ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN TECNOLÓGICA

DESARROLLO DE UN SISTEMA DE CONTROL DE TRANSPORTE MARÍTIMO EN ARQUITECTURA WEB PARA “PETROECUADOR”

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

BALLESTEROS VINUEZA MARIELA CRISTINA
BASTIDAS CÓNDOR IVONNE ALEXANDRA

DIRECTOR: ING. ROSA NAVARRETE, MSc

Quito, febrero 2006

DECLARACIÓN

Nosotras, Ballesteros Vinueza Mariela Cristina, Bastidas Cóndor Ivonne Alexandra, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de la Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Mariela Ballesteros **Ivonne Bastidas**

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Ballesteros Vinueza Mariela Cristina, Bastidas Cóndor Ivonne Alexandra, bajo mi supervisión.

ING. ROSA NAVARRETE, MSc

DIRECTOR DE PROYECTO

AGRADECIMIENTO

Agradecemos en primer lugar a Dios que nos dio la voluntad y fuerza para no decaer en el transcurso de nuestro trabajo, a nuestros padres que nos brindaron su apoyo incondicional, a nuestros amigos que nos ayudaron cuando lo necesitamos.

DEDICATORIA

Este trabajo lo dedicamos a nuestros padres, quienes se han esforzado por darnos lo mejor y nos han motivado a seguir adelante a lo largo de nuestra vida.

TABLA DE CONTENIDOS

DECLARACIÓN II

CERTIFICACIÓN III

AGRADECIMIENTO IV

DEDICATORIA V

TABLA DE CONTENIDOS VI

CAPITULO 1 1

INTRODUCCIÓN 1

1.1 AMBITO 1
1.2 PLANTEAMIENTO DEL PROBLEMA 2
1.3 OBJETIVOS 2
1.3.1 OBJETIVO GENERAL 2
1.3.2 OBJETIVOS ESPECÍFICOS 2
1.4 JUSTIFICACIÓN 3
1.5 ALCANCE 4
1.6 PRESUPUESTO 5

CAPITULO 2 6

ASPECTOS METODOLÓGICOS 6

2.1 PARADIGMA ESPIRAL ORIENTADO A LA WEB 6
2.1.1 ETAPAS 6
2.1.1.1 Planificación 6
2.1.1.2 Comunicación con el Cliente o Formulación 7
2.1.1.3 Análisis De Riesgos 7
2.1.1.4 Ingeniería 7
2.1.1.5 Generación De Páginas 7
2.1.1.6 Evaluación Cliente 8

2.2 MODELO OOHDM (OBJECT-ORIENTED HYPERMEDIA DESIGN MODEL) O MÉTODO DE DISEÑO DE HIPERMEDIA ORIENTADO A OBJETOS 8
2.2.1 DISEÑO CONCEPTUAL 9
2.2.2 DISEÑO NAVEGACIONAL 9
2.2.3 DISEÑO DE INTERFAZ ABSTRACTA 10
2.2.4 IMPLEMENTACIÓN 11
2.3 UML 11
2.3.1 DIAGRAMAS DE CASOS DE USO 13
2.3.1.1 Actor 13
2.3.1.2 Caso de Uso 14
2.3.1.3 Relaciones 14
2.3.1.3.1 Asociación 14
2.3.1.3.2 Dependencia o Instanciación 14
2.3.1.3.3 Generalización 14
2.3.2 DIAGRAMAS DE CLASES 15
2.3.2.1 Clase 15
2.3.2.1.1 Relaciones entre clases 16
2.3.2.1.2 (Herencia/Generalización) 16
2.3.2.1.3 Agregación 16
2.3.2.1.4 Asociación 17
2.3.2.1.5 Depedencia o Instancia 17
2.3.3 DIAGRAMA DE OBJETOS 17
2.3.4 DIAGRAMA DE SECUENCIA 18
2.3.4.1 Objeto/Actor 18
2.3.4.2 Mensaje a otro Objeto 18
2.3.4.3 Mensaje al mismo Objeto 18
2.3.5 DIAGRAMA DE COLABORACIÓN 19
2.3.5.1 Objeto 19
2.3.5.2 Enlaces 19
2.3.5.3 Flujo de Mensajes 19
2.3.5.4 Marcadores de Creación y Destrucción de Objetos 19
2.3.5.5 Objeto Compuesto 20

2.3.6 DIAGRAMA DE ACTIVIDADES 20
2.3.6.1 Estados de Actividad y Estados de Acción 20
2.3.6.2 Transiciones 21
2.3.6.3 Bifurcaciones 21
2.3.6.4 División y Unión 22
2.3.6.5 Calles 22

CAPITULO 3 23

ASPECTOS TEÓRICOS 23

3.1 INGENIERÍA WEB 23
3.1.1 DISEÑO WEB CENTRADO EN EL USUARIO 24
3.1.2 PLANIFICACIÓN 25
3.1.3 DISEÑO 25
3.1.4 MODELADO DEL USUARIO 25
3.1.5 DISEÑO CONCEPTUAL 26
3.1.6 PROTOTIPADO 27
3.1.7 EVALUACIÓN 27
3.1.7.1 Método por Inspección: Evaluación Heurística 28
3.1.7.2 Método de Test con usuarios 28
3.1.7.3 Implementación y Lanzamiento 28
3.1.7.4 Mantenimiento y Seguimiento 29
3.2 ARQUITECTURA DE DESARROLLO WEB 29
3.2.1 EL NAVEGADOR WEB, BROWSER 30
3.2.2 EL SERVIDOR WEB 30
3.2.3 APLICACIONES MULTINIVEL 31
3.3 HERRAMIENTAS DE DESARROLLO 33
3.3.1 LINUX 33
3.3.1.1 Distribuciones Linux 34
3.3.1.1.1 Linux RedHat 8.0 34
3.3.2 APACHE 37
3.3.2.1 ARQUITECTURA APACHE 38
3.3.3 COLDFUSION 38
3.3.3.1 Coldfusion y sus Características 39

3.3.3.2 Componentes Coldfusion 40
3.3.3.2.1 Tecnología del servidor 40
3.3.3.2.2 Herramientas de desarrollo 40
3.3.3.2.3 Ambiente de programación 40
3.3.4 ACTIONSCRIPT 43
3.3.4.1 Características generales del ActionScript 43
3.3.5 JAVASCRIPT 44
3.3.5.1 Que no es Javascript 45
3.3.5.2 Diferencias entre Javascript y Java 46
3.3.6 ORACLE 47
3.3.6.1 Historia 48
3.3.6.2 Esquema de la Base de Datos 48
3.3.6.3 Estructuras de Proceso 50
3.3.6.3.1 System Monitor, SMON 51
3.3.6.3.2 Process Monitor, PMON 51
3.3.6.3.3 Database Writer, DBWR 51
3.3.6.3.4 Log Writer, LGWR 52
3.3.6.3.5 Checkpoint, CKPT 53
3.3.6.3.6 Archiver, ARCH 53
3.3.6.3.7 Recovered, RECO 54
3.3.6.3.8 Lock, LCK 54

CAPITULO 4 55

CONCLUSIONES Y RECOMENDACIONES 55

4.1 CONCLUSIONES 55
4.2 RECOMENDACIONES 56

REFERENCIAS BIBLIOGRÁFICAS 57

ANEXOS 58

ANEXO NO.1 MANUAL TECNICO 59

ANEXO NO.2 MANUAL DE USUARIO 60

CAPITULO 1

INTRODUCCIÓN

1.1 AMBITO

Petroecuador se creó el 26 de septiembre de 1989. De acuerdo al modelo internacional se creo un holding conformado por una matriz y tres filiales: Petroproducción, Petroindustrial y Petrocomercial, a todo el conjunto se lo denominó sistema Petroecuador.

A la empresa le corresponde la planificación y coordinación de sus actividades y la de sus filiales; administrar, fiscalizar y controlar los contratos de prestación de servicios y explotación de hidrocarburos con compañías internacionales; y comercializar el crudo y sus derivados en el mercado interno y externo.

La matriz la conforman: La Presidencia Ejecutiva, El Consejo de Administración, las Gerencias de: Economía y Finanzas, Administrativa, Comercio Internacional, Oleoducto, Administración de Contratos y Protección Ambiental. Además de la Unidad Coordinadora de las Rondas de Licitación Petrolera. En el año 2003 se integró la Unidad de Aviación.

La Gerencia de Comercio Internacional se encarga de la comercialización externa que es la actividad de exportar petróleo y derivados, así como importar derivados para abastecer el mercado interno.
Su objetivo es lograr recursos para el país y satisfacer los requerimientos del consumo interno.
Son políticas de comercialización externa de crudo:

- Colocar las exportaciones de hidrocarburos en contratos a largo plazo.
- Buscar y asegurar mercados para la comercialización de crudos pesados.
- Diversificar geográficamente las exportaciones.

- Negociar con empresas de probada solvencia.
- Vincularse con empresa petroleras estatales de la región.
- Programar el abastecimiento de hidrocarburos, como parte del Comité Nacional e importar derivados en función de las necesidades del país.

La Gerencia de Comercio Internacional a través del la unidad de Trasporte Marítimo se encarga de exportar fuel oil además de nafta debutanizada, además se encarga de la programación y ejecución del transporte interno de productos en sus buques sea para cabotajes y alijes.

1.2 PLANTEAMIENTO DEL PROBLEMA

Actualmente la Unidad de Transporte Marítimo tiene un control parcial de las operaciones que se realizan en cada puerto, derivadas de un viaje, así como también de los viajes que se pueden generar a partir de un puerto internacional ya que estos viajes no proviene de ningún Programa Buques, es decir que no está Planificados y no se tiene ninguna información acerca de este.
No tienen un control ordenado de los gastos que cada una de éstas operaciones generan. La información requerida es enviada manualmente ya sea por fax u otros medios retardando así el proceso de la generación de un nuevo viaje.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Apoyar el control del Transporte Marítimo en PETROECUADOR y su integración al Sistema de Comercio Internacional, a través del desarrollo y posterior implementación de un sistema en arquitectura web.

1.3.2 OBJETIVOS ESPECÍFICOS

I. Ingresar la información de la operación de buques en las regionales (Esmeraldas, Libertad, Tres Bocas) para disponer de la misma en forma inmediata y en línea y ser utilizada en la matriz para empezar el proceso de viajes.

II. Controlar las operaciones de viajes internos y externos, es decir, todas las cargas, descargas y abastecimientos de productos que se realizan desde los barcos de exportaciones e importaciones hacia barcos pequeños o hacia tierra mediante la instalación del sistema en las regionales, ya que en este se ingresará toda la información necesaria, y esta estará disponible online para Petroecuador Matriz.

III. Controlar pagos sea generación de facturas, gastos por viaje y órdenes de pago que se efectúan tanto a FLOPEC, SUIMBA, ADUANAS, COMPAÑÍAS DE EXPORTACIÓN E IMPORTACIÓN DE PRODUCTOS y otras instituciones que están vinculadas al proceso de Transporte Marítimo.

IV. Controlar los desvíos a otros puertos de carga que no fueron planificados y los gastos que se deriven de esta operación para luego cargarlos a la compañía contratante.

V. Controlar de las cargas de combustible y de los blendings (mejoramiento de la calidad del producto), además del control de las estadías y sobrestadías del los buques durante el viaje.

VI. Controlar los subarrendamientos de los buques y carga de productos fuera del territorio nacional.

1.4 JUSTIFICACIÓN

De acuerdo a las necesidades y requerimientos de la Gerencia de Comercio Internacional y de sus regionales, se hace necesario el diseño del Sistema de Transporte Marítimo de tal manera que se integre al sistema general de Comercio Internacional con el fin de tener un sistema que vincule los embarques de exportaciones e importaciones de productos, el transporte de los barcos pertenecientes a esta unidad y proporcionar a las regionales un rol más activo en el control de sus operaciones para que les permita el ingreso y consulta de la información de la operación de embarques y de buques en línea a través de la Intranet.

Dentro de las operaciones de viajes los barcos tienen que arribar a diferentes puertos con productos específicos lo cual genera costos por diferentes conceptos los cuales derivan en la generación de facturas por parte de FLOPEC y otros organismos las mismas que no vienen ordenadas ni por puerto ni por viaje lo cual dificulta en gran medida su control mas aún cuado todo es manual, por lo que se hace necesario el desarrollo del sistema.

1.5 ALCANCE

Este módulo se encargará del control de viajes de los buques tanto a nivel nacional e internacional. Todas las operaciones nacen a partir de convenios y de contratos, los cuales son ejecutados a través de las operaciones de embarques, por lo que en el sistema se va a considerar el control de dichos contratos y convenios.

El Sistema hará el control de pagos de cada uno de los gastos que se generen en cada puerto por donde sea establecida la ruta y además los gastos por estadía, sobrestadía, demorajes, abastecimiento de combustibles, desvíos y su posterior cobro a las compañías que adquirieron el producto en el caso de ser productos de exportación (fuel oil y nafta).

Para los viajes internos se llevará el control de gastos y de las facturas y órdenes de pago que se generen por los mismos.

1.6 PRESUPUESTO

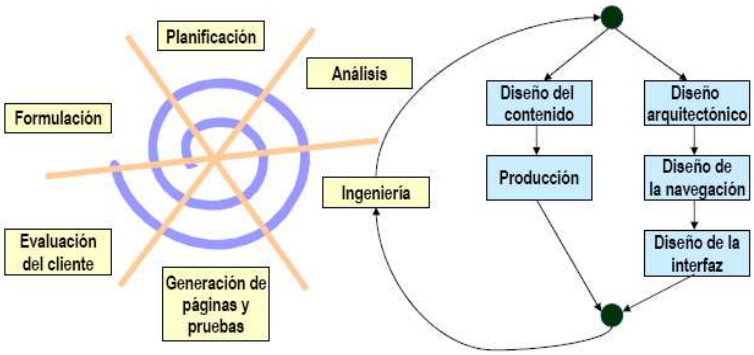
Hardware	
2 Computadores Pentium IV 2.4 GHz	3400
Impresora hp Laser Jet 1300n PCL 6	300
Infraestructura	
Insumos de Impresión	10
Software	
Windows XP	500
Macromedia Dreamweaver MX	850
ColdFusion MX	6000
Oracle	70000
Recursos Humanos	
Programadores	2000
Analistas	1200
Total	84260

CAPITULO 2

ASPECTOS METODOLÓGICOS

2.1 PARADIGMA ESPIRAL ORIENTADO A LA WEB

Modelo orientado al riesgo. Es el más versátil y flexible, pero también el más complejo. Cada vuelta de la espiral (ciclo) supone una refinación en el desarrollo.



[1]

2.1.1 ETAPAS

A continuación se indican las etapas que presenta este Modelo:

2.1.1.1 Planificación

Estima el coste global del proyecto, evalúa los riesgos asociados con el esfuerzo del desarrollo, y define una planificación del desarrollo bien granulada para el incremento final de la WebApp, con una planificación mas toscamente granulada para los incrementos subsiguientes.

---

[1] (Pressman, 525)

2.1.1.2 Comunicación con el Cliente o Formulación

Las tareas requeridas para establecer comunicación entre el desarrollador y el cliente. Actividad que identifica 'las metas y lo objetivos de la WebApp, y establece el ámbito del primer incremento.

2.1.1.3 Análisis De Riesgos

Establece los requerimientos técnicos para la WebApp e identifica los elementos del contenido que se van a incorporar. También se define los requisitos de diseño gráfico (estética). Es decir se identifica los datos y requisitos funcionales y de comportamiento para la aplicación web.

2.1.1.4 Ingeniería

Aquí se incorpora dos tareas paralelas, el Diseño del Contenido y la Producción, son tareas llevadas a cabo por personas no técnicas del equipo IWeb. El objetivo de estas tareas es diseñar, producir y/o adquirir todo el contenido de texto, gráfico y video que se vayan a integrar en la WebApp. Al mismo tiempo se lleva a cabo un conjunto de tareas de diseño.

2.1.1.5 Generación De Páginas

La generación de páginas es una actividad de construcción que hace mucho uso de las herramientas automatizadas para la creación de la WebApp. El contenido definido en la actividad de ingeniería se fusiona con los diseños arquitectónicos, de navegación y de la interfaz para la elaboración de páginas Web ejecutables en HTML, XML y otros lenguajes orientados a procesos (java). Durante esta actividad también se lleva a cabo la integración con el software intermedio (Middleware) de componentes (es decir: CORBA, DCaM o JavaBEan). Las pruebas ejercitan la navegación, intentan descubrir los errores de los applets, guiones y formularios, y ayuda a asegurar que la WeApp funcionará correctamente en diferentes entornos (por ejemplo, con diferentes navegadores).

2.1.1.6 Evaluación Cliente

En este punto es donde se solicitan cambios, tiene lugar la ampliación del ámbito. En esta activiadad se revisan los incrementos producidos como parte del proceso.[2]

2.2 MODELO OOHDM (OBJECT-ORIENTED HYPERMEDIA DESIGN MODEL) O MÉTODO DE DISEÑO DE HIPERMEDIA ORIENTADO A OBJETOS

Las metodologías tradicionales de ingeniería de software, o las metodologías para sistemas de desarrollo de información, no contienen una buena abstracción capaz de facilitar la tarea de especificar aplicaciones hipermedia. El tamaño, la complejidad y el número de aplicaciones crecen en forma acelerada en la actualidad, por lo cual una metodología de diseño sistemática es necesaria para disminuir la complejidad y admitir evolución y reusabilidad.

Producir aplicaciones en las cuales el usuario pueda aprovechar el potencial del paradigma de la navegación de sitios *web*, mientras ejecuta transacciones sobre bases de información, es una tarea muy difícil de lograr. En primer lugar, la navegación posee algunos problemas. Una estructura de navegación robusta es una de las claves del éxito en las aplicaciones hipermedia. Si el usuario entiende dónde puede ir y cómo llegar al lugar deseado, es una buena señal de que la aplicación ha sido bien diseñada.

Construir la interfaz de una aplicación *web* es también una tarea compleja; no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

En hipermedia existen requerimientos que deben ser satisfechos en un entorno de desarrollo unificado. Por un lado, la navegación y el comportamiento funcional de la aplicación deberían ser integrados. Por otro lado, durante el proceso de diseño se debería poder desacoplar las decisiones de diseño relacionadas con la

---

[2] (Pressman,526)

estructura navegacional de la aplicación, de aquellas relacionadas con el modelo del dominio.

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

2.2.1 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 hipermedia convencionales, cuyos componentes de hipermedia 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.

2.2.2 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 hipermedia. 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 hipermedia, y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos. La principal estructura primitiva del espacio navegacional es la noción de contexto navegacional. Un contexto navegacional es un conjunto de nodos, enlaces, clases de contextos, y otros contextos navegacionales (contextos anidados). Pueden ser definidos por comprensión o extensión, o por enumeración de sus miembros.

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 contexto particular.

2.2.3 DISEÑO DE 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, cómo los objetos de interfaz activarán la navegación y el resto de la funcionalidad de la aplicación, qué transformaciones de la interfaz son pertinentes y cuándo es necesario realizarlas.

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 que las modernas metodologías tienden a descuidar. En OOHDM se utiliza el diseño de interfaz abstracta para describir la interfaz del usuario de la aplicación de hipermedia.
El modelo de interfaz ADVs (Vista de Datos Abstracta5) 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.

2.2.4 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 es tenido 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.[3]

2.3 UML

El Lenguaje Unificado de Modelado preescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML es una consolidación de muchas de las notaciones y conceptos más usadas orientados a objetos. Empezó como una consolidación del trabajo de Grade

---

[3]http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r22_art5_c.pdf#search=%22oohdm%20metodologia%22

Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares. UML, en su versión 1.0, fue propuesto como una respuesta a esta petición en enero de 1997. Hubo otras cinco propuestas rivales. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1. Este documento fue aprobado por el OMG en Noviembre de 1997. El OMG llama a este documento OMG UML versión 1.1. El OMG está actualmente en proceso de mejorar una edición técnica de esta especificación, prevista su finalización para el 1 de abril de 1999.

Los inconvenientes de UML son los siguientes:

- UML no es una metodología.
- No cubre todas las necesidades de especificación de un proyecto software. Por ejemplo, No define los documentos textuales.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

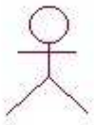
- Diagramas de Casos de Uso para modelar los procesos.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.

- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

2.3.1 DIAGRAMAS DE CASOS DE USO

[4] Estos permiten modelar el comportamiento de un sistema, el cual sirve para visualizar gráficamente cada uno de los componentes del sistema, la relación que existe entre ellos, quien hará uso de dicho comportamiento sin importar como estén implementados, esto nos ayuda a describir las funciones del sistema desde el punto de vista de los actores (usuarios, operadores, sistemas externos). En estos diagramas se especifica cada función que realizará el sistema, obteniendo así el comportamiento total del sistema en desarrollo.

2.3.1.1 Actor



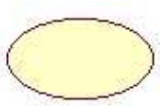
Un **Actor** es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de Vendedor con respecto al sistema puede ser realizado por un Vendedor o bien por el Jefe de Local.

---

[4] http://www.dcc.uchile.cl/~psalinas/uml/casosuso.html

2.3.1.2 Caso de Uso

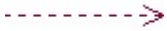


Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

2.3.1.3 Relaciones

2.3.1.3.1 Asociación

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

2.3.1.3.2 ***Dependencia o Instanciación*** 

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

2.3.1.3.3 ***Generalización***

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de **Uso** (<<uses>>) o de **Herencia** (<<extends>>).

Este tipo de relación esta orientado exclusivamente para casos de uso (y no para actores).

extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

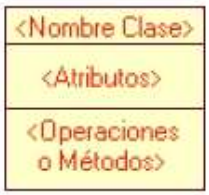
uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica. [5]

2.3.2 DIAGRAMAS DE CLASES

El Diagrama de Clases es el principal para el análisis y diseño del sistema.
Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso debería aportar información para establecer las clases, objetos, atributos y operaciones.

2.3.2.1 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.).



Tanto los atributos como los métodos pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno que puden ser:

- Public: es accesible desde todos lados, dentro y fuera de la clase.
- Private: solo será accesible dentro de la clase.
- Protected: no séra accesible desde fuera de la clase, pero si de las que se deriven de esta.[6]

---

[5] (César Liza Avila, 34)

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

2.3.2.1.1 Relaciones entre clases

Previo daremos el concepto de cardinalidad de relaciones, la cardinalidad indica el grado y nivel de dependencia que se anotan el los extremos de cada relación, y son los siguientes:
Uno o muchos 1..* (1..n)
0 muchos 0..* (0..n)
Número fijo m (m denota el número)

2.3.2.1.2 (Herencia/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).

2.3.2.1.3 Agregación

Para modelar objetos complejos, no 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 las siguientes 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 comunmente llamada **Composición** (el Objeto base se contruye 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 comunmente llamada **Agregación** (el objeto base utiliza al incluido para su funcionamiento).

2.3.2.1.4 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.

2.3.2.1.5 Depedencia o Instancia

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.

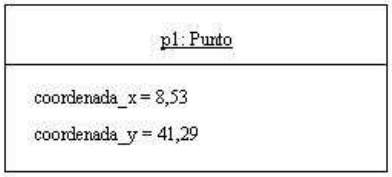
El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra.[7]

2.3.3 DIAGRAMA DE OBJETOS

Un objeto se representa de la misma forma que una clase. En el compartimento superior aparecen el nombre del objeto junto con el nombre de la clase subrayados, según la siguiente sintaxis: *nombre_del_objeto: nombre_de_la_clase* Puede representarse un objeto sin un nombre especifico, entonces solo aparece el nombre de la clase.[8]

Todas las relaciones son las mismas que en el diagrama de Clases por lo que no se las volverá a mencionar, la única diferencia que posee es que los atributos están instanciados como se muestra a continuación en la figura:

Objeto *p1* de la clase **Punto**



[7] (César Liza Avila, 72)

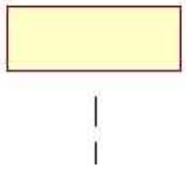
[8] http://www.clikear.com/manuales/uml/diagramasestructuraestatica.asp

Una clase describe un conjunto de objetos con características y comportamiento idéntico.

2.3.4 DIAGRAMA DE SECUENCIA

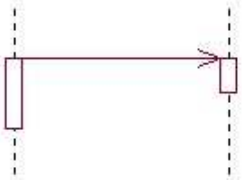
Muestra la secuencia de mensajes entre objetos durante un escenario concreto. Cada objeto viene dado por una barra Vertical. El tiempo transcurre de arriba abajo. Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua.

2.3.4.1 Objeto/Actor



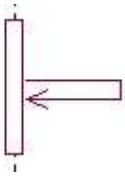
El rectángulo representa una instancia de un Objeto en particular, y la línea punteada representa las llamadas a métodos del objeto.

2.3.4.2 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.

2.3.4.3 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.[9]

2.3.5 DIAGRAMA DE COLABORACIÓN

Son útiles en la fase exploratoria para identificar objetos. La distribución de los objetos en el diagrama permite observar adecuadamente la interacción de un objeto con respecto de los demás. La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces.

2.3.5.1 Objeto

Un objeto se representa con un rectángulo, que contiene el nombre y la clase del objeto en un formato *nombreObjeto: nombreClase*.

2.3.5.2 Enlaces

Un enlace es una instancia de una asociación en un diagrama de clases. Se representa como una linea contínua que une a dos objetos. Esta acompañada por un número que indica el orden dentro de la interacción y por un estereotipo que indica que tipo de objeto recibe el mensaje. Pueden darse varios niveles de subindices para indicar anidamiento de operaciones. Los estereotipos indican si el objeto que recibe el mensaje es un atributo (association y se asume por defecto), un parámetro de un mensaje anterior, si es un objeto local o global.

2.3.5.3 Flujo de Mensajes

Expresa el envío de un mensaje. Se representa mediante una flecha dirigida cercana a un enlace.

2.3.5.4 Marcadores de Creación y Destrucción de Objetos

Puede mostrarse en la gráfica cuáles objetos son creados y destruidos, agregando una restricción con la palabra *new* o *delete*, respectivamente, cercana al rectángulo del objeto.

---

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

2.3.5.5 Objeto Compuesto

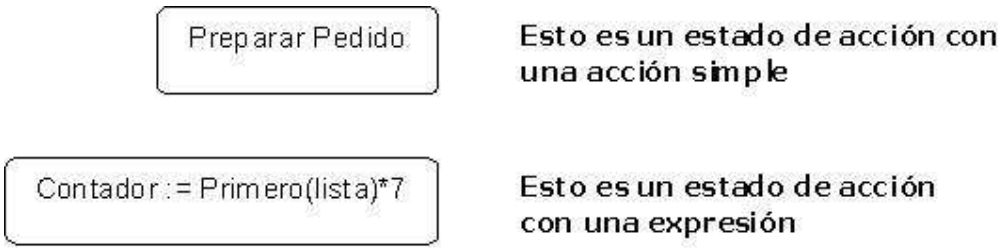
Es una representación alternativa de un objeto y sus atributos. En esta representación se muestran los objetos contenidos dentro del rectángulo que representa al objeto que los contiene. [10]

2.3.6 DIAGRAMA DE ACTIVIDADES

Un diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos.

2.3.6.1 Estados de Actividad y Estados de Acción

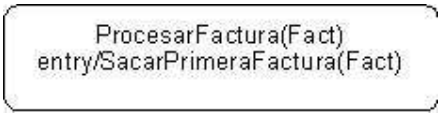
La representación de ambos es un rectángulo con las puntas redondeadas, en cuyo interior se representa bien una actividad o bien una acción. La forma de expresar tanto una actividad como una acción, no queda impuesta por UML, se podría utilizar lenguaje natural, una especificación formal de expresiones, un metalenguaje, etc. 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” En la Figura, podemos ver ejemplos de estados de acción.



En cambio un estado de actividad, sí puede descomponerse en más sub-actividades representadas a través de otros diagramas de actividades. Además

[10] http://www.cs.ualberta.ca/~pfiguero/soo/uml/colaboracion01.html

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.



2.3.6.2 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 el arco dirigido que marca la transición. Como todo flujo de control debe empezar y terminar en algún momento.

2.3.6.3 Bifurcaciones

Un flujo de control no tiene porqué ser siempre secuencial, puede presentar caminos alternativos. Para poder representar dichos caminos alternativos o bifurcación se utilizará como símbolo el rombo. Dicha bifurcación tendrá una transición de entrada y dos o más de salida. En cada transición de salida se colocará una expresión booleana que será evaluada una vez al llegar a la

bifurcación, las guardas de la bifurcación han de ser excluyentes y contemplar todos los casos ya que de otro modo la ejecución del flujo de control quedaría interrumpida. Para poder cubrir todas las posibilidades se puede utilizar la palabra ELSE, para indicar una transición obligada a un determinado estado cuando el resto de guardas han fallado.

2.3.6.4 División y Unión

No sólo existe el flujo secuencial y la bifurcación, también hay algunos casos en los que se requieren tareas concurrentes. UML representa gráficamente el proceso de división, que representa la concurrencia, y el momento de la unión de nuevo al flujo de control secuencial, por una línea horizontal ancha. En la Figura 25 podemos ver cómo se representa gráficamente.

2.3.6.5 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. [11]

---

[11] http://www.cs.ualberta.ca/~pfiguero/soo/uml/actividades01.html

CAPITULO 3

ASPECTOS TEÓRICOS

3.1 INGENIERÍA WEB

Durante la última década hemos podido ver el crecimiento vertiginoso del desarrollo y uso de aplicaciones y sistemas *Web* cada vez más complejos y sofisticados, para lo que se han creado nuevas herramientas para poder cubrir éstas necesidades.

Podemos definir la *Ingeniería Web* como el proceso utilizado para crear, implantar y mantener aplicaciones y sistemas *Web* de alta calidad. Esta breve definición nos lleva a abordar un aspecto clave de cualquier proyecto como es determinar que tipo de proceso es más adecuado en función de las características del mismo.

La aplicación de principios de ingeniería pueden evitar el caos potencial al que nos enfrentamos, y poner bajo control el desarrollo de las aplicaciones *Web*, minimizando riesgos y mejorando el mantenimiento y calidad. [12]

La mayoria de personas que hayan navegado pueden describir lo que es una buena WebApp, que cumpla con sus expectativas, pero los criterios pueden ser muy variados, las características mas relevantes son:

- Usabilidad
- Fiabilidad
- Eficiencia y
- Capacidad de mantenimiento

Estos requisitos de calidad proporcionan una base útil para evaluar la calidad de los sistemas basados en Web.

---

[12] (http://ucc.guegue.com/IngenieriaWeb/index.htm)

A continuación se describe cada uno de ellos.

- La usabilidad se refiere a la capacidad de comprensión del sitio global, a los servicios de ayuda y realimentación en línea , capacidades estéticas y de interfaz, servicios especiales.
- Funcionalidad: Capacidad de recuperación y de búsqueda, servicios de búsqueda y navegación, servicios relacionados con el dominio de aplicación.
- Fiabilidad: Proceso correcto de enlace, recuperación de errores, validación y recuperación de la entrada del usuario.
- Eficiencia: Rendimiento del tiempo de respuesta, velocidad de generación de páginas, velocidad de generación de gráficos.
- Capacidad de mantenimiento: Facilidad de corrección, adaptabilidad, extensibilidad.

3.1.1 DISEÑO WEB CENTRADO EN EL USUARIO

El Diseño Web Centrado en el Usuario se caracteriza por asumir que todo el proceso de diseño y desarrollo del sitio web debe estar conducido por el usuario, sus necesidades, características y objetivos. Centrar el diseño en sus usuarios implica involucrar desde el comienzo a los usuarios en el proceso de desarrollo del sitio; conocer cómo son, qué necesitan, para qué usan el sitio; probar el sitio con los propios usuarios; investigar cómo reaccionan ante el diseño, cómo es su experiencia de uso; e innovar siempre con el objetivo claro de mejorar la experiencia del usuario.

El proceso de Diseño Web Centrado en el Usuario se divide en varias fases o etapas, algunas de las cuales tienen carácter iterativo.

Las fases de "diseño", "prototipado" y "evaluación" son cíclicas e iterativas. Esto quiere decir que todo lo que se diseñe debe ser constantemente evaluado a través de su prototipado, para así poder corregir errores de usabilidad desde los primeros momentos del desarrollo

3.1.2 PLANIFICACIÓN

Todo proyecto debe comenzar por una correcta planificación. En esta etapa se identifican los objetivos, así como las necesidades, requerimientos y objetivos de la audiencia potencial.

Confrontando esta información se definen los requerimientos, entre los que podemos contar requerimientos técnicos (back-end y front-end), recursos humanos y perfiles profesionales necesarios, y adecuación del presupuesto disponible.

El diseñador debe obtener información precisa tanto de las necesidades y objetivos del proveedor como del usuario. En el primer caso, mediante entrevistas y reuniones con los responsables, será relativamente fácil obtener dicha información. Más dificultoso, pero al mismo tiempo más importante, es obtener esta información del usuario. Cuanto más conozcamos a la audiencia, más adaptado será el diseño y más satisfactoria la experiencia del usuario final.

La etapa de planificación se basa casi completamente en la recaudación, análisis y ordenación de toda la información posible, con el objetivo de tener una base sólida sobre la que poder tomar decisiones de diseño en las siguientes etapas del proceso.

3.1.3 DISEÑO

La etapa de Diseño es el momento del proceso de desarrollo para la toma de decisiones acerca de cómo diseñar o rediseñar, en base siempre al conocimiento obtenido en la etapa de planificación, así como a los problemas de usabilidad descubiertos en etapas de prototipado y evaluación.

3.1.4 MODELADO DEL USUARIO

Toda la información obtenida de los estudios de usuarios realizados en la anterior fase de planificación debe servir como base para comenzar el diseño.

El diseño debe estar orientado al usuario, organizando y estructurando la información según los modelos definidos de usuarios. El problema de esta técnica

de modelado de usuario es que cuando los usuarios son muchos y sus necesidades heterogéneas.

Es demasiado común que el diseñador se imagine a sí mismo usando el sitio y por tanto sea incapaz de comprender por qué a alguien le puede resultar difícil, incomodo y hasta frustrante su uso. Estos arquetipos de usuarios conseguirán precisamente que el diseñador tenga en mente a un usuario 'real', con limitaciones, habilidades y necesidades reales.

3.1.5 DISEÑO CONCEPTUAL

El objetivo de la fase de Diseño Conceptual es definir el esquema de organización, funcionamiento y navegación del sitio. No se especifica qué apariencia va a tener el sitio, sino que se centra en el concepto mismo del sitio: su arquitectura de información.

La "estructura" del sitio web se refiere precisamente a las conexiones y relaciones entre páginas, a la topología de la red de páginas, así como a la granularidad de los elementos de información contenidos en las páginas; y la "navegación" a las posibilidades y forma en que cada página presenta las opciones de desplazamiento hacia otras páginas.

Una vez definida la estructuración del sitio es necesario documentarla, para así tener un modelo de referencia sobre el que sustentar el desarrollo del sitio. La forma de documentar arquitecturas se suele hacer a través de grafos y esquemas, con el objetivo de que sean de fácil y rápida comprensión por todos los miembros del equipo de desarrollo.

Si la arquitectura es ascendente normalmente se documentará a través de diagramas entidad-relación. Por otro lado, cuando la arquitectura a documentar es la descendente, para sitios web proponemos el uso del vocabulario gráfico de **Garret**. A través de unas sencillas convenciones gráficas para la diagramación de la arquitectura, podemos definir la estructura de la información así como la navegación del sitio.

Otras tareas a llevar a cabo por el Arquitecto de Información o diseñador en la fase de Diseño Conceptual son: Definir sistemas de clasificación para los

contenidos; Elaborar índices y mapas del sitio; Aplicar metadatos a cada una de las páginas y sub-elementos de información.

3.1.6 PROTOTIPADO

La evaluación de la usabilidad del sitio web se debe realizar desde las primeras etapas de diseño través de prototipos.

La etapa de prototipado se basa en la elaboración de modelos o prototipos de la interfaz del sitio. Su aspecto no se corresponde exactamente con el que tendrá el sitio una vez finalizado, pero pueden servir para evaluar la usabilidad del sitio sin necesidad de esperar a su implementación.

En las primeras etapas de desarrollo del sitio web se puede hacer uso del prototipado en papel o de bajo coste, que consiste en reproducir los aspectos básicos de la interfaz del sitio en papel.

Otra forma de realizar prototipos es mediante la reproducción del aspecto del sitio a través de herramientas software. Mediante el procesador de textos o un simple editor HTML podemos esbozar cómo será la interfaz del sitio.

Hay que recordar que estos prototipos son reproducciones, no estados tempranos de implementación de la interfaz. Una vez que el prototipo se ha utilizado se tira, no es parte del sitio web.

La utilidad real del prototipado se fundamenta en que no tendría sentido empezar a implementar una interfaz web si no nos hemos asegurado antes de que el diseño es usable.

3.1.7 EVALUACIÓN

Existe una gran diversidad de métodos para evaluación de usabilidad como los que presentamos a continuación:

3.1.7.1 Método por Inspección: Evaluación Heurística

Los métodos de inspección de la usabilidad de un sitio web son aquellos realizados por el experto en usabilidad, y que se basan en el recorrido y análisis del sitio identificando errores y problemas de diseño.

3.1.7.2 Método de Test con usuarios

El test con usuarios es una prueba de usabilidad que se basa en la observación y análisis de cómo un grupo de usuarios reales utiliza el sitio web, anotando los problemas de uso con los que se encuentran para poder solucionarlos posteriormente.

La ventaja que ofrecen los test de usuarios frente a otro tipo de evaluaciones es que por un lado es una demostración con hechos, por lo que sus resultados son más fiables, y por otro porque posibilitan el descubrimiento de errores de diseño imposibles o difíciles de descubrir mediante la evaluación heurística.

3.1.7.3 Implementación y Lanzamiento

En la implementación del sitio es recomendable utilizar estándares (HTML, XHTML...) para asegurar la futura compatibilidad y escalabilidad del sitio. Esto se debe a que, aunque puede ser tentador utilizar tecnologías propietarias, el panorama tecnológico puede hacerlas desaparecer o cambiar en poco tiempo.

En esta etapa del desarrollo se debe llevar, así mismo, un control de calidad de la implementación, supervisando que todo funcione y responda a cómo había sido planificado, ya que la usabilidad del sitio depende directamente de la funcionalidad.

Una vez implementado el sitio y testada su funcionalidad se procede al lanzamiento del sitio, que consiste en su puesta a disposición para los usuarios.

El primer encuentro entre usuario y el sitio web modelará en gran medida la percepción que el usuario tendrá del sitio en posteriores visitas. Por ello es necesario que durante los primeros meses a partir del lanzamiento, el sitio tenga

un diseño y contenidos adaptados a este importante momento de su ciclo de vida. Es el momento de explicar a los usuarios el sitio, de enseñarles a usarlo.

3.1.7.4 Mantenimiento y Seguimiento

Un sitio web no es una entidad estática, es un objeto vivo cuyos contenidos cambian; cuya audiencia, necesidades y perfiles cambian, y que por lo tanto requiere de continuos rediseños y mejoras.

Estos rediseños deben ser muy sutiles, no se puede cambiar el aspecto y diseño de forma drástica de un día para otro, pues aunque estos cambios estén fundamentados en problemas de usabilidad descubiertos post-lanzamiento, los cambios pueden resultar dramáticos para los actuales usuarios que ya estaban acostumbrados y familiarizados con el actual diseño.[13]

3.2 ARQUITECTURA DE DESARROLLO WEB

El modo de generar páginas dinámicas ha evolucionado, desde la utilización del CGI ,*Common Gateway Interface*, hasta los *servlets* pasando por tecnologías tipo *JavaServer Pages*. Todas estas tecnologías se encuadran dentro de aquellas conocidas como *Server Side*, ya que se ejecutan en el servidor web.
Otro aspecto que completa el panorama son las inclusiones del lado del cliente, *Client Side*, que se refieren a las posibilidades de que las páginas lleven incrustado código que se ejecuta en el cliente, como por ejemplo JavaScript y programas Java.
En la figura que se presenta a continuación se puede apreciar cada tipo de tecnología involucrada en la generación e interacción de documentos Web.

---

[13] (http://www.hipertext.net/web/pag206.htm)

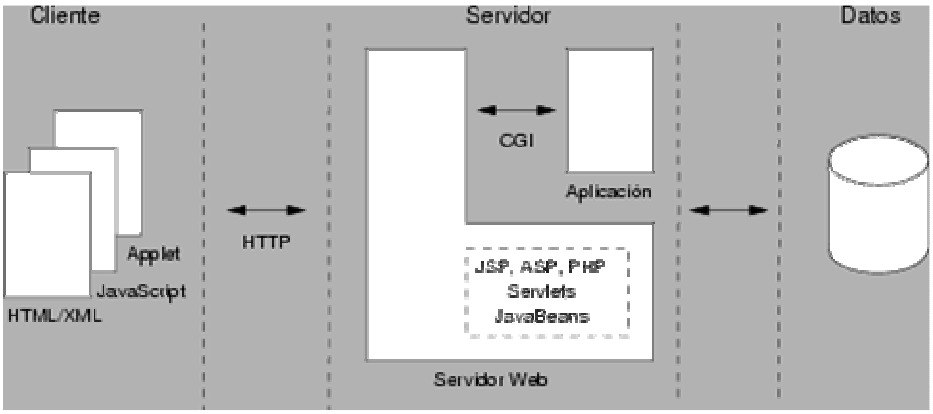


Figura: Esquema general de las tecnologías Web.

3.2.1 EL NAVEGADOR WEB, BROWSER

El navegador puede considerarse como una interfaz de usuario universal. Dentro de sus funciones están la petición de las páginas Web, la representación adecuada de sus contenidos y la gestión de los posibles errores que se puedan producir.

Para todo esto, los fabricantes de navegadores les han dotado de posibilidades de ejecución de programas de tipo *script*, con modelos de objetos que permiten manipular los contenidos de los documentos. Estos lenguajes de programación son VBScript, JScript (ambas de Microsoft) y JavaScript (de Netscape), y proporcionan las soluciones llamadas del lado del cliente, *client side* y permiten realizar validaciones de datos recogidos en las páginas antes de enviarlos al servidor y proporcionan un alto grado de interacción con el usuario dentro del documento.

3.2.2 EL SERVIDOR WEB

El servidor Web es un programa que corre sobre el servidor que escucha las peticiones que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado al cliente o navegador que realizó la petición.

El servidor Web va a ser fundamental en el desarrollo de las aplicaciones del lado del servidor, *server side applications*, que vayamos a construir, ya que se ejecutarán en él.

3.2.3 APLICACIONES MULTINIVEL

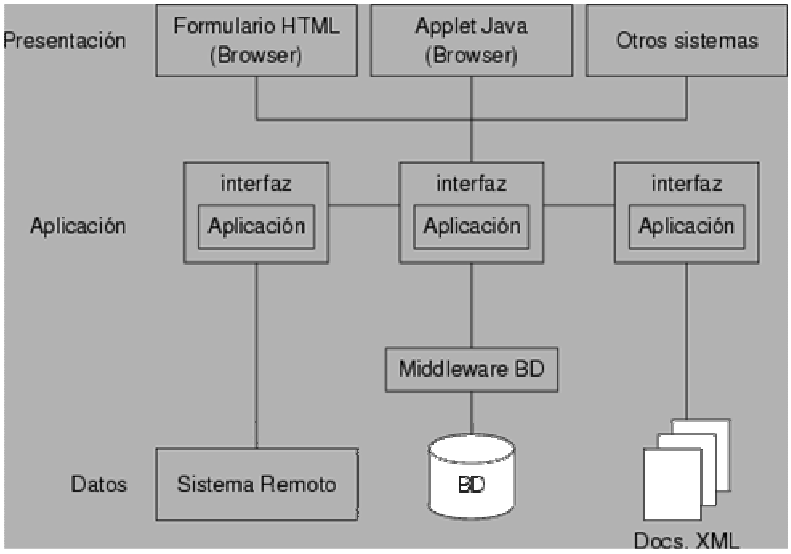
Al hablar del desarrollo de aplicaciones Web resulta adecuado presentarlas dentro de las aplicaciones multinivel. Los sistemas típicos cliente/servidor pertenecen a la categoría de las aplicaciones de dos niveles. La aplicación reside en el cliente mientras que la base de datos se encuentra en el servidor.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

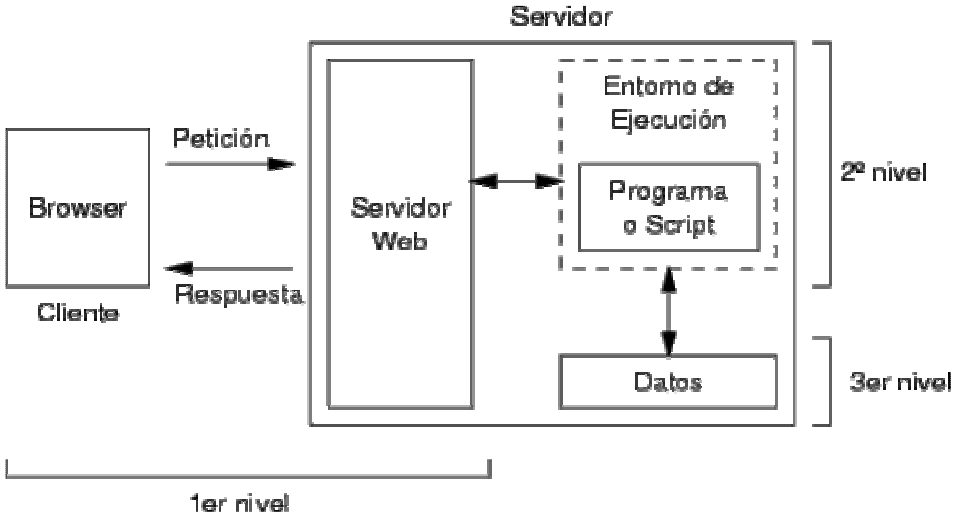
Convertir un sistema de tres niveles a otro multinivel es fácil ya que consiste en extender la capa intermedia permitiendo que convivan múltiples aplicaciones en lugar de una sola como se muestra en la figura.



Arquitectura Multinivel.

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).



Arquitectura Web de tres niveles.

[14]

3.3 HERRAMIENTAS DE DESARROLLO

3.3.1 LINUX

Linux es la denominación de un sistema operativo y el nombre de un núcleo. Es uno de los paradigmas del desarrollo de software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente estudiarlo, usarlo, modificarlo y redistribuirlo.

El término *Linux* estrictamente se refiere al núcleo Linux, pero es más comúnmente utilizado para describir al sistema operativo tipo Unix (que implementa el estándar POSIX), que utiliza primordialmente filosofía y metodologías libres (también conocido como GNU/Linux) y que está formado mediante la combinación del núcleo Linux con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software (libre o no libre). El núcleo no es parte oficial del proyecto GNU (el cual posee su propio núcleo en

[14] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/pordocente.html

desarrollo, llamado Hurd), pero es distribuido bajo los términos de la licencia GNU GPL.

La expresión *Linux* también es utilizada para referirse a las distribuciones Linux, colecciones de software que suelen contener grandes cantidades de paquetes además del núcleo. El software que suelen incluir consta de una enorme variedad de aplicaciones, como: entornos gráficos, suites ofimáticas, servidores web, servidores de correo, servidores FTP, etcétera. Coloquialmente se aplica el término *Linux* a éstas, aunque en estricto rigor sea incorrecto, dado que la distribución es la forma más simple y popular para obtener un sistema Linux.

La marca *Linux* (Número de serie: 1916230) pertenece a Linus Torvalds y se define como "*un sistema operativo para computadoras que facilita su uso y operación*".

Desde su lanzamiento, Linux ha incrementado su popularidad en el mercado de servidores. Su gran flexibilidad ha permitido que sea utilizado en un rango muy amplio de sistemas de cómputo y arquitecturas: computadoras personales, supercomputadoras, dispositivos portátiles, etc.

Los sistemas Linux funcionan sobre más de 20 plataformas diferentes de hardware; entre ellas las más comunes son las de los sistemas compatibles con PCs x86 y x86-64, computadoras Macintosh, PowerPC, Sparc y MIPS.

Asimismo, existen Grupos de Usuarios de Linux en casi todas las áreas del planeta. [15]

3.3.1.1 Distribuciones Linux

En este caso haremos referencia a Linux RedHat .

3.3.1.1.1 Linux RedHat 8.0

---

[15] http://es.wikipedia.org/wiki/Linux

Red Hat Linux es instalado con un ambiente gráfico llamado *Anaconda*, diseñado para su fácil uso por novatos. También incorpora una herramienta llamada *Lokkit* para configurar las capacidades de Cortafuegos.

La versión 8.0 fue la primera en incluir el entorno de escritorio gráfico Bluecurve.

Red Hat Linux carece de muchas características debido a posibles problemas de copyright y patentes. Por ejemplo, el soporte al formato MP3 está desactivado tanto en Rhythmbox como en XMMS; en su lugar, Red Hat recomienda usar Ogg Vorbis, que no tiene patentes. Sin embargo, el soporte para MP3 puede ser instalado luego, aunque se requiere el pago de regalías en los Estados Unidos. El soporte al formato NTFS también está ausente, pero también puede ser instalado libremente.[16]

El estándar para la estructura del sistema de archivos en Linux es el FHS (Filesystem Hierarchy Standard). Dos categorías de archivos pueden definirse en Linux: Los que se pueden y no compartir, y estáticos y no estáticos. Servirá para entender la estructura del árbol de directorios.

En una red es normal que existan documentos y aplicaciones que puedan compartirse para ahorrar espacio. Pero existe información relativa a un ordenador, que no debe compartirse. Los datos estáticos solo son modificables por el administrador de sistema o root.

En el directorio raíz contiene toda la estructura jerárquica del sistema.

/bin

Este directorio contiene archivos binarios esenciales del sistema, disponibles para todos los usuarios. Por lo que es estático y no se puede compartir. Normalmente no incluye carpetas.

/boot

Los archivos necesarios para el arranque de Linux se encuentran en esta carpeta. Estático y no puede compartirse.

/dev

Archivos de dispositivos del sistema. Linux trata todo como si fuera un archivo incluido el Hardware. La disquetera es el fichero /dev/fdo. Si el disco duro está

---

[16] http://es.wikipedia.org/wiki/Red_Hat_Linux

conectado como maestro al IDE primario es el fichero hda y hdd si es esclavo del secundario, etc. Estático y no puede compartirse.

/etc

Aquí están los archivos de configuración locales, o sea, los específicos del ordenador donde corre el sistema. Estático y no puede compartirse. Algunos de los ficheros que contiene:

Passwd: usuarios del sistema y sus contraseñas

Inittab: Configuración de init

Bashrc:Configuración de shell

/home

Aquí están los directorios personales del sistema y los usuarios. No es estático pudiéndose o no compartirse.

/lib

Dentro están las librerías compartidas, se necesitan para ejecutar los comandos y arrancar el sistema. También los módulos esenciales del kernel. Estático, no compartible.

/mnt

Se encuentran los puntos de montaje de los sistemas de archivos temporales.

/opt

Archivos de paquetes y aplicaciones.

/proc

Sistema de archivo virtual. (procesos de la memoria y el kernel).

/root

Dentro está el directorio personal del administrador del sistema. No estático, incompartible.

/sbin

Archivos binarios necesarios para arrancar el sistema. Estático, no compartible.

/tmp

Archivos temporales de aplicaciones. No estático, incompartible.

/usr

Directorio principal de almacenamiento de aplicaciones. Se encuentra en una partición diferente al resto del sistema. Dispone de su propia estructura jerárquica.

/usr/X11R6.

Para X Window.

/usr/bin.

Contiene los ejecutables no necesarios para la adm. y mantenimiento del sistema.

/usr/sbin.

Librerías necesarias para los binarios.

/usr/local.

Archivos de las aplicaciones.

/var

Contiene archivos que pueden modificarse durante el uso del sistema pero que deben quedar como estaban al volver a arrancar Linux.

Los archivos de Linux pueden tener todos los caracteres que quieras, no existe un formato estándar para sus nombres y no necesitan extensión porque en sus propiedades especifican el tipo de archivo.

Para Linux todo es un archivo. Una partición, un directorio, etc.[17]

3.3.2 APACHE

Originalmente basado en el servidor mas popular de la época: NCSA httpd a principios de 1995. El nombre Apache es "A PatCH y server".

Desde su origen a evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad.

Apache es de acuerdo al estudio hecho por Netcraft, el servidor WWW mas popular del momento.

Apache a demostrado ser substancialmente mas rápido que muchos otros servidores libres y compite de cerca con los mejores servidores comerciales. [18]

---

[17] http://structio.sourceforge.net/guias/AA_Linux_colegio/administracion-de-archivos.html

[18](http://es.tldp.org/Presentaciones/200102linuxcol/linux-serv-internet/linux-serv-internet/apache.html)

3.3.2.1 ARQUITECTURA APACHE

El servidor Apache es un software que esta estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

- **Módulos Base:** Módulo con las funciones básicas del Apache
- **Módulos Multiproceso:** son los responsables de la unión con los puertos de la máquina, acepando las peticiones y enviando a los hijos a atender a las peticiones
- **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código. El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.[19]

3.3.3 COLDFUSION

ColdFusion fue desarrollado por Allaire Corporation para ser una alternativa poderosa y simple de usar a Perl y otras tecnologías CGI. Con cientos de miles de desarrolladores en el mundo que actualmente lo usan , se puede asegurar que ColdFusion ha sido exitoso.

---

[19] **(http://www.desarrolloweb.com/articulos/1112.php).**

3.3.3.1 Coldfusion y sus Características

ColdFusion, según Macromedia, combina un lenguaje intuitivo, basado en tags, rico, con herramientas visuales y un servidor de aplicaciones web probadamente confiable, para entregar la manera más rápida de desarrollar poderosas aplicaciones web. ColdFusion es una herramienta que corre en forma concurrente con la mayoría de los servidores web de Windows, Linux y Solaris (también en servidores web personales en Windows 98 y puede ser usado para intranets). El servidor de aplicaciones web de ColdFusion trabaja con el servidor HTTP para procesar peticiones de páginas web. Cada vez que se solicita una página de ColdFusion, el servidor de aplicaciones ColdFusion ejecuta el script o programa contenido en la página. ColdFusion es un lenguaje de programación, puede crear y modificar variables igual que en otros lenguajes de programación. Posee controles de flujo de programas, como IF, Switch Case, Loop, etc. No es un lenguaje de bases de datos, pero interactúa de manera simple con bases de datos (Sybase, Oracle, MySQL,SQL, o Access). Usando SQL estándar, las páginas y aplicaciones web pueden fácilmente recuperar, guardar, formatear y presentar información dinámicamente. ColdFusion es un lenguaje basado en tags. Muchas de las funciones poderosas de ColdFusion, como leer desde y escribir en discos duros del servidor, son basadas en tags. Así como el tag <Table> puede tener argumentos como 'width' o 'align', el tag <CFFILE> tiene argumentos que especifican 'action=read/write/copy/delete', path=' etc.ColdFusion integra tecnologías. El tag <CFFORM> construirá automáticamente todo el código JavaScript para verificar los campos requeridos antes de hacer submit al form. Es escalable. ColdFusion fue diseñado para desarrollar sitios complejos y de alto tráfico. A veces, el problema más grande para un diseñador web es que su sitio se vuelve popular. ColdFusion está diseñado para correr en máquinas multi-procesador, y permite construir sitios que pueden correr en clusters de servidores. Es un lenguaje server-side. A diferencia de JavaScript y Applets Java, que corren en el cliente o en browsers, ColdFusion corre en el servidor web. Esto significa que los scripts escritos en ColdFusion correrán de la misma manera en cualquier browser.[20]

---

[20] (Macromedia ColdFusionMX, 251)

3.3.3.2 Componentes Coldfusion

ColdFusion posee los siguientes componentes los cuales lo hacen una herramienta muy eficaz para cualquier tipo de aplicación.

3.3.3.2.1 Tecnología del servidor

En el corazón de cada aplicación de ColdFusion hay un servidor ColdFusion, el cual, combina una arquitectura abierta y extensible que se integra fácilmente con sistemas existentes, así como también con aplicaciones built-in y servicios de infraestructura que ayudan a presentar la información de manera elegante y lograr un alto nivel de desempeño y confiabilidad.

3.3.3.2.2 Herramientas de desarrollo

Macromedia ofrece dos herramientas para el desarrollo. Para desarrolladores web, ColdFusion provee una tecnología poderosa de edición de código. Para diseñadores y desarrolladores web Dreamweaver UltraDev ofrece el mejor ambiente visual de desarrollo.

3.3.3.2.3 Ambiente de programación

ColdFusion soporta un poderoso lenguaje de scripting en el lado del servidor, ColdFusion Markup Language (CFML), que es extremadamente fácil de aprender y se integra limpiamente con todos los lenguajes y tecnologías web populares. ColdFusion trabaja con múltiples arquitecturas a través de la integración de COM, CORBA y EJB. También puede ser fácilmente extendido con nuevos componentes creados con Java Servlets, clases Java, o C/C++.

- Permite construir aplicaciones web rápidamente
- Acelera el desarrollo con un conjunto poderoso de herramientas poderosas de diseño, programación, depuración e implantación.
- Permite a los equipos de desarrollo colaborar de manera más efectiva compartiendo el mismo servidor y trabajando local o remotamente. Ensambla soluciones poderosas fácilmente
- Integración completa con la empresa, se conecta con todo el rango de sistemas backend, incluyendo bases de datos, servidores de mail,

directorios, y aplicaciones empaquetadas. Se integra con tecnologías de empresa y de internet, incluyendo COM, CORBA, EJB, XML, C/C++ y Java.

- Posee inteligencia de negocios. Permite crear planillas y reportes tabulares de calidad profesional.
- Completa búsqueda de texto. Permite indexar fácilmente y buscar muchos tipos de contenido, incluyendo páginas web y documentos Microsoft Office 2000.
- Arquitectura de alto desempeño. Asegura que las aplicaciones sean de implantación multiplataforma, entrega un avanzado thread pooling, caching de páginas built-in, consultas persistentes y pooling de conexiones a bases de datos.
- Administración fácil. Simplifica la implantación y la administración del servidor a través de una poderosa consola de administración basada en web, reportes robustos de servidor y herramientas de análisis, además de integración con los sistemas de administración de la empresa.
- Clustering de servidor. Provee balance de carga y recuperación automática para asegurar que las aplicaciones se mantengan consistentemente disponibles y se escala fácilmente para manejar tráfico creciente.

ColdFusion puede ser usado en un sitio cada vez que se necesita interacción con el usuario. Procesa formularios, hace seguras algunas partes del sitio, y recolecta o publica datos. Se puede usar para construir diarios murales, clientes de POP mail, calendarios en línea, y salas de chat. Se pueden escribir scripts para rastrear estadísticas.Usando ColdFusion se ahorra dinero en mantenimiento. Se gasta mucho más dinero en ajustar el software a nuevos requerimientos que en el desarrollo inicial, ColdFusion es experto en el área de mantenimiento sobre otras herramientas middleware para crear sitios web dinámicos, ya que: Esconde la complejidad, usa menos líneas y son más intuitivas para alcanzar resultados, permite al usuario migrar a otros servidores web y motores de bases de datos con pocos cambios y sin plug-ins externos .Permite setear y olvidarse de los defaults para el acceso a bases de datos, el programador puede setear los detalles de la conexión a la base de datos una vez y después sólo referirse a la fuente de los

datos con un simple nombre. En cambio, con ASP, los detalles de la conexión, como username y password, deben ser repetidos en el código cada vez que se utiliza la fuente de los datos. Permite setear y olvidarse de la administración de sesiones. Un archivo global, que es transparentemente incluido al comienzo de cada página puede asignar un ID de sesión y un símbolo que hace difícil perder la sesión y puede asociarla transparentemente (através de cookies o URLs) con todas las páginas accesadas por un cliente. La creación de símbolos y la verificación de que el símbolo está amarrado a el ID de la sesión se maneja automáticamente. Otros middleware forzan al programador a manejar estos detalles. ColdFusion simplifica el almacenamiento de variables, el programador puede manipular fácilmente las variables apropiadas a su sesión lógica. Lo mismo pasa con las variables en el servidor, en la aplicación y al nivel de página. Otros middleware necesitan más compromiso del programador y más trabajo para que sea escalable .ColdFusion hace loop implícito sobre consultas y listas. Sabe como hacer loop sobre las columnas de una consulta. Obtiene automáticamente la siguiente columna cuando no hay nada más que hacer con la columna actual. Se detiene automáticamente cuando no hay más columnas. Estos detalles no se codifican y no necesitan ser revisados durante el mantenimiento. ColdFusion genera y envía javascript transparentemente on the fly cuando ciertos tags de input son utilizados. Esto facilita el chequeo de inputs del lado del cliente sin forzar al programador a escribir, revisar y modificar javascript para hacer esto. ColdFusion usa menos líneas de código y son más intuitivas. supongamos que se necesita desplegar información de una base de datos

ColdFusion tiene un buen manejo de errores y depuración. Cuando un motor de bases de datos arroja un error, ColdFusion sugiere causas posibles. Le permite al programador crear sus propios manejadores de errores cuando se necesita cuidado especial. Con ColdFusion se pueden personalizar los mensajes de error para situaciones específicas y puede proveer un nivele de detalle para los usuarios y uno diferente para los mantenedores. Las aplicaciones en ColdFusion pueden cambiar de plataformas y motores de bases de datos. Se pueden cambiar las aplicaciones de ColdFusion a diferentes sistemas operativos y servidores web con pequeños cambios, y se puede incluso cambiar los motores de bases de

datos con un poco más de esfuerzo. Los proyectos pequeños en ColdFusion pueden crecer sin abandonar código ni empezar del comienzo. Ventajas inigualables de ColdFusion viene con habilidades que otros middleware no pueden alcanzar sin necesitar agregados. Viene con un motor para indexar sitios web. Se puede realizar balance de carga dinámico, se pueden mantener porciones de código como propietarias. Se pueden hacer consultas persistentes para mayor velocidad. Se integra con el monitor de desempeño y el monitor de seguridad de NT. Se puede modificar el registro. Permite continuar usando scripts CGI existentes. El código ColdFusion puede ser generado on the fly con herramientas que vienen con su usual editor, ColdFusion Studio. El lenguaje es extensible, convierte datos hacia y desde XML y permite conectar un sitio con otros como en un browser para extender el alcance de la aplicación. En resumen, según CFM resources: Dado que la gramática de ColdFusion lleva a menos líneas de código más entendibles, los tiempos de revisión y ajuste se acortan en un tercio o más. Esto gracias a aspectos del lenguaje que esconden la complejidad mientras proveen poder. [21]

3.3.4 ACTIONSCRIPT

El ActionScript es el lenguaje de programación que ha utilizado Macromedia Flash desde sus comienzos, y que por supuesto, emplea Flash MX 2004. A grandes rasgos, podemos decir que el ActionScript nos permitirá realizar con Flash MX 2004 todo lo que nos propongamos, ya que nos da el control absoluto de todo lo que rodea a una película Flash.

3.3.4.1 Características generales del ActionScript

Es el lenguaje de programación propio de Flash, tal y como el Lingo lo es de Macromedia Director, por ejemplo. El ActionScript está basado en la especificación ECMA-262, al igual que otros lenguajes como Javascript.

Es un lenguaje de script, esto quiere decir que no hará falta crear un programa completo para conseguir resultados, normalmente la aplicación de fragmentos de

---

[21] (http://www.tejedoresdelweb.com/307/article-5812.html)

código ActionScript a los objetos existentes en nuestras películas nos permiten alcanzar nuestros objetivos.

Es un lenguaje de programación orientado a objetos, tiene similitudes, por tanto, con lenguajes tales como los usados en el Microsoft Visual Basic, en el Borland Delphi etc... y aunque, evidentemente, no tiene la potencia de un lenguaje puramente orientado a objetos derivado del C o del Pascal como los anteriores, cada versión se acerca más a un lenguaje de este tipo. Así, la versión 2.0 recientemente estrenada en el Flash MX 2004 es mucho más potente y mucho más "orientado a objetos" que su anterior versión 1.0 presenta muchísimos parecidos con el Javascript.[22]

3.3.5 JAVASCRIPT

JavaScript es un lenguaje interpretado, al igual que VisualBasic, Perl, TCL... (lenguajes de *script*) sin embargo, posee una característica que lo hace especialmente idóneo para trabajar en Web, ya que son los navegadores que utilizamos para viajar por ella los que interpretan (y por tanto ejecutan) los programas escritos en JavaScript. De esta forma, podemos enviar documentos a través de la Web que llevan incorporados el código fuente de programas, convirtiéndose de esta forma en documentos dinámicos, y dejando de ser simples fuentes de información estáticas.

Los programas en JavaScript no son la primera forma que conoce la Web para transformar información, dado que el uso de CGIs está ampliamente difundido. La diferencia básica que existe entre un programa CGI y uno escrito en JavaScript es que el CGI se ejecuta en el servidor de páginas Web mientras que el programa en Javascript se ejecuta en el cliente (es decir, en el navegador). Por regla general, el CGI necesita unos datos de entrada (que normalmente se proporcionan mediante un formulario), los procesa y emite un resultado en forma de documento HTML. Esto implica tres transacciones en la red:

- Cargar la página del formulario
- Enviar los datos al servidor
- Recibir la nueva página que ha generado el CGI.

---

[22]http://www.aulaclic.es/flashmx_2004/t_17_1.htm

Por el contrario, los programas escritos en JavaScript se ejecutan en el navegador del cliente, sin necesidad de que intervenga el servidor. De esta forma, una sola transacción basta para cargar la página en la que se encuentra tanto el formulario, para los datos de entrada, como el programa en JavaScript que proporciona los resultados. Sin embargo, esto no significa que los CGI vayan a ser substituidos por JavaScript.

Las dos principales características de JavaScript son, por un lado que es un lenguaje basado en objetos (es decir, el paradigma de programación es básicamente el de la programación dirigida a objetos, pero con menos restricciones), y por otro JavaScript es además un lenguaje orientado a eventos, debido por supuesto al tipo de entornos en los que se utiliza (Windows y sistemas X-Windows). Esto implica que gran parte de la programación en JavaScript se centra en describir objetos (con sus variables de instancia y métodos de "clase") y escribir funciones que respondan a movimientos del ratón, pulsación de teclas, apertura y cerrado de ventanas o carga de una página, entre otros eventos.

Por último, es necesario resaltar que hay dos tipos de JavaScript. Por un lado está el que se ejecuta en el cliente. Este es el JavaScript propiamente dicho, aunque técnicamente se denomina *Navigator JavaScript*. Pero también existe un JavaScript que se ejecuta en el servidor. Este, más reciente, se denomina *LiveWire JavaScript*. Nos centraremos solo en el primero, el Navigator JavaScript.

3.3.5.1 Que no es Javascript

Ante todo no es un lenguaje de propósito general. No permite un control absoluto sobre los recursos del ordenador, tal y como estamos acostumbrados al crear un programa (aunque las últimas innovaciones de los navegadores en cuestiones de seguridad le están dando un mayor operatibilidad) . Cada programa en JavaScript solo tiene acceso al documento HTML en el que va inmerso y, si acaso, a las ventanas en las que se ejecuta el navegador dentro del cual se está ejecutando el programa en JavaScript.

Como ya se mencionó, tampoco es un sustituto de los CGIs. Ciertas cosas no se pueden hacer en JavaScript, especialmente las relacionadas con acceso a ficheros en todos sus sentidos. Sin embargo es muy útil para depurar errores en los datos antes de pasárselos al CGI que los trata, por lo que el uso combinado

de JavaScript y CGIs redunda en un mejor tratamiento de datos y un menor número de transacciones en la Web cuando se usan los CGIs.
Por último, no es un lenguaje orientado a objetos, ya que, por ejemplo, no existe el concepto de clase. Es basado en objetos, de modo que se trabaja directamente con instancias de objetos.

3.3.5.2 Diferencias entre Javascript y Java

Aunque pudiera pensarse que son una misma cosa, son dos lenguajes totalmente diferentes. Ciertos detallas son similares para ambos (lenguajes muy relacionados con la Web, basados en un paradigma de programación orientada a objetos, permiten trabajar con objetos inmersos en un documento HTML...), pero en esencia son dos lenguajes independientes.
He aquí algunas de las diferencias más notables entre Java y Javascript:

- Java es un lenguaje de próposito general, y por tanto se puede utilizar para programar aunque no sea para la Web. Javascript es específico de la Web.
- JavaScript es un lenguaje *interpretado* y ejecutado por el cliente. Java es un lenguaje *compilado* por el servidor y que puede ser ejecutado en el cliente.
- JavaScript es *basado* en objetos. Java es *orientado* a objetos. Los objetos usados en JavaScript existen cuando empieza a ejecutarse el programa; no existen clases ni por tanto ninguna de las características que de ello se derivan (herencia o encapsulación).

El código del programa en JavaScript va inscrito en la página HTML en la que se ejecuta.
Javascript es un lenguaje *débilmente tipificado*, no es necesario declarar el tipo de los datos y se realizan conversiones de unos a otros automáticamente cuando es necesario y si es viable. Java es, por contrario, *fuertemente tipificado*, como C o Pascal.
El hecho de que algunas personas confundan estos dos lenguajes puede deberse a que el compilador de Java crea un código intermedio que debe ser nuevamente traducido por *java virtual machine*, dándose la circunstancia de que los navegadores pueden traducir dicho código intermedio y, por tanto, pueden ejecutar programas escritos en Java.

JavaScript se ejecuta tanto los navegadores de Netscape como en los Internet Explorer de Microsoft (NS e IE respectivamente). En lo básico son compatibles, pero desde la última versión del lenguaje, la 1.2, han surgido ciertas diferencias que deben ser tenidas en cuenta por aquellos que diseñen páginas html (dichas diferencias se centran especialmente en la gestión del nuevo modelo de eventos). [23]

3.3.6 ORACLE

Oracle es un sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias). Otro aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL,

---

[23] http://geneura.ugr.es/~victor/cursillos/javascript/js_intro.html

MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

3.3.6.1 Historia

Oracle surge a finales de los 70 bajo el nombre de Relational Software a partir de un estudio sobre SGBD (Sistemas Gestores de Base de Datos) de George Koch. Computer World definió este estudio como uno de los más completos jamás escritos sobre bases de datos. Este artículo incluía una comparativa de productos que erigía a Relational Software como el más completo desde el punto de vista técnico. Esto se debía a que usaba la filosofía de las bases de datos relacionales, algo que por aquella época era todavía desconocido.

En la actualidad, Oracle (Nasdaq: ORCL) todavía encabeza la lista. La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo y en las oficinas de 98 de las 100 empresas Fortune 100. Oracle es la primera compañía de software que desarrolla e implementa software para empresas 100 por ciento activado por Internet a través de toda su línea de productos: base de datos, aplicaciones comerciales y herramientas de desarrollo de aplicaciones y soporte de decisiones. Oracle es el proveedor mundial líder de software para administración de información, y la segunda empresa de software independiente más grande del mundo.[24]

3.3.6.2 Esquema de la Base de Datos

Un esquema es una colección de objetos lógicos, utilizados para organizar de manera más comprensible la información y conocidos como objetos del esquema. Una breve descripción de los objetos que lo componen es la siguiente:

Tabla: Es la unidad lógica básica de almacenamiento. Contiene filas y columnas (como una matriz) y se identifica por un nombre. Las columnas también tienen un

---

[24] http://es.wikipedia.org/wiki/Oracle

nombre y deben especificar un tipo de datos. Una tabla se guarda dentro de un *tablespace* (o varios, en el caso de las tablas particionadas).

Cluster: Un cluster es un grupo de tablas almacenadas en conjunto físicamente como una sola tabla que comparten una columna en común. Si a menudo se necesita recuperar datos de dos o más tablas basado en un valor de la columna que tienen en común, entonces es más eficiente organizarlas como un cluster, ya que la información podrá ser recuperada en una menor cantidad de operaciones de lectura realizadas sobre el disco.

Indice: Un índice es una estructura creada para ayudar a recuperar datos de una manera más rápida y eficiente. Un índice se crea sobre una o varias columnas de una misma tabla. De esta manera, cuando se solicita recuperar datos de ella mediante alguna condición de búsqueda (cláusula *where* de la sentencia), ésta se puede acelerar si se dispone de algún índice sobre las columnas-objetivo.

Vista: Una vista implementa una selección de varias columnas de una o diferentes tablas. Una vista no almacena datos; sólo los presenta en forma dinámica. Se utilizan para simplificar la visión del usuario sobre un conjunto de tablas, haciendo transparente para él la forma de obtención de los datos.

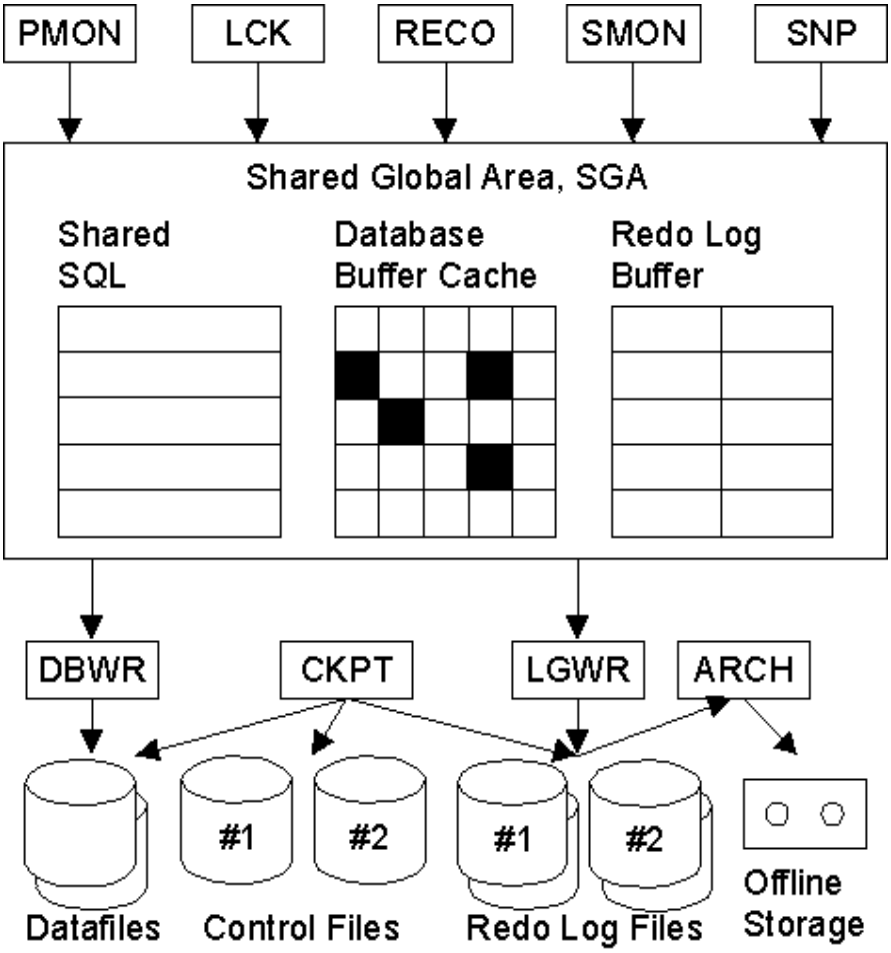
Proced. Almacenado: Son programas que permiten independizar el manejo de datos desde una aplicación y efectuarla directamente desde el motor de base de datos, disminuyendo así el tráfico de información a través de la red y mejorando el rendimiento de los procesos implementados mediante estos programas.
Trigger: Un *trigger* es un procedimiento que se ejecuta en forma inmediata cuando ocurre un evento especial. Estos eventos sólo pueden ser la inserción, actualización o eliminación de datos de una tabla.

Secuencias: El generador de secuencias de Oracle se utiliza para generar números únicos y utilizarlos, por ejemplo, como claves de tablas. La principal

ventaja es que libera al programador de obtener números secuenciales que no se repitan con los que pueda generar otro usuario en un instante determinado.[25]

3.3.6.3 Estructuras de Proceso

El servidor se vale de una serie de procesos que son el enlace entre las estructuras físicas y de memoria. A continuación se describen cada proceso y el papel que juega en la gestión de laBD. Todo esto se puede ver en la siguiente figura.



[25] http://www.lawebdelprogramador.com/cursos/enlace.php?idp=1185&id=66&texto=Oracle

3.3.6.3.1 System Monitor, SMON

El SMON es el supervisor del sistema y se encarga de todas las recuperaciones que sean necesarias durante el arranque. Esto puede ser necesario si la BD se paró inesperadamente por fallo físico, lógico u otras causas. Este proceso realiza la recuperación de la instancia de BD a partir de los ficheros *redo log*. Además límpia los segmentos temporales no utilizados y compacta los huecos libres contiguos en los ficheros de datos. Este proceso se despierta regularmente para comprobar si debe intervenir.

3.3.6.3.2 Process Monitor, PMON

Este proceso restaura las transacciones no validadas de los procesos de usuario que abortan, liberando los bloqueos y los recursos de la SGA. Asume la identidad del usuario que ha fallado, liberando todos los recursos de la BD que estuviera utilizando, y anula la transacción cancelada. Este proceso se despierta regularmente para comprobar si su intervención es necesaria.

3.3.6.3.3 Database Writer, DBWR

El proceso DBWR es el responsable de gestionar el contenido de los buffers de datos y del caché del diccionario. Él lee los bloques de los ficheros de datos y los almacena en la SGA. Luego escribe en los ficheros de datos los bloques cuyo contenido ha variado. La escritura de los bloques a disco es diferida buscando mejorar la eficiencia de la E/S.

Es el único proceso que puede escribir en la BD. Esto asegura la integridad. Se encarga de escribir los bloques de datos modificados por las transacciones, tomando la información del *buffer* de la BD cuando se valida una transacción. Cada validación no se lleva a la BD física de manera inmediata sino que los bloques de la BD modificados se vuelcan a los ficheros de datos periodicamente o cuando sucede algún *checkpoint* o punto de sincronizaión: grabación *diferida*:

- Los bloques del *buffer* de la BD (bloques del segmento de *rollback* y bloques de datos) menos recientemente utilizados son volcados en el disco continuamente para dejar sitio a los nuevos bloques.
- El bloque del segmento de *rollback* se escribe SIEMPRE antes que el correspondiente bloque de datos.
- Múltiples transacciones pueden solapar los cambios en un sólo bloque antes de escribirlo en el disco.

Mientras, para que se mantenga la integridad y coherencia de la BD, todas las operaciones se guardan en los ficheros de *redo log*. El proceso de escritura es asíncrono y puede realizar grabaciones multibloque para aumentar la velocidad.

3.3.6.3.4 Log Writer, LGWR

El proceso LGWR es el encargado de escribir los registros *redo log* en los ficheros *redo log*. Los registros *redo log* siempre contienen el estado más reciente de la BD, ya que puede que el DBWR deba esperar para escribir los bloques modificados desde el buffer de datos a los ficheros de datos.
Conviene tener en cuenta que el LGWR es el único proceso que escribe en los ficheros de *redo log* y el único que lee directamente los buffers de *redo log* durante el funcionamiento normal de la BD.
Coloca la información de los *redo log buffers* en los ficheros de *redo log*. Los *redo log buffers* almacenan una copia de las transacciones que se llevan a cabo en la BD. Esto se produce:

- a cada validación de transacción, y antes de que se comunique al proceso que todo ha ido bien,
- cuando se llena el grupo de *buffers* de *redo log*
- cuando el DBWR escribe *buffers* de datos modificados en disco.

Así, aunque los ficheros de DB no se actualicen en ese instante con los buffers de BD, la operación queda guardada y se puede reproducir. Oracle no tiene que consumir sus recursos escribiendo el resultado de las modificaciones de los datos

en los archivos de datos de manera inmediata. Esto se hace porque los registros de *redo log* casi siempre tendrán un tamaño menor que los bloques afectados por las modificaciones de una transacción, y por lo tanto el tiempo que emplea en guardarlos es menor que el que emplearía en almacenar los bloques sucios resultado de una transacción; que ya serán trasladados a los ficheros por el DBWR. El LGWR es un proceso único, para asegurar la integridad. Es asíncrono. Además permite las grabaciones multibloque.

3.3.6.3.5 Checkpoint, CKPT

Este proceso escribe en los ficheros de control los *checkpoints*. Estos puntos de sincronización son referencias al estado coherente de todos los ficheros de la BD en un instante determinado, en un punto de sincronización. Esto significa que los bloques sucios de la BD se vuelcan a los ficheros de BD, asegurándose de que todos los bloques de datos modificados desde el último *checkpoint* se escriben realmente en los ficheros de datos y no sólo en los ficheros *redo log*; y que los ficheros de *redo log* también almacenan los registros de *redo log* hasta este instante. La secuencia de puntos de control se almacena en los ficheros de datos, *redo log* y control. Los *checkpoints* se produce cuando:

- un espacio de tabla se pone inactivo, *offline*,
- se llena el fichero de *redo log* activo,
- se para la BD,
- el número de bloques escritos en el *redo log* desde el último *checkpoint* alcanza el límite definido en el parámetro LOG_CHECKPOINT_INTERVAL,
- cuando transcurra el número de segundos indicado por el parámetro LOG_CHECKPOINT_TIMEOUT desde el último *checkpoint*.

Está activo si el parámetro CHECKPOINT_PROCESS tiene un valor verdadero.

3.3.6.3.6 *Archiver, ARCH*

El proceso archivador tiene que ver con los ficheros *redo log*. Por defecto, estos ficheros se reutilizan de manera cíclica de modo que se van perdiendo los

registros *redo log* que tienen una cierta antiguedad. Cuando la BD se ejecuta en modo ARCHIVELOG, antes de reutilizar un fichero *redo log* realiza una copia del mismo. De esta manera se mantiene una copia de todos los registros *redo log* por si fueran necesarios para una recuperación. Este es el trabajo del proceso archivador.

3.3.6.3.7 Recovered, RECO

El proceso de recuperación está asociado al servidor distribuido. En un servidor distribuido los datos se encuentran repartidos en varias localizaciones físicas, y estas se han de mantener sincronizadas. Cuando una transacción distribuida se lleva a cabo puede que problemas en la red de comunicación haga que una de las localizaciones no aplique las modificaciones debidas. Esta transacción *dudosa* debe ser resuelta de algún modo, y esa es la tarea del proceso recuperador. Está activo si el parámetro DISTRIBUTED_TRANSACTIONS tiene un valor distinto de 0.

3.3.6.3.8 Lock, LCK

El proceso de bloqueo está asociado al servidor en paralelo. [26]

[26] http://www.infor.uva.es/~jvegas/cursos/bd/orarq/orarq.html

CAPITULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- En el desarrollo de este proyecto el uso de la metodología OOHDM fue de gran ayuda ya que contemplando todas las fases que esta metodología implica nos permiten un mayor control para alcanzar un producto de calidad, incluyendo las pruebas de funcionamiento.
- La herramienta ColdFusion MX permitió el desarrollo de un software con una interfase amigable lo cual satisface al usuario y permiten su fácil entendimiento, otra de las ventajas es el ahorro de líneas de código, la seguridad que ofrece ya que es una aplicación orientada a la web la necesita, además de su fácil aprendizaje.
- Como nos hemos dado cuenta en la actualidad las nuevas tecnologías en el desarrollo de sistemas basados en Web, son cada vez mas utilizadas por los programadores por lo que pretenden aprovechar todas las ventajas que ofrece este medio, en cuanto al intercambio de información entre usuarios, así como la facilidad que brinda la intranet, para dar o recibir información .
- El presente trabajo, presenta algunos resultados de aplicaciones propuestas para el módulo de Comercio Internacional las cuales son, registro de viajes y pagos en los cuales están inmersos la ruta, las operaciones de los puertos y todo lo que esto implica.

4.2 RECOMENDACIONES

- Se recomienda que en las regionales en las que se vaya usar la aplicación se dote del equipo necesario para su óptimo funcionamiento así como la capacitación a a los usuarios.
- Es necesario que se implemente las seguridades necesarias para el acceso al árbol ya que no todos podrán tener acceso a todas las opciones mucho menos modificar la información.
- Es recomendable realizar todas las pruebas de funcionamiento con los usuarios que vayan a utilizar la aplicación y que ellos vayan revisandola a medida que se avanza en el desarrollo, ya que son los que mejor nos podrán ayudar para su mejoramiento.

REFERENCIAS BIBLIOGRÁFICAS

- Liza Ávila C., Modelando con UML Principios y Aplicaciones, (2001), Primera Edición.
- Pressman Roger, (2002), Ingeniería de Software, Quinta Edición.
- Eric Ladd, Macromedia ColdFudion MX Development.
- http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r22_art5_c.pdf#search=%22oohdm%20metodologia%22
- http://www.dcc.uchile.cl/~psalinas/uml/casosuso.html
- http://www.dcc.uchile.cl/~psalinas/uml/modelo.html
- http://www.clikear.com/manuales/uml/diagramasestructuraestatica.asp
- http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html
- http://ucc.guegue.com/IngenieriaWeb/index.htm)
- http://www.cs.ualberta.ca/~pfiguero/soo/uml/actividades01.html
- http://www.hipertext.net/web/pag206.htm
- http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/pordocente.html
- http://es.wikipedia.org/wiki/Linux
- http://es.wikipedia.org/wiki/Red_Hat_Linux
- http://structio.sourceforge.net/guias/AA_Linux_colegio/administracion-de-archivos.html
- http://es.tldp.org/Presentaciones/200102linuxcol/linux-serv-internet/linux-serv-internet/apache.html
- http://www.desarrolloweb.com/articulos/1112.php
- http://www.tejedoresdelweb.com/307/article-5812.html)
- http://www.aulaclic.es/flashmx_2004/t_17_1.htm
- http://geneura.ugr.es/~victor/cursillos/javascript/js_intro.html
- http://es.wikipedia.org/wiki/Oracle
- http://geneura.ugr.es/~victor/cursillos/javascript/js_intro.html
- http://www.lawebdelprogramador.com/cursos/enlace.php?idp=1185&id=66&texto=Oracle
- http://www.infor.uva.es/~jvegas/cursos/bd/orarq/orarq.html

ANEXOS

ANEXO No.1 MANUAL TECNICO

ANEXO No.2 MANUAL DE USUARIO