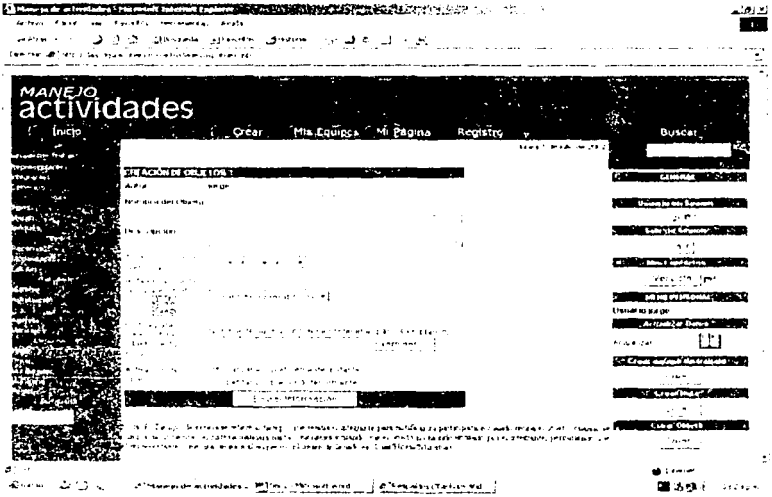


Fig. 5.2 La Pantalla que permite subir diversos tipos de documentos como objetos en el sistema.

En ella encontrará una forma que solicita ingresar un nombre del objeto, la descripción de cada objeto es necesaria e importante por ello también se solicita, además se debe asignar el tiempo de espera máximo para que cada participante pueda revisar el objeto.

Cada objeto se desplaza sobre una ruta por ello se desplegará el nombre de las rutas existentes en el sistema para que el representante elija sobre cual de sus rutas enviará el objeto para su revisión o aprobación.

Ahora es necesario indicar donde se encuentra el objeto, para poder subirlo al sistema por ello se presenta el cuadro de examinar para que el usuario pueda buscar en su computadora el documento que desea enviar a revisión, una vez elegido el documento se debe decidir si será enviado inmediatamente o se enviará después.

Si la decisión es enviarle inmediatamente llegará un mensaje a los participantes de la ruta, indicándoles que hay un objeto para su revisión, si no es su oportunidad para revisarlo deberán esperar a que sea revisado por la primera persona indicada en el bloque y así sucesivamente hasta que sea su turno para la revisión o decisión.

Esta pantalla aparecerá en el caso de que sea un URL lo que se va enviar a revisión.

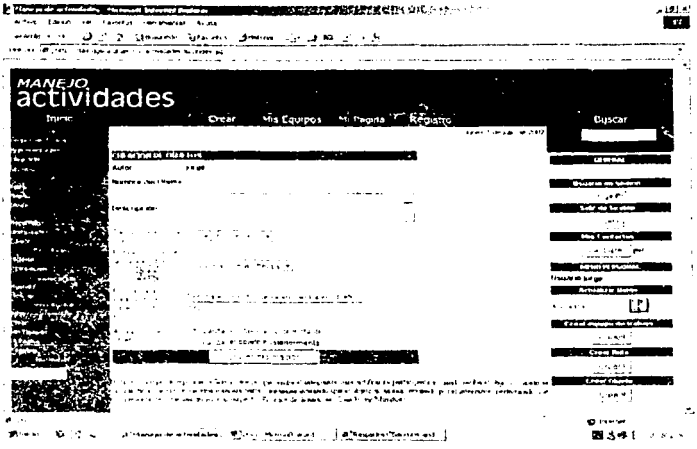


Fig. 5.2 Pantalla que aparecera en el caso de que sea elegido como objeto un url.

Si la decisión fue que el envío fuera inmediato entonces aparecerá un mensaje indicando que el objeto ha sido creado satisfactoriamente.

Cuando a un participante le llega un objeto el menú contará con una nueva tarea la cual le indicará que se necesita de su participación, en ese mensaje le llegará la información del

objeto y se le pedirá que envíe su aprobación para realizar la tarea de revisión o el rechazo del objeto.

Lo que verá el usuario será una pantalla como la siguiente, la cual contiene:

- Los datos del Objeto.
- Los datos del bloque actual que recorre.
- La decisión de Aprobación o Rechazo.

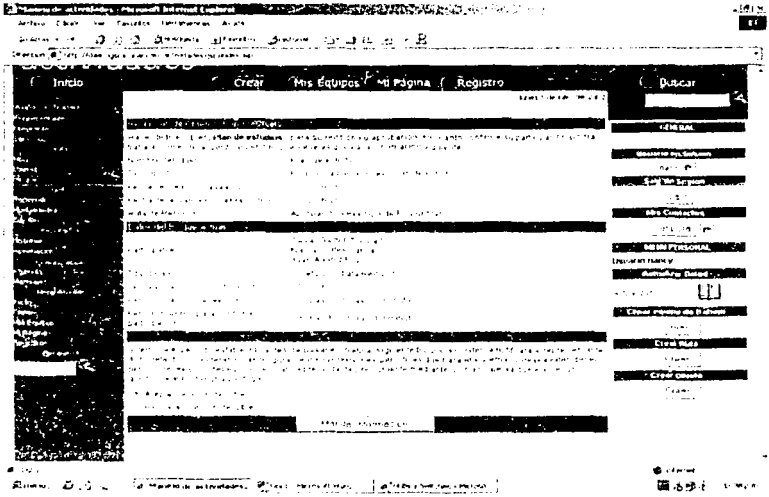


Fig. 5.28 Pantalla que presenta el sistema a la participantes para confirmar la revisión del objeto.

El participante deberá confirmar su decisión al representante del equipo que le envió el documento para realizar sobre el la tarea.

Al contestar el icono el objeto cambiará y se le presentará al responsable de la ruta un icono que al dar clic sobre él, permitirá no solo realizar la acción de decisión sobre el objeto, le permitirá revisarlo , analizarlo con calma y agregar a el, Anexos, Notas, Podrá agregarle otro archivo, podrá agregarle una liga o URL.

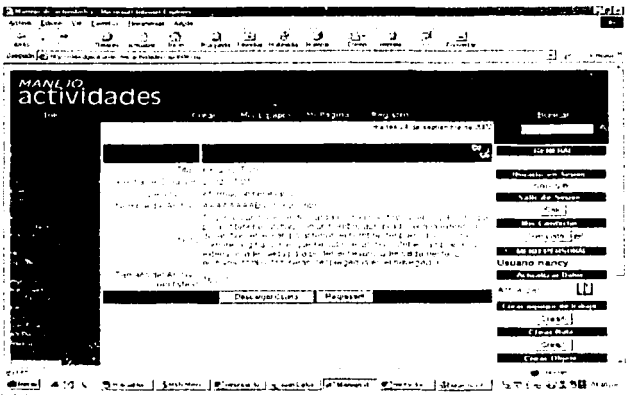


Fig.. 5.29. Pantalla que aparece para permitir al participante la revisión del objeto.

Las notas que agregue el usuario deberán ser realizadas indicando en cada una de ellas un título, una descripción de la nota y además un objetivo, es decir indicar el porque se agregó la nota al objeto.

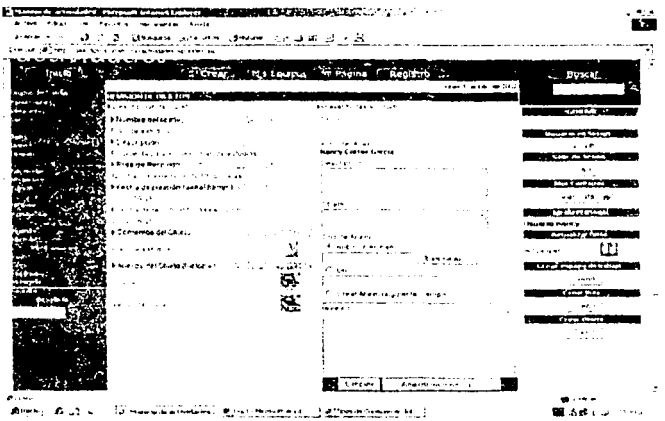


Fig. 5.30 Pantalla que permite hacer la revisión del documento, agregar notas y anexos al objeto

Esto permite el análisis del documento pero hasta el momento el participante no ha tomado una decisión sobre él.

Si el participante desea revisar los anexos agregados al objeto lo podrá hacer de una forma similar a la revisión del objeto, solo dando clic, en las imágenes de anexos.

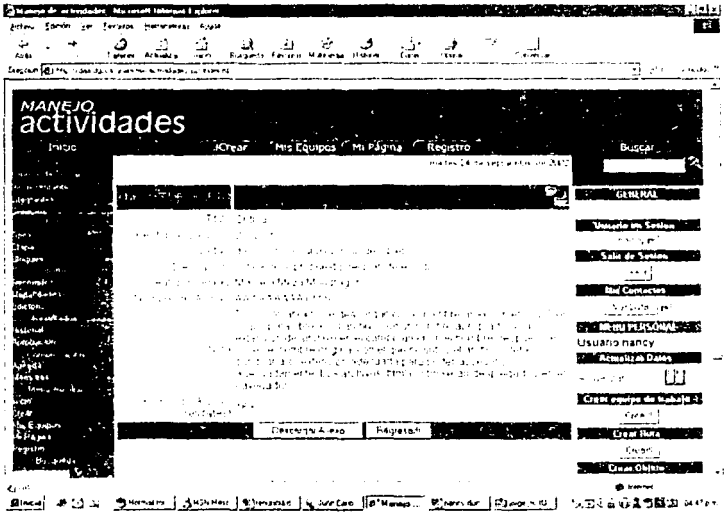


Fig. 5.31 Pantalla que permite descargar la información de los anexos del objeto.

Lo que podrán ver será el objeto enviado y presentado como un archivo JSP, esto para permitir la visualización del objeto por medio de la red.

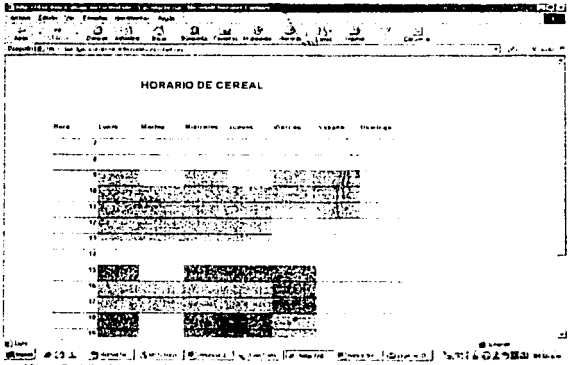


Fig. 5.32. Pantalla que permite ver el desplegado de un objeto.

En el caso de los Anexos podrán ver información como la siguiente:

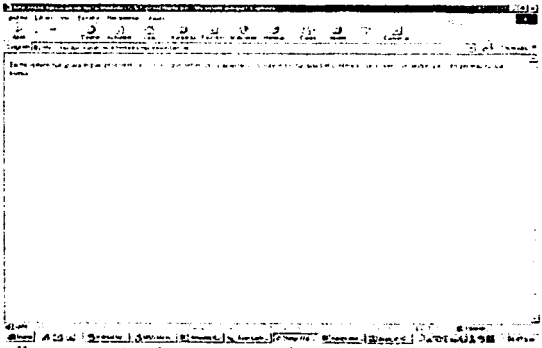


Fig. 5.33. Pantalla que permite observar como el usuario podrá ver los anexos y comentarios del objeto.

Las decisiones son lo que llamamos modos de operación del participante, el representante al crear la ruta indica el tipo de revisión que realizará cada participante, por ello cuando el objeto llega a manos del participante que esta en turno de revisión, la revisión ya viene indicada.

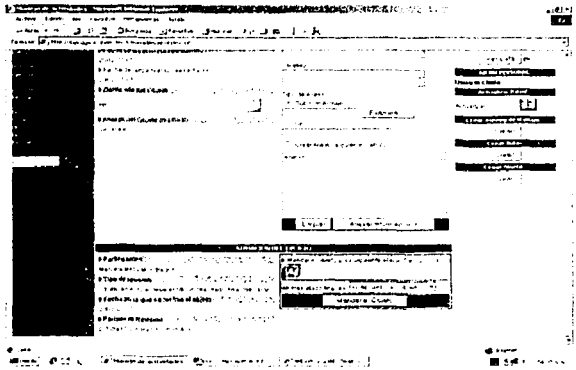


Fig. 5.34 Pantalla para tomar una decisión sobre el objeto.

Esta pantalla aparecerá en el caso de que el tipo de revisión sea de decisión, la pantalla presenta un menú en el cual tendrá dos opciones aprobar el objeto o no aprobarlo y hacerlo que vaya a donde indico el responsable en caso de rechazo.

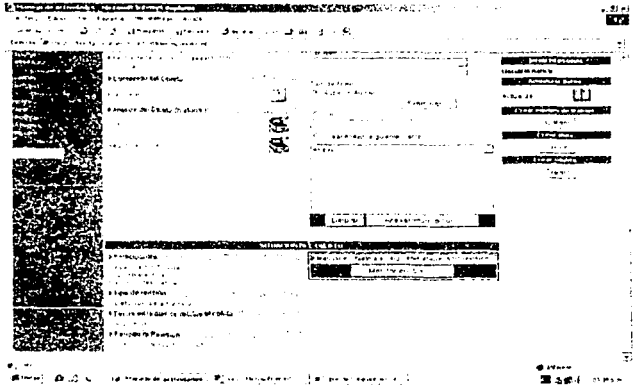


Fig. 5.35 Pantalla para el caso de revisión de un objeto.

Esta pantalla solo se presenta en el caso de que el usuario solo tenga la tarea de revisión sobre un objeto

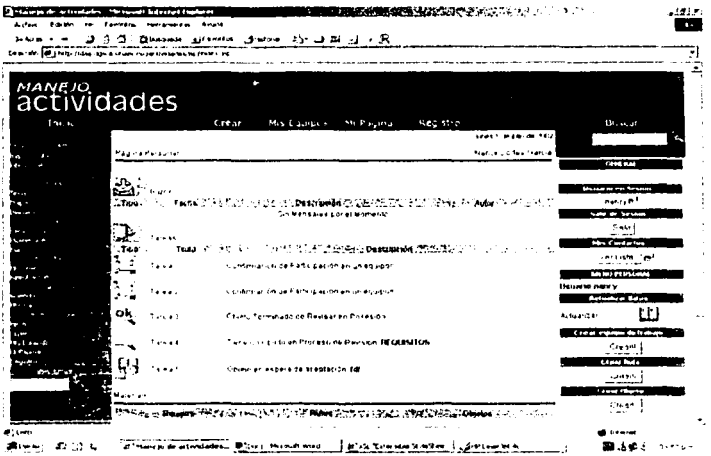


Fig. 5.36 Pantalla que muestra que la tarea de revisión de objeto ha sido realizada por este participante

Cuando ha terminado el proceso de revisión por parte del usuario el icono del objeto cambiará en la sección de tareas. Para que el objeto salga de la ruta tendrá que haber sido revisado por todos los responsables de los bloques.

Cuando el icono del objeto cambie y se vea un sobre con la letra P indicará que el objeto esta en proceso de revisión.

Cuando termina el proceso de revisión de todos los participantes, se obtendrá una decisión analizada por todos, entonces el objeto ha cumplido su cometido, ha recolectado entre el grupo de trabajo comentarios que servirán para tomar una decisión en equipo.

Los objetos permiten crear un mecanismo de colaboración, convirtiéndose en una herramienta que ayuda a los individuos a trabajar juntos en un modo cualitativamente mejor que el planteado por los esquemas de organización tradicionales.

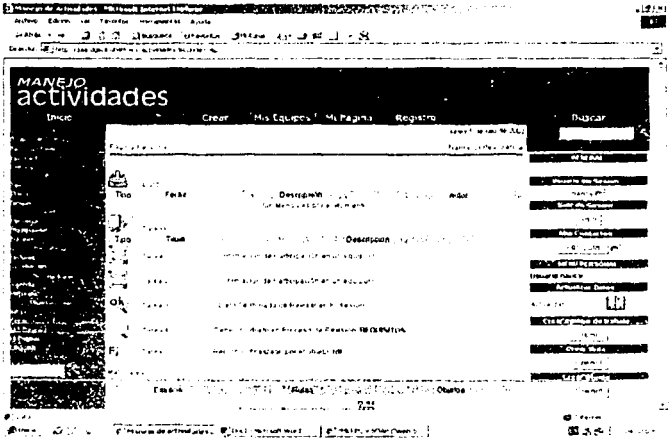


Fig. 5.37. Pantalla que indica el fin del recorrido de un objeto en el menú de tareas del responsable de la ruta y del equipo.

Cuando termina el recorrido de un objeto sobre una ruta, el representante del equipo observará en su sección de tareas un icono y sobre el una letra F la cual indicará el fin del recorrido del objeto por la ruta asociada.

En este momento el representante del equipo tiene la posibilidad de analizar el historial del objeto, lo que se convierte en un reporte final, este reporte incluirá todas las anotaciones, y notas que los responsables incluyeron con este reporte será posible la toma de decisiones dentro del desarrollo

Lo que podrá ver el responsable será un reporte que contenga el número de bloques que recorrio el objeto, el nombre de los responsables que lo revisaron y sus decisiones y comentarios sobre el.

Las siguientes pantallas serán el resultado del camino que recorrió el objeto sobre la ruta.

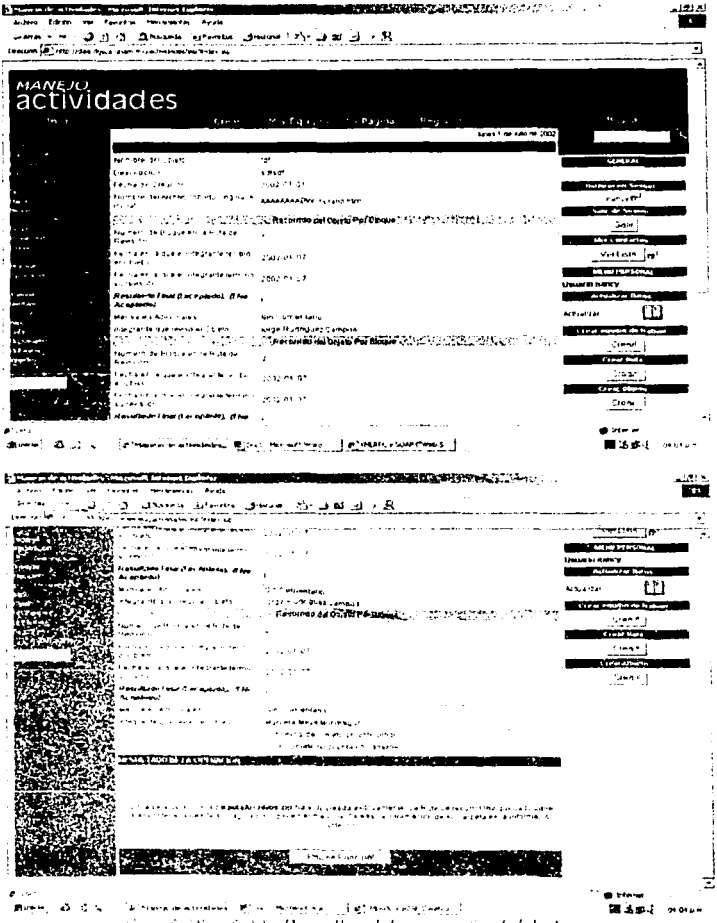


Fig. 5.58 y 5.59 Pantallas del reporte final del objeto.

Cuando un objeto termina su objetivo, que es conseguir las opiniones, notas, anexos y comentarios del equipo de trabajo, no ha terminado su trabajo, su nuevo trabajo consiste ahora en ser parte de la documentación técnica del proyecto, por ello el representante de la ruta tenga que tomar la decisión de que se hará con el objeto.

El sistema le ofrece la oportunidad de hacer con el una carpeta donde guarde toda la información obtenida en el recorrido.

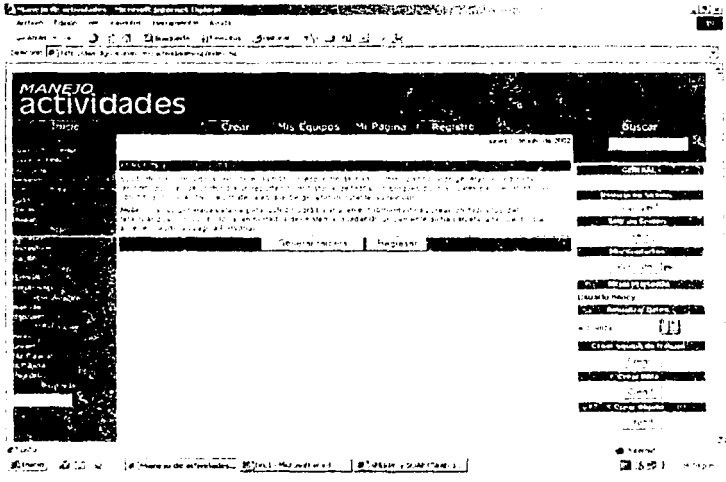


Fig. 5.40. Pantalla que permite generar la carpeta de un objeto.

El representante podrá generar la carpeta por medio de la opción generar carpeta esto le regresará como resultado una carpeta en la sección de tareas, la cual incluirá la descripción indicando que se trata de la carpeta final e indicará el nombre del objeto

El icono de carpeta proporciona otra opción más al representante del equipo, esta opción es bajar la carpeta a un directorio donde se este recopilando toda la información del objeto o la otra opción es borrar el objeto del sistema.

Para lo cual presenta al usuario la siguiente pantalla:

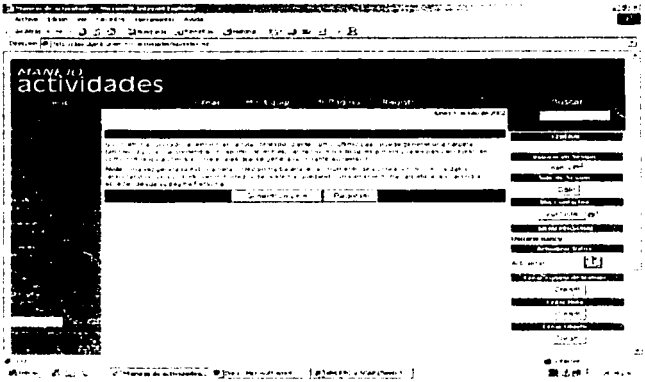


Fig. 5.41. Pantalla para generar una carpeta del objeto.

El representante tomará la decisión sobre que hacer con la carpeta del objeto, y para ello se le presentará las opciones que tiene:

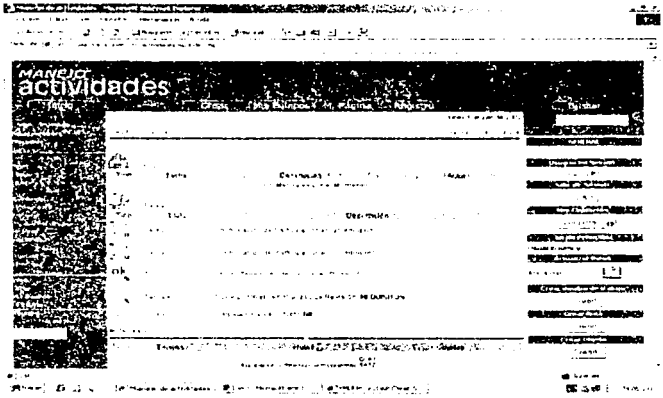


Fig. 5.42. Pantalla que tiene en su menú de tareas la carpeta generada para un objeto.

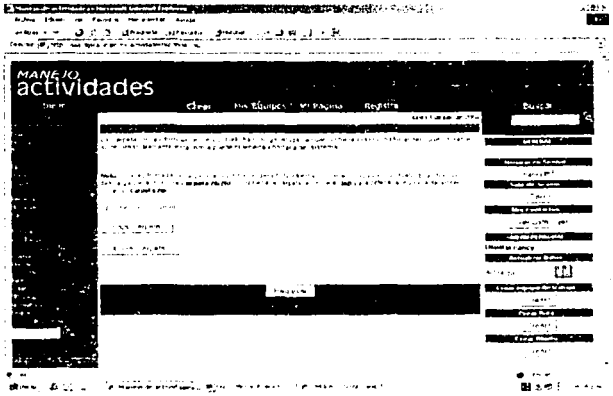


Fig. 5.43. Pantalla que permite tomar la decisión de bajar o borra una carpeta de un objeto del sistema.

Si su opción es bajar la carpeta al sistema deberá elegir la opción bajar carpeta.

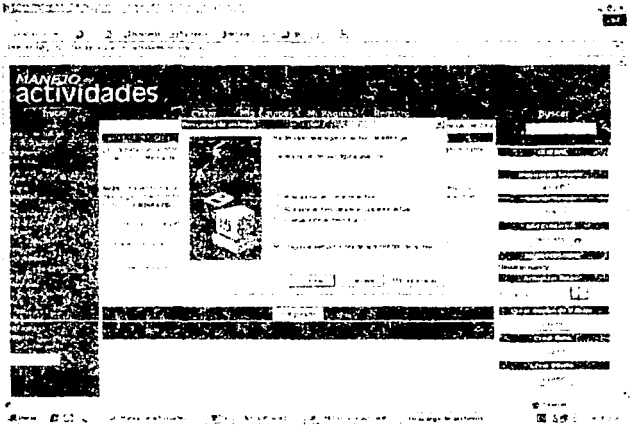


Fig. 5.44. Pantalla que permite al usuario bajar la carpeta que el sistema genera del objeto.

Si por otro lado la opción del usuario es borrar el objeto del sistema deberá elegir la opción borrar y aparecerá el mensaje indicando que el objeto ha sido borrado del sistema.

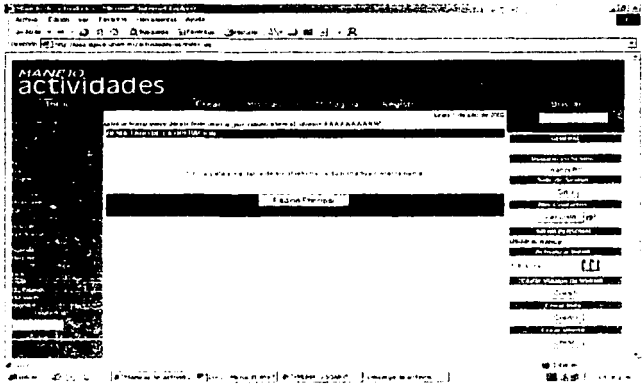


Fig.. 5.45. Pantalla que permite borra del sistema la carpeta de información que se genera de un objeto al terminar de recorrer una ruta.

Este el proceso que siguen los objetos en el sistema, su objetivo es permitir la comunicación automática a los participantes de los equipo del sistema.

Lo que se busca es obtener un desarrollo de calidad combinando las herramientas, con los conocimientos de las participantes.

Los objetos nos permiten ademas de llevar un control de los avances y de los cambios hechos en el desarrollo, realizar una buena documentación para el proyecto, uno de los problemas que presentan los desarrollos que hoy funcionan es no contar con una buena documentación y el tiempo hombre que se necesita invertir para entender el sistema cada vez que se desea realizar una actualización o un cambio en su funcionamiento es realmente grande.

Cuando se termina un desarrollo siempre se debe pensar que requerirá modificaciones futuras por ello si se cuenta con una buena documentación se podrá entender a futuro el sistema sin problemas. Con esta herramienta, se pretende que los equipos de trabajo se encuentren en todo momento comunicados e informados de los avances del proyecto, que su participación sea mas dinámica pero sobre todo se pretende ofrecer un espacio virtual donde el equipo integre no solo su trabajo si no también sus conocimientos y propuestas para encontrar mejores alternativas y lograr un trabajo de calidad.

CAPÍTULO VI
CONCLUSIONES

Capítulo VI CONCLUSIONES.

Conclusiones:

El trabajo que se ha implementado a lo largo de esta tesis muestra un sistema que tiene las siguientes características:

1) Es un sistema que se utiliza con éxito.

El sistema de Manejo de Actividades se diseño en el Departamento de Aplicaciones Avanzadas de la Dirección General de Cómputo Académico (DGSCA) como una herramienta de apoyo para el desarrollo de sistemas de software.

El sistema de Manejo de Actividades que se ha implementado es una herramienta que apoya la planeación de proyectos de software, coordina y comunica de una manera sencilla a los integrantes de un equipo de trabajo.

Manejo de Actividades permite a los responsables de desarrollos, plasmar la planeación y los avances del proyecto en un sistema que puede ser consultado en cualquier momento, que puede dar una estimación del avance general en cada actividad.

Los resultados obtenidos por el sistema han sido satisfactorios, ya que se logra una mejor comunicación entre los diferentes miembros de equipos de desarrollo. El sistema permite almacenar la información que se va recolectando y nos permite administrarla. Además ha permitido tener un constante monitoreo de los avances en los proyectos así como permitir el análisis de la información que se recolecta, con esta herramienta no es necesario generar gran cantidad de reportes sobre los avances de cada uno de los participantes en los desarrollos, ya que sus avances son visibles en el sistema. Permite además implementar planes de acción si existiera retraso en los desarrollos.

Por medio de este software el seguimiento de cada proyecto es más sencillo ahora y ha mejorado. las reuniones de trabajo son mas dinámicas pues todo el equipo está bien informado de todas las actividades, los equipos de trabajo presentan más propuestas y los desarrollos se terminan en los tiempos esperados.

2) Es portable

Este sistema es fácilmente portable ya que:

- La independencia de la plataforma es una de las ventajas más representativas que tiene Java sobre otros lenguajes de programación, en particular para los sistemas que necesitan funcionar en varias plataformas. Java mantiene esta independencia de la plataforma tanto a nivel del código fuente como en el binario.

- A nivel código fuente, los tipos primitivos de datos de Java tienen tamaños consistentes, en todas las plataformas de desarrollo. Los fundamentos de bibliotecas de Java facilitan la escritura de código, el cual puede desplazarse de plataforma a plataforma sin necesidad de volver a escribirlo para que funcione con esa plataforma.

- JSP (JavaServer Pages) es transportable a otros sistemas operativos y servidores, por ello su portabilidad se podrá realizar de una manera muy sencilla

- La base de datos PostgresSQL ha tenido tanta aceptación que está disponible no solo para UNIX actualmente se está probando una versión que funcionará de la misma manera en un sistema operativo Windows.

3) Se realiza el mantenimiento del software fácilmente.

El control que realizan los participantes es la única administración que necesita.

Cada participante será administrador de su página personal, cada representante será responsable de sus equipos, de sus rutas y de tomar decisiones sobre las tareas que asigne.

La creación de nuevas funciones del sistema, si requerirá manos expertas pero al ser un sistema creado con una orientación a objetos, facilita de gran manera la inserción de nuevas funciones.

La base de datos podrá ser administrada, por el administrador del servidor donde se encuentre funcionando el sistema y la base de datos.

En el caso de que el equipo de cómputo sea apagado o reiniciado solamente se tendrá que realizar la actividad de levantar los demonios correspondientes a la base de datos PostgresSQL y el servidor de aplicaciones Tomcat para que el sistema siga funcionando.

Expectativas a futuro:

Aunque esta herramienta ya está terminada en su totalidad, permite agregar más funciones al sistema.

Entre las funciones que se podrían agregar al sistema se encuentran:

- Calendario con las fechas de entrega de las actividades y el nombre del responsable.
- Diagrama de tareas visible para los participantes del equipo.
- Página personal con más información .

BIBLIOGRAFÍA

[1] Serrano Pérez Jorge, Programación con ASP 3, Editorial Anaya Multimedia.

[2] Pressman R.S., Ingeniería de software un enfoque práctico, cuarta edición, editorial Mc-Graw Hill.

[3] Matt Hayden,Aprendiendo redes en 24 horas, Prentice Hall.

[4] Craig Hunt, TCP/IP Network Administration, editorial

[5]Yves Lepage and Raul Larrera,Unix Systems Administrator's bible,IDG Books Worldwide.Inc.

[6] Codd J.Date, An Introduction to Database Systems , the systems programming serie,third Edition, Addison Wesley.

[7] David M.Kroenke,Database Processing Fundamental, Design and Implementation, fifh Edition, Prentice Hall.

[8] Judith M Myerson, Enterprise Systems Integration, second Edition, Averbach Publications.

[9] Herbert Schildt, Manual de referencia Java 2,cuarta edición, Editorial Osborne McGraw-Hill.

[10] Marthy Hall, Servlets y JavaServlets Pages Guía práctica,editorial Prentice Hall.

[11] Steven Holzner Java 2 la biblia, Editorial Anaya Multimedia.

[12] Ivor Horton , Beginning Java 2, Editorial Wrox Pressltd.

[13] Richard L Petersen, UNIX clearly Explaines, AP Professional

[14] Ken Arnold, James Gosling, David Holmes, El lenguaje de programación Java, tercera edición, Editorial Addison Wesley.

APÉNDICES

I. Consultas a la base de datos (queries).

Para interactuar con la base de datos el sistema realiza sentencias sql (queries) correspondientes al manejador de la base de datos. Los queries que puede realizar el participante al usar el sistema son los de un usuario final u ocasional, pero los realiza por medio de pantallas lo que lo hace ser para la base de Manejo de Actividades un usuario ingenio (debido a que solo realizará los queries usando los métodos del sistema).

Los participantes del sistema usan solo el Lenguaje de Manipulación de Datos (DML), con el cual recuperan los datos del sistema.

Entre los queries que se realizan en el sistema se encuentran queries como los siguientes:

1.- Dar login y password de un usuario.

Este query permite obtener de la base de datos el id_usuario del participante con el cual corresponden el login y el password.
Antes de poder hacer alguna petición de información a la base, se tiene que instanciar la clase Base.java.

```
objetoBase=new Base(out);
  int row1=objetoBase.ejecutaSelect("select id_usuario from usuario where
login='"+login+"'");
  int row2=objetoBase.ejecutaSelect("select id_usuario from usuario where
password='"+pwd+"'");
```

2.- Insertar a la base de datos un registro para un nuevo usuario.

Estas líneas llaman al método getLlaveString contenido en el archivo Base.java, el cual se encarga de conseguir la última llave que existe en el sistema para el campo id_usuario, con esta información genera la siguiente llave y después inserta el nuevo registro del nuevo usuario en el sistema, la instrucción es ejecutada con el método ejecutaUpdate contenido en el script Base.java.

```
 String llaveUsuario=objetoBase.getLlaveString("usuario");
  String query1="insert into usuario "+

"(id_usuario,nombre,apellido_pat,apellido_mat,email,telefono,ocupacion,logi
n,password,status) values"+

"('"+llaveUsuario+"','"+nombre+"','"+apepat+"','"+apemat+"','"+email+"','"+
tel+"','"+ocupacion+"','"+login+"','"+pwd+"','R')";

int row3=objetoBase.ejecutaUpdate(query1);
```

3.- Modificar el estado de un objeto después de que ha sido revisado.

Este query permite cambiar el estado de un objeto después de que ha sido revisado por algún integrante del equipo asignado para su revisión.

```
String query1="update integrantes set situacion='"+revision+"' where
objeto.id_objeto='"+
   idObjeto+"' and objeto.id_objeto=objetos_por_camino.id_objeto and "+
   "objetos_por_camino.id_camino=camino.id_camino and
camino.id_camino=bloques_por_camino.id_camino and "+
   "bloques_por_camino.id_bloque=bloque.id_bloque and
bloque.id_bloque=integrantes.id_bloque and "+
   "integrantes.id_usuario='"+idUsuario+"' and integrantes.posesion=TRUE";
```

4.- Dar información de los usuarios integrantes de un equipo para enviar un mensaje dentro del sistema.

El sistema realiza las acciones de recuperación de la información, pero para poder hacer visible los resultados de la ejecución de un query son necesarias dos cosas:

- Realizar la ejecución del query

Por ejemplo : `objeto.ejecutaSelect(query3);`

con lo que será posible manipular los datos pero no visualizarlos

- Si se desea visualizar los resultados de la consulta es necesario además obtener cada elemento de la manera siguente:

```
destino=GeneraCadena.getCadena(objeto.getQuery(),"","id_usuario","");
```

Donde se podrá indicar un nuevo nombre al resultado obtenido.

```
//mensaje al Representante
  String query3="select distinct i.id_usuario from integrantes i where
objeto.id_objeto='"+
  idObjeto+"' and objeto.id_objeto=objetos_por_camino.id_objeto and "+
  "objetos_por_camino.id_camino=camino.id_camino and
camino.objeto_activo=TRUE and "+
  "camino.id_camino=bloques_por_camino.id_camino and "+
  "bloques_por_camino.id_bloque=bloque.id_bloque and
bloque.id_bloque=integrantes.id_bloque "+
  "and i.rol_usuario='R'";

  objeto.ejecutaSelect(query3);

  String
destino=GeneraCadena.getCadena(objeto.getQuery(),"","id_usuario","");
  String query4="insert into mensajes
(id_mensaje,autor,destino,titulo_mensaje,"+
```

```
  "mensaje,tipo_mensaje,fecha_mensaje,status) values("+

"'"+objeto.getLlaveString("mensajes")+"','"+idUsuario+"','"+destino+"','Con
firmaci&oacute;r"+" de Revisi&oacute;n de Objeto','"+
  "El usuario ha <b>"+decision+"</b> su participaci&oacute;n en el proceso
"+
  "de revisi&oacute;n del Objeto: <b>"+nombreObjeto+"</b>.<br>"+
  "Fecha de Confirmaci&oacute;n : "+Fecha.getFecha("MEDIUM")+ " a la hora:
"+
   Fecha.horaDeHoy()+"','S','"+Fecha.getFechaActualPosgres()+"','N')";

  objeto.ejecutaUpdate(query2);
  objeto.ejecutaUpdate(query4);
```

En este ejemplo la variable destino contiene el valor de la llave id_usuario, correspondiente al usuario al que se le enviará un mensaje.

5.- Eliminar un usuario de un equipo por medio de su llave de identificación.

Esta instrucción permite borrar un integrante de un equipo en el sistema.

```
row1+=objeto.ejecutaUpdate("delete from equipo where
id_usuario='"+idBorra+"' and id_equipo='"+idEquipo+"'");
```

6.- Eliminar un diagrama de ruta de la base de datos.

Esta otra permite borrar una ruta y su diagrama de la base de datos del sistema.

```
for(int i=0;i<row2;i++)
 {
  String query3="delete from historia_revision where
id_bloque='"+bloque[i]+"'";
  objeto.ejecutaUpdate(query3);

  query3="delete from integrantes where id_bloque='"+bloque[i]+"'";
  objeto.ejecutaUpdate(query3);

  query3="delete from bloque where id_bloque='"+bloque[i]+"'";
  objeto.ejecutaUpdate(query3);

  query3="delete from bloques_por_camino where id_bloque='"+bloque[i]+"'";
  objeto.ejecutaUpdate(query3);

 }

  String query3="delete from camino where id_camino='"+idRuta+"'";
  objeto.ejecutaUpdate(query3);

 //Borrado del directorio de la Ruta

 File liga=new File(rutaAbs+"archivos_actividades/"+idRuta+"/");
```

7.- Modificar datos de usuario por medio de su password .

Este query permite cambiar los datos a los participantes que deseen actualizar sus datos en el sistema.

```
Base objetoBase=new Base(out);
  int row1=objetoBase.ejecutaSelect("select id_usuario from usuario where
login='"+login+"'");
  int row2=objetoBase.ejecutaSelect("select id_usuario from usuario where
password='"+clave+"'");
  int row3=objetoBase.ejecutaSelect("select status from usuario where
id_usuario='"+idUsuario+"'");
  String
status=GeneraCadena.getCadena(objetoBase.getQuery(),"","status","");

String query1=" update usuario  set
nombre='"+nombre+"',apellido_pat='"+apepat+

"',apellido_mat='"+apemat+"',email='"+email+"',telefono='"+tel+"',ocupacion
='"+ocupacion+
  "',login='"+login+"',password='"+clave+"',status='R' where
id_usuario='"+idUsuario+"'";
```

8.- Archivo JSP con queries que llama los métodos de la clase Base.java

Aunque los queries son escritos en los archivos jsp, no son realizados hasta que se solicita su ejecución a un archivo java, los cuales fueron creados como librerías para controlar las interacciones del sistema con la base de datos.

Las sentencias sql son de gran importancia, ya que gracias a ellas el sistema puede lograr la interacción con la base de datos.

Esta sección de script contiene los queries que permiten obtener los datos de un usuario agregado al sistema por un representante y enviarle un correo de notificación de participación en un equipo.

```
String query3="select nombre_equipo from equipo where
id_equipo='"+idEquipo+"'";
  int row10=objetoBase.ejecutaSelect(query3);
  ResultSet resultado3=objetoBase.getQuery();
  String
nombreEquipo=GeneraCadena.getCadena(resultado3,"","nombre_equipo","");

  String llaveUsuario=objetoBase.getLlaveString("usuario");
  String query1="insert into usuario"+

"(id_usuario,nombre,apellido_pat,apellido_mat,email,telefono,ocupacion,logi
n,password,status)"+
```

```
"values('"+llaveUsuario+"','"+nombre+"','"+appat+"','"+apmat+"','"+email+"'
,'"+tel+"','"+ocupacion+"','"+llaveUsuario+"','"+llaveUsuario+"','I')";

  String query2="insert into equipo
(id_equipo,id_usuario,rol_usuario,nombre_equipo,status)  values"+
  "('"+idEquipo+"','"+llaveUsuario+"','I','"+nombreEquipo+"','I')";

  //ENVIO DE CORREO

  String mensaje=""+
  "El usuario fue creado y agregado exitosamente a su Equipo, el sistema ha
enviado un Correo para notificarle.<br>";

   //CREACION DEL USUARIO Y ENVIO DE CORREO, SI EXISTE EL CORREO SE CREA EL
USUARIO


   Mail.enviaCorreo(email,"Nuevo Usuario",MailMensajes.altaUsuario(nombre+"
"+appat+" "+apmat,representante,nombreEquipo,llaveUsuario,llaveUsuario));
   objetoBase.ejecutaUpdate(query1);
   objetoBase.ejecutaUpdate(query2);
   out.print(Mensaje.getMensaje(mensaje+otro,""));
   objetoBase.cierraConexion();
```

II. Algunos programas del sistema Manejo de Actividades.

Este programa se llama Base.java fue programado para permitir la conexión a la base de datos, en el se incluye el nombre de la base de datos, el drive que controla el tipo de manejador de bases de datos que se empleara, también se coloca el nombre de usuario y el password con el que se conectara a la base.

Este script también permite controlar los métodos para ejecutar un select, un update y un insert sobre la base. Además realiza la creación de llaves para cada uno de los elementos del sistema y verifica la llave anterior para controlar la secuencia.

Las conexiones a la base de datos se harán siempre con la ayuda de llamadas a este programa.

```
package paquete1;
import java.sql.*;
import javax.servlet.jsp.*;
import java.io.*;
public class Base
{
 ResultSet resultadoQuery;
 Connection conexion;
 Statement cadenaQuery;
 String URLBase;
 String sentenciaSQL;
 ResultSet resultadoSelect;
 int resultadoUpdate;


 //CONSTRUCTOR

public Base(JspWriter out) throws IOException
 {

URLBase="jdbc:postgresql://titan.dgsca.unam.mx/actividades?user=pathadmin&p
assword=path.104.sql";

  try

   Class.forName("org.postgresql.Driver");
   conexion= DriverManager.getConnection(URLBase);
  }

  catch (SQLException e)
  {
    out.println("IMPOSIBLE CONECTARSE A LA BASE DE DATOS " + e);
    System.exit(-1);
  }
  catch(ClassNotFoundException e)
  {
   out.println("EL DRIVE NO FUE ENCONTRADO." + e);
  }
```

```
 }//constructor

 public void finalize() throws Throwable
 {
  if(conexion!=null)
    conexion.close();
 }//finalize


 public void cierraConexion()throws SQLException
 {
  if(conexion!=null)
    conexion.close();
 }

//METODO QUE EJECUTA UN SELECT DEVUELVE EL NUMERO DE RENGLONES
SELECCIONADOS

 public int ejecutaSelect(String sentenciaSQL) throws SQLException
 {
  Statement query=conexion.createStatement();
  this.resultadoSelect=query.executeQuery(sentenciaSQL);
  boolean sig=resultadoSelect.next();
  int indice=0;
  while(sig)
  {
   indice++;
   sig=resultadoSelect.next();
  }
  resultadoSelect.first();

  return indice;
 }

 //REGRESA UN OBJETO TIPO ResultSet

 public ResultSet getQuery()
 {
  return resultadoSelect;
 }

 public int ejecutaUpdate(String sentenciaSQL) throws SQLException
 {
  int row;
  Statement query=conexion.createStatement();
  row=query.executeUpdate(sentenciaSQL);
  return row;
 }//update


 public String getLlaveString(String tabla) throws SQLException
 {
  String llaveValor="";
  Statement query=conexion.createStatement();
  ResultSet resultado=query.executeQuery("select
nextval('sec_"+tabla+"')");
```

```
  ResultSet resultado2=query.executeQuery("select last_value from
sec_"+tabla);
  resultado2.next();
  int nextval= resultado2.getInt("last_value");
  for(int i=0;i<10;i++)
  {
    llaveValor=(char) (65+ nextval%26)+llaveValor;
    nextval=(int)(nextval/26);
  }
  return llaveValor;
 }

 public int getLlaveInt(String tabla) throws SQLException
 {
  Statement query=conexion.createStatement();
  ResultSet resultado=query.executeQuery("select
nextval('sec_"+tabla+"')");
  ResultSet resultado2=query.executeQuery("select last_value from
sec_"+tabla);
  resultado2.next();
  return(resultado2.getInt("last_value"));
 }


}//class
```

Este programa se llama Cadenas.java y se programo para controlar la creación de cadenas en el diagrama de rutas. Permite controlar la posición de los elementos en la ruta.

```
package paquete1;
import java.util.*;

public class Cadenas
{
 public static String generaTabla(int[][] matriz,Vector usuariosPorBloque)
 {
  int cuenta=1;
  String encabezado=""+
  " <table width=\"100%\" border=\"3\" bordercolor=\"#000066\"
bgcolor=\"#f3f3f3\" height=\"460\">\n"+
  " <tr>"+
    "<td bgcolor=\"#000066\"><font color=\"f3f3f3\" face=\"Arial,
Helvetica, sans-serif\"><b>Ruta Generada\n"+
   "</b></font></td>\n"+
  "</tr>\n"+
  "<tr>\n"+
    "<td>\n"+
      "<table width=\"43%\" border=\"1\" bgcolor=\"#ffffff\"
bordercolor=\"f3f3f3\">\n";

  String celdaVacia="<td bgcolor=\"#F3f3f3\"> </td>\n";

  String etapa1=""+

   "   <td width=bgcolor=\"#FFFFFF\">\n"+
            "<div align=\"center\"><img src=\"imagenes/mono.gif\"
width=\"50\" height=\"50\">\n"+
             " <font face=\"Arial, Helvetica, sans-serif\"
color=\"#000099\" size=\"2\">";
   String etapa2="</font>\n</div>\n</td>\n";

  String inicio=""+
   "   <td height=\"32\" nowrap><b><img src=\"imagenes/inicio.gif\"
width=\"65\" height=\"30\"></b></td>\n";

   String flecha=""+
        " <td height=\"32\"  nowrap><img src=\"imagenes/flecha3.gif\"
width=\"50\" height=\"30\"></td>\n";


   String cuadro1= ""+
    "<td width=\"10%\" bgcolor=\"#000099\" height=\"32\" nowrap>\n"+
          " <div align=\"center\"><b><font face=\"Arial, Helvetica, sans-
serif\" color=\"f3f3f3\">\n"+
           " Bloque ";
   String cuadro2="</font></b></div>\n</td>\n";

    String pregunta=""+
     "<td bgcolor=\"#FFFFFF\" height=\"32\" nowrap>\n"+
          "<div align=\"center\"><b><img src=\"imagenes/Qman.gif\"
width=\"32\" height=\"32\"></b></div>\n"+
```

```
          "</td>\n";

     String fin="<td height=\"32\" nowrap><b><img src=\"imagenes/fin.gif\"
width=\"85\" height=\"30\"></b></td>\n";

     String flechaNo="<td width=\"6%\" bgcolor=\"#FFFFFF\">\n"+
           "<div align=\"center\"><img src=\"imagenes/flechano.gif\"
width=\"30\" height=\"50\">\n"+
          "</div>\n"+
          "</td>\n";

     String mono="    <td width=\"10%\" bgcolor=\"#FFFFFF\">\n"+
           "<div align=\"center\"><img src=\"imagenes/mono.gif\"
width=\"50\" height=\"50\">\n"+
            "<font face=\"Arial, Helvetica, sans-serif\"
color=\"#000099\" size=\"2\">Continua al </font>\n"+
           "</div>\n"+
          "</td>\n";


  String tabla=encabezado;
  int[] pila=new int[(matriz[0].length)];
  for(int i=0;i<matriz.length;i++)
  {
   tabla+="<tr>";
   for(int j=0;j<matriz[i].length;j++)
   {
    if(matriz[i][j]==-1)
       tabla+=inicio;
    else if(matriz[i][j]==-2)
       tabla+=flecha;
    else if(matriz[i][j]==-3)
       tabla+=pregunta;
    else if(matriz[i][j]==-4)
       tabla+=flechaNo;
    else if(matriz[i][j]==-5)
       tabla+=fin;
    else if(matriz[i][j]==-6)
       tabla+=mono;

    else if(matriz[i][j]<=-70)
    {
       tabla+=cuadro1;
       cuenta=-1*(70+matriz[i][j]);
       tabla+=cuenta;
       tabla+=cuadro2;
    }

    else if(matriz[i][j]==-7)
    {
       tabla+=cuadro1;
       cuenta=matriz[i-1][j];
       tabla+=cuenta;
       tabla+=cuadro2;
    }
    else if(matriz[i][j]>0)    //usuarios
    {
```

```
      String nombreInt;
      String[] nombre=(String[])usuariosPorBloque.elementAt(matriz[i][j]-1);
      nombreInt=nombre[(pila[j])];
      if(nombre.length > pila[j])
       pila[j]++;
       tabla+=etapa1;
       tabla+=nombreInt;
       tabla+=etapa2;
      }

      else
        tabla+=celdaVacia;
     }//for
     tabla+="</tr>";
    }
  tabla+="</table></td></tr><tr><td bgcolor=#00066>"+
        "<div align=\"left\"><b><font face=\"Arial, Helvetica, sans-serif\"
size=\"2\" color=\"white\">"+
         "<form>"+
         "<center><input type=button value= \"Regresar \"
onclick=javascript:history.go(-1);></form>"+
       "</font></b></div></center>"+
      "</td></tr></table>";

  return tabla;
  }
 }
```

Estos dos scripts llamados Sesiones.java y MiSesion.java permiten controlar la as actividades de cada usuario en el sistema.

Una vez que el usuario se ha registrado en Manejo de Actividades, el sistema le asignará una llave, por medio de esa llave, el usuario será reconocido por el sistema desde el momento en que inicie su sesión con su login y password y hasta que de fin a su sesión.

```
package paquetel;
public class Sesiones
{
 String idUsuario;
 String login;
 public Sesiones(String login,String idUsuario)
 {
  this.login=login;
  this.idUsuario=idUsuario;
 }

 public String getLogin()
 {
  return login.trim();
 }

 public String getIdusuario()
 {
  return idUsuario.trim();
 }
}//class
```

```
package paquetel;
import javax.servlet.*;
import javax.servlet.http.*;
public class MiSesion
{
 HttpServletRequest request;
 HttpSession sesion;
 Sesiones miSesion;

 public MiSesion(HttpServletRequest request)
 {
  this.request=request;
  this.sesion=request.getSession(true);
  synchronized(sesion)
  {
   this.miSesion=(Sesiones)sesion.getValue("sesionActividades");
  }
 }


 public void setSesion(String loginUsuario,String idUsuario)
 {
  synchronized(sesion)
  {
   Sesiones nuevaSesion=new Sesiones(loginUsuario,idUsuario);
```

```
    sesion.putValue("sesionActividades",nuevaSesion);
   }
  }

  public boolean isSesion()
  {
   if(miSesion==null)
    return false;
   else
     return true;
  }

  public String getLogin()
  {
   if(isSesion())
    return miSesion.getLogin().trim();
   else
   return null;
   }

  public String getIdUsuario()
  {
   if(isSesion())
    return miSesion.getIdUsuario().trim();
   else
   return null;
  }

  public void finSesion()
  {
   if(isSesion())
   sesion.invalidate();
  }

}
```

Estos scripts Mail.java y MailMensajes.java permiten enviar mensajes de correo electrónico para los usuarios que están siendo invitados a participar en el sistema.

Mail.java interactua con el sistema de envió de correo del servidor donde se encuentra el sistema y se encarga de darle la orden de enviar el correo.

MailMensajes.java es el mensaje que le como invitación al usuario.

```
package paquete1;
import javax.mail.*;
import java.util.Properties;
import javax.mail.internet.*;

public class Mail
{

  public static void enviaCorreo(String host,String from,String to,String
titulo,String info) throws SendFailedException,MessagingException
  {

   Properties propiedades=System.getProperties();
   propiedades.put("mail.smtp.host",host);
   Session miSesion=Session.getDefaultInstance(propiedades,null);
   MimeMessage mensaje=new MimeMessage(miSesion);
   mensaje.setFrom(new InternetAddress(from));
   mensaje.addRecipient(Message.RecipientType.TO,new InternetAddress(to));
   mensaje.setSubject(titulo);
   mensaje.setContent(info,"text/html");
   Transport.send(mensaje);
  }

  public static void enviaCorreo(String to,String titulo,String info)
throws SendFailedException,MessagingException
  {
   String host="titan.dgsca.unam.mx";
   String from="actividades@titan.dgsca.unam.mx";
   Properties propiedades=System.getProperties();
   propiedades.put("mail.smtp.host",host);
   Session miSesion=Session.getDefaultInstance(propiedades,null);
   MimeMessage mensaje=new MimeMessage(miSesion);
   mensaje.setFrom(new InternetAddress(from));
   mensaje.addRecipient(Message.RecipientType.TO,new InternetAddress(to));
   mensaje.setSubject(titulo);
   mensaje.setContent(info,"text/html");
   Transport.send(mensaje);
  }

 }//class

package paquete1;
public class MailMensajes
{

public static String  altaUsuario(String to,String rep,String equipo,String
login,String clave)
```

```
{
 Stri. newUser=""+

"<html> <head>"+
"<title>-tividades</title>"+
"<meta ht>-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-
1\">"+
"</head>"+
"<body bgco.or=\"#FFFFFF\" text=\"#000000\">"+
"<table widt.=\"88%\" border=\"1\" height=\"91\">"+
  "<tr>"+
    "<td bgcolor=\"#000099\" width=\"94%\">"+
      "<div al.gn=\"left\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"F3F3F3\"><b>SISTEMA"+
       " MANEJO DE ACTIVIDADES</b></font></div>"+
    "</td>"+
    "<td bgcolor=\"#000099\" width=\"6%\"><img
src=http://daa.dgsca.unam.mx/actividades/jsp/imagenes/Boxin.gif
width=\"32\" height=\"32\"></td>"+
  "</tr>"+
  "<tr>"+
    "<td bgcolor=\"f3f3f3\" colspan=\"2\">"+
      "<p><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Hola <b>"+to+"</b>:</font></p>"+
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Este"+
       "es un mensaje del sistema manejo de Actividades</font></p>"+
      "<p align=\"left\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">El"+
       " usuario <b> "+rep+" </b> lo ha inscrito en el equipo
<b>"+equipo+" </b>dentro"+
       " del sitio &quot;Manejo de Actividades&quot; con la finalidad de
que participe"+
       " en las actividades de su equipo. Debe visitar el sitio para
confirmar"+
       " su participaci&oacute;n, en la secci&oacute;n de tareas de su
p&aacute;gina"+
       " personal. El sistema dio de alta una cuenta para usted con los
siguientes"+
       " datos:</font></p>"+
      "<ul>"+
       "<li><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Login <b>"+login+"</b></font></li>"+
       "<li><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Clave <b>"+clave+"</b></font></li>"+
      "</ul>"+
      "<p><font face=\"Arial, Helvetica, sans-serif\" color=\"#000066\">Con
estos datos"+
       " podr&aacute; entrar a su Pagina Personal en la cu&aacute;l
estar&aacute;"+
       " en comunicaci&oacute;n constante con su equipo, asi como las
tareas que"+
       " se le han asignado. Le recomendamos cambiar su Login y su clave a
la brevedad"+
       " posible en el menu personal, ya que el sistema hara referencia a
su login"+
       " todo el tiempo que este en sesion.</font></p>"+
```

```
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#990000\">Visitar"+
        "el sitio</font></p>"+
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">"+
        "<a
href=\"http://daa.dgsca.unam.mx/actividades\">http://daa.dgsca.unam.mx/acti
vidades</a></font></p>"+
      "</td>"+
  "</tr>"+
  "<tr>"+
    "<td bgcolor=\"#000099\" colspan=\"2\">"+
      "<div align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"F3F3F3\"><b>"+
       "actividades@titan.dgsca.unam.mx</b></font></div>"+
    "</td>"+
  "</tr>"+
"</table>"+
"</body>"+
"</html>";

 return newUser;
 }

public static String  agregaIntegrante(String to,String rep,String equipo)
{
 String newUser=""+

"<html>\n<head>"+
"<title>Actividades</title>"+
"<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-
1\">"+
"</head>"+
"<body bgcolor=\"#FFFFFF\" text=\"#000000\">"+
"<table width=\"98%\" border=\"1\" height=\"91\">"+
  "<tr>"+
    "<td bgcolor=\"#000099\" width=\"94%\">"+
      "<div align=\"left\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"F3F3F3\"><b>SISTEMA"+
        " MANEJO DE ACTIVIDADES</b></font></div>"+
    "</td>"+
    "<td bgcolor=\"#000099\" width=\"6%\"><img
src=http://daa.dgsca.unam.mx/actividades/jsp/imagenes/Boxin.gif
width=\"32\" height=\"32\"></td>"+
  "</tr>"+
  "<tr>"+
    "<td bgcolor=\"f3f3f3\" colspan=\"2\">"+
      "<p><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Hola <b>"+to+"</b>:</font></p>"+
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Este"+
        "es un mensaje del sistema manejo de Actividades</font></p>"+
      "<p align=\"left\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">El"+
        " usuario <b> "+rep+" </b> lo ha inscrito en el equipo
<b>"+equipo+" </b>dentro"+
```

```
        " del sitio &quot;Manejo de Actividades&quot; con la finalidad de
que participe"+
        " en las actividades de su equipo. Debe visitar el sitio para
confirmar"+
        " su participaci&oacute;n, en la secci&oacute;n de tareas de su
p&aacute;gina"+
        " personal, mientras no se reporte, permanecer&aacute; como
integrante Inactivo </font></p>"+
         "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#990000\">Visitar"+
        "el sitio</font></p>"+
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">"+
      "<a
href=\"http://daa.dgsca.unam.mx/actividades\">http://daa.dgsca.unam.mx/acti
vidades</a></font></p>"+
        "</td>"+
  "</tr>"+
  "<tr>"+
    "<td bgcolor=\"#000099\" colspan=\"2\">"+
      "<div align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"F3F3F3\"><b>"+
       "actividades@titan.dgsca.unam.mx</b></font></div>"+
    "</td>"+
  "</tr>"+
"</table>"+
"</body>"+
"</html>";

 return newUser;
 }


public static String  getRecuerda(String clave,String login)
{
 String newUser=""+

"<html>\n<head>"+
"<title>Actividades</title>"+
"<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-
1\">"+
"</head>"+
"<body bgcolor=\"#FFFFFF\" text=\"#000000\">"+
"<table width=\"66%\" border=\"1\" height=\"91\">"+
  "<tr>"+
    "<td bgcolor=\"#000099\" width=\"94%\">"+
      "<div align=\"left\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"F3F3F3\"><B>SISTEMA"+
       " MANEJO DE ACTIVIDADES</b></font></div>"+
    "</td>"+
    "<td bgcolor=\"#000099\" width=\"6%\"><img
src=http://daa.dgsca.unam.mx/actividades/jsp/imagenes/BoxIn.gif
width=\"32\" height=\"32\"></td>"+
  "</tr>"+
  "<tr>"+
    "<td bgcolor=\"f3f3f3\" colspan=\"2\">"+
```

```
      "<p><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Hola <b>"+login+"</b>:</font></p>"+
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">Este"+
        "es un mensaje del sistema manejo de Actividades</font></p>"+
      "<p align=\"left\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">"+
        " El motivo de este correo es debido a que solicitó recuperar su
clave de "+
        " usuario dentro del sistema Manejo de Actividades.<br>"+
        " Su clave es:<br>"+
        " <center><b>"+clave+"</b><br>"+
        " con la cual podrá seguir accesando al sistema.<br>"+
         "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#990000\">Visitar"+
        "el sitio</font></p>"+
      "<p align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"#000066\">"+
        "<a
href=\"http://daa.dgsca.unam.mx/actividades\">http://daa.dgsca.unam.mx/acti
vidades</a></font></p>"+
      "</td>"+
  "</tr>"+
  "<tr>"+
    "<td bgcolor=\"#000099\" colspan=\"2\">"+
      "<div align=\"center\"><font face=\"Arial, Helvetica, sans-serif\"
color=\"F3F3F3\"><b>"+
       "actividades@titan.dgsca.unam.mx</b></font></div>"+
    "</td>"+
  "</tr>"+
"</table>"+
"</body>"+
"</html>";

 return newUser;
 }

;
```

Con los script Registro.jsp y RegistroProceso.jsp, el nuevo participante puede registrarse en el sistema de Manejo de Actividades.

Estos scripts son los encargados de recibir y verificar los datos que los usuarios ingresan a un formulario HTML. Reciben los datos y son los encargados de ingresar en la base de datos los nuevos valores.

```
<head>
<script language=javascript>
function validaRegistro()
{
 if(document.forma.nombre.value==0)
   alert('Debe especificar Un Nombre');
 else if(document.forma.apepat.value==0)
   alert('Debe especificar El apellido Paterno');
 else if(document.forma.apemat.value==0)
   alert('Debe especificar Un Apellido Materno');
 else if(document.forma.tel.value==0)
   alert('Debe especificar Un Telefono de referencia');
  else if(document.forma.email.value.indexOf("@")<1 ||
document.forma.email.value.indexOf(".")<1 ||
document.forma.email.value.indexOf(" ")>0)
    alert('Debe especificar Una Direccion de Correo valida!');
  else if(document.forma.ocupacion.value==0)
    alert('Debe especificar Una ocupacion');
  else if(document.forma.login.value==0)
    alert('Debe especificar Un Login!');
  else if(document.forma.pwd.value==0)
    alert('Debe especificar Un Password!');
  else if(document.forma.pwd.value != document.forma.pwd2.value)
    alert('Los Passwords no coinciden!');
  else
    document.forma.submit();
 }
</script>
</head>
 <form name="forma" method="post" action="index.jsp">
  <table width="100%" border="1" bgcolor="f3f3f3" bordercolor="#FFFFFF"
height="266">
    <tr>
      <td colspan="2" bgcolor="#000066" height="15"><font
color="F3F3F3"><b><font face="Arial, Helvetica, sans-serif">REGISTRO
        DE USUARIOS</font></b></font></td>
  </tr>
  <tr>
      <td width="41%" height="12">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Nombre(s)</font></b></div>
    </td>
      <td width="59%" height="12">
        <input type="text" name="nombre" maxlength="50">

    </td>
  </tr>
  <tr>
      <td width="41%" height="9">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#003366">Apellido
          Paterno</font></b></div>
```

```
    </td>
      <td width="59%" height="9">
        <input type="text" name="apepat" maxlength="50">

    </td>
  </tr>
  <tr>
      <td width="41%" height="4">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Apellido
        Materno</font></b></div>
    </td>
      <td width="59%" height="4">
        <input type="text" name="apemat" maxlength="50">
    </td>
  </tr>
  <tr>
      <td width="41%" height="19">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Correo
        Electr&oacute;nico</font></b></div>
    </td>
      <td width="59%" height="19">
        <input type="text" name="email" size="35" maxlength="100">
    </td>
  </tr>
  <tr>
      <td width="41%" height="11">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Tel&eacute;fono</font></b></div>
    </td>
      <td width="59%" height="11">
        <input type="text" name="tel" maxlength="30">
    </td>
  </tr>
  <tr>
      <td width="41%" height="37">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Ocupaci&oacute;n</font></b></div>
    </td>
      <td width="59%" height="37">
        <textarea name="ocupacion" cols="30" rows="5"></textarea>
    </td>
  </tr>
  <tr>
      <td width="41%" height="8">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Login</font></b></div>
    </td>
      <td width="59%" height="8">
        <input type="text" name="login" size="10" maxlength="10">
    </td>
  </tr>
  <tr>
    <td width="41%" height="24">
      <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Password</font></b></div>
```

```
    </td>
    <td width="59%" height="24">
        <input type="password" name="pwd" size="10" maxlength="10">
    </td>
  </tr>
  <tr>
      <td width="41%" bgcolor="f3f3f3" height="20">
        <div align="right"><b><font face="Arial, Helvetica, sans-serif"
color="#000066">Confirma
        Password</font></b></div>
    </td>
      <td bgcolor="f3f3f3" width="59%" height="20">
        <input type="password" name="pwd2" size="10" maxlength="10">
    </td>
  </tr>
  <tr>
      <td bgcolor="#000066" colspan="2" height="2">
        <div align="center"> <font color=#f3f3f3>
            <input type=hidden name=opcionCen value=registroProceso.jsp>
          <input type=button value="Enviar datos"
onClick='validaRegistro()';>
          <input type="reset" name="Submit2" value="Limpiar"></font>

        </div>
    </td>
  </tr>
</table>
</form>
```


Script RegistroPoceso.jsp

```
<%@ page import="paquete1.*,java.sql.*" %>

<%
  String nombre=  request.getParameter("nombre");
  String apepat=  request.getParameter("apepat");
  String apemat=  request.getParameter("apemat");
  String ocupacion=request.getParameter("ocupacion");
  String tel=     request.getParameter("tel");
  String email=   request.getParameter("email");
  String login=   request.getParameter("login");
  String pwd=     request.getParameter("pwd");

  Base objetoBase;

  try
  {
  objetoBase=new Base(out);
  int row1=objetoBase.ejecutaSelect("select id_usuario from usuario where
login='"+login+"'");
  int row2=objetoBase.ejecutaSelect("select id_usuario from usuario where
password='"+pwd+"'");
  String mensaje="";

  if(row1>0)
    mensaje="<br><center>EL LOGIN QUE ELIGIO YA EXISTE, DEBE CAMBIARLO";
  if(row2>0)
```

```
    mensaje="<br><center>EL PASSWORD  QUE ELIGIO YA EXISTE, DEBE
CAMBIARLO";
  if(row1>0 && row2>0)
     mensaje="<br><center>EL LOGIN Y EL PASSWORD QUE ELIGIO YA EXISTEN,
DEBE CAMBIARLOS";


  if(row1>0  || row2>0)
  {
   objetoBase.cierraConexion();
   throw new MiException(mensaje);
  }

  String llaveUsuario=objetoBase.getLlaveString("usuario");
  String query1="insert into usuario "+

"(id_usuario,nombre,apellido_pat,apellido_mat,email,telefono,ocupacion,logi
n,password,status) values"+

"('"+llaveUsuario+"','"+nombre+"','"+apepat+"','"+apemat+"','"+email+"','"+
tel+"','"+ocupacion+"','"+login+"','"+pwd+"','R')";

  int row3=objetoBase.ejecutaUpdate(query1);
  out.print(Mensaje.getMensaje("SUS DATOS FUERON ACEPTADOS. USUARIO
REGISTRADO!!","",""));
  objetoBase.cierraConexion();
 }
 catch(MiException e)
 {
   out.print(Mensaje.getMensaje(e.toString(),""));
 }
 catch(SQLException e)
 {
  out.print(Mensaje.getMensaje(e.toString()+"ERROR EN LA CONEXION EN LA
BASE DE DATOS","",""));
 }

%>
```

Este archivo nos permite hacer uso de las clases Sesiones.java y MiSesion.java y nos permite conseguir una sesión cuando nos registramos en el sistema.

```
<!-- ------------------------- CODIGO JAVA ------------->


<%@ page import="java.sql.*,paquete1.*" %>
<%
 String login=request.getParameter("login");
 String password=request.getParameter("password");

 String query1="select login from usuario where login='"+login+"'";
 String query2="select password from usuario where
password='"+password+"'";
 String query3="select id_usuario from usuario where login='"+login+"' and
password='"+password+"'";

 Base objetoBase;

 try
 {
  objetoBase=new Base(out);

  if(objetoBase.ejecutaSelect(query1)==0)
   throw new MiException("SU LOGIN ES INCORRECTO");
  if(objetoBase.ejecutaSelect(query2)==0)
   throw new MiException("SU PASSWORD ES INCORRECTO");

  int row1=objetoBase.ejecutaSelect(query3);
 if(row1==0)
   throw new MiException("SU LOGIN O SU  PASSWORD ES INCORRECTO");

  ResultSet resultado3=objetoBase.getQuery();
  String idUsuario=GeneraCadena.getCadena(resultado3,"","id_usuario","");
  idUsuario=idUsuario.trim();

  //OBTENCION DE SESION

  MiSesion sesionUsuario=new MiSesion(request);
   if(!sesionUsuario.isSesion())
   {
    sesionUsuario.setSesion(login,idUsuario);
    out.print("<script language=javascript>alert('SU SESION HA SIDO
ACTIVADA BIENVENIDO(A) " +login+"');</script>");
   }

/*   if(!(login.equals(sesionUsuario.getLogin())) ||
!(idUsuario.equals(sesionUsuario.getIdUsuario())))
   {

    throw new MiException("LOS DATOS DE LA SESION DE SU COMPUTADORA NO
CORRESPONDEN A LOS QUE PROPORCIONO<br> DEBE HACER LOGOUT PARA ELIMINIAR LA
SESION ANTERIOR");
   }*/
```

```
   out.print("<script language=javascript>
window.location.href='index.jsp';</script>");

 }//try

 catch(SQLException e)
 {
  out.print(Mensaje.getMensaje("<center>ERROR EN LA CONEXION"+e,""));
 }

 catch(MiException e)
 {
  out.print(Mensaje.getMensaje(e.toString(),""));

 }

  %>

<!-- ------------------------------------------------------------------------>
```

Este es un script JSP con el es posible subir archivos al sistema Manejo de Actividades. Permite agregar objetos a una ruta, siempre y cuando no haya ningún objeto en la ruta.

```
<%@ page import="java.sql.*,paquete1.*,java.util.*,java.io.*" %>

<%
  try
 {
  MiSesion sesionObjeto3=new MiSesion(request);
  if(!sesionObjeto3.isSesion())
   throw new MiException("Para poder crear un Objeto debe estar
registrado");

 String idUsuario=sesionObjeto3.getIdUsuario();
 String nombreObjeto=request.getParameter("nombreObjeto");
 String descripcionObjeto=request.getParameter("descripcionObjeto");
 String tipoObjeto=request.getParameter("tipoObjeto");
 String idRutaEleccion=request.getParameter("idRutaEleccion");

 Base objeto2=new Base(out);

//OBTENCION DEL id_bloque PRINCIPAL

   String query2="select id_bloque from integrantes where
id_camino='"+idRutaEleccion+"' and rol_usuario='R'";
   if(objeto2.ejecutaSelect(query2)<=0)
     throw new MiException("NO SE ENCONTRO EL REPRESENTANTE CORRESPONDIENTE
A LA RUTA");

   String
idBloqueInicio=GeneraCadena.getCadena(objeto2.getQuery(),"","id_bloque","")
;


//OBTENCION DE LA RUTA CORRESPONDIENTE
 String query3="select diagrama from camino where
id_camino='"+idRutaEleccion+"'";
 int row3=objeto2.ejecutaSelect(query3);
 if(row3==0)
   throw new MiException("NO SE ENCONTRO EL DIRECTORIO CORRESPONDIENTE A LA
RUTA");

 String
rutaHome=GeneraCadena.getCadena(objeto2.getQuery(),"","diagrama","");



 String llaveObjeto=objeto2.getLlaveString("objeto");
 File directorio=new File(rutaHome+"/"+llaveObjeto+"_OBJETO");
 if(!directorio.exists())
  {

  directorio.mkdir();
  }
  directorio=new
File(rutaHome+"/"+llaveObjeto+"_OBJETO",llaveObjeto+".obj");
```

```
DataOutputStream liga = new DataOutputStream(
                        new BufferedOutputStream(
                        new FileOutputStream(directorio)));

liga.writeChars(contenido);

liga.close();


String fecha=Fecha.getAnio()+"-"+Fecha.getMes()+"-"+Fecha.getDia();

String query1="insert into objeto
(id_objeto,id_camino,nombre_objeto,descripcion,fecha_creacion,tipo_objeto,d
ireccion_objeto,status)"+

"values('"+llaveObjeto+"','"+idRutaEleccion+"','"+nombreObjeto+"','"+descri
pcionObjeto+"','"+fecha+"','"+tipoObjeto+"','"+rutaHome+"/"+llaveObjeto+"_O
BJETO','L')";

objeto2.ejecutaUpdate(query1);

 out.print(Mensaje.getMensaje("<br> Su objeto ha sido Creado exitosamente,
presione siguiente para ",""));

objeto2.cierraConexion();
}//try

catch(MiException e)
{
 out.print(Mensaje.getMensaje(e.toString(),"regresar"));
}
catch(SQLException e)
{
 out.print(Mensaje.getMensaje(e.toString(),""));
}
catch(IOException e)
{
 out.print(Mensaje.getMensaje(e.toString(),""));
}


%>
```