# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNOLOGOS

Diseño e implementación de una aplicación web con acceso a datos para la “DIRECCIÓN DE RECURSOS HUMANOS DE LA ESCUELA POLITÉCNICA NACIONAL”

**PROYECTO PREVIO A LA OBTENCIÓN DEL TITULO DE TECNÓLOGO EN ANÁLISIS DE SISTEMAS INFORMÁTICOS**

**MARÍA ELENA TIRIRA ARÉVALO**
**mary_a19@hotmail.com**

**DIRECTORA: ING. ROSA NAVARRETE**

**rosa.navarrete@epn.edu.ec**

**Quito, Noviembre 2008**

# DECLARACIÓN

Yo Tirira Arévalo María Elena, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Tirira Arévalo María Elena

# CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Tirira Arévalo María Elena, bajo mi supervisión.

_____

Ing. Rosa Navarrete.

DIRECTOR DEL PROYECTO

## AGRADECIMIENTOS

Agradezco a Dios por brindarme la vida, una familia maravillosa, y la fortaleza para seguir siempre adelante y poder culminar una etapa en mi vida profesional.

A mis padres y hermanos por el gran apoyo que me han brindado siempre.

A mis amigos que han formado parte de mi familia, gracias por haberme brindado su respaldo incondicional.

A mi novio (Christian) por su gran ayuda y apoyo durante el desarrollo de este proyecto.

*Mary*

## DEDICATORIA

*Este trabajo dedico con todo mi amor y cariño*

*A ti Dios que me diste la oportunidad de vivir y brindarme una familia maravillosa.*

*Con todo cariño principalmente a mis padres que me dieron la vida y me han apoyado en todo momento. Gracias por todo papá y mamá por darme una carrera para mi futuro y por creer en mí. Gracias por el ejemplo que me han brindado y los valores que me han inculcado siempre.*

*A mis queridos amigos quienes siempre me demostraron su amistad en los momentos en los que más los necesite.*

*Mary*

# ÍNDICE

# 1 CAPITULO 1: INTRODUCCIÓN

## 1.1 ÁMBITO

La Dirección de Recursos Humanos maneja gran cantidad de información del personal de la Escuela Politécnica Nacional, en lo que respecta a datos personales, nombramientos, acciones de personal, contratos, solicitudes de pago, vacaciones, permisos ocasionales, horas de los profesores a tiempo parcial, ascensos de nivel escalafonario, control de asistencia del personal administrativo, servicios con contrato, nombramientos y becarios (ayudantes y auxiliares de laboratorio), etc.

## 1.2 OBJETIVOS

### 1.2.1 OBJETIVO GENERAL

Permitir que el personal de la Escuela Politécnica Nacional, tenga un acceso permanente a su información personal y de historia laboral, a través del desarrollo e implementación de una aplicación web.

### 1.2.2 OBJETIVOS ESPECÍFICOS

a) Desarrollar un módulo que permita la consulta y actualización de datos personales del empleado.
b) Desarrollar un módulo de consultas de datos referentes al historial laboral del empleado.
c) Desarrollo de un módulo para zonas de descargas (manuales de procedimientos, cronogramas de capacitación, reglamentos, resoluciones, etc.).
d) Consultar información acerca de actividades que realiza la Dirección de Recursos Humanos.

## 1.3 ALCANCE Y LIMITACIONES

### 1.3.1 ALCANCE

El sitio web permitirá:

- Acceder a información de eventos Institucionales y de interés del personal de la Escuela Politécnica Nacional.
- Consultar datos referentes al historial laboral.
- Consultar los datos personales y actualizarlos.
- Permitir la descarga de: manuales de procedimientos, cronogramas de capacitación, reglamentos y resoluciones.

### 1.3.2 LIMITACIONES

El sitio web no permitirá:

- Transacciones financieras de cobro o pago.
- Servicio de Chat.
- Servicio de foro.

# 2 CAPITULO 2: MARCO DE REFERENCIA

## 2.1 INGENIERÍA WEB

Durante la última década hemos asistido al crecimiento vertiginoso del desarrollo y uso de aplicaciones y sistemas Web cada vez más complejos y sofisticados.

Desafortunadamente, dicha complejidad no parece estar acompañada de los mecanismos adecuados que garanticen la calidad de unos sistemas de los que cada día tenemos mayor dependencia a nivel social, funcional y económico.

Esta carencia de calidad ha venido generando una preocupación creciente entre la comunidad científica y técnica involucrada en el desarrollo Web. Así pues, en los últimos años surgen varias iniciativas con el objetivo de poner cierto orden dentro de la maraña que estamos creando y en la que nos movemos habitualmente.

### 2.1.1 EL PROCESO DE INGENIERÍA WEB

Características como inmediatez y evolución y crecimiento continuo, nos llevan a un proceso incremental y evolutivo, que permite que el usuario se involucre activamente, facilitando el desarrollo de productos que se ajustan mucho a lo que éste busca y necesita.

Según Pressman, las actividades que formarían parte del marco de trabajo incluirían las tareas a bajo numeradas. Dichas tareas serían aplicables a cualquier aplicación W*eb,* independientemente del tamaño y complejidad de la misma. Las actividades que forman parte del proceso son: formulación, planificación, análisis, modelización, generación de páginas, test y evaluación del cliente.

### 2.1.2 CONTROL Y GARANTÍA DE LA CALIDAD

“Una de las tareas colaterales que forman parte del proceso es el Control y Garantía de la Calidad (CGC). Todas las actividades CGC de la ingeniería de software tradicional son: establecimiento y supervisión de estándares, revisiones técnicas formales, análisis, seguimiento y registro de informes, etc, son igualmente aplicables a la Ingeniería Web. Sin embargo, en la Web toman especial relevancia para valorar la calidad aspectos como: Usabilidad, Funcionabilidad, Fiabilidad, Seguridad, Eficiencia y Mantenibilidad” (OLS, 266).

### 2.1.3 CONTROL DE LA CONFIGURACIÓN

“Establecer mecanismos adecuados de control de la configuración para la *Ingeniería Web* es uno de los mayores desafíos a los que esta nueva disciplina se enfrenta. La *Web* tiene características únicas que demandan estrategias y herramientas nuevas. Hay cuatro aspectos importantes a tener en cuenta en el desarrollo de tácticas de control de la configuración para la *Web “*: (DAR99,191).

http://www.informandote.com/jornadasIngWEB/articulos/jiw01.pdf

*Contenido*: Considerando la dinámica con la que el contenido se genera, es tarea compleja organizar racionalmente los objetos que forman la configuración y establecer mecanismos de control.

*Personal*: Cualquiera realiza cambios. Hay mucho personal no especializado que no reconoce la importancia que tiene el control del cambio.

*Escalabilidad*: Es común encontrar aplicaciones que de un día para otro crecen considerablemente. Sin embargo, las técnicas de control no escalan de forma adecuada.

Política: ¿Quién posee la información? ¿Quién asume la responsabilidad y coste de mantenerla?

### 2.1.4 LA GESTIÓN DEL PROCESO

En un proceso tan rápido como es el proceso de *Ingeniería Web*, donde los tiempos de desarrollo y los ciclos de vida de los productos son tan cortos, ¿merece la pena el esfuerzo requerido por la gestión? La respuesta es que dada su complejidad es imprescindible. Entre los aspectos que añaden dificultad a la gestión destacamos:

- alto porcentaje de contratación a terceros,
- el desarrollo incluye una gran variedad de personal técnico y no técnico trabajando en paralelo,
- el equipo de desarrollo debe dominar aspectos tan varios como, software basado en componentes, redes, diseño de arquitectura y navegación, diseño gráfico y de interfaces, lenguajes y estándares en Internet, test de aplicaciones *Web*, etc, lo que hace que el proceso de búsqueda y contratación de personal sea arduo.

### 2.1.5 DIFERENCIAS

- *Confluencia de disciplinas:* Sistemas de Información, Ingeniería Software y Diseño Gráfico que requiere equipos multidisciplinares y polivalentes.

- *Ciclos de vida y tiempo de desarrollo muy cortos*

- *Cambio continuo:* Necesidad de soluciones que permitan flexibilidad y adaptación conforme el proyecto cambia.

Requisitos fuertes de seguridad, rendimiento y usabilidad.

http://www.webengineering.org/beta1/wsls.aspx?vdc=beta1&vurl=start/about

## 2.2 APLICACIONES WEB

Inicialmente la Web era simplemente una colección de páginas estáticas, documentos, etc., que podían consultarse o descargarse.

El siguiente paso en su evolución fue la inclusión de un método para confeccionar páginas dinámicas que permitiesen que lo mostrado fuese dinámico (generado o calculado a partir de los datos de la petición). Dicho método fue conocido como CGI (Common Gateway Interface) y definía un mecanismo mediante el cual podíamos pasar información entre el servidor HTTP y programas externos. Los CGI siguen siendo muy utilizados, puesto que la mayoría de los servidores Web los soportan debido a su sencillez. Además, nos proporcionan total libertad a la hora de escoger el lenguaje de programación para desarrollarlos.

El esquema de funcionamiento de los CGI tenía un punto débil: cada vez que recibíamos una petición, el servidor Web lanzaba un proceso que ejecutaba el programa CGI.

En ingeniería de software una aplicación Web es aquella que los usuarios usan accediendo a un servidor Web a través de Internet o de una intranet. Las aplicaciones Web son populares debido a la practicidad del navegador Web como cliente ligero.

La idea fundamental es que los navegadores, browsers, presentan documentos escritos en HTML que han obtenido de un servidor Web. Estos documentos HTML habitualmente presentan información de forma estática, sin más posibilidad de interacción con ellos.

El modo de crear los documentos HTML ha variado a lo largo de la corta vida de las tecnologías Web pasando desde las primeras páginas escritas en HTML almacenadas en un fichero en el servidor Web hasta aquellas que se generan al vuelo como respuesta a una acción del cliente y cuyo contenido varía según las circunstancias.

http://www.uoc.edu/masters/esp/img/873.pdf

http://www.sonork.com/esp/web_app.html

El esquema general muestra cada tipo de tecnología involucrada en la generación e interacción de documentos Web.

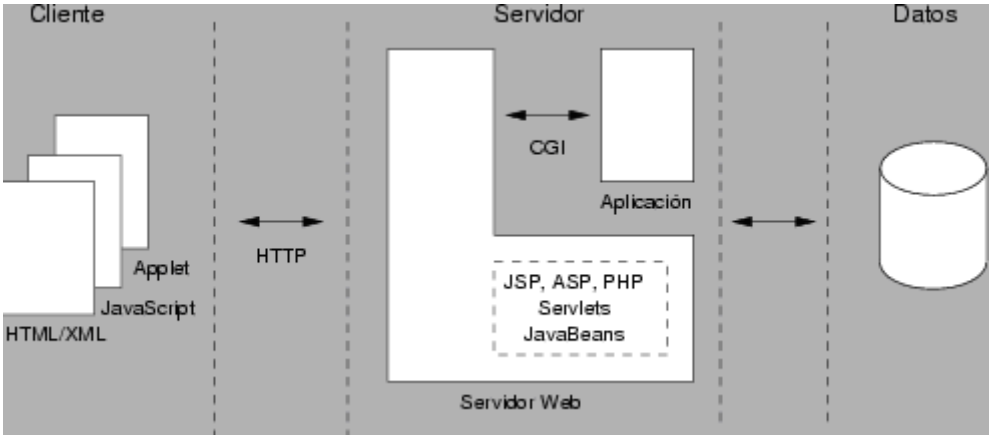



**Figura 2-1 Esquema general de las tecnologías Web**
Fuente: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html

### 2.2.1 ARQUITECTURA WEB

La fácil creación de hojas HTML y en general de sitios Web, usando herramientas simples, ha hecho que el desarrollo de este tipo de aplicaciones se haga sin un trabajo serio de análisis y diseño.

Cualquier sistema de complejidad no trivial, necesita ser analizado y modelado. Las aplicaciones Web, al igual que otras aplicaciones, necesitan métodos formales de análisis y diseño.

**Cuál es la diferencia entre un sitio Web y una aplicación Web?**

Una aplicación Web es un sitio Web donde la navegación a través del sitio, y la entrada de datos por parte de un usuario, afectan el estado de la lógica del

negocio. En esencia, una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica.

La arquitectura de un sitio Web tiene tres componentes principales: un servidor Web, una conexión de red, y uno o más clientes (browsers).
El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP.

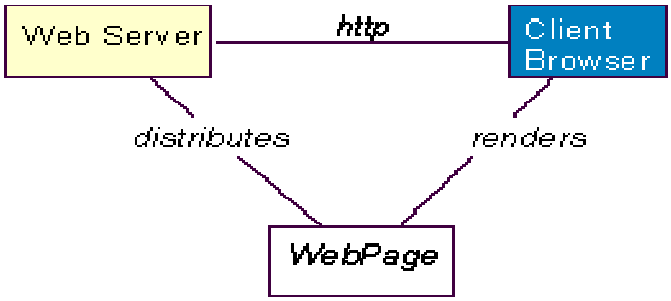



**Figura 2-2 Arquitectura de un sitio Web**
Fuente: http://uca.guegue.com/AnalisisyDisenoWeb/web-app.pdf

### Arquitectura básica de una aplicación / sitio Web

La información mostrada en las páginas está típicamente almacenada en archivos. Sin embargo, muchas veces esta información está almacenada en una base de datos, y las páginas son creadas dinámicamente.

Los sitios Web que usan este esquema, son llamados sitios dinámicos.

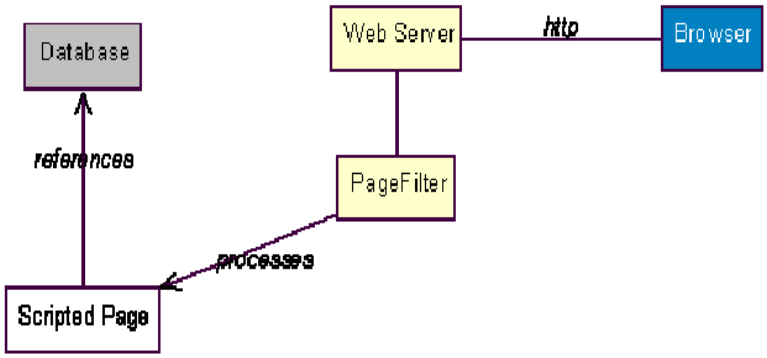



**Figura 2-3 Arquitectura básica de un sitio Web dinámico**
Fuente: http://uca.guegue.com/AnalisisyDisenoWeb/web-app.pdf

**Páginas Web:** Las páginas Web son el componente principal de una aplicación o sitio Web. Los browsers piden páginas (almacenadas o creadas dinámicamente) con información a los servidores Web.

En algunos ambientes de desarrollo de aplicaciones Web, las páginas contienen código HTML y scripts dinámicos, que son ejecutados por el servidor antes de entregar la página.

Una vez que se entrega una página, la conexión entre el browser y el servidor Web se rompe (a diferencia de otros esquemas tipo cliente / servidor). Es decir que la lógica del negocio en el servidor solamente se activa por la ejecución de los scripts de las páginas solicitadas por el browser (en el servidor, no en el cliente).

**Scripts en el cliente:** Cuando el browser ejecuta un script en el cliente, éste no tiene acceso directo a los recursos del servidor.

Hay otros componentes que no son scripts, como los applets o los componentes ActiveX. Los scripts del cliente son por lo general código JavaScript o VBSscript, mezclados con código HTML.

**Formularios:** La forma más común de capturar la información dada por el usuario, es a través de formularios. Un formulario (form) es una colección de campos de entrada: textbox, text area, checkbox, radio button group, button y selection list.

Cuando un formulario es llenado, se envía al servidor usando una operación submit solicitada por el usuario típicamente al hacer click en un botón.

**Servidor Web:** Un servidor Web es un programa que atiende y responde a las diversas peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o el protocolo HTTPS (la versión segura, cifrada y autenticada de HTTP).

A partir del esquema anterior se han diseñado y construido todos los programas servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGI, Servlets, etc.) que pueden atender, en función de que sean o no multi-proceso, multi-hilados, etc.

En muchas aplicaciones Web hay una capa intermedia, compuesta por un conjunto de componentes, que se ejecutan no necesariamente en el servidor Web, sino en otros servidores de aplicaciones. Esta capa encapsula la lógica del negocio, y, al ser componentes compilados puede contener objetos, con sus métodos y atributos (llamados business objects).

http://www.uoc.edu/masters/esp/img/873.pdf

### 2.2.2 APLICACIONES MULTINIVEL

Al hablar del desarrollo de aplicaciones Web resulta adecuado presentarlas dentro de las aplicaciones multinivel. Los sistemas típicos cliente / servidor pertenecen a la categoría de las aplicaciones de dos niveles. La aplicación reside en el cliente mientras que la base de datos se encuentra en el servidor.

En este tipo de aplicaciones el peso del cálculo recae en el cliente, mientras que el servidor hace la parte menos pesada, y eso que los clientes suelen ser máquinas menos potentes que los servidores. Además, está el problema de la actualización y el mantenimiento de las aplicaciones, ya que las modificaciones a la misma han de ser trasladada a todos los clientes. Para solucionar estos problemas se ha desarrollado el concepto de arquitecturas de tres niveles: interfaz de presentación, lógica de la aplicación y los datos.

La capa intermedia es el código que el usuario invoca para recuperar los datos deseados. La capa de presentación recibe los datos y los formatea para mostrarlos adecuadamente. Esta división entre la capa de presentación y la de la lógica permite una gran flexibilidad a la hora de construir aplicaciones, ya que se pueden tener múltiples interfaces sin cambiar la lógica de la aplicación.

La tercera capa consiste en los datos que gestiona la aplicación. Estos datos pueden ser cualquier fuente de información como una base de datos o documentos XML.

Convertir un sistema de tres niveles a otro multinivel es fácil ya que consiste en extender la capa intermedia permitiendo que convivan múltiples aplicaciones en lugar de una sola.

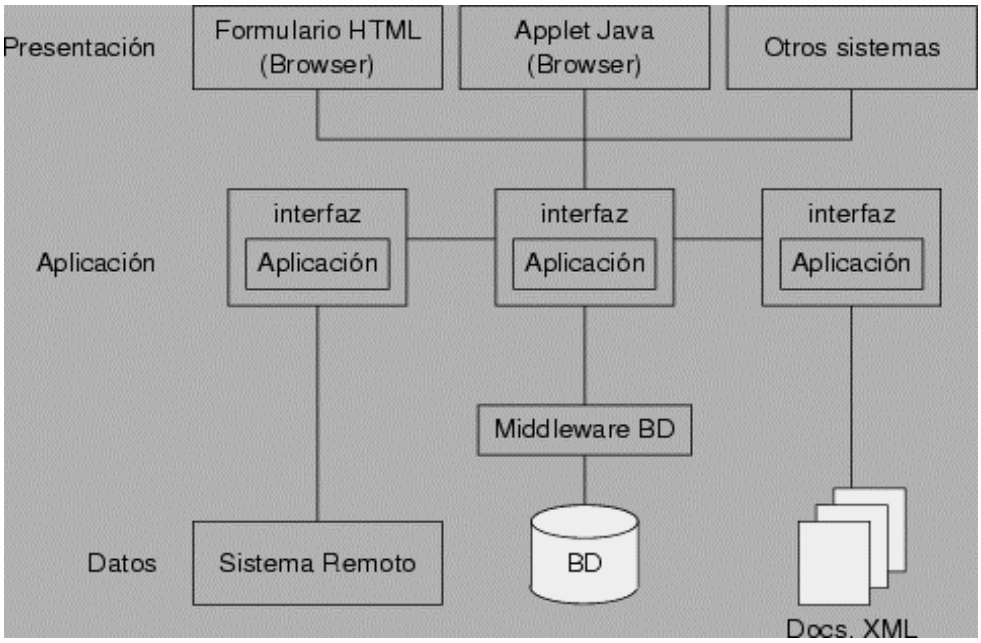



**Figura 2-4 Arquitectura Multinivel**
Fuente: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node21.html

La arquitectura de las aplicaciones Web suelen presentar un esquema de tres niveles. El primer nivel consiste en la capa de presentación que incluye no sólo el navegador, sino también el servidor Web que es el responsable de dar a los datos un formato adecuado. El segundo nivel está referido habitualmente a algún tipo de programa o script. Finalmente, el tercer nivel proporciona al segundo los datos necesarios para su ejecución.

Una aplicación Web típica recogerá datos del usuario (primer nivel), los enviará al servidor, que ejecutará un programa (segundo y tercer nivel) y cuyo resultado será formateado y presentado al usuario en el navegador (primer nivel otra vez).

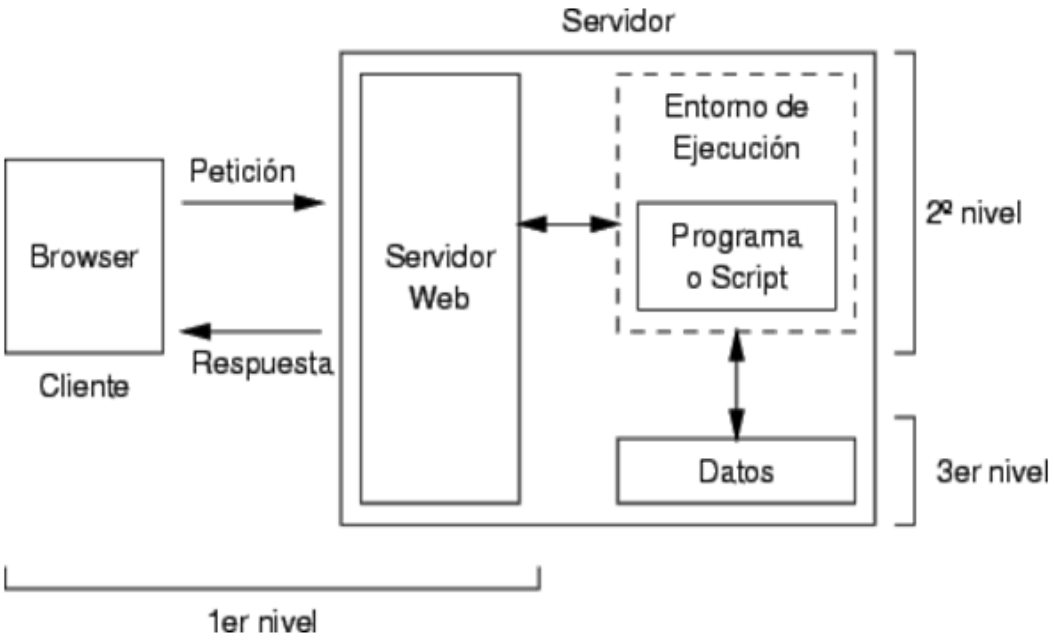



**Figura 2-5 Arquitectura Web de tres niveles**
Fuente: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node21.html

Lamentablemente, el uso de toda esta tecnología pasa por el dominio de técnicas de programación y de acceso a bases de datos.

## 2.3 HERRAMIENTAS

### 2.3.1 JAVASCRIPT

JavaScript es un lenguaje de programación utilizado para crear programas encargados de realizar acciones dentro del ámbito de una página Web siendo el siguiente paso, después del HTML, que puede dar un programador de la Web que decida mejorar sus páginas y la potencia de sus proyectos.
JavaScript se inicia en al año de 1995 cuando Netscape introduce la versión 2.0 de Navigator e incluye JavaScript bajo el nombre de Mocha, cuando aparece esta versión de Navigator se le llamaba LiveScript.

Finalmente se le bautiza con el nombre de JavaScript para llamar la atención de los medios y la industria de la informática.
Lo que quería Netscape es que JavaScript fuera un lenguaje de guiones, fácil de usar y que cualquier persona pudiera utilizarlo. Después de 2 años JavaScript se convirtió en una de las herramientas mas utilizadas por los desarrolladores Web, incluso se utiliza mas que el propio Java y Activex.

En el año de 1996 Microsoft empieza a tener gran interés por lograr competir con JavaScript por lo que lanza su lenguaje llamado Jscript, pero que desafortunadamente no ha tenido el gran éxito por lo que no ha podido competir directamente con este lenguaje.

A mediados de 1997, Netscape promocionó JavaScript y lanzó la versión 1.2 de este lenguaje. Esta nueva versión incluye nuevos componente que dan gran potencial al lenguaje, pero lamentablemente esta versión solo funcionaba en la ultima versión del Navigator en ese momento. En la actualidad esta versión de JavaScript es soportada en la mayoría de los navegadores y es utilizada en casi todo los sitios de Internet existentes.

http://www.ortizmania.com/online/articulo.asp?art=6

### 2.3.2 HTML

HTML (*HyperText Markup Language*) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con *enlaces* (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas, y con *inserciones* multimedia (gráficos, sonido, etc). La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.), así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho

hipertexto se realice por un programa especializado (como Mosaic, o Netscape).

http://etsit.upm.es/~alvaro/manual/manual.html

### 2.3.3 APACHE WEB SERVER

Este servidor cuyo mantenimiento está a cargo de Apache Software Foundation, es el servidor Web más popular debido a su estabilidad, costo, eficiencia y portabilidad. Es un producto de código fuente abierta que se ejecuta bajo las plataformas Unix, Linux y Windows.

El nombre apache tiene un origen un poco discutido, algunos dicen que viene de "**a patchy** Server" debido a numerosos patchs, otros dicen de una manera más seria que los instigadores de este proyecto tomaron el nombre en memoria de los Apaches por su gran adaptabilidad al terreno.

Según los estudios de netcraft en mayo de 2003, apache es utilizado en el 62.57 % de los servidores, seguidos por IIS con un 27.45%.

http://es.tldp.org/LinuxFocus/pub/mirror/LinuxFocus/Castellano/May2000/article122.shtml

### 2.3.4 PHP 5

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos Web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un

CGI escrito en C que permitía la interpretación de un número limitado de comandos.

El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

La siguiente gran contribución al lenguaje se realizó a mediados del 97 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP.

Actualmente PHP se encuentra en su versión 5, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son su rapidez gracias a que primero se compila y luego se ejecuta, mientras que antes se ejecutaba mientras se interpretaba el código, su mayor independencia del servidor Web creando versiones de PHP nativas para más plataformas y un API más elaborado y con más funciones.

http://www.desarrolloweb.com/articulos/436.php?manual=12

Existen tres campos en los que se usan scripts escritos en PHP:

*Scripts del lado del servidor*. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor Web y un navegador. Es necesario correr el servidor Web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor Web.

*Scripts en la línea de comandos*. Puede crear un script PHP y correrlo sin ningún servidor Web o navegador. Solamente necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto.

Escribir aplicaciones de interfaz gráfica. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas. También es posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal.

### 2.3.5 SQL (Lenguaje de Consulta Estructurado)

El SQL (Structure Query Language), es un lenguaje de consulta estructurado establecido claramente como el lenguaje de alto nivel estándar para sistemas de base de datos relacionales. Los responsables de publicar este lenguaje como estándar, fueron la ANSI (Instituto Americano de Normalización) y la ISO (Organismo Internacional de Normalización). Es por lo anterior que este lenguaje se lo va a encontrar en cualquiera de los DBMS relacionales que existen en la actualidad, por ejemplo, ORACLE, SYBASES, SQL SERVER por mencionar algunos.

Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre la misma. Es un lenguaje de cuarta generación (4GL).

**Características generales**

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.

Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizas en un lenguaje de bajo nivel.

**Optimización**

SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.
El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a cabo una optimización antes de la ejecución de la misma. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos. Dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.

Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa. Este lenguaje ha sido implementado a nivel experimental y está evolucionando rápidamente.

http://148.228.56.77/Descargas/Descargas/SQL.pdf

## 2.4 HERRAMIENTAS DE APOYO

### 2.4.1 DREAMWEAVER 8

Dreamweaver 8 es un software fácil de usar que permite crear páginas web profesionales. Las funciones de edición visual de Dreamweaver 8 permiten agregar rápidamente diseño y funcionalidad a las páginas, sin la necesidad de programar manualmente el código HTML.

Se puede crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, etc., de una forma muy sencilla y visual.

#### 2.4.1.1 Características principales de Dreamweaver 8:

Dreamweaver 8 aporta nuevas características sobre la versión anterior, Dreamweaver MX 2004, entre las que se destacan las siguientes:

- Mejoras CSS: esta última versión ha mejorado mucho respecto a la compatibilidad y manejo de estilos de cascada. De esta forma se ha mejorado el panel de estilos CSS, donde ahora podrás acceder a la configuración de cada uno de los estilos desde una lista mucho mejor dotado de una cuadrícula editable desde donde se podrá modificar sus propiedades. Además, Dreamweaver 8, añade una nueva barra de herramientas que proporciona la reproducción inmediata de los estilos para diferentes medios (pantalla, impresora, webTV, PDAs).
- Transferencia de archivos: Con Dreamweaver 8 se puede seguir trabajando con archivos mientras el programa se comunica con el servidor e incluye los archivos creados o modificados recientemente. Su sincronización ha mejorado notablemente siendo posible una mejor gestión de cambios, además de permitir en uso de bloqueo/desbloqueo de archivos para que estos no se sobrescriban.
- Interfaz mejorada: Los usuarios con problemas visuales podrán acceder a una opción de aumento de la pantalla en vista de diseño para analizar o trabajar con difíciles anidamientos de tablas. Además de la inclusión de información visual gracias a las guías que permitirán la medición píxel a píxel de todos los elementos.

- Compatibilidad: Se tiene compatibilidad con otras versiones como: PHP5, Coldfusion MX 7 y Video Flash.

http://www.aulaclic.es/dreamweaver8/t_1_1.htm

### 2.4.2 MACROMEDIA FLASH

Flash se refiere tanto al programa de edición multimedia como a Macromedia Flash Player, escrito y distribuido por Macromedia, que utiliza gráficos vectoriales e imágenes de mapa de bits, sonido, código de programa, flujo de vídeo y audio bidireccional (el flujo de subida sólo está disponible si se usa conjuntamente con Macromedia Flash Communication Server). En sentido estricto, Macromedia Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos Flash.

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia. Son también ampliamente utilizados en anuncios de la web.

En versiones recientes, Macromedia Flash está más allá de las animaciones simples, convirtiéndolo en una herramienta de desarrollo completa, para crear principalmente elementos multimedia e interactivos para Internet.

http://www.blog.pucp.edu.pe/item/1691

### 2.4.3 POWER DESIGNER

PowerDesigner es un único conjunto de herramientas de modelamiento que combina distintas técnicas estándar de modelamiento: modelamiento de aplicación a través de UML, técnicas de Modelamiento de Procesos Empresariales y técnicas tradicionales de modelamiento de base de datos.

PowerDesigner combina únicamente varios estándares y técnicas junto con ambientes principales del desarrollo tales como .NET, espacio de trabajo, PowerBuilder, Java, eclipse, etc., para traer análisis de negocio y soluciones formales del diseño de base de datos al ciclo de vida tradicional del desarrollo de software.

PowerDesigner es una suite de aplicaciones de Powersoft para la construcción, diseño y modelado de datos a través de diversas aplicaciones. Es la herramienta para el análisis, diseño inteligente y construcción sólida de una base de datos y un desarrollo orientado a modelos de datos a nivel físico y conceptual, que dan a los desarrolladores Cliente / servidor la más firme base para aplicaciones de alto rendimiento.

http://www.pcm.gob.pe/portal_ongei/publicaciones/cultura/lib5103/Libro.pdf

### 2.4.4 RATIONAL ROSE

Rational Rose es la mejor solución de modelado visual en el mundo, y la mejor herramienta para traducir requisitos de alto nivel a una arquitectura flexible basada en componentes.

Rational Rose se encuentra a la cabeza en cuanto al desarrollo del Unified Modeling Language (UML), que se ha convertido en la notación estandarizada empleada en Rational Rose para especificar, visualizar y construir desarrollos de software y sistemas.

**Características:**

- ✓ Mantiene la consistencia de los modelos del sistema.
- ✓ Chequeo de la sintaxis UML.
- ✓ Generación de documentación automática.
- ✓ Generación de código a partir de los modelos.
- ✓ Ingeniería inversa (crear modelo a partir código).

**El desarrollo de Software abarca las disciplinas de:**

- ✓ Modelado de casos de uso y datos
- ✓ Modelado de arquitectura
- ✓ Modelado de componentes,
- ✓ Construcción de claves y;
- ✓ Pruebas de unidades.

http://www.abits.cl/rational/desa_soft.htm

### 2.4.5 CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un lenguaje que permite a los autores y usuarios enlazar estilos (Fonts, spacing, etc.) a documentos estructurados (Documentos HTML y aplicaciones XML). Mediante la separación de estilo de los documentos de su contenido CSS simplifica el desarrollo Web y el mantenimiento del sitio.

Las hojas de estilo describen cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los *Estilos* definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

http://www.w3c.es/Divulgacion/Guiasbreves/HojasEstilo

# 3 CAPITULO 3: ASPECTOS METODOLÓGICOS

## 3.1 PARADIGMA ESPIRAL ORIENTADO A LA WEB

El modelo espiral para la ingeniería de software ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos.

El modelo espiral orientado a la web, define seis actividades principales:

*Formulación:* Actividad que identifica las metas y los objetivos de la Web y establece el ámbito del primer incremento.

*Planificación*: Estima el costo global del proyecto, evalúa los riesgos asociados con el esfuerzo del desarrollo bien granulado para el incremento final de la web con la determinación de objetivos, alternativas y restricciones.

*Análisis de riesgo:* establece los requisitos técnicos para la web e identifica los elementos del contenido que van a incorporar análisis de alternativas e identificación/resolución de riesgos.

*Ingeniería:* desarrollo del producto del "siguiente nivel". La actividad de ingeniería incorpora dos tareas paralelas. El diseño del contenido y la producción son tareas llevadas acabo por personas no técnicas del equipo web. El objetivo de estas tareas es diseñar, producir todo el contenido de texto, gráficos y videos que se vayan a integrar en la web.

*Generación de Páginas:* Es una actividad de construcción que hace mucho uso de las herramientas automatizadas para la creación de la web. El contenido definido en la actividad de ingeniería se fusiona con los diseños arquitectónicos, de navegación y de la interfaz para elaborar páginas Web ejecutables en HTML, XML y otros lenguajes orientados a procesos (por ejemplo Java). Durante esta actividad se lleva a cabo la integración con el software intermedio (Middle ware) de componentes (CORBA, DCOM O Java Beans). Las pruebas ejercitan la navegación, intentan descubrir los errores de las Apples, guiones y formularios y ayudan a asegurar que la Web funcionara correctamente en diferentes entornos (por ejemplo, con diferentes navegadores).

*Evaluación del cliente:* Cada incremento producido como parte del proceso Web se revisa durante la actividad de evaluación del cliente en este punto en donde se solicitan cambios (tiene lugar ampliaciones del ámbito). Estos cambios se integran en la siguiente ruta mediante el flujo incremental del proceso.

El paradigma del modelo en espiral para la ingeniería de software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala.
Utiliza un enfoque evolutivo para la ingeniería de software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción de riesgo, pero permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución de prototipos. (PRE00, 525)

## 3.2 OOHDM: "OBJECT ORIENTED HYPERMEDIA DESIGN METHOD"

**Introducción:**

OOHDM es una propuesta metodológica ampliamente aceptada para el desarrollo de aplicaciones web. En sus comienzos no contemplaba la fase de captura y definición de requisitos, pero actualmente propone el uso de user interaction diagrams (UIDs). Esta propuesta parte de los casos de uso que considera una técnica muy difundida, ampliamente aceptada y fácilmente entendible por los usuarios y clientes no expertos, pero que resulta ambigua para el equipo de desarrollo en fases posteriores del ciclo de vida. Igualmente, resalta la necesidad de empezar el diseño del sistema, especialmente en los entornos web, teniendo un claro y amplio conocimiento de las necesidades de interacción, o lo que es lo mismo de la forma en la que el usuario va a comunicarse con el sistema.

Partiendo de estas dos premisas, OOHDM propone que la comunicación con el usuario se realice utilizando los casos de uso y a partir de ellos los analistas elaboran los UIDs. Estos UIDs son modelos gráficos que representan la interacción entre el usuario y el sistema, sin considerar aspectos específicos de la interfaz. El proceso de transformación de un caso de uso a un UIDs es descrito detalladamente en la propuesta, y se basa en detectar la interacción

necesaria para la realización del caso de uso. OOHDM centra el desarrollo de un sistema de información web entorno del modelo conceptual de clases.

Este diagrama debe surgir de los requisitos que se definan del sistema, pero los casos de uso resultan demasiado ambiguos para ello. Así, propone refinar el proceso de desarrollo descrito en UML de forma que de los casos de uso se generen los UIDs que concreticen más la definición de los requisitos para, a partir de ellos, obtener el diagrama conceptual. En algunos de los primeros trabajos OOHDM propone la descripción de escenarios en forma textual y gráfica para cada tipo de usuario como etapa previa al diseño de la navegación.

OOHDM propone el desarrollo de aplicaciones Web a través de un proceso compuesto por cuatro etapas: diseño conceptual, diseño navegacional, diseño de interfaces abstractas e implementación.

**Diseño Conceptual:**

Durante esta actividad se construye un esquema conceptual representado por los objetos del dominio, las relaciones y colaboraciones existentes establecidas entre ellos. En las aplicaciones Web convencionales, cuyos componentes Web no son modificados durante la ejecución, se podría usar un modelo de datos semántico estructural (como el modelo de entidades y relaciones). De este modo, en los casos en que la información base pueda cambiar dinámicamente o se intenten ejecutar cálculos complejos, se necesitará enriquecer el comportamiento del modelo de objetos.

En OOHDM, el esquema conceptual está construido por clases, relaciones y subsistemas. Las clases son descritas como en los modelos orientados a objetos tradicionales. Sin embargo, los atributos pueden ser de múltiples tipos para representar perspectivas diferentes de las mismas entidades del mundo real.

Se usa notación similar a UML (Lenguaje de Modelado Unificado) y tarjetas de clases y relaciones similares a las tarjetas CRC (Clase Responsabilidad Colaboración). El esquema de las clases consiste en un conjunto de clases conectadas por relaciones. Los objetos son instancias de las clases. Las clases son usadas durante el diseño navegacional para derivar nodos, y las relaciones que son usadas para construir enlaces.

**Diseño Navegacional:**

La primera generación de aplicaciones web fue pensada para realizar navegación a través del espacio de información, utilizando un simple modelo de datos de Hypertexto. En OOHDM, la navegación es considerada un paso crítico en el diseño aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual, admitiendo la construcción de modelos diferentes de acuerdo con los diferentes perfiles de usuarios. Cada modelo navegacional provee una vista subjetiva del diseño conceptual.

El diseño de navegación es expresado en dos esquemas: el esquema de clases navegacionales y el esquema de contextos navegacionales. En OOHDM existe un conjunto de tipos predefinidos de clases navegacionales: nodos, enlaces y estructuras de acceso. La semántica de los nodos y los enlaces son las tradicionales de las aplicaciones Web, y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos.

Los contextos navegacionales juegan un rol similar a las colecciones y fueron inspirados sobre el concepto de contextos anidados. Organizan el espacio navegacional en conjuntos convenientes que pueden ser recorridos en un orden particular y que deberían ser definidos como caminos para ayudar al usuario a lograr la tarea deseada.

Los nodos son enriquecidos con un conjunto de clases especiales que permiten de un nodo observar y presentar atributos (incluidos las anclas), así como métodos (comportamiento) cuando se navega en un particular contexto.

**Diseño de la Interfaz Abstracta:**

Una vez que las estructuras navegacionales son definidas, se deben especificar los aspectos de interfaz.

Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, como objetos de interfaz. Activan la navegación y el resto de funcionalidad de la aplicación, transformaciones de la interfaz pertinentes.

Una clara separación entre diseño navegacional y diseño de interfaz abstracta permite construir diferentes interfaces para el mismo modelo navegacional, dejando un alto grado de independencia de la tecnología de interfaz de usuario.

El aspecto de la interfaz de usuario de aplicaciones interactivas (en particular las aplicaciones web) es un punto crítico en el desarrollo de las modernas metodologías que tienden a descuidar. En OOHDM se utiliza el diseño de interfaz abstracta para describir la interfaz de usuario de la aplicación Web.

El modelo de interfaz ADVs (Vista de Datos Abstracta) especifica la organización y comportamiento de la interfaz, pero la apariencia física real o de los atributos, y la disposición de las propiedades de las ADVs en la pantalla real son hechas en la fase de implementación.

**Implementación:**

En esta fase, el diseñador debe implementar el diseño. Hasta ahora, todos los modelos fueron construidos en forma independiente de la plataforma de implementación; en esta fase se tiene en cuenta el entorno particular en el cual se va a correr la aplicación.

Al llegar a esta fase, el primer paso que debe realizar el diseñador es definir los ítems de información que son parte del dominio del problema. Debe identificar

también, cómo son organizados los ítems de acuerdo con el perfil del usuario y su tarea; decidir qué interfaz debería ver y cómo debería comportarse. A fin de implementar todo en un entorno web, el diseñador debe decidir además qué información debe ser almacenada.

http://viait.com.ar/docs/Metodologias_de_desarrollo_web.pdf
http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r22_art5_c.pdf

**Pruebas:**

En este caso a medida que se codifiquen las tareas, se van a ir aplicando las pruebas unitarias a cada una de ellas, con el fin de ir constatando el buen funcionamiento de los módulos, para luego poder ensamblar el sistema y así realizar las pruebas definitivas con el cliente.

### 3.2.1 UML (UNIFIED MODELING LANGUAGE)

El Lenguaje de Modelado Unificado (UML) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s. Directamente unifica los métodos de Booch, Rumbaugh (OMT), y Jacobson, y algo más.
UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso.

http://agamenon.uniandes.edu.co/~pfigueroa/soo/uml
UML Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

Es un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes.

Existían diversos métodos y técnicas orientadas a objetos, con muchos aspectos en común pero utilizando distintas notaciones, se presentaban inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc., además de pugnas entre enfoques, lo que generó la creación del UML como estándar para el modelamiento de sistemas de software principalmente, pero con posibilidades de ser aplicado a todo tipo de proyectos.

http://www.creangel.com/uml/intro.php

#### 3.2.1.1 Diagramas de Caso de Uso

Un diagrama de Caso de Uso describe lo que hace un sistema desde el punto de vista de un observador externo, debido a esto, un diagrama de este tipo generalmente es de los más sencillos de interpretar en UML, ya que su razón de ser se concentra en un Qué hace el sistema, a diferencia de otros diagramas UML que intentan dar respuesta a un Cómo logra su comportamiento el sistema.

Un Caso de Uso esta muy relacionado con lo que pudiera ser considerado un escenario en el sistema, esto es, lo que ocurre cuando alguien interactúa con el sistema: "Acude un mesero a colocar la orden, la orden es tomada por el cocinero, y posteriormente se abona a la cuenta del cliente".

Un Caso de Uso es empleado con más frecuencia en alguna de las siguientes etapas:

*Determinación de Requerimientos:* Por lo general nuevos requerimientos de sistema generan nuevos usos-casos, conforme es analizado y diseñado el sistema.

*Comunicación con el Cliente:* Debido a la sencillez de este tipo de diagramas, son fáciles de emplear para comunicarse con el cliente final del proyecto.

*Generación de pruebas de Sistemas:* A través de los diagramas de caso de uso se pueden generar una serie de pruebas de sistema.

Los casos de uso se usan para especificar el comportamiento del sistema sin definir su estructura, la forma de que un modelo de Caso de uso es realizado en términos de objetos que son definidos por clases dentro del sistema se puede describir con diagramas de colaboración.

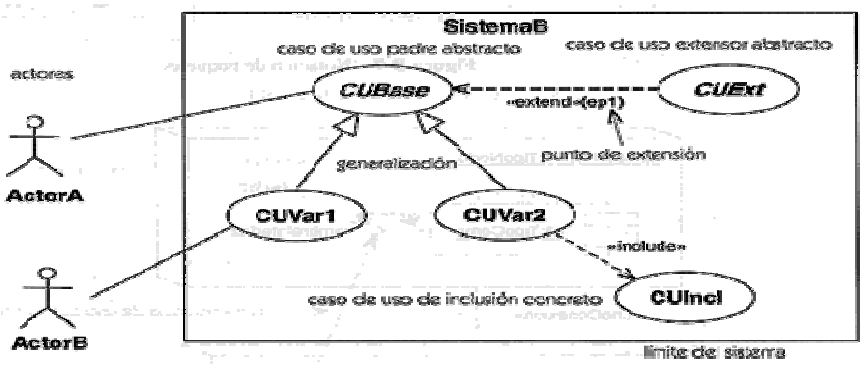



**Figura 3-1 Diagrama de Casos de Uso**
Fuente: http://www.creangel.com/uml/casouso.php

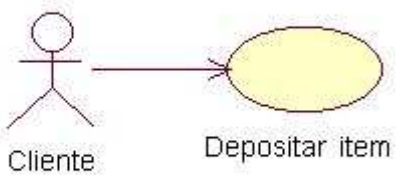
*3.2.1.1.1 Composición*

- **Actor**: Un actor representa quien o que inicia una acción dentro del sistema, en otras palabras, es simplemente un rol que es llevado acabo por una persona o cosa. Un Actor en un diagrama Uso-Caso es representado por una figura en forma de persona.
- **Uso-Caso**: El uso-caso en sí es representado por un ovalo que describe la funcionalidad a groso modo que se requiere por el sistema.
- **Comunicación**: Este elemento representa la relación que existe entre un Uso-Caso y un Actor, dicho elemento es representado simplemente por una línea recta que se extiende de la figura del actor hacia el ovalo del uso-caso.
- **Limite de Sistema (System Boundry)**: Empleado para delimitar los límites del sistema, y representado por un rectángulo con color de fondo distintivo.
- **Generalización:** Una generalización indica que un uso-caso (ovalo) es un caso especial de otro caso, en otros términos, representa una relación padre-hijo, donde el hijo puede ser suplido directamente por el padre en cualquier momento. Este elemento es representado por una línea con flecha que se extiende del uso-caso hijo hacia el uso caso padre (general).
- **Inclusión**: Una inclusión es utilizada para indicar que un uso-caso (ovalo) depende de otro caso, dicho de otra manera, significa que la funcionalidad de determinado caso se requiere para realizar las tareas de otro. Este elemento es representado por una linea punteada con flecha y comentario <<include>> que se extiende del uso-caso base hacia el uso caso de inclusión.

- **Extensión:** Una extensión representa una variación de un uso-caso a otro, aunque similar a una **generalización**, una extensión representa una dependencia especifica, mientras una generalización no implica que los usos-casos dependen uno del otro. Este elemento es representado por una linea punteada con flecha y comentario <<extend>> que origina del uso-caso base hacia el uso caso de extensión.

*3.2.1.1.2 Notación del Diagrama de Caso de uso*

Un diagrama de caso de uso es una gráfica de actores, un conjunto de casos de uso, posiblemente algunas interfaces y las relaciones entre estos elementos. Las relaciones son asociaciones entre los actores y los casos de uso generalizaciones entre los actores y generalizaciones, extensiones e inclusiones entre los casos de uso.




*3.2.1.1.3 Relaciones de los Casos de uso*

Hay varias relaciones estándar entre los casos de uso o entre los actores y los casos de uso.

**Asociación ( ____ ).** La participación de un actor en el Caso de uso, i. e. instancias de el actor e instancias de el caso de uso se comunican entre si. Esta es la única relación entre los actores y los Casos de uso.

**Extensión (__e__).** Una relación de extensión entre el caso de uso A y el caso de uso B indica que una instancia de el caso de uso B puede ser aumentada (Dependiendo de ciertas condiciones especificadas en la extensión) por el comportamiento de especificado en el Caso de uso B. El comportamiento es insertado el punto definido como punto de extensión del Caso de uso B, el cual es referenciado por la relación externa.

**Generalización (◄——).** Una generalización de un caso de uso A hacia el caso de uso B indica que A es una especialización de B.

**Inclusión (__i__).** Una relación de inclusión del caso de uso A hacia el Caso de uso B indica que una instancia del caso de uso A también contendrá el comportamiento especificado por B. El comportamiento es incluido en el punto definido en A.

http://www.geocities.com/txmetsb/#Que es un Use Case
http://www.osmosislatina.com/lenguajes/uml/casos.htm

### 3.2.1.2 Diagrama de Clases

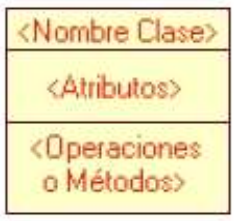
Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido.
Un diagrama de clases esta compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

#### *3.2.1.2.1 Clase*

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

En UML, una clase es representada por un rectángulo que posee tres divisiones:




En donde:

**Superior**: Contiene el nombre de la Clase
**Intermedio**: Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser private, protected o public).
**Inferior**: Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).
Ejemplo:
Una Cuenta Corriente que posee como característica:
Balance

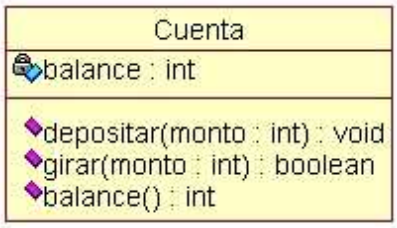
Puede realizar las operaciones de:

Depositar

Girar

y Balance

El diseño asociado es:




*3.2.1.2.2 Atributos*

Los atributos o características de una Clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:

**Public** (+,  ): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.

**Private** (-,  ): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden accesar).

**Protected** (#,  ): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

*3.2.1.2.3 Métodos*

Los métodos u operaciones de una clase son la forma en como ésta interactúa con su entorno, éstos pueden tener las características:

**Public** (+,  ): Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.

**Private** (-,  ): Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden accesar).

**Protected** (#,  ): Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

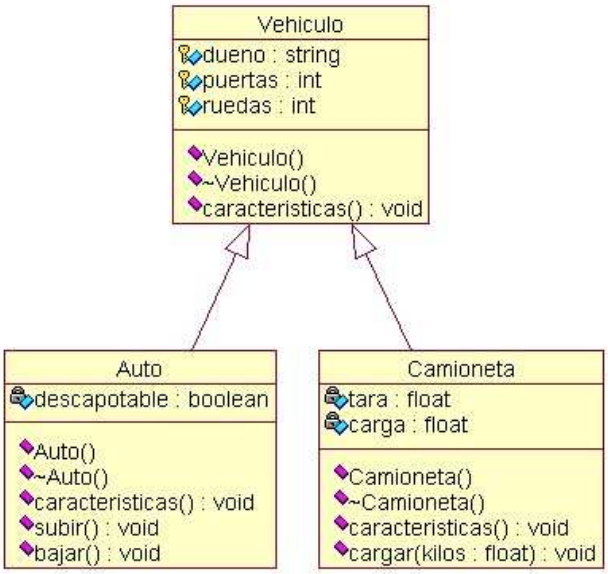
http://www.clikear.com/manuales/uml/diagramasestructuraestatica.asp

*3.2.1.2.4 Relaciones entre Clases*

Ahora ya definido el concepto de Clase, es necesario explicar como se pueden interrelacionar dos o más clases (cada uno con características y objetivos diferentes).
Antes es necesario explicar el concepto de cardinalidad de relaciones:
En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

- **uno o muchos**: 1..* (1..n)
- **0 o muchos**: 0..* (0..n)
- **número fijo**: m (m denota el número).

**Herencia (Especialización/Generalización)**: 
Indica que una subclase hereda los métodos y atributos especificados por una Super Clase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected), ejemplo:




En la figura se especifica que Auto y Camión heredan de Vehículo, es decir, Auto posee las Características de Vehículo (Precio, VelMax, etc) además posee algo particular que es Descapotable, en cambio Camión también hereda las características de Vehiculo (Precio, VelMax, etc) pero posee como particularidad propia Acoplado, Tara y Carga.

Cabe destacar que fuera de este entorno, lo único "visible" es el método Características aplicable a instancias de Vehículo, Auto y Camión, pues tiene definición pública, en cambio atributos como Descapotable no son visibles por ser privados.

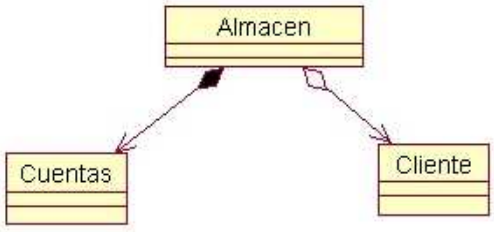
**Agregación**: 

Para modelar objetos complejos, n bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:

**Por Valor**: Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido esta condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada:

*Composición* (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").

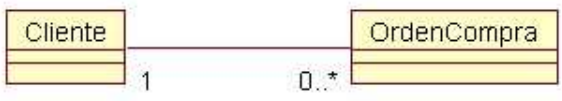
**Por Referencia:** Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada *Agregación* (el objeto base utiliza al incluido para su funcionamiento).
Un Ejemplo es el siguiente:



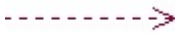

En donde se destaca que: Un almacén posee Clientes y Cuentas (los rombos van en el objeto que posee las referencias).

Cuando se destruye el Objeto almacén también son destruidos los objetos Cuenta asociados, en cambio no son afectados los objetos Cliente asociados.
La composición (por Valor) se destaca por un rombo relleno.
La agregación (por Referencia) se destaca por un rombo transparente.
La flecha en este tipo de relación indica la navegabilidad del objeto referenciado. Cuando no existe este tipo de particularidad la flecha se elimina.

**Asociación**:  La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.
Ejemplo:



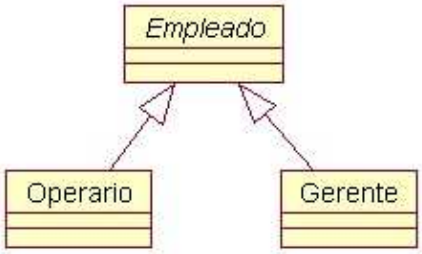

Un cliente puede tener asociadas muchas Ordenes de Compra, en cambio una orden de compra solo puede tener asociado un cliente.

**Dependencia o Instanciación (uso)**:  Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada.
El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación grafica que instancia una ventana (la creación del Objeto Ventana esta condicionado a la instanciación proveniente desde el objeto Aplicación):


Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

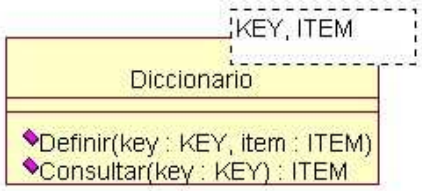
*3.2.1.2.5 Casos Particulares*

**Clase Abstracta**:




Una clase abstracta se denota con el nombre de la clase y de los métodos con letra "itálica". Esto indica que la clase definida no puede ser instanciada pues posee métodos abstractos (aún no han sido definidos, es decir, sin implementación). La única forma de utilizarla es definiendo subclases, que implementan los métodos abstractos definidos.

**Clase parametrizada**:



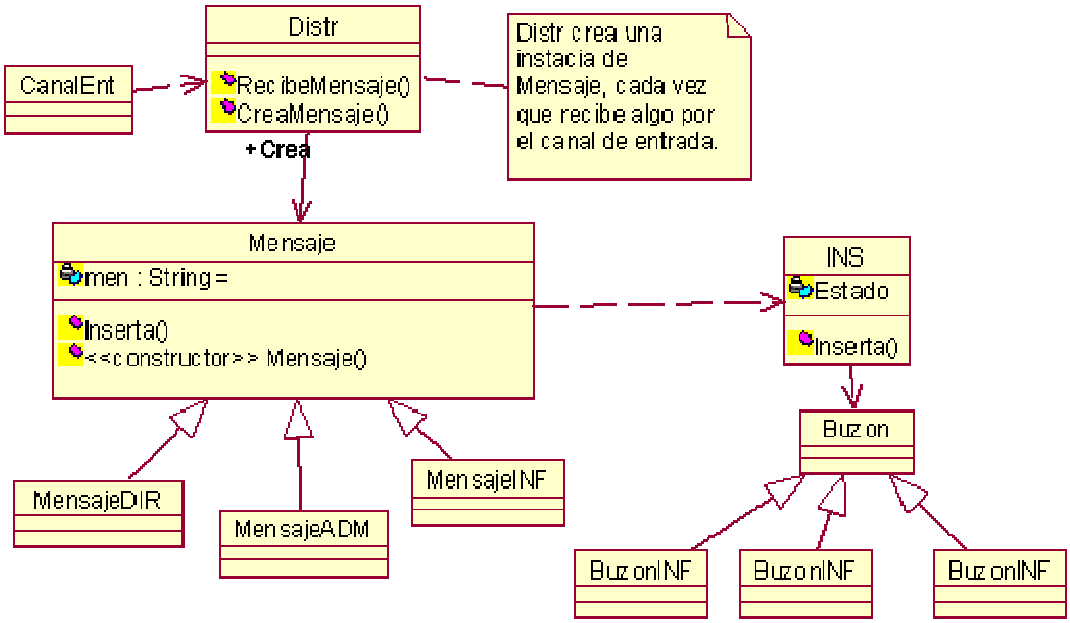

Una clase parametrizada se denota con un subcuadro en el extremo superior de la clase, en donde se especifican los parámetros que deben ser pasados a la clase para que esta pueda ser instanciada. El ejemplo más típico es el caso de un Diccionario en donde una llave o palabra tiene asociado un significado, pero en este caso las llaves y elementos pueden ser genéricos.

http://www.dcc.uchile.cl/~psalinas/uml/modelo.html

#### 3.2.1.3 Diagrama de Objetos

Forma parte de la vista estática del sistema. En este diagrama se modelan las instancias de las clases del diagrama de clases. Muestra a los objetos y sus relaciones, pero en un momento concreto del sistema. Estos diagramas contienen objetos y enlaces. En los diagramas de objetos también se pueden incorporar clases, para mostrar la clase de la que es un objeto representado.

En este diagrama se muestra un estado del diagrama de eventos. Para realizar el diagrama de objetos primero se debe decidir que situación queremos representar del sistema. Es decir si disponemos de un sistema de mensajería, deberemos decidir que representaremos el sistema con dos mensajes entrantes, los dos para diferentes departamentos, dejando un departamento inactivo. Para el siguiente diagrama de clases:




Tendríamos un diagrama de objetos con dos instancias de Mensaje, mas concretamente con una instancia de MensajeDIR y otra de MensajeADM, con todos sus atributos valorados. También tendríamos una instancia de cada una de las otras clases que deban tener instancia. Como CanalEnt, INS, Distr, y el buzón correspondiente a la instancia de mensaje que se está instanciando. En la instancia de la clase INS se deberá mostrar en su miembro Estado, que esta ocupado realizando una inserción.

En un diseño no podemos encontrar con una multitud de diagramas de objetos, cada uno de ellos representando diferentes estados del sistema.

http://usuarios.lycos.es/oopere/uml.htm

#### 3.2.1.4 Diagramas de Interacción

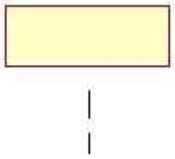
Los diagramas de interacción, representan la forma en: como un Cliente (Actor) u Objetos (Clases) se comunican entre si en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente.

Dichos diagramas pueden ser obtenidos de dos partes, desde el Diagrama Estático de Clases o el de Casos de Uso (son diferentes).

Los componentes de un diagrama de interacción son:

- Un Objeto o Actor.
- Mensaje de un objeto a otro objeto.
- Mensaje de un objeto a si mismo.

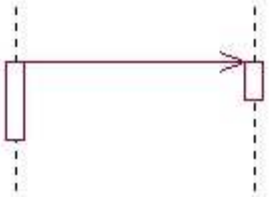
**Objeto/Actor**:



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

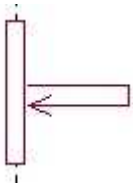
http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html

**Mensaje a Otro Objeto**:



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

**Mensaje al Mismo Objeto**:



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

Hay dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular: Diagramas de Secuencia y Diagramas de Colaboración.

#### 3.2.1.5 Diagrama de Secuencia

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

http://www.clikear.com/manuales/uml/diagramainteraccion.asp

El diagrama de Secuencia Muestra las interacciones entre los objetos organizadas en una secuencia temporal. En particular muestra los objetos participantes en la interacción y la secuencia de mensajes intercambiados.

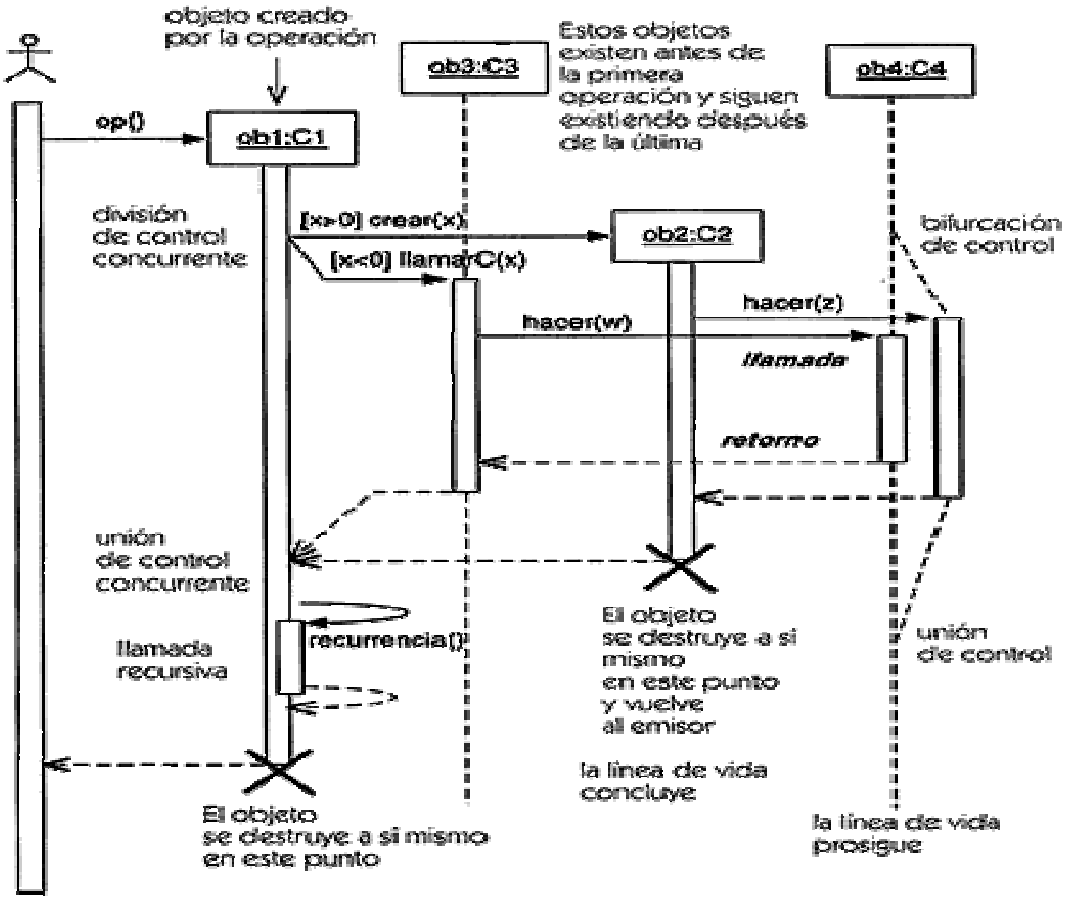



**Figura 3-2 Diagrama de Secuencia**
Fuente: http://www.creangel.com/uml/secuencia.php

### 3.2.1.6 Diagrama de Colaboración

Un Diagrama de Colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los Diagramas de Secuencia, los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia.

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

Un Diagrama de Colaboración muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran explícitamente las relaciones de los roles.

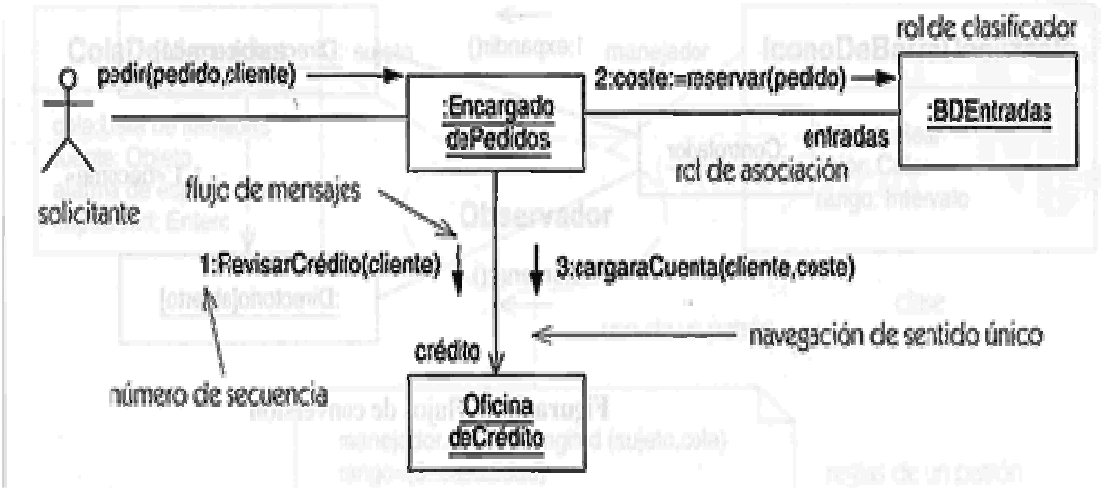



**Figura 3-3 Diagrama de Colaboración**
Fuente: http://www.creangel.com/uml/colaboracion.php#

### 3.2.1.7 Diagrama de Estados

Un Diagrama de Estados muestra la secuencia de estados por los que pasa bien un caso de uso, bien un objeto a lo largo de su vida, o bien todo el sistema. En él se indican qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera.
En cuanto a la representación, un diagrama de estados es un grafo cuyos nodos son estados y cuyos arcos dirigidos son transiciones etiquetadas con los nombres de los eventos. Un estado se representa como una caja redondeada con el nombre del estado en su interior. Una transición se representa como una flecha desde el estado origen al estado destino.

La caja de un estado puede tener 1 o 2 compartimentos.

- En el primer compartimento aparece el nombre del estado.
- El segundo compartimento es opcional, y en él pueden aparecer acciones de entrada, de salida y acciones internas.

Una acción de entrada aparece en la forma entrada/acción _ asociada donde acción _ asociada es el nombre de la acción que se realiza al entrar en ese estado. Cada vez que se entra al estado por medio de una transición la acción de entrada se ejecuta.

http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x98.html

Una acción de salida aparece en la forma salida/acción_asociada. Cada vez que se sale del estado por una transición de salida la acción de salida se ejecuta.

Una acción interna es una acción que se ejecuta cuando se recibe un determinado evento en ese estado, pero que no causa una transición a otro estado. Se indica en la forma nombre_de_evento/acción_asociada.

Un diagrama de estados puede representar ciclos continuos o bien una vida finita, en la que hay un estado inicial de creación y un estado final de destrucción (finalización del caso de uso o destrucción del objeto).

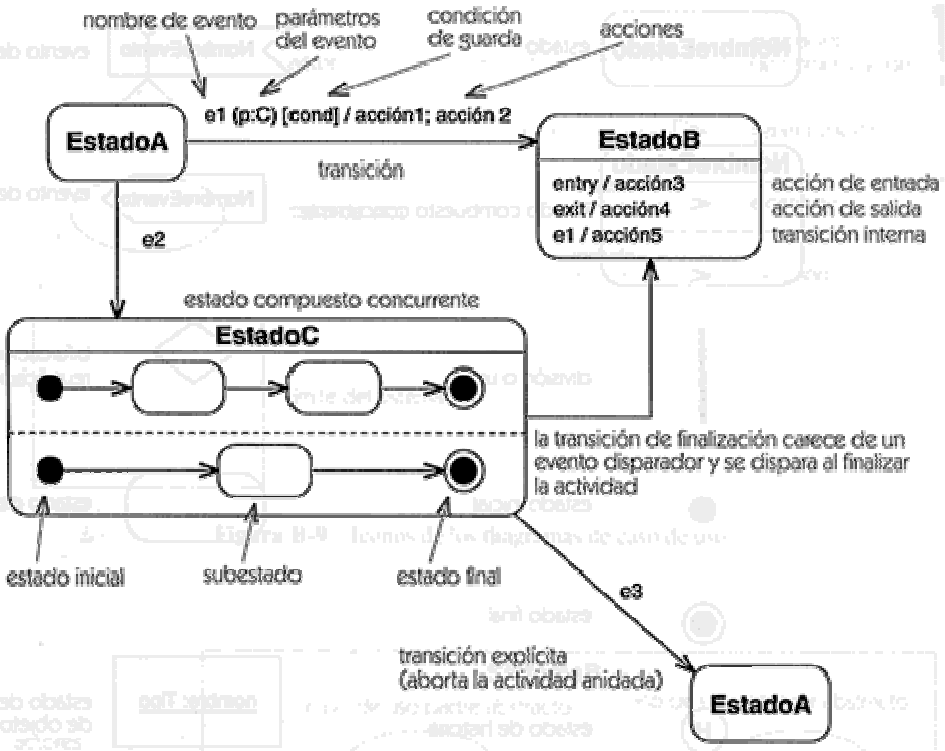



**Figura 3-4 Diagrama de Estados**
Fuente: http://www.clikear.com/manuales/uml/diagramasestados.asp

El estado inicial se muestra como un círculo sólido y el estado final como un círculo sólido rodeado de otro círculo. En realidad, los estados inicial y final son pseudoestados, pues un objeto no puede “estar” en esos estados, pero nos sirven para saber cuáles son las transiciones inicial y final(es).

http://www.clikear.com/manuales/uml/diagramasestados.asp

### 3.2.1.8 Diagrama de Actividades

Un diagrama de actividad muestra la realización de operaciones para conseguir un objetivo. Presentan una visión simplificada de lo que ocurre en un proceso, mostrando los pasos que se realizan, constituyéndose en uno de los diagramas que modelan los aspectos dinámicos del sistema.

Los diagramas de actividad son una extensión de los diagramas de estado. Los diagramas de estado resaltan los estados y muestran las actividades que dan lugar a cambios de estado, mientras que los diagramas de actividad resaltan justamente las actividades.

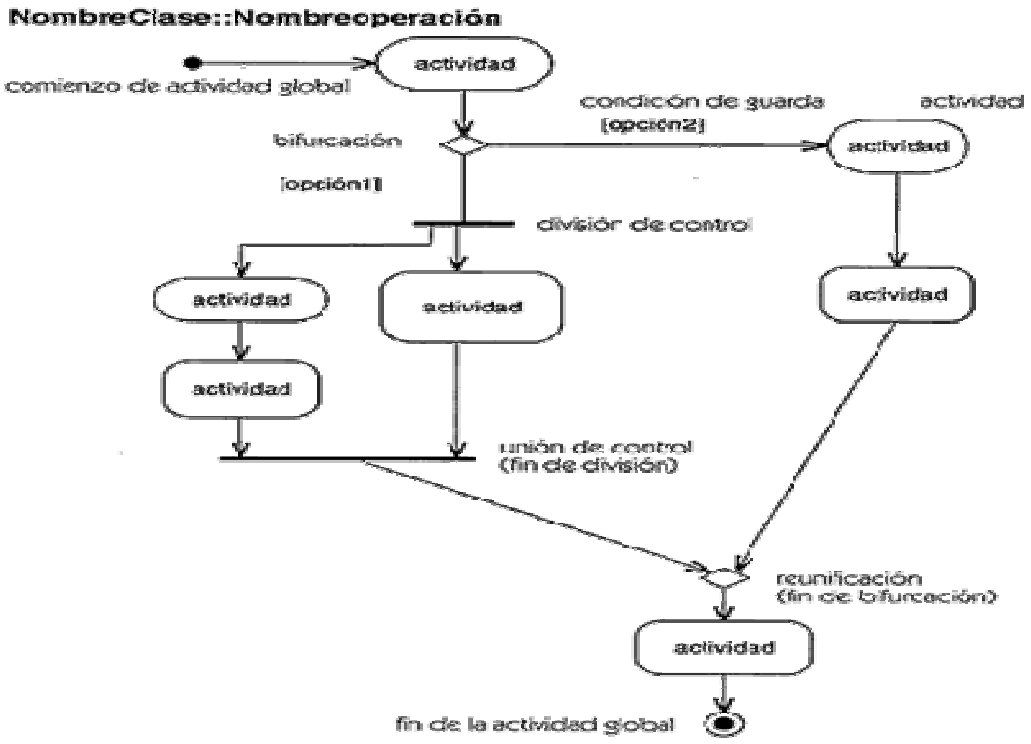



**Figura 3-5 Diagrama de Actividades**

Fuente: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x277 .html

Básicamente un diagrama de actividades contiene:

- Estados de acción
- Estados de actividad

- Inicio y Terminación
- Transiciones
- Bifurcaciones
- Calles
- Señales.

#### *3.2.1.8.1 Estado de Acción*

La representación es un rectángulo con las puntas redondeadas, en cuyo interior se representa una acción. La idea central es la siguiente:

*"Un estado que represente una acción es atómico, lo que significa que su ejecución se puede considerar instantánea y no puede ser interrumpida"*

#### *3.2.1.8.2 Estados de Actividad*

Un estado de actividad representa la realización de una o varias tareas que causa un cambio en el sistema. La representación es un rectángulo con las puntas redondeadas.
Un estado de actividad puede descomponerse en más sub-actividades representadas a través de otros diagramas de actividades. Además estos estados sí pueden ser interrumpidos y tardan un cierto tiempo en completarse.
En los estados de actividad podemos encontrar otros elementos adicionales como son: acciones de entrada (entry) y de salida (exit) del estado en cuestión.
http://www.embarcadero.com/support/what_is_uml.asp

#### *3.2.1.8.3 Inicio y Terminación*

Un flujo de control debe empezar en algún sitio y terminar El inicio se representa mediante un círculo completamente relleno.

 Inicio del flujo

 Término de flujo

Mientras que el término se representa con un círculo relleno dentro de una circunferencia.

*3.2.1.8.4 Transiciones*

Las transiciones reflejan el paso de un estado a otro, bien sea de actividad o de acción. Esta transición se produce como resultado de la finalización del estado del que parte la transición.
Como todo flujo de control debe empezar y terminar en algún momento, podemos indicar esto utilizando dos disparadores de inicio y fin.
Se representa mediante una línea dirigida que va de la actividad que finaliza hacia la actividad a ejecutar. Las flechas que indican la transición no deben ser etiquetadas pues son disparadas automáticamente por la finalización de la primera actividad.

*3.2.1.8.5 Bifurcaciones (Decisiones)*

En los diagramas de actividad se pueden incluir caminos alternativos según se cumpla alguna condición. Estas condiciones se representan mediante un rombo al cual llega la transición de la actividad origen y del cual salen las múltiples transiciones a cada actividad destino, cada una con una condición diferente.

Se pueden incluir la palabra else que indica que ninguna de las expresiones de guarda es verdad.
Una expresión de guarda es aquella que puede tomar un valor de verdadero o falso. Las alternativas de la bifurcación han de ser excluyentes y contemplar todos los casos ya que de otro modo la ejecución quedaría interrumpida.

Fuente: http://www.monografias.com/trabajos5/insof/insof.shtml

*3.2.1.8.6 Barras de Sincronización*

Al modelar flujos de procesos podemos encontrar actividades que se pueden realizar simultáneamente. A una barra de sincronización se la representa como una línea vertical u horizontal gruesa.

Las barras de sincronización muestran actividades concurrentes es decir que la ejecución de procesos es al mismo tiempo; de forma paralela.

*3.2.1.8.7 Calles*

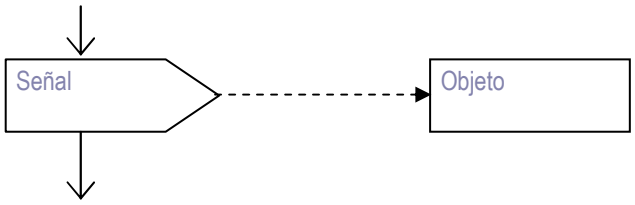
Cuando se modelan flujos de trabajo de organizaciones, es especialmente útil dividir los estados de actividades en grupos, cada grupo tiene un nombre concreto y se denominan calles. Cada calle representa a la parte de la organización responsable de las actividades que aparecen en esa calle.

*3.2.1.8.8 Señales*

Algunos objetos pueden enviar señales para la realización de alguna actividad. Estas señales pueden se de dos tipos: señal de envió y señal de recibo, aunque no son indispensables para la construcción de diagramas de actividades y pueden evitarse.

*3.2.1.8.9 Señal de envío*

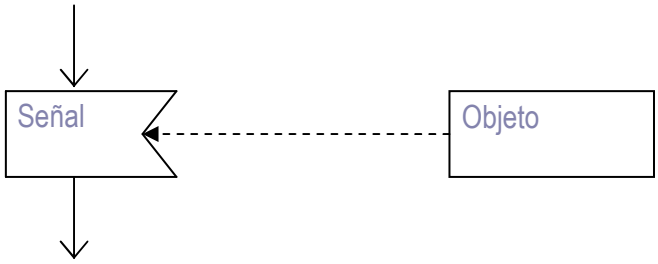
El envió es una señal, desde nuestro diagrama de actividades hacia un objeto, puede ser mostrado como un pentágono convexo que luce como un rectángulo con un triangulo apuntando hacia fuera. El significado de esta señal es mostrada dentro del símbolo.



Esta señal se coloca dentro de dos transiciones una entrando desde una actividad y otra saliendo hacia otra actividad. Opcionalmente se puede dibujar una flecha de línea discontinua desde el pentágono convexo, hacia el objeto receptor de la señal

*3.2.1.8.10 Señal de recibo*

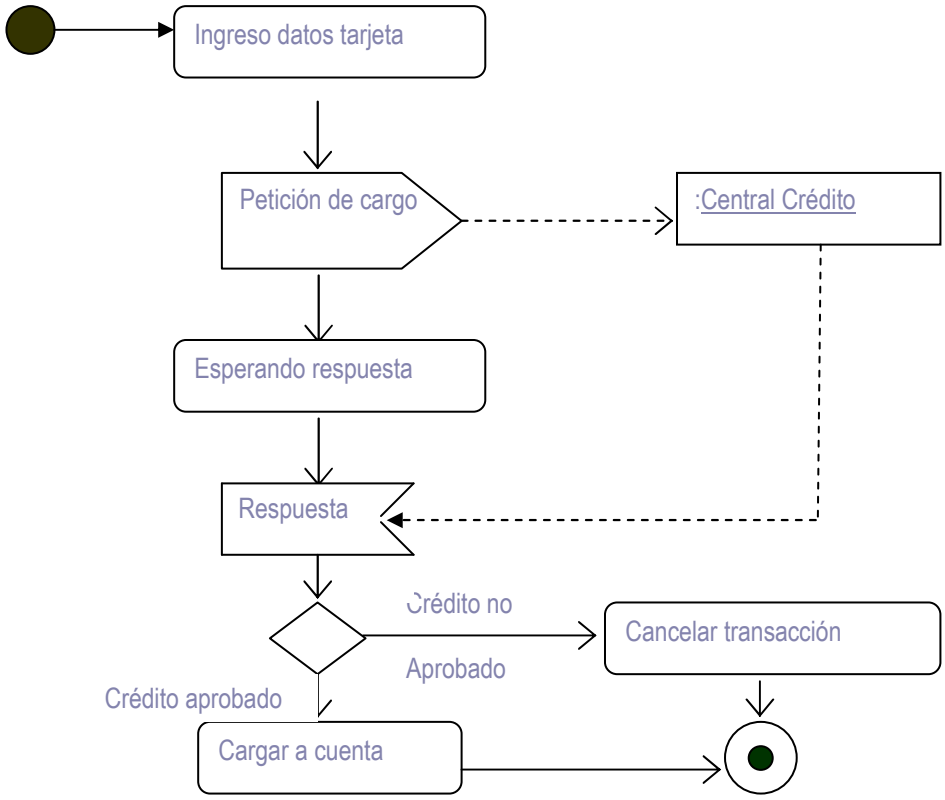
El recibo de una señal, por nuestro diagrama de actividades desde un objeto puede ser mostrado como un pentágono cóncavo que luce como un rectángulo con un triangular entrante en uno de sus lados. El significado de esta señal es mostrado dentro del símbolo.




Esta señal se coloca dentro de dos transiciones una entrando desde una actividad y la otra saliendo hacia otra actividad. Opcionalmente se puede dibujar una flecha de línea discontinua desde el objeto que envía la señal hacia el pentágono cóncavo.

**Ejemplo:**

La siguiente descripción corresponde a un proceso típico de pago con tarjeta de crédito. Cuando se realizan pagos con tarjeta de crédito debe introducirse los datos de la tarjeta, luego se dispara una señal de petición de cargo, hacia la central de crédito mientras esto ocurre estaremos esperando a que la central de crédito nos envié la señal de autorizado, para finalmente hacer el cargo respectivo a la tarjeta.




### 3.2.1.9 Diagrama Navegacional

Inicialmente, se clasifica el sitio Web a ser evaluado dentro de un Dominio Web especifico (académico, cultural, comercio-electrónico etc.) ya que un sitio puede atender una multiplicidad de perfiles de usuario y sus distintos requisitos. A continuación se identifican cada uno de estos perfiles, que llamamos Agentes. La unión de todos los mapas navegacionales compone un modelo navegacional del sitio Web.

Un Mapa Navegacional representa la visión global del sistema que tiene un determinado agente. Éste se representa como un grafo dirigido en el cual los nodos son los contextos navegacionales y los arcos son los vínculos de navegación. Un contexto navegacional es el punto de vista que tiene un agente de una parte del sitio web y los vínculos de navegación definen las relaciones existentes entre estos contextos.

Un contexto navegacional se compone por tres partes: área de definición, área navegacional y área de características avanzadas. El área de definición

muestra cuál es el tipo de contexto. Estos pueden ser contextos de exploración o de secuencia. Mientras los contextos de exploración pueden ser alcanzados desde cualquier parte del sitio web, los de secuencia pueden solamente ser alcanzados siguiendo una secuencia predefinida de enlaces de navegación. El área navegacional esta compuesta por un conjunto de clases navegacionales y relaciones entre ellas.

En un contexto navegacional aparece una clase principal, llamada clase directora desde la cual empieza la navegación y las otras clases son llamadas clases complementarias que contribuyen en aportar información adicional a las instancias de la clase directora.
Estas clases pueden estar conectadas por dos tipos de relación:

- La de contexto.
- la de dependencia contextual.

La relación de contexto es una relación binaria unidireccional que puede ser definida sobre una relación de agregación o de herencia existente entre dos clases navegacionales. Indica la dirección de navegación entre el contexto actual y el contexto destino indicado en la relación. En este tipo de relación se pueden especificar los siguientes tipos de atributos: de contexto, de enlace, de filtro y de rol.

Un atributo de contexto indica el nombre del contexto destino de la relación. Un atributo de enlace especifica el atributo de la clase destino que servirá de enlace de la navegación. En una página web, este atributo normalmente aparece en forma de enlace. Un atributo de filtro permite un filtrado de la clase destino. El filtrado es útil en las aplicaciones web para realizar búsquedas restringidas de datos conocidos a priori. Un atributo de rol indica el rol al que se hace referencia en una relación entre dos clases. Su uso sólo es imprescindible cuando existe más de una relación entre dos clases.

Una relación de dependencia de contexto, también es una relación binaria unidireccional definida sobre una relación entre clases. Pero esta relación no

implica un salto o enlace navegacional a otro contexto. Su utilidad se halla restringida a ‘complementar’ la información de una clase navegacional.

En un servicio de una clase se puede especificar enlaces de servicio. Éste está asociado a un contexto de navegación, que indica el contexto destino del salto después de la ejecución del servicio definido. Esta aproximación también posibilita definir filtros para un contexto navegacional así como su presentación. Estas primitivas permiten capturar y representar la semántica asociada a los requisitos de navegación de las aplicaciones web.

http://paginas.fe.up.pt/ipc/suporte/praticas/JISBD-01.pdf

#### 3.2.1.10 Diagrama Arquitectónico

Un modelo arquitectónico es un diagrama rico y riguroso, creado usando los estándares disponibles, en los cuales la preocupación primaria es ilustrar un sistema específico de compensaciones inherentes en la estructura y el diseño de un sistema o de un ecosistema. Los arquitectos del software utilizan modelos arquitectónicos para comunicar con otros y para buscar la regeneración del par.

Elementos claves de la definición de un modelo arquitectónico del software:
Rico - para el punto de vista en la pregunta, debe haber suficiente información para describir el área detalladamente. La meta es reducir al mínimo malentendidos, para no perpetuarlos.
Riguroso - el arquitecto ha aplicado una metodología específica para crear este modelo particular, y el modelo resultante “mira” una manera particular.

Diagrama - un modelo puede referir generalmente a cualquier abstracción que simplifique algo por la dirección de un punto de vista particular. Subclases “modelos arquitectónicos” de esta definición específicamente al subconjunto de descripciones modelo que se representan como diagramas.

Los estándares - trabajo de los estándares cuando cada uno los sabe y cada

uno los utilizan. Esto permite un nivel de comunicación que no pueda ser alcanzado cuando cada diagrama es substancialmente diferente de otro. UML es el estándar lo más a menudo posible cotizado.

Preocupación primaria - es fácil ser detallado también incluyendo muchas diversas necesidades en un solo diagrama. Esto debe ser evitado. Es mejor dibujar los diagramas múltiples, uno para cada punto de vista, que dibujar un “diagrama mega” que sea tan rico en contenido que requiere un curso del estudio de dos años entenderlo.

Ilustrar - la idea detrás de crear un modelo es comunicar y buscar la regeneración valiosa. La meta del diagrama debe ser contestar que una pregunta específica y compartir esa respuesta con otras considera si convienen, y guía su trabajo.
Regla empírica: saber que cuáles es quieres decir, y de quién trabajo te prepones influenciar con él.

Sistema del específico de compensaciones - podemos crear un diseño genérico, pero por otra parte necesitamos alterarlo a las situaciones específicas basadas en los requisitos del negocio.

Compensaciones inherentes en la estructura y el diseño - un componente no es una compensación. Las compensaciones son los primeros principios que produjeron los modelos del diseño.
Sistema o ecosistema - el modelado en general se puede hacer en diversos niveles de abstracción. Es útil modelar la arquitectura de un uso específico, completa con los componentes y las interacciones. Es también razonable modelar los sistemas de entrega necesaria los usos un proceso de negocio completo.

http://camara123.com/?u=/wiki/Software_Architectural_Model

# 4 CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES

## 4.1 Conclusiones

- El sitio web desarrollado para la Dirección de Recursos Humanos de la Escuela Politécnica Nacional proveerá un medio eficaz de información para el personal de la Institución en cuanto a los servicios que presta, actividades y procesos administrativos, además al permitir recibir inquietudes y sugerencias, se podrá mejorar continuamente los niveles de calidad del servicio.

- La aplicación web permitirá que el personal de la Institución pueda consultar de manera ágil y oportuna su información en lo referente a datos personales e historia laboral.

- La aplicación web permitirá que el personal de la Institución pueda modificar ciertos datos personales con lo cual se consigue que la actualización de esa información sea oportuna.

- El contar con una aplicación web con acceso a datos permite que la información esté disponible cuando el personal de la Institución lo requiera.

## 4.2 Recomendaciones

- Se recomienda mantener actualizada la base de datos para que el empleado pueda consultar información actualizada.

- Se sugiere implementar un módulo para los reportes de asistencia de los empleados, para que se mantengan informados semanalmente de sus

registros de entrada y salida. Por esta razón el empleado conocerá si tiene o no atrasos y faltas, sin la necesidad de acudir al departamento de Recursos Humanos.

- Se sugiere mantener actualizado las leyes, reglamentos e instructivos, formularios y manuales de procedimiento conforme vayan teniendo cambios dicha información.

# BIBLIOGRAFÍA:


- [DAR99] S. Dart, “Containing the Web Crisis Using Configuration Management,” Proc. 1st ICSE Workshop on Web Engineering, ACM, Los Angeles, May 1999.

- [OLS01] L. Olsina, G. Lafuente, G. Rossi. “Specifying Quality Characteristics and Attributes for Websites.” Lecture Notes in Computer Science 2016 Springer 2001,.

- [PRE00] R. Pressman, “Software Engineering: A Practitioner´s Approach. 5th edition,,” Mc Graw-Hill 2000. Chapter 29, “Web Engineering”.

- http://www.informandote.com/jornadasIngWEB/articulos/jiw01.pdf
- http://www.webengineering.org/beta1/wsls.aspx?vdc=beta1&vurl=start/about
- http://www.uoc.edu/masters/esp/img/873.pdf
- http://www.sonork.com/esp/web_app.html
- http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html
- http://uca.guegue.com/AnalisisyDisenoWeb/web-app.pdf
- http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node21.html
- http://www.ortizmania.com/online/articulo.asp?art=6
- http://etsit.upm.es/~alvaro/manual/manual.html
- http://es.tldp.org/LinuxFocus/pub/mirror/LinuxFocus/Castellano/May2000/article122.shtml
- http://www.desarrolloweb.com/articulos/436.php?manual=12
- http://148.228.56.77/Descargas/Descargas/SQL.pdf
- http://www.aulaclic.es/dreamweaver8/t_1_1.htm
- http://www.blog.pucp.edu.pe/item/1691
- http://www.pcm.gob.pe/portal_ongei/publicaciones/cultura/lib5103/Libro.pdf
- http://www.abits.cl/rational/desa_soft.htm

- http://www.w3c.es/Divulgacion/Guiasbreves/HojasEstilo
- http://viait.com.ar/docs/Metodologias_de_desarrollo_web.pdf
- http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r22_art5_c.pdf
- http://agamenon.uniandes.edu.co/~pfigueroa/soo/uml
- http://www.creangel.com/uml/intro.php
- http://www.creangel.com/uml/casouso.php#
- http://www.geocities.com/txmetsb
- http://www.osmosislatina.com/lenguajes/uml/casos.htm
- http://www.clikear.com/manuales/uml/diagramasestructuraestatica.asp
- http://www.dcc.uchile.cl/~psalinas/uml/modelo.html
- http://usuarios.lycos.es/oopere/uml.htm
- http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html
- http://www.clikear.com/manuales/uml/diagramainteraccion.asp
- http://www.creangel.com/uml/secuencia.php
- http://www.creangel.com/uml/colaboracion.php#
- http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x98.html
- http://www.clikear.com/manuales/uml/diagramasestados.asp
- http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x277 .html
- http://www.embarcadero.com/support/what_is_uml.asp
- http://www.monografias.com/trabajos5/insof/insof.shtml
- http://paginas.fe.up.pt/ipc/suporte/praticas/JISBD-01.pdf
- http://camara123.com/?u=/wiki/Software_Architectural_Model