Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Aplicaciones en Internet

Jossie Zambrano
Eugenio Scalise

ND 2012-02

Centro de Ingeniería de Software y Sistemas (ISYS)
Caracas, marzo 2012

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Aplicaciones en Internet

Prof. Jossie Zambrano
jossie.zambrano@ciens.ucv.ve
Prof. Eugenio Scalise
eugenio.scalise@ciens.ucv.ve

Centro de Investigación de Ingeniería de Software y Sistemas (ISYS)

Caracas, Marzo 2012

Contenido

La evolución de las Tecnologías de Información y Comunicación ha otorgado a la red Internet y al World Wide Web un papel predominante en el mundo tecnológico, social, educativo y económico. Las Aplicaciones en Internet han modificado la forma cotidiana de obtener información, realizar negocios e intercambios comerciales, recibir instrucción y comunicarse. La red Internet es un importante medio para organizaciones e individuos, quienes interactúan empleando aplicaciones desarrolladas con tecnología Internet, las cuales se han vuelto muy populares ya que poseen ventajas significativas sobre las aplicaciones tradicionales en una variedad de escenarios. Es por ello que esta nota docente presenta un compendio relacionados a Aplicaciones en Internet, cuyo propósito es proveer al estudiante un panorama de los conceptos y fundamentos tecnológicos de la red Internet y el World Wide Web, así como los métodos y técnicas en el desarrollo de estas aplicaciones.

1. Ingeniería Web

Los cambios rápidos en la tecnología e inclusive en los requerimientos traen como consecuencia que prácticas y procesos tradicionales de la ingeniería de software no sean totalmente adecuados para el desarrollo de aplicaciones web. Actualmente tiene gran auge la ingeniería web (IWeb) como una disciplina de la ingeniería de software que estudia procesos, métodos, técnicas y herramientas para la especificación, implementación, operación y mantenimiento de aplicaciones web. Según [14] la ingeniería web puede ser considerada una rama independiente de la ingeniería de software que consiste en la aplicación de enfoques sistemáticos y cuantitativos para el análisis de requerimientos, diseño, implementación, prueba, operación y mantenimiento de aplicaciones web de manera rentable.

Los sistemas y aplicaciones basados en tecnologías web, ofrecen un complejo arreglo de contenido y funcionalidad a una amplia población de usuarios finales. La ingeniería web es el proceso con el que se crean aplicaciones web de alta calidad. La IWeb no es un clon perfecto de la IS, pero toma en cuenta muchos conceptos y principios fundamentales de ella. Además, el proceso de IWeb acentúa actividades técnicas y administrativas similares. Existen sutiles diferencias en la manera como se dirigen dichas actividades, pero el método primordial dicta un enfoque disciplinado para el desarrollo de un sistema computacional.

La IWeb es de vital importancia ya que se integran cada vez más en las estrategias de negocio para pequeñas y grandes empresas (por ejemplo el comercio electrónico), crece en importancia la necesidad de construir sistemas confiables, prácticos y adaptables. Por tanto es necesario un enfoque disciplinado en cuanto el desarrollo de aplicaciones web.

Al igual que cualquier disciplina de ingeniería, la IWeb aplica un enfoque genérico que se suaviza mediante estrategias, tácticas y métodos especializados. El proceso IWeb comienza con una formulación del problema que se resolverá con la aplicación. Se planea el proyecto IWeb y se modelan los requerimientos y el diseño de la aplicación. El sistema se construye con herramientas especializadas asociadas con la web para entregar a los usuarios finales y así evalúar mediante criterios tanto técnicos como empresariales. Dado que las aplicaciones evolucionan continuamente, se deben establecer mecanismos para el control de configuraciones, el aseguramiento de la calidad y el soporte continuo. En ocasiones es difícil asegurar la calidad de la aplicación web; sin embargo, se aplican prácticas que aseguran la calidad de software para valorarla en los modelos IWeb, el contenido y la función global del sistema, la facilidad de uso, el desempeño y la seguridad.

1.1 Atributos de las Aplicaciones Web

Según [12] en la mayoría de las aplicaciones web se encuentran los siguientes atributos:

- **Intensidad de Red:** Una aplicación web reside en una red y debe satisfacer las necesidades de una variada comunidad de clientes. Una aplicación web puede residir en la Internet, Extranet o Intranet.
- **Concurrencia:** Un gran número de usuarios puede tener acceso a la aplicación web al mismo tiempo. En muchos casos, los patrones de uso entre los usuarios finales variarán enormemente.
- **Carga impredecible:** El número de usuarios de la aplicación web puede variar en órdenes de magnitud diariamente.
- **Desempeño:** Si un usuario de aplicaciones web debe esperar demasiado puede decidir irse a cualquier otra parte.
- **Disponibilidad:** Los usuarios de aplicaciones web populares frecuentemente demandan acceso "24 horas, 7 días de la semana, los 365 días del año".
- **Gobernada por los datos:** La función primordial de muchas aplicaciones es usar hipermedia para presentar contenido de texto, gráfico, audio y video al usuario final, además, por lo general, las aplicaciones web se utilizan para tener acceso a la información que existe en la BD que originalmente no eran parte integral del ambiente basado en web. (ej: comercio electrónico, aplicaciones financieras).
- **Sensibilidad al contenido:** La calidad y naturaleza estética del contenido sigue siendo un importante determinante de la calidad de una aplicación web.
- **Evolución continua:** A diferencia del software de aplicación convencional, que evoluciona a lo largo de una serie de liberaciones espaciadas cronológicamente, las aplicaciones web evolucionan continuamente.
- **Inmediatez:** La inmediatez se refiere a la apremiante necesidad de poner software en el mercado rápidamente.
- **Seguridad:** Con la finalidad de proteger el contenido confidencial y ofrecer modos seguros de transmisión de datos**,** se deben implementar fuertes medidas de seguridad a lo largo de la infraestructura que sustenta una aplicación web.
- **Estética:** Una parte de la apariencia de una aplicación web es su presentación y la disposición de sus elementos. Cuando la aplicación se diseña para comercializar o vender productos o ideas, la estética puede tener tanto que ver con el éxito como el diseño técnico.

A continuación se describe algunos fundamentos de los proceso de desarrollo de aplicaciones web, tomando en cuenta perspectivas de la ingeniería de software.

1.2 Proceso de Desarrollo de Aplicaciones Web

El proceso de desarrollo de aplicaciones web suele ser rápido debido a que los tiempos de desarrollo y los ciclos de vida de los productos son cortos. Entre los aspectos que añaden dificultad a la gestión se encuentran: alto porcentaje de contratación a terceros, el desarrollo incluye una gran variedad de personal técnico y no técnico trabajando en paralelo, el equipo de desarrollo debe dominar aspectos tan variados como, software basado en componentes, redes, diseño de arquitectura y navegación, diseño gráfico y de interfaces, usabilidad, accesibilidad, lenguajes y estándares en Internet, pruebas de aplicaciones web, entre otros, lo que hace que el proceso de desarrollo sea arduo.

La efectividad de cualquier proceso de ingeniería depende de su adaptabilidad. Esto es, la organización del equipo de trabajo del proyecto, los modos de comunicación entre miembros del equipo, las actividades de ingeniería y las tareas que deben realizarse, la información que se recolecte y se cree, y los métodos empleados para producir un producto de alta calidad deben estar adaptados a la gente que realiza el trabajo, el plazo y las restricciones del proyecto, y al problema que se quiere resolver.

Antes de definir un marco de trabajo de proceso para la IWeb se debe reconocer que las aplicaciones web con frecuencia se entregan de manera incremental, los cambios ocurren frecuentemente y los plazos son cortos.

El ciclo de desarrollo de aplicaciones web propuesto por [14] consta de las siguientes etapas:

- **Comunicación con el cliente:** La comunicación con el cliente es vital para el desarrollo de las aplicaciones web. La formulación es una actividad de recopilación de requisitos que involucran a todos los participantes.
- **Planeación:** Se crea el plan del proyecto para el incremento de la aplicación web. El plan consiste de una definición de tareas y un calendario de plazos respecto al período establecido para el desarrollo del proyecto.
- **Modelado:** Las labores convencionales de análisis diseño de la ingeniería del software se adaptan al desarrollo de las aplicaciones web, se mezclan y luego se funden en una actividad de modelado de la IWeb. El intento es desarrollar análisis rápido y modelos de diseño que definan requisitos y al mismo tiempo representen una aplicación web que los satisfaga.
- **Construcción:** Las herramientas y la tecnología IWeb se aplican para construir la aplicación web que se ha modelado. Una vez que se construye el incremento se dirige a una serie de pruebas rápidas para asegurar que se descubran los errores en el diseño.
- **Entrega y Retroalimentación:** Las aplicaciones web se configuran para su ambiente operativo, se entrega a los usuarios finales y luego comienza un período de evaluación. La retroalimentación acerca de la evaluación para realizar los procesos respectivos.

Las etapas descritas anteriormente se representan en la Figura 1 que se encuentran a continuación.

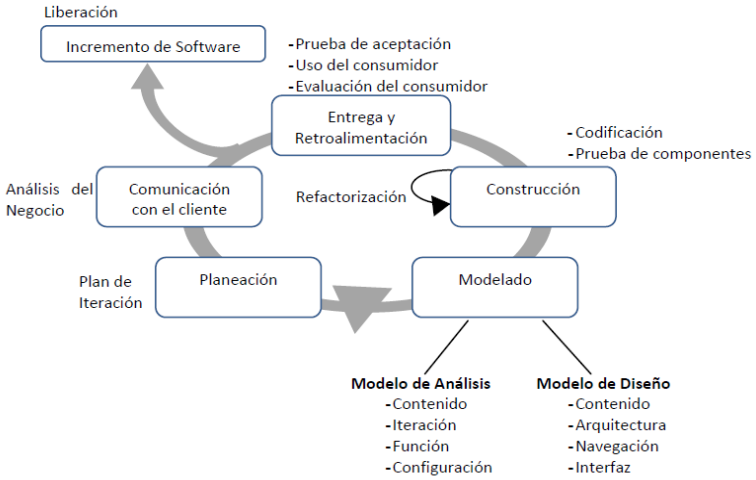

Figura 1: Ciclo de desarrollo IWeb [14]

Los métodos de la IWeb abarcan un conjunto de labores técnicas que permiten al Ingeniero web comprender, caracterizar y luego construir una aplicación Web de alta calidad.

En los últimos años se han ido produciendo metodologías para a ser incluidos en el modelo de desarrollo de aplicaciones web, quedando éstas principalmente centradas en las etapas de análisis y diseño, así como en la implementación.

En la Figura 2, se presentan algunas de las metodologías desarrolladas en el tiempo para apoyar el desarrollo de aplicaciones web, tomando en cuenta las diferentes etapas del proceso de desarrollo de software.

De estos métodos se describirán algunos, con el propósito de presentar una noción de las características de estos, tomando en cuenta la descripción de ellos en las etapas de Análisis, Diseño e Implementación del proceso de desarrollo de software.

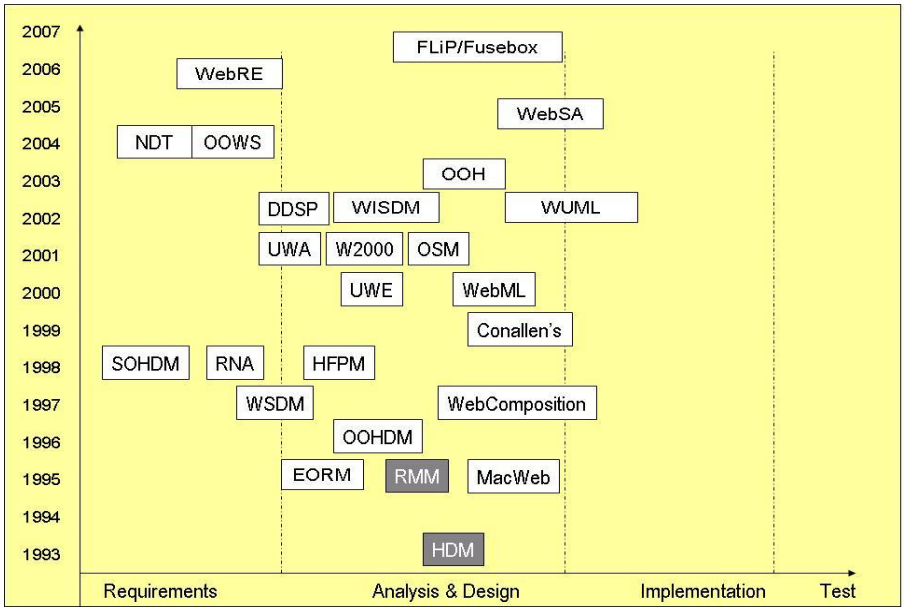

Figura 2: Métodos para la ingeniería Web [12]

1.2.1 UWE UML

UML-Based Web Engineering [5] es una herramienta para modelar aplicaciones web, utilizada en la ingeniería web, prestando especial atención en sistematización y personalización (sistemas adaptativos).

UWE es una propuesta basada en el proceso unificado y UML (Unified Modeling Language) pero adaptado a la web. En requisitos separa las fases de captura, definición y validación. Hace además una clasificación y un tratamiento especial dependiendo del carácter de cada requisito. Consiste en una notación que se basa en UML para aplicaciones web en general y adaptativas en particular. El método consta de seis modelos:

1. Modelo de casos de uso para capturar los requisitos del sistema
2. Modelo conceptual para el contenido (modelo del dominio)
3. Modelo de usuario: modelo de navegación que incluye modelos estáticos y dinámicos
4. Modelo de estructura de presentación, modelo de flujo de presentación
5. Modelo abstracto de interfaz de usuario y modelo de ciclo de vida del objeto
6. Modelo de adaptación

1.2.3 WSDM

Web Site Design Modeling centra la generación del diseño en el usuario más que en los datos. Para esto trata de definir las “clases de usuarios” que visitarán el sitio. Según estas futuras visitas, y la forma en que estos usuarios recorrerán el sitio, se establecen los parámetros de diseño. El método está compuesto por cuatro fases: modelado de los usuarios, diseño conceptual, diseño de implementación, e implementación. En la primera fase se define el tipo de información que buscarán los usuarios cuando ingresen al sitio. En el diseño conceptual se modela la información requerida y se detallan las clases de

usuarios. También se crea el diseño de navegación. En el diseño de la implementación se especifican los requisitos y restricciones del diseño gráfico del sitio, según lo definido en el diseño conceptual. Finalmente, en la implementación se selecciona el ambiente de desarrollo y se implementa el sitio [9].

1.2.4 WebML

Web Modeling Language es el Lenguaje de Modelado Web que consiste en conceptos visuales simples para expresar el hipertexto como un conjunto de páginas enlazadas que contienen unidades y operaciones acerca de los datos que esta presenta, cada concepto tiene una representación gráfica y diagramas para hacer las especificaciones [16].

WebML describe las aplicaciones web en tres niveles: El contenedor de Objetos, la organización de la presentación al usuario, y la visualización de la información (look and feel). El contenedor de objetos es especificado a través del modelo de datos Entidad-Relación o alguno equivalente como los diagramas de clases de UML. La presentación a los usuarios utiliza el modelo del hipertexto, el cual tiene una organización jerárquica

WebML provee primitivas para realizar los modelados del hipertexto, adoptando la idea del modelo de Entidad-Relación de usar conceptos para hacer especificaciones simples y expresivas, soportados por una representación gráfica intuitiva. Por lo tanto este debe ser percibido por el desarrollador como una extensión natural del modelo de Entidad-Relación, lo que le permitirá ampliar el esquema de los datos de la aplicación con la especificación del hipertexto usado para publicar y manipular los datos.

Los conceptos claves del WebML son las páginas, unidades y enlaces organizados en módulos llamados áreas y vistas.

- Las unidades son piezas atómicas de contenido publicable. Estas son bloques de construcción de las páginas, que serán presentados al usuario como elementos de la interfaz.
- Las páginas son normalmente construidas por un conjunto de unidades de varios tipos.
- Los enlaces representan la posibilidad de navegación de un punto a otro en el hipertexto, y el pase de parámetros de una unidad a otra.
- Las vistas son agrupaciones de páginas que representan un conjunto coherente de requerimientos, datos y necesidades específicas para un grupo de usuarios.
- Las áreas son agrupaciones de vistas dispuestas jerárquicamente, con un propósito común.

1.2.5 WebSA

Web Software Architecture propone la inclusión de modelos de arquitectura de software para complementar la especificación de las aplicaciones web, y el uso del estándar MDA (Model-Driven Architecture) para formalizar y describir los modelos [17].

WebSA se basa en la separación de la descripción de la arquitectura web en varias vistas simultáneas. Estas vistas están basadas en el estándar MDA.

El modelo de vistas muestra los enlaces entre las diferentes vistas (consideradas como un conjunto de artefactos creados durante el proceso de desarrollo del software) que conforman la arquitectura de software de la aplicación web.

En WebSA el modelo de la aplicación web está formado por ocho (8) vistas agrupadas en puntos de vistas.

- El punto de vista de requerimientos tiene la información necesaria para especificar el sistema, consta de:
    1. Requerimientos funcionales y,
    2. Requerimientos no funcionales.
- El Punto de vista funcional define las funcionalidades de la aplicación web, consta de las vistas de:
    3. Conceptos, las cuales capturan la estructura de la información del sistema,
    4. Procesos que tienen lo correspondiente a las actividades y flujos de los procesos,
    5. Presentación, refleja todo lo concerniente a la apariencia de la aplicación y,
    6. Navegación, que especifica qué puede hacer el usuario y los pasos que debe seguir para hacerlo en los diferentes escenarios de la aplicación.
- El punto de vista de la arquitectura a través de las vistas de:
    7. Arquitecturas Lógicas que agrupan los componentes lógicos del sistema (módulos y componentes de software) y sus relaciones y,
    8. Arquitecturas Físicas que agrupan los componentes físicos que integran la representación final (clientes, servidores, redes, etc.).

2. Aplicaciones Web

En la ingeniería software se denomina aplicaciones web a aquellas que los usuarios usan accediendo a un servidor web a través de Internet o de una Intranet mediante un navegador web.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones web para comercio electrónico, enciclopedias, comunidades virtuales, y un sinfín de servicios que se ofrecen en Internet.

Existen diversas clasificaciones de las aplicaciones web propuestas [7] por diversos autores, entre ellas destaca la propuesta en la que se tipifican las aplicaciones web en base a dos dimensiones o criterios: la complejidad y el momento de su aparición tal como se puede apreciar en la Figura 3. Bajo esta clasificación se observa que existe una relación proporcional entre la complejidad de un tipo de aplicación web y su ubicación en el tiempo; es decir, en la medida que las tecnologías y el desarrollo de aplicaciones web evolucionan, los requerimientos también. Entre los tipos de aplicaciones se destacan las orientadas a documentos, las interactivas, las transaccionales, las basadas en un flujo de trabajo, las orientadas a portales, las colaborativas, las orientadas hacia una red social, las ubicuas y las que implementan técnicas de la web semántica.

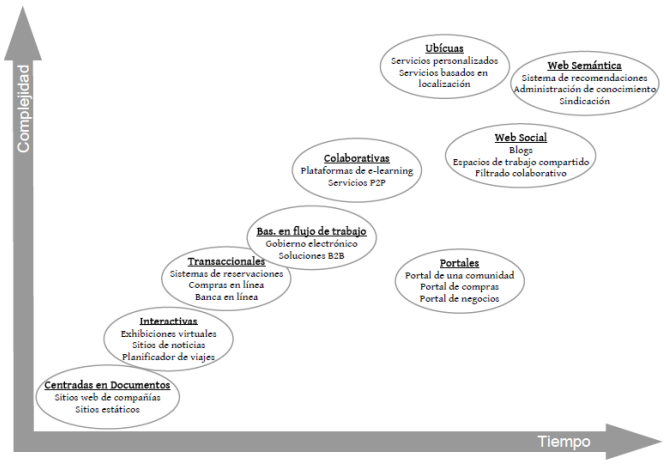

Figura 3: Clasificación de Aplicaciones Web [14]

2.1 Arquitectura de Aplicaciones Web

La arquitectura de aplicaciones web es similar a la arquitectura de software de un sistema de computación [5], consiste en su estructura, conformada por componentes de software, propiedades externas visibles de esos componentes y las relaciones entre ellos. En general, para definir la arquitectura de una aplicación se requieren decisiones sobre:

- La organización del sistema de software, ya sea estructura estática (relación entre los elementos que la componen) o su estructura dinámica (relaciones que cambian en el tiempo)
- La selección de los elementos estructurales y sus interfaces
- Los componentes que forman y la aplicación y las interacciones entre ellas
- La organización de las distintas unidades ejecutables que deben ser desplegadas o instaladas.

Existen diversos estilos arquitectónicos, uno de ellos es el de cliente-servidor, el cual se describe a continuación.

2.1.1 Cliente Servidor

Se puede definir como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. En la arquitectura cliente servidor (ver Figura 4), el cliente envía una petición solicitando un determinado servicio a un servidor, y éste envía una o varias respuestas.

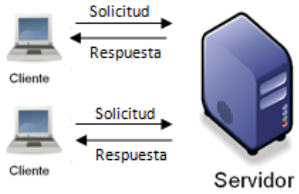

Figura 4: Cliente - Servidor

2.1.1.1 Cliente

Es el proceso que permite al usuario formular los requerimientos y enviarlos al servidor, también se le conoce con el término front-end. El cliente habitualmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces de usuario, además de acceder a los servicios distribuidos en cualquier parte de una red.

2.1.1.2 Servidor

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor también se le conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Entre las principales características de la arquitectura cliente/servidor se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz de comunicación.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Para realizar la comunicación entre el cliente y el servidor es necesario utilizar un protocolo de transferencia. Las aplicaciones web se basan en una arquitectura cliente-servidor, donde el cliente es un navegador web y el servidor es el servidor web, en este contexto el protocolo de transferencia es el protocolo de hipertexto (HTTP) el cual se describe a continuación.

2.1.1.3 Protocolo de Transferencia de Hipertexto HTTP

Es un protocolo cliente-servidor que articula los intercambios de información entre los clientes web y los servidores HTTP. Fue desarrollado por el consorcio W3C y la IETF, colaboración que culminó en 1999 con la publicación de una serie de RFC, siendo el más importante de ellos el RFC 2616, que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo sin estado orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. A la información transmitida se la llama recurso y se identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, entre otros.

El método HTTP le indica al servidor que hacer con el URL (Uniform Resource locator), por último la versión simplemente indica el número de versión del protocolo que el cliente entiende. Una petición habitual utiliza el método GET para pedirle al servidor que devuelva el URI solicitado (`GET /index.html HTTP/1.0`). Dentro de los métodos HTTP se encuentran los siguientes:

- GET: Devuelve el recurso identificado en la URL pedida
- HEAD: Funciona como el GET, pero sólo devuelve la información de cabecera
- POST: Indica al servidor que se prepare para recibir información del cliente. Suele usarse para enviar información desde formularios.
- PUT: Envía el recurso identificado en la URL desde el cliente hacia el servidor
- OPTIONS: Pide información sobre las características de comunicación proporcionadas por el servidor. Le permite al cliente negociar los parámetros de comunicación
- TRACE: Inicia un ciclo de mensajes de petición. Se usa para depuración y permite al cliente ver lo que el servidor recibe
- DELETE: Solicita al servidor que borre el recurso identificado con el URL
- CONNECT: Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel.

2.1.1.4 Servidor Web

Es un servidor que implementa el protocolo HTTP, su objetivo es devolver el recurso solicitado y manejar el protocolo. Uno de los servidores web más conocidos es APACHE HTTP SERVER que presenta entre otras características la flexibilidad en la configuración, bases de datos de autenticación y protocolos con transmisión cifrada como HTTPS (Hypertext Transfer Protocol Secure).

2.1.1.5 Servidor de Aplicaciones

Es un componente encargado de procesar solicitudes de páginas dinámicas con alguna tecnología del lado del servidor. Este puede cumplir las funciones de un servidor web más el despacho y ejecución de aplicaciones del lado del servidor. El servidor de aplicaciones recibe la petición, y tiene reglas internas para procesarla, y el resultado lo envía al cliente web.

De acuerdo a la tecnología existen variantes de servidor de aplicaciones. Algunos ejemplos para tecnologías específicas se enuncian a continuación:

- En el caso de CGI (Common Gateway Interface) el servidor HTTP puede configurarse con intérpretes de acuerdo a un módulo para ejecutar los scripts.
- En el caso de PHP los archivos .php son procesados por un módulo que se integra al APACHE HTTP SERVER.
- Para la tecnología ASP los archivos .asp son procesados por el servidor web y de aplicaciones IIS (Intenet Information Server).
- En Java, se puede usar TOMCAT como el contenedor de servlets y las asociaciones son por especificaciones en archivos XML de configuración. En este caso, la definición general es "Contenedor de Servlets", y el término servidor de aplicaciones lo usan para referirse a un servidor de aplicaciones JEE en los cuales se pueden correr componentes distribuidos.

2.2 Arquitecturas de Software para Aplicaciones Empresariales

En la actualidad, es común que la arquitectura de software de una aplicación cliente/servidor se organice descomponiendo sus partes en lo que comúnmente se conoce como tiers (niveles) o layers (capas). Esta separación facilita el desempeño de aplicaciones de alta demanda por parte de los usuarios, así como también facilita el diseño e implementación, además de promover el mantenimiento y la separación de responsabilidades.

En general, una aplicación dividida en múltiples niveles se denomina aplicación cliente/servidor n-niveles (o n-capas). La diferencia fundamental entre los dos términos tiene que ver con la forma como se plantean las divisiones al proponer la arquitectura, la cual puede ser física o lógica.

Una partición física se refiere a que la las diferentes partes de una aplicación están instaladas en diferentes máquinas que se comunican mediante una red. Cuando se establece una partición lógica, la idea principal consiste en estructurar el código de manera que cada nivel posee una responsabilidad principal y se definen funciones para las responsabilidades de cada capa; todo esto independiente de la configuración física o despliegue utilizado.

El estilo que se utiliza con mayor frecuencia es de tres o más niveles. Las tres responsabilidades que de manera estándar se establecen son: presentación, lógica de negocio y datos. Esta división fomenta el bajo acoplamiento entre los componentes del sistema y facilita el mantenimiento, la escalabilidad y robustez del sistema [5].

A continuación se presenta un escenario usual de divisiones físicas y lógicas para una aplicación web (ver Tabla 1).

	Físico	**Lógico**
Presentación	Navegador/Browser	Contiene scripts del lado cliente que se ejecutan en el navegador y scripts que se ejecutan en el contexto del servidor web.
Lógica de Negocio	Web/Application Server	Contiene componentes que encapsulan la lógica del problema, los cuales se ejecutan en el contexto del servidor web/aplicaciones.

Datos	Servidor de Datos	Contiene componentes para el acceso a los datos (ejecutándose en el servidor Web/aplicaciones) y stored procedures que se ejecutan en el contexto del servidor de Base de Datos (Suponiendo que el servidor de datos es un SMBD)

Tabla 1: Niveles lógicos y físicos para una aplicación Web [5]

2.3 Patrones

Cada patrón describe un problema que ocurre con frecuencia o regularidad, y describe la solución, de tal forma que esta pueda ser utilizada cada vez que se presente el problema, sin tener que hacer la misma solución y de la misma manera dos veces. Los patrones están enfocados en soluciones particulares que sean comunes y efectivas para solucionar uno o más problemas recurrentes [13].

Una característica clave de los patrones es que están muy presentes en la práctica. Se puede conseguir patrones en lo que la gente hace a diario, al observar sus trabajos y actividades, y determinando cómo son las soluciones a las situaciones que se les presentan. Esto no es un proceso fácil, pero una vez que se consiguen algunos buenos patrones, ellos se convierten en algo valioso. Para trabajar con patrones lo más importante es saber qué son, qué problemas ellos resuelven y cómo lo resuelven.

Aunque los patrones son soluciones exitosas a problemas recurrentes, siempre debe buscarse la forma de aplicarlos bajo las circunstancias particulares de cada problema, por lo que ellos no pueden ser aplicados a priori o a ciegas, ya que aunque se tenga la misma solución varias veces, está no es exactamente igual.

2.3.1 Patrón de Diseño Arquitectónico Modelo Vista Controlador

Es un patrón de diseño aplicado a arquitecturas de software que establece tres componentes que interactúan entre sí (Modelo, Vista y Controlador). El modelo representa los datos y operaciones/lógica de negocio. La Vista es una representación del modelo en interfaces de usuario, y el controlador representa la lógica de control que obtiene las peticiones de la vista para cambiar o acceder al modelo. MVC desacopla estas partes para mejorar el mantenimiento y promover la reusabilidad. Hoy en día este patrón es muy usado y sirve como base en muchos frameworks web porque define responsabilidades específicas, lo cual permite modificaciones en cualquiera de los componentes con bajo impacto en los otros.

La arquitectura de las aplicaciones web describe una infraestructura que permite a un sistema o aplicación basada en web lograr sus objetivos de negocio.

Las aplicaciones deben construirse con el uso de capas en la que se tomen en cuenta las diferentes preocupaciones; en particular, los datos de la aplicación se deben separar de los contenidos de las páginas, y éstos a su vez deben estar claramente separados de la apariencia y la percepción de la interfaz.

La arquitectura MVC (ver Figura 5), es uno de los modelos de infraestructura sugerido para aplicaciones web el cual permite desacoplar la interfaz de usuario de la funcionalidad y de contenido de información.

- El modelo contiene todo el contenido específico de la aplicación y la lógica de procedimientos como el acceso a fuentes de datos.
- La vista contiene todas las funcionalidades específicas de la interfaz y habilita la presentación del contenido y la lógica de procesamiento.
- El controlador gestiona el acceso al modelo y a la vista, coordina el flujo de datos entre ellos.

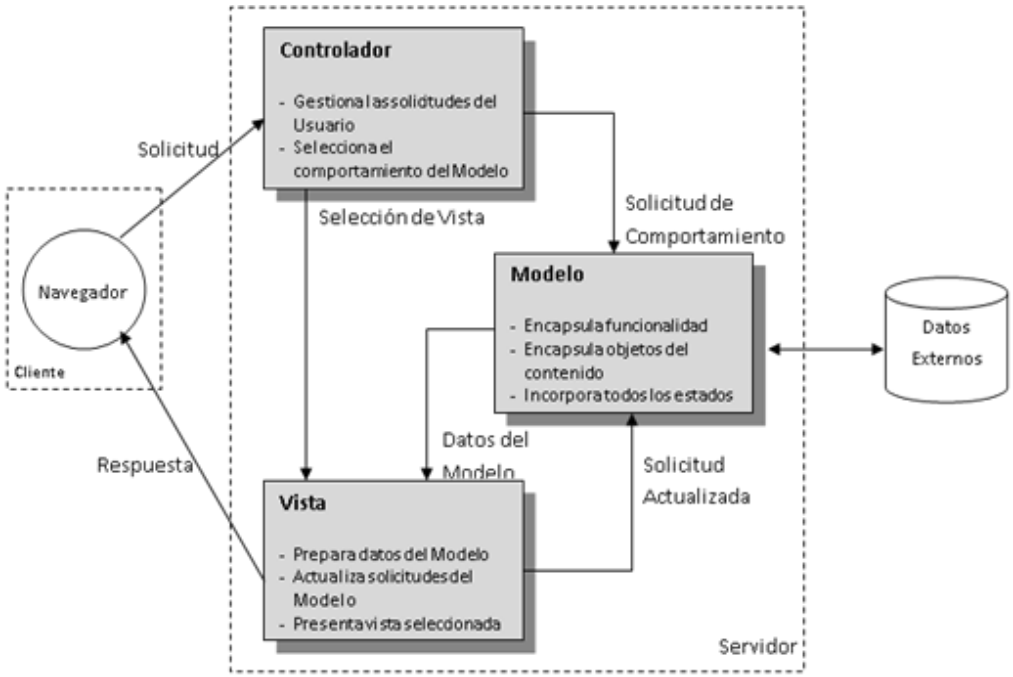

Figura 5: Modelo Vista Controlador [14]

2.3.2 Patrón de Diseño Arquitectónico Active Record

ActiveRecord es un patrón en el cual, el objeto contiene los datos que representan a un registro de la tabla o vista, además de encapsular la lógica necesaria para acceder a la base de datos. De esta forma el acceso se presenta de manera uniforme a través de la aplicación [13].

Los objetos mantienen tanto los datos como la gestión de los mismos. Muchos de estos datos son persistentes lo que implica que es necesario almacenarlos en la base de datos, situando el acceso en el propio objeto de negocio. De este modo es evidente como manipular la persistencia a través del mismo.

Una clase Active Record consiste en el conjunto de propiedades que representa las columnas de la tabla más los típicos métodos de acceso como las operaciones CRUD, búsqueda (Find), validaciones, y métodos de negocio.

Gran parte de este patrón viene de un Domain Model y esto significa que las clases están muy cercanas a la representación en la base de datos. Cada Active Record es responsable de si mismo, tanto en lo relacionado con persistencia como en su lógica de negocio.

La estructura del Active Record podría ser igual a la de la base de datos, existiendo un atributo en la clase para cada columna de la tabla correspondiente. El Active Record Generalmente proveerá métodos como:

- Construir una instancia de la clase desde una fila de un resultado de una consulta.
- Construir una nueva instancia para insertarlo posteriormente en la tabla.
- Un método estático para buscar.
- Actualización e inserción en la tabla.
- Getter y Setters o bien propiedades.
- Implementará algunas piezas de la lógica de negocio.

2.3.3 Patrón de Diseño Arquitectónico Table Data Gateway (TDG)

Incorporar sentencias SQL (Structured Query Language) en la lógica de las aplicaciones puede causar problemas de rendimiento, ya que los desarrolladores normalmente no son expertos, ni conocen las mejores prácticas, técnicas y métodos para realizar los accesos a las bases de datos [13].

En este patrón se utiliza un objeto para que actúe como un intermediario de accesos a una tabla de la base de datos. Este objeto maneja todos los accesos a una tabla o una vista, permitiendo ejecutar sentencias SQL: Select, Insert, Update, y Delete, pero otros objetos deben invocar los métodos de éste para realizar toda la interacción con la base de datos.

El Table Data Gateway (TDG) tiene una interfaz simple normalmente con un conjunto de métodos de búsquedas de datos en la base de datos y métodos para realizar las operaciones de Insert, Update y Delete. Cada método toma los parámetros de entrada y los utiliza para completar sentencias SQL, las cuales se ejecutan a través de una conexión a base de datos. Normalmente, una instancia del TDG es un objeto sin estados ya que su único rol es ejecutar sentencias en el manejador Base de Datos.

2.3.4 Patrón de Diseño Arquitectónico Data Mapper

Objetos y bases de datos relacionales tienen diferentes mecanismos para estructurar los datos. Muchas características de los objetos tales como, colecciones y herencia no están presentes en las bases de datos relacionales. El esquema de objetos y el de la base de datos relacional se diferencian, cuando se construye un modelo de objetos con una lógica de negocio asociada, se logra mejorar los mecanismos para organizar los datos y el comportamiento que estos tienen, teniendo como consecuencia que la transferencia de datos entre los dos esquemas se haga muy complejo [13].

El Data Mapper es una capa de software que permite separar los objetos de memoria de las bases de datos, pero asumiendo éste la responsabilidad de la transferencia de datos entre ellos, y aislar uno del otro, logrando así que los objetos en memoria no necesiten conocer que hay una base de datos presente. Por lo tanto la separación entre los objetos del dominio y la fuente de datos es la función principal de este patrón.

2.3.5 Patrón Data Access Object (DAO)

Debido a que las aplicaciones cada vez más deben hacer persistentes los datos, estás están utilizando con mayor frecuencia sistemas manejadores de bases de datos relacionales, orientadas objetos, sistemas de directorios como LDAP (Lightweight Directory Protocol), archivos planos, y otros tales como: servicios externos, donde cada uno de ellos tiene un mecanismo propio de acceso [3].

Al tener tal variedad de fuentes de datos para permitir la persistencia y la necesidad de romper la dependencia de código para acceder a las fuentes de datos persistentes se hace necesario, implementar mecanismos de accesos y manipulación de datos que desacoplen la aplicación, del sistema de almacenamiento persistente, que provea una forma de acceso único a los datos independiente del tipo de almacenamiento.

Para lograr esto el Core J2EE tiene el patrón Data Access Object (DAO) que se usa para abstraer y encapsular todo el acceso al almacenamiento persistente. El DAO gestiona las conexiones con las fuentes de datos para obtener y almacenar los datos. El DAO implementa el mecanismo de acceso requerido para los datos, sin importar la fuente de datos y manteniendo un API (Application Programming Interface) única al cliente, lo que permite exponer interfaces que no cambian así cambie la capa de las fuentes de datos.

El DAO es implementado sin estados, lo que evita hacer almacenamiento temporal de cualquier ejecución de consultas y cualquier dato que el cliente requiera en situaciones posteriores, haciendo esto que los DAO sean objetos ligeros y evite problemas en el manejo de hilos y concurrencia.

2.3.6 Patrón Domain Store

Algunos sistemas tienen modelos de objetos complicados que requieren de estrategias de persistencia sofisticadas. Con la adición del Contenedor-Gestionador de relaciones (container-managed relationships – CRM) al contenedor EJB 2.x de manejo de persistencia (CMP), se hace mucho más factible emplear el CMP como una estrategia de persistencia para modelos complejos. Sin embargo, algunos desarrolladores evitan el uso de los *entity beans* u optan por ejecutar una aplicación en un contenedor web y, prefieren separar la persistencia del modelo de objetos [4].

Con este patrón se persigue evitar colocar los detalles de la persistencia en los objetos de negocio, evitar el uso de *entity beans*, y hacer persistencia en modelos de objetos que tiene herencia y relaciones complejas.

La implementación del *Domain Store* puede hacerse de dos formas: Escribiendo un framework persistencia propio ó usando un producto de persistencia. Las siguientes son las opciones de persistencia en J2EE:

1. Leer y escribir los datos directamente en los recursos de datos por cada operación CRUD (Create, Read, Update, Delete) – *Transaction Script* de Martin Fowler.
2. Usar *entity beans* del Container Manager Persistente (CMP).
3. Usar *entity beans* del Bean Manager Persistence (BMP).
4. Código persistente en el modelo de objeto *Plain Old Java Object* (POJO) – *Active Record* de Martin Fowler.
5. Separar la persistencia del modelo de objeto POJO.

3. Tecnologías Web

Son aquellas que hacen posible la distribución de información basada en hipertexto o hipermedias enlazados y accesibles a través de Internet que pueden contener textos, datos, imágenes y videos. Se pueden concentrar en dos grupos, tecnologías del lado del cliente y tecnologías del lado del servidor, a continuación se describen.

3.1 Tecnologías del lado del Cliente

Las tecnologías del lado de cliente en una aplicación web son aquellas que se ejecutan en el navegador web y generalmente están asociadas con la interfaz de usuario.

Una de las organizaciones que trabajan en la estandarización de estas tecnologías es el W3C (World Wide Web Consortium) [3] cuya misión es guiar la web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la web.

El W3C hace referencia a este objetivo como "interoperabilidad web". Al publicar estándares abiertos (no propietarios) para lenguajes web y protocolos, a continuación se describen algunos de ellos.

3.1.1 Lenguajes de Marcado

Con el desarrollo de los programas que procesan texto (editores y procesadores que agilizan el proceso de edición) surgen los primeros lenguajes informáticos especializados en tareas de descripción y

estructuración de información: los lenguajes de marcas. Paralelamente, también, surgen orientados a la representación, almacenamiento y consulta eficiente de grandes cantidades de datos: lenguajes y sistemas de bases de datos [2].

Los lenguajes de marcas surgieron, por el conjunto de códigos de formato que los procesadores de texto introducen en los documentos para dirigir el proceso de presentación. Como en el caso de los lenguajes de programación, inicialmente estos códigos de formato estaban ligados a las características de una máquina, programa o procesador de textos concreto y, en ellos, no había nada que permitiese al programador (formateador de documentos en este caso) abstraerse de las características del procesador de textos y expresar de forma independiente la estructura y la lógica interna del documento. En los años 70 cambia esta situación cuando Goldfarb y su equipo, trabajando en IBM, introducen el concepto de *marcado descriptivo* que, en la década de los 80, alcanza la categoría de estándar con la definición del lenguaje SGML (ISO/IEC IS 8879 Standard Generalized Markup Language). SGML introduce tres conceptos básicos:

- El concepto de lenguaje de marcado generalizado como un metalenguaje que sirve para definir lenguajes concretos que pueden adaptarse a cada dominio mediante una gramática que describe formalmente un tipo específico de documento o DTD (Document Type Definition);
- El concepto de marcado descriptivo describe, mediante las marcas o etiquetas definidas en la DTD, la estructura lógica de la información. La idea clave es que las marcas no determinan el procesamiento del documento de manera fija, ya que dicho procesamiento se determina a partir de las necesidades concretas, y se beneficia de la estructura lógica del documento caracterizada a través de sus marcas;
- El concepto de independencia de la plataforma. Como los documentos SGML únicamente contienen texto, éstos pueden ser procesados en distintas plataformas, trascendiendo el uso de dichos documentos a los sistemas que los crearon y utilizaron originariamente.

El paradigma documental es especialmente apropiado para la construcción de aplicaciones que, como las hipermedia, manejan cantidades apreciables de contenidos estructurados. A continuación de hace una breve descripción de los lenguajes de marcado para hipermedia más utilizados.

3.1.1.1 HTML

Siglas de (HyperText Markup Language) Lenguaje de Marcas de Hipertexto, [2] es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", (< >). La estructura de un documento HTML se muestra a continuación:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 <html lang="es">
 <head>
 <title>Ejemplo</title>
 </head>
 <body>
 <p>ejemplo “Hola Mundo” </p>
 </body>
 </html>
```

3.1.1.2 XML

XML, siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), [2] es un metalenguaje extensible de etiquetas que permite definir la gramática de lenguajes específicos. Por lo

tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, entre otros.

XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible.

Una etiqueta de XML consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma **<nombre>**, donde nombre es el nombre del elemento que se está señalando. A continuación se muestra un ejemplo de un documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Lista_datos_mensaje.dtd"
[<!ELEMENT Edit_Mensaje (Mensaje)*>]>
 <Ejemplo>
 <Mensaje>
 <Remitente>
  <Mail> Correo del remitente </Mail>
 </Remitente>
 <Destinatario>
  <Mail>Correo del destinatario</Mail>
 </Destinatario>
 <Texto>
  <Asunto>
  Este es un xml con una estructura muy sencilla
  </Asunto>
  <Parrafo>
           Este es un xml con una estructura muy sencilla
           </Parrafo>
 </Texto>
 </Mensaje>
 </Ejemplo>
```

La estructura de un archivo XML es definida formalmente por un documento DTD o XSD en los cuales se especifica que etiquetas van dentro de otras, atributos, tipos, etc. A continuación el código del DTD del documento "Ejemplo":

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Este es el DTD de Ejemplo -->

<!ELEMENT Mensaje (Remitente, Destinatario, Texto)*>
 <!ELEMENT Remitente (Nombre, Mail)>
 <!ELEMENT Mail (#PCDATA)>

<!ELEMENT Destinatario (Nombre, Mail)>
 <!ELEMENT Mail (#PCDATA)>

<!ELEMENT Texto (Asunto, Parrafo)>
 <!ELEMENT Asunto (#PCDATA)>
 <!ELEMENT Parrafo(#PCDATA)>
```

Uno de los principales usos de XML es estructurar datos, recibirlos o enviarlos, pero también es posible guardar datos en documentos para que sean tratados luego con otro lenguaje.

Las Transformaciones XSL (XSLT) presentan una forma de transformar documentos XML en otros e incluso a formatos que no son XML. XSLT realiza la transformación del documento utilizando una o varias reglas de plantilla. La unión de XML y XSLT permite separar contenido y presentación, aumentando así la productividad.

Otro uso es la redifusión de noticias: ATOM/RSS/RDF son formatos que tienen como base, documentos XML, en este caso es un servicio que permite obtener información de un documento XML generado automáticamente por un sistema de publicación.

Se dice que un documento XML está bien formado cuando este cumple las reglas de sintaxis definidas por la W3C: una sola etiqueta raíz, etiquetas bien anidadas, valor de atributos entre comillas, entre otros. Se dice que un documento XML es válido cuando aparte de ser bien formado cumple la estructura definida por un DTD o XSD.

3.1.1.3 XHTML

XHTML, acrónimo en inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto) [6], está pensado para sustituir a HTML como estándar para las páginas web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.

La siguiente lista muestra algunas reglas de XHTML 1.0 que lo diferencian de HTML 4.01. Muchas de estas diferencias vienen por ajustarse a las condiciones de un documento XML bien formado:

- Los elementos vacíos y no vacíos deben cerrarse siempre
- Los elementos anidados deben tener un correcto orden de apertura
- Los valores de los atributos deben siempre ir encerrados entre comillas
- Los nombres de elementos y atributos deben ir en minúsculas
- No está permitida la minimización de atributos
- Los atributos desaprobados en HTML 4.01 no forman parte de XHTML.

3.1.1.4 HTML 5.0

HTML 5.0 (HyperText Markup Language, versión 5) [2] es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML (XHTML) (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.
HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada, como los elementos <audio> y <video>. Mejoras en el elemento <canvas>, capaz de producir en algunos navegadores elementos 3D.
Algunos elementos de HTML 4.01 han quedado obsoletos, incluyendo elementos puramente de presentación, como <font> y , cuyos efectos son manejados por el CSS. También hay un renovado énfasis en la importancia del scripting DOM (Document Object Model) para el comportamiento de la web.

3.1.2 CSS

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) [2] son un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo “style”.

3.1.2.1 Diagramado de página en CSS

Antes de las CSS, una forma de componer espacialmente una página era mediante tablas. Aunque es una técnica cómoda y versátil, se está usando un elemento con una semántica particular, que es la de expresar información tabular, solamente por su efecto en la presentación. Para definir la situación de los elementos en una página Web, se utilizan las capas, que permiten agrupar o dividir el documento en partes lógicas (cabecera, menú, contenido, pie, entre otros) a las que se aplica un posicionamiento a través de las hojas de estilo. (Ver Figura 6).

Para crear las capas se utilizan etiquetas <DIV>, en las que se introducen los elementos que deben aparecer en la página. Los elementos dentro de los <DIV> se pueden anidar, para heredar las propiedades y posicionamiento de las capas padre organizando el contenido de una manera más natural.

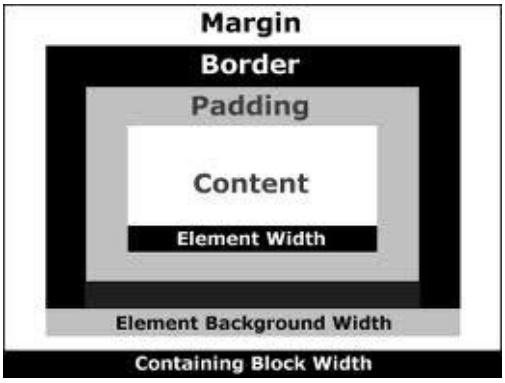

Figura 6: Box Model [3])

Entre las ventajas del diagramado con CSS se encuentran:

- La separación del contenido de la página y del estilo o aspecto con el que se deben mostrar.
- Ahorro en la transferencia
- Facilidad para alterar el aspecto de la página sin tocar el código HTML.

Y como desventajas del diagramado con CSS se encuentran:

- Incompatibilidad con navegadores antiguos
- Diferencias entre navegadores
- Requiere de un entrenamiento mayor.

3.1.3 Lenguajes de Scripting

A continuación se presenta algunos lenguajes de scripting del lado del cliente.

3.1.3.1 JavaScript

Es un lenguaje de programación interpretado, es decir, que no requiere compilación, [2] muy utilizado en páginas Web, con una sintaxis semejante a la de Java y C. Se ejecuta en el cliente al mismo tiempo que las sentencias HTML. En junio de 1997 fue adoptado como un estándar ECMA (European Computer Manufacturers' Association), con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

El núcleo de JavaScript contiene un conjunto central de objetos, tales como Array (arreglos), Date (fechas) y Math (objetos matemáticos), además de un conjunto central de elementos del lenguaje tales como los operadores, estructuras de control y sentencias. El núcleo de JavaScript puede ser extendido para una variedad de propósitos complementándolo con objetos adicionales; por ejemplo:

JavaScript del lado Cliente

Extiende el núcleo del lenguaje proporcionando objetos para el control del navegador (Navigator o cualquier Web browser) y su Modelo Objeto Documento [Document Object Model] (DOM). Por ejemplo, las extensiones del lado del cliente permiten a una aplicación ubicar elementos en un formulario HTML y responder a los eventos de usuario tales como los clics del ratón, entradas del formulario y navegación de páginas.

El Modelo Objeto del Documento es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo sobre cómo pueden combinarse dichos objetos, y una interfaz para acceder a ellos y manipularlos.

El DOM define la manera en que objetos y elementos se relacionan entre sí en el navegador y en el documento. Cualquier lenguaje de programación adecuado para el desarrollo de aplicaciones Web puede ser utilizado. En el caso de JavaScript, cada objeto tiene un nombre, el cual es exclusivo y único. Cuando existe más de un objeto del mismo tipo en un documento Web, estos se organizan en un arreglo.

El DOM está basado en una estructura de objeto muy parecida a la estructura del documento que modela. Por ejemplo, considere esta tabla, tomada de un documento XHTML:

```
<table>
 <tbody>
 <tr>
 <td>Shady Grove</td>
 <td>Aeolian</td>
 </tr>
 <tr>
 <td>Sobre el río, Charlie</td>
 <td>Dorian</td>
 </tr>
 </tbody>
</table>
```

A continuación en la Figura 7 una representación gráfica DOM de la tabla del ejemplo anterior.

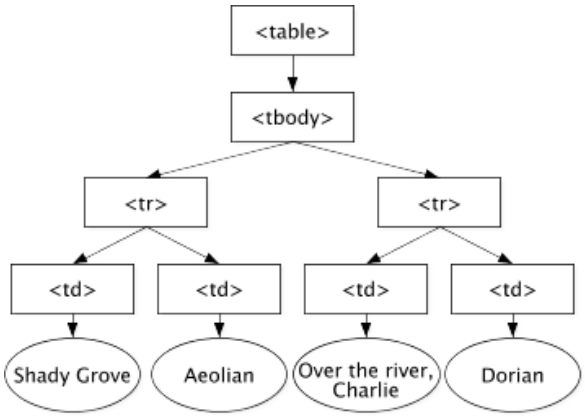

Figura 7: Representación DOM [3]

JavaScript del lado Servidor

Extiende el núcleo del lenguaje proporcionando objetos relevantes para la ejecución de JavaScript en un servidor. Por ejemplo, las extensiones del lado del servidor permiten que una aplicación se comunique con una base de datos relacional, proporcionar continuidad de la información desde una invocación de la aplicación a otra o efectuar la manipulación de archivos en un servidor (Ejemplos: Server Side, JavaScript, JavaScript Server Page, node.js, Wakanda y Rhino, entre otros).

3.1.4 Aplicaciones de Internet Enriquecidas

Las RIA (Rich Internet Applications) son un tipo de aplicaciones desarrolladas con estándares de la Web, pero con interactividad más avanzada tal como las que se ven en las aplicaciones de escritorio.

Una de las desventajas de las tradicionales aplicaciones Web es la poca interactividad que posee. Las RIAs proveen a los usuarios actualizaciones de estados sin necesidad que este lo solicite, ofreciéndole también un conjunto de servicios, alternativas o sugerencias según el próposito del sitio.

Hay muchas herramientas para la creación de entornos RIAs. Entre éstas se pueden mencionar las plataformas Adobe Flash, Adobe Flex y Adobe AIR de Adobe, uniPaaS de Magic Software, la técnica AJAX, OpenLaszlo, Silverlight de Microsoft, Java Script, Bindows de MB Technologies y JavaScript. A continuación se describen brevemente Ajax, JQuery y CoffeScript.

3.1.4.1 AJAX

Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), [2] es una técnica de desarrollo Web para crear aplicaciones interactivas o RIA. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas completamente, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de AJAX mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales (ver Figura 8). AJAX es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones como JavaScript.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor Web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

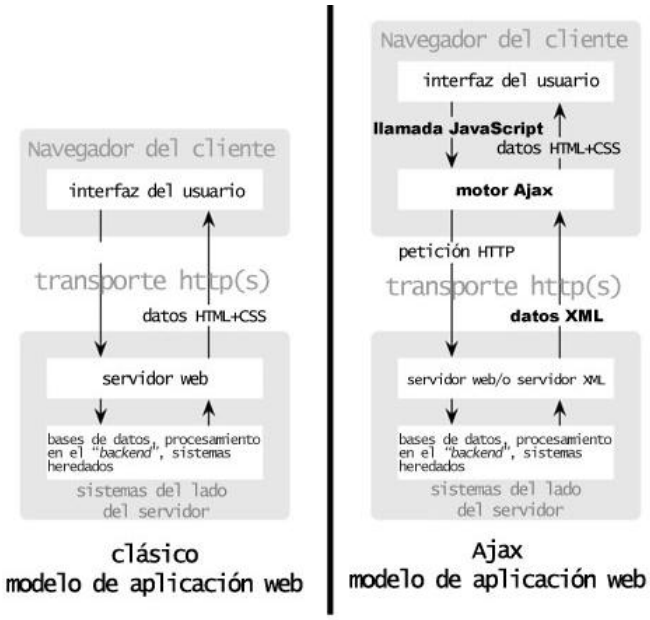

Figura 8: Funcionamiento Ajax

3.1.4.2 JQuery

Jquery creado inicialmente por John Resig, permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas Web. JQuery aporta en colocar código no intrusivo JavaScript en las páginas. Su diseño permite especificar el comportamiento e interacción de los elementos del HTML, separado de las etiquetas valiéndose del DOM.

3.1.4.3 CoffeeScript

CoffeeScript [1] es un lenguaje que se compila en JavaScript. JavaScript tiene un buen modelo de objetos en su núcleo, pero suele ser engorroso por sus llaves ({}) y puntos y comas (;). CoffeeScript es un intento de exponer las características buenas de JavaScript de una manera sencilla evitando la sintaxis engorrosa.

La regla de oro de CoffeeScript es: "Es simplemente JavaScript". El código CoffieScript se traduce línea a línea en un código JavaScript equivalente, y no hay una interpretación en tiempo de ejecución. Se puede usar cualquier librería JavaScript sin problemas. El resultado compilado es legible y presentable, pasa sin alertas a través de JavaScript Lint, y funciona en las distintas implementaciones de JavaScript. CoffeeScript tiende a correr tan rápido o más rápido que el código JavaScript escrito a mano equivalente (ver Figura 9).

```coffeescript
# Assignment:
number   = 42
opposite = true

# Conditions:
number = -42 if opposite

# Functions:
square = (x) -> x * x

# Arrays:
list = [1, 2, 3, 4, 5]

# Objects:
math =
  root:   Math.sqrt
  square: square
  cube:   (x) -> x * square x

# Splats:
race = (winner, runners...) ->
  print winner, runners

# Existence:
alert "I knew it!" if elvis?

# Array comprehensions:
cubes = (math.cube num for num in list)
```

```javascript
var cubes, list, math, num, number, opposite, race, square;
var __slice = Array.prototype.slice;
number = 42;
opposite = true;
if (opposite) number = -42;
square = function(x) {
  return x * x;
};
list = [1, 2, 3, 4, 5];
math = {
  root: Math.sqrt,
  square: square,
  cube: function(x) {
    return x * square(x);
  }
};
race = function() {
  var runners, winner;
  winner = arguments[0], runners = 2 <= arguments.length ?
__slice.call(arguments, 1) : [];
  return print(winner, runners);
};
if (typeof elvis !== "undefined" && elvis !== null) alert("I
knew it!");
cubes = (function() {
  var _i, _len, _results;
  _results = [];
  for (_i = 0, _len = list.length; _i < _len; _i++) {
    num = list[_i];
    _results.push(math.cube(num));
  }
  return _results;
})();
```

run: cubes

Figura 9: Ejemplo CoffeeScript (Lado izquierdo CoffeeScript; Lado derecho Javascript generado) [1]

3.2 Tecnologías del lado del Servidor

Una página Web, es un documento adaptado para la Web y normalmente forma parte de una aplicación Web. Su principal característica son los hiperenlaces a otras páginas, siendo esto el fundamento de la WWW. Una página Web está compuesta por información texto o multimedia e hiperenlaces; además puede contener o asociar datos de estilo para especificar cómo debe visualizarse, o aplicaciones embebidas para hacerla interactiva.

El contenido de la página puede ser predeterminado "página web estática" (ver Figura 10), en cuyo caso se refiere a la solicitud de un recurso y éste es devuelto por el servidor HTTP. El recurso puede ser un documento HTML, CSS, JS, imagen o cualquier otro que el usuario requiera y pueda obtener a través del cliente.

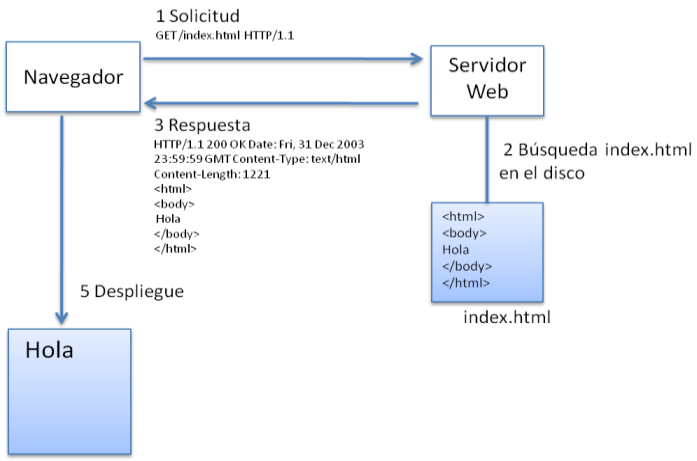

Figura 10: Página Estática

También el contenido puede ser generado al momento de solicitar a un servidor web lo cual se conoce como "página web dinámica"(ver Figura 11), es indica que es dinámica si el contenido HTML, JS, CSS es generado en tiempo de ejecución, y por lo general es personalizado para ese usuario. Ejemplo de ello es el cliente web de correo (Webmail), en el cual el usuario ve los correos que le pertenecen. Para que una página pueda ser dinámica se necesita de algunas de las tecnologías del lado del servidor.

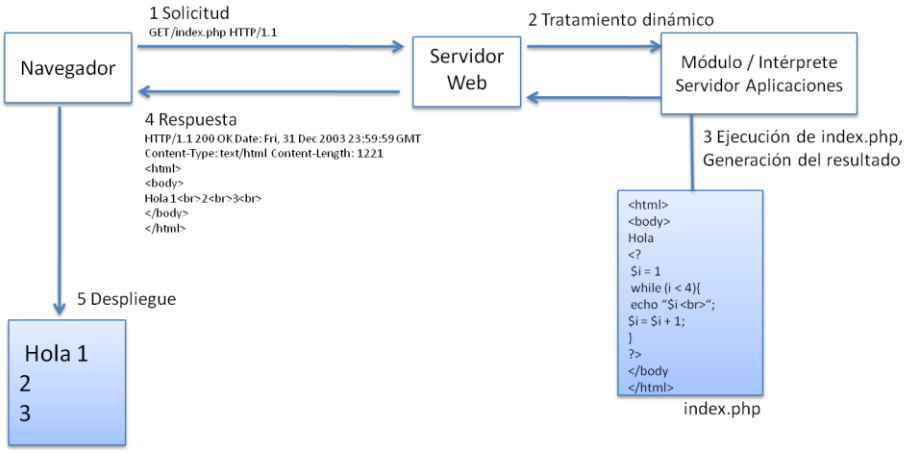

Figura 11: Página Dinámica

3.2.1 Taxonomía de Aplicaciones Web según Enfoque Tecnológico.

No es ni conveniente ni práctico para diseñar y desarrollar cada nueva aplicación web comenzar desde cero. Se tendría que seguir construyendo las mismos componentes funcionales que acepten e interpreten los datos, autentiquen y autoricen a los usuarios, accedan y transformen los datos y construyan y transmitan las respuestas finales. Muchos de estos componentes serán idénticos en diferentes aplicaciones. Los servidores web ofrecen a los desarrolladores de aplicaciones funcionalidades bien definidas para aceptar las peticiones y la transmisión de las respuestas. Algunos servidores ofrecen las bases para la construcción y desarrollo de aplicaciones web. En la práctica estas bases son muy básicas para facilitar el desarrollo e implementación de aplicaciones sofisticadas.

En esta sección se clasifican y analizan varias estrategias para la construcción de aplicaciones web, distinguiendo entre enfoques y frameworks [11]. Un enfoque (approach) se define como una librería de componentes funcionales (funcionalidades) que toman las bondades de los servidores web que pueden ser reutilizadas en múltiples y diferentes aplicaciones; éstas normalmente están íntimamente ligadas a los lenguajes de programación, con APIs complementarias y paquetes que proveen las funcionalidades necesarias de las aplicaciones web.

Los frameworks van más allá de los enfoques en el desarrollo de aplicaciones web. Ellos proveen una infraestructura consistente que incluye un conjunto de servicios, eliminando la necesidad de escribir código redundante para funciones comunes de las aplicaciones. Normalmente incluye soporte integrado para acceder a Bases de Datos, autenticación, manejo de estados y sesiones. Y más importante aún, los frameworks promueven y facilitan la separación del contenido de la presentación, lo cual es un objetivo de las buenas prácticas en el desarrollo de aplicaciones web. Se propone lograr este objetivo al hacer responsable a los programadores por la lógica de negocio y el acceso al contenido, mientras el diseñador trabaja en la presentación de la información y el formato de las páginas. El mejor framework es aquel que permite al desarrollador y al diseñador trabajar en módulos separados, tal que el trabajo de ellos no interfiriera con el trabajo del otro. Las soluciones de desarrollo de aplicaciones web se dividen en cuatro grandes categorías:

- Enfoques en programación o scripting
- Enfoques en plantillas
- Enfoques híbridos
- Frameworks

3.2.1.1 Enfoques en programación o scripting

En este enfoque, el código asociado a las páginas consiste principalmente de código escrito en lenguajes de script tales como Perl, Python, etc. o en lenguajes de propósito general como Java. El código puede ser intercalado con las definiciones del formato del contenido y de la presentación. Naturalmente este enfoque está dirigido hacia los programadores. La mayoría del código de la página implementa la lógica de la aplicación, mientras que el formato del contenido y de la presentación es normalmente generado por las instrucciones de salida de datos del lenguaje de programación. Entre los enfoques que entran en esta categoría están los CGI y los Servlets.

El mayor de los temas en el enfoque de programación para el desarrollo de aplicaciones web está en su naturaleza de código centralizado. HTML y otros métodos de presentación de contenidos y datos están embebidos con la lógica del programa. Esto limita la creatividad que el diseñador gráfico pueda tener en la construcción de la página. El diseñador gráfico puede generar una maqueta, pero el programador debe traducir esta al lenguaje correspondiente e integrarlo con el código de la lógica del programa. La intervención del programador es necesaria para hacer casi cualquier modificación del aspecto de la página, o de la lógica del programa. A continuación se define brevemente algunas tecnologías que ejemplifican este enfoque.

A. CGI

Common Gateway Interface (CGI) específica según la W3C [2] un estándar para transferir datos entre el cliente y un programa que está en el servidor. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME (Multipurpose Internet Mail Extensions[1]). Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

Las aplicaciones CGI fueron una de las primeras maneras prácticas de crear contenido dinámico para las páginas web. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar escrito en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La salida de estos programas es enviada al cliente en lugar del archivo estático tradicional.

CGI es un tipo de tecnología donde se establece un mecanismo de llamada a través del servidor web que invoca a programas o scripts hechos en Perl, C++, entre otros. A continuación se presenta un ejemplo usando Perl:

```
#!/usr/local/bin/perl
print "Content-type: text/html\n\n";
print "<HTML>";
print "<HEAD>";
print "<TITLE> Hola Mundo en CGI </TITLE>";
print "</HEAD>";
print "<BODY>";
print "Hola Mundo";
print "</BODY>";
print "</HTML>";
```

[1] Consiste en una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario.

B. Servlets

Un servlet [10] es un objeto que se ejecuta en un servidor o contenedor JEE (Java Enterprise Edition), especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente HTML.

Un servlet implementa la interfaz `javax.servlet.Servlet` o hereda alguna de las clases más convenientes para un protocolo específico (ej: `javax.servlet.HttpServlet`). Al implementar esta interfaz el servlet es capaz de interpretar los objetos de tipo `HttpServletRequest` y `HttpServletResponse` quienes contienen la información de la página que invocó al servlet. Entre el servidor de aplicaciones y el servlet existe un estándar que determina cómo han de interactuar según su ciclo de vida. El ciclo de vida del Servlet se divide en los siguientes puntos:

1. El cliente solicita una petición a un servidor vía URL.
2. El servidor recibe la petición.
1. Si es la primera, se utiliza el motor de Servlets para cargarlo y se llama al método `init()`.
2. Si ya está iniciado, cualquier petición se convierte en un nuevo hilo. Un Servlet puede manejar múltiples peticiones de clientes.
3. Se llama al método `service()` para procesar la petición devolviendo el resultado al cliente.
4. Cuando se detiene el motor de un Servlet se llama al método `destroy()`, que lo destruye y libera los recursos abiertos.

A continuación un ejemplo de servlet que genera una página web dinámica Hola Mundo

```
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.*;
public class HelloServlet extends HttpServlet {
 public void doGet (HttpServletRequest req,
   HttpServletResponse res)
 throws ServletException, IOException
 {
 PrintWriter out = res.getWriter();
 out.println("<HTML>");
 out.println("<HEAD>");
 out.println("<TITLE> Hola Mundo en Servlet </TITLE>");
 out.println("</HEAD>");
 out.println("<BODY>");
 out.println("Hola Mundo");
 out.println("</BODY>");
 out.println("</HTML>");
 out.close();
 }
}
```

En estas tecnologías tanto CGI como Servlet se mezclan la lógica de control con la de presentación, por lo que su uso se ha reducido a la aplicación en problemas muy puntuales y se combinan con otras tecnologías.

Con los CGI el proceso de arranque de cada proceso domina el tiempo de ejecución, en cambio con los servlets el uso de hilos hace que el tiempo de ejecución dependa directamente de la ejecución del proceso. Adicionalmente para los CGI N peticiones simultáneas implican N cargas en memoria del CGI, lo que no ocurre con los servlets.

Para los servlets es sencillo mantener datos entre peticiones simplificando el seguimiento de sesiones y operaciones de caché, permitiendo además compartir datos entre diferentes servlets, a diferencia de los CGI que requieren de soluciones de programación externas al propio CGI para realizar estás acciones.

3.2.1.2 Enfoques basados en plantillas

Este utiliza objetos de código que consisten predominantemente en estructuras de presentación de contenido y datos, con un limitado conjunto de estructuras y código que agregan la capacidad de programación. El objetivo de estos objetos de código es la presentación del contenido y de los datos y no la lógica del programa. Este enfoque está dirigido a los diseñadores gráficos cuya experiencia está en el área de diagramación de la página más que en la programación.

Este enfoque se centra en la estructura de la página, la presentación del contenido y no en el código. Con esto en mente a este enfoque se le llama centrado en la página. Esto objetos de códigos son las plantillas, que consisten en HTML, acoplado con estructuras y código que soportan el procesamiento condicional y la presentación iterativa de resultados.

Server-Site Include (SSI) es un mecanismo para agregar fácilmente plantillas a las páginas web. Entre las plantillas mejor conocidas se encuentran el Cold Fusion y Velocity.

3.2.1.3 Enfoques híbridos

Este enfoque emplea objetos de código que combinan las estructuras y códigos de presentación y de lógica del programa. Se tiene mayor énfasis en la lógica de programación que en las plantillas. Estas tecnologías combinan en general código del lado del cliente (HTML / JS / CSS) con trozos de código embebido que se ejecutan del lado del servidor. El objetivo es retornar un código producto de la combinación y ejecución de los trozos dinámicos con los estáticos.

Existe un procesador o intérprete que analiza el código combinado en el servidor, para ejecutar la característica dinámica de la página. La tecnología establece delimitadores para indicar que se está escribiendo código a ejecutar del lado del servidor. A continuación se describen brevemente algunas de las tecnologías basadas en lenguaje de scripting embebido.

A. Hypertext Pre-Processor (PHP)

PHP [8] es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Además puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas.

B. Java Server Page (JSP)

JSP [10] utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Éstas pueden ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas externas e incluso personalizadas. El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se retorna al navegador o cliente. En la Tabla 2 se puede observar un ejemplo de la codificación de los lenguajes de scripting embebidos.

PHP	JSP
<HTML> <HEAD> <TITLE> Hola Mundo en PHP </TITLE> </HEAD> <BODY> **<?** **print("Hola mundo");** **?>** </BODY> </HTML>	<HTML> <HEAD> <TITLE> Hola Mundo en JSP </TITLE> </HEAD> <BODY> **<%** **out.println("Hola Mundo");** **%>** </BODY> </HTML>

Tabla 2: Lenguajes de Scripting Embebidos

Las ventajas que permiten este tipo de tecnologías es que de una forma simple y rápida se pueden crear páginas dinámicas. Sin embargo estas tecnologías limitan el código del lado del servidor pero no separan estrictamente la lógica de negocio con la de presentación, además en aplicaciones con muchas funcionalidades se dificulta su mantenimiento, como alternativa a esta problemática se recomienda el uso de frameworks web, los cuales se describen a continuación.

3.2.1.4 Frameworks Web

Los frameworks web representan el próximo nivel de sofisticación en el desarrollo de aplicaciones web. En lugar de combinar la presentación con la lógica de negocio en un solo módulo, estos siguen el principio de separar del contenido de la presentación. Los módulos responsables de la generación del contenido (el Modelo) son distintos de los módulos que presentan el contenido en formato definido (las Vistas).

Los frameworks permiten trabajar en una aplicación web a partir de una estructura, ofreciendo un conjunto de facilidades como librerías, funciones, clases, estructura de directorios y estándares definidos alrededor de cómo implementar la funcionalidad en una aplicación web.

En general los frameworks para desarrollo web están basados en el patrón de diseño arquitectónico MVC el cual fue descrito en la sección 2.2. A continuación se presenta la Figura 12 que ilustra el patrón MVC en entornos web.

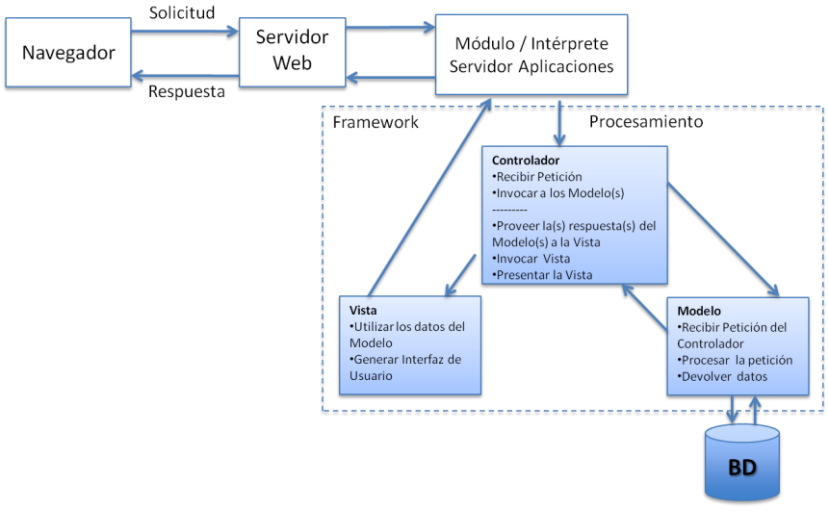

Figura 12: Framework Web MVC

Entre los frameworks más conocidos se encuentra Ruby on Rails que se describe brevemente a continuación:

A. Ruby on Rails (RoR)

RoR es [15] un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo-Vista-Controlador. Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

La ventaja de trabajar en un framework es que las tareas comunes están implementadas o se hacen con menos esfuerzo y además, propone un estándar de programación lo que permite un mejor mantenimiento y reusabilidad. También poseen implementaciones de patrones, por ejemplo el patrón Modelo-Vista-Controlador (MVC), y el patrón Active Record.

Los frameworks proveen una estructura básica para los desarrollos, lineamientos que contribuyen en la organización de los componentes del software. Algunos directorios que lo componen corresponden a aplicación, librerías, bitácoras, configuración, complementos y pruebas los cuales varían según el producto.

3.2 Servicios Web

Un servicio web (Web Service) [4] es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de computadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios web se ha creado el organismo WS-I (Web Services Interoperability Organization), encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Entre los más utilizados se encuentran los que se describen a continuación:

a) REST (Representational State Transfer): estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la World Wide Web. En resumen, es un conjunto de principios para el diseño de redes, que es utilizado comúnmente para definir una interfaz de transmisión sobre HTTP de manera análoga a como lo hace SOAP. Aunque REST como tal no es un estándar, posee un conjunto de estándares tales como HTML, URL, XML, GIF, JPG y tipos MIME. Los principios de REST son:

- Escalabilidad de la interoperabilidad con los componentes.
- Generalidad de Interfaces.
- Puesta en funcionamiento independiente.
- Compatibilidad con componentes intermedios.

b) RPC (Remote Procedure Calls): Es una tecnología de software que permite ejecutar una rutina en un equipo o segmento de red de manera remota. Es un paradigma popular para la implementación de sistemas distribuidos bajo arquitecturas cliente servidor.

c) XML-RPC: Es un protocolo de llamada remota que utiliza XML como lenguaje de codificación y HTTP como mecanismo de transporte. Es un protocolo sencillo ya que solo define algunos tipos de datos y comandos. Existen implementaciones de XML-RPC específicas para ActionScript, Delphi, C++, .NET, OClam, Common LISP, PHP y otros.

d) XML (eXtended Markup Language): XML es uno de los lenguajes más utilizados para el intercambio de datos sobre la web. Un documento XML es un objeto de datos que está bien formado, y se dice que lo está cuando tomado en su conjunto coincide con la producción del documento etiquetado, reúne todas las especificaciones de formato definidas y cada una de las entidades que se llaman directa o indirectamente están también bien definidas. El XML es un lenguaje etiquetado, característica que le permite definir objetos de datos estructurados en partes bien definidas llamadas elementos. Una etiqueta es una señal realizada dentro del documento XML que delimita un segmento definido y con sentido de este documento.

e) SOAP (Simple Object Access Protocol): es un protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes de computadores. Básicamente es una vía de transmisión entre un SOAP Sender y un SOAP Receiver, pero los mensajes SOAP deben interactuar con un conjunto de aplicaciones para que se pueda generar un “diálogo” a través de mensajes SOAP. Un mensaje SOAP es la unidad fundamental de una comunicación entre nodos SOAP. SOAP es básicamente un paradigma de una sola vía pero con la ayuda de las aplicaciones se puede llegar a crear patrones más complejos. SOAP básicamente está constituido por:

- Un marco que describe el contenido del mensaje e instrucciones de proceso.
- Un conjunto de reglas para representar los tipos de datos definidos.
- Convenciones para representar llamadas a procedimientos remotos y respuestas.
- Y algunos lineamientos entre SOAP y HTTP.

Referencias

[1] CoffeeScript. CoffeeScript. Octubre 2011. http://jashkenas.github.com/coffee-script/.

[2] Consortium W.W.W W3C. Mayo 2011. http://www.w3c.es/.

[3] D. Alur, J. Crupi, D. Malks. Core J2EE PAtterns. Best Practice and Design Strategies USA. Pretince Hall PTR. Mayo 2003.

[4] D. Barry. Web Services and Service Oriented Architectures. Morgan Kaufmann Publishers.Abril 2003.

[5] E. Scalise Diseño de Aplicaciones Empresariales Multiformato utilizando Estándares y Tecnología Web. 2003.

[6] F. Miller, A.Vandome, J. McBrewster Markup Language: Annotation, Troff, LaTeX, XML, Presentation Semantics, Hypertext, HTML, World Wide Web, Standard Generalized Markup Language, IBM Generalized Markup Language, XHTML, Lightweight Markup Language. Alphascript Publishing. Junio 2010.

[7] G. Kappel. Web Engineering: The Discipline of Systematic Development of Web Applications. USA John Wiley. Julio 2006.

[8] Group The PHP PHP.net . Octubre de 2011. http://www.php.net/.

[9] Guerrero L. A. Modelando Interfaces para Aplicaciones Web. Marzo 2007.

[10] JavaServer Technologies. Octubre de 2011 http://java.sun.com/developer/technicalArticles/.

[11] L. Shklar, R. Rosen Web Application Architecture: Principles, Protocols and Practices. Junio 2009.

[12] M. Escalona. Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software.Octubre 2004.

[13] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford. Patterns of Enterprise Application Architecture. Addison Wesley. Febrero 2002.

[14] R. Pressman. Ingeniería de Software Un enfoque Sistemático. Mayo 2007.

[15] Ruby on Rails. Octubre de 2011. http:// www.rubyonrails.org.

[16] S. Ceri Applications Designing Data-Intensive Web. IET Software. Febrero 2007.

[17] S. Melia Using MDA in Web Software Architectures. Septiembre 2008.