Capítulo 1. Marco teórico

1.1 Sistemas de información

De acuerdo a la definición de Samuelson[2] un sistema de información es la combinación de recursos humanos y materiales que resultan de las operaciones de almacenar, recuperar y usar datos con el propósito de realizar una gestión eficiente en las operaciones de las organizaciones. En la figura 1 se esquematizan los componentes básicos que conforman un sistema.

Figura 1. Sistema de información.

1.1.1 Sistemas de Administración de Contenidos (CMS)

[Un Sistema de Administración de Contenidos, (CMS, Content Management System) es un sistema que permite crear una estructura de soporte para la creación y administración de contenidos.

Para el caso de un sistema administrador de contenidos de las páginas de un sitio web, un CMS consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio; a su vez permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la publicación en el sitio de manera sencilla y controlada.][3]

Un ejemplo de un CMS de páginas web es el siguiente: una interfaz que permita administrar los eventos y seminarios del sitio del Instituto de Astronomía X, en donde el usuario que funge como administrador tiene la posibilidad de crear eventos y/o seminarios de manera sencilla a través del llenado de formularios que solicitan los campos que conforman un evento o seminario. El CMS se encarga de darle forma al contenido por medio del uso de plantillas predeterminadas. También por medio del CMS se puede editar el contenido y eliminar eventos y seminarios.

1.2 Programación Orientada a Objetos (POO)

La POO se basa en la idea natural de la existencia de una realidad llena de objetos y que la resolución de los problemas se realiza en términos de objetos. De forma que la estructura de los datos pasa a ser el eje central de la POO junto con las relaciones entre ellos.

En la POO se definen los programas en términos de clases de objetos, objetos que son entidades que combinan estado, comportamiento e identidad. La POO expresa un programa como un conjunto de

[2] Samuelson, Kjell. *Information Systems and Networks*. Holanda, 1977.
[3] Basado en Wikipedia, http://es.wikipedia.org/wiki/Sistema_de_gestión_de_contenidos

estos objetos, que colaboran entre ellos para realizar tareas.

De esta forma, un objeto contiene todos los atributos que permiten definirlo e identificarlo frente a otros objetos pertenecientes a otras clases. A su vez, los objetos disponen de mecanismos de interacción, es decir, métodos que favorecen la comunicación entre objetos a través de mensajes. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan datos y procesamiento.

Las principales características de la POO son:

- Abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella se puede armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere afrontar.
- Encapsulamiento. Es el mecanismo básico para ocultar los detalles internos de un objeto de los demás objetos. El encapsulamiento permite hacer una distinción entre la interfaz y la implementación. La interfaz se refiere al aspecto externo de una clase, es decir, a los atributos y métodos que podrán ser usados por otras clases. Mientras que la implementación es todo el código, variables y métodos necesarios para el correcto funcionamiento de los procedimientos publicados en la interfaz.
- Polimorfismo. Se refiere a la capacidad de que un método se comporte de distinta forma de acuerdo con la clase que lo implementa. Mediante el polimorfismo se definen múltiples funciones con nombre e interfaces similares sólo que en distintas clases.
- Herencia. Permite crear una jerarquía de objetos en un programa. La herencia permite a una clase heredar el conjunto de métodos y atributos de una clase sin tener que reescribir el código de dicha clase.

Las ventajas de la orientación a objetos aplicada en la programación son las siguientes:

- Reutilización y extensión del código**.** Un programa orientado a objetos consta de módulos independientes, por lo que se pueden reutilizar en distintos programas e incluso extender el código, lo que se traduce en ahorro de tiempo de desarrollo.
- Mantenimiento**.** Se refiere a que un programa debe ser capaz de adaptarse a un medio que cambia constantemente. Un programa basado en la POO que está sujeto a cambios en las condiciones externas (como la definición de una nueva variable) implicará pocas modificaciones del programa.
- Creación de sistemas complejos**.** Permite crear sistemas más complejos capaces de relacionar un sistema con el mundo real, así como la generación de ambientes gráficos y construcción de prototipos con mayor rapidez y sencillez.

La POO también tiene algunos inconvenientes:

- Rendimiento**.** Los lenguajes que utilizan el paradigma de la POO tienen un rendimiento menor en comparación con otros lenguajes estructurados.
- Curvas de aprendizaje largas. Al hacer la transición hacia un paradigma orientado a objetos la mayoría de los programadores deben aprender a resolver problemas en términos de objetos.

- Determinación de las clases**.** Una clase es un molde que se utiliza para crear nuevos objetos. En consecuencia es importante crear el conjunto de clases adecuado para un proyecto. En caso de que las clases que se establecieron no sean posibles de desarrollar será necesario reestructurar la jerarquía de clases, cambiando de manera significativa la planificación original.

Los lenguajes de programación orientados a objetos que destacan actualmente son: ActionScript 3, C++, C#, Delphi, Scala, Java, Ruby y Smalltalk.

1.3 Plataforma Java EE

La tecnología Java es tanto un lenguaje de programación como una plataforma. El lenguaje de programación es un lenguaje de alto nivel orientado a objetos que tiene una sintaxis y estilo particular. Una plataforma Java es un ambiente particular en el cual las aplicaciones Java se ejecutan.

1.3.1 Plataformas de Java

Existen tres plataformas del lenguaje Java:

- Java SE[4]
- Java EE[5]
- Java ME[6]

Todas las plataformas se componen de:

- Una máquina virtual (VM[7]). Es un programa para una plataforma particular de hardware y software que ejecuta aplicaciones Java.
- Una API[8]. Es una colección de componentes de software que se pueden usar para crear otros componentes de software y aplicaciones.

Cada plataforma provee de una máquina virtual y una API, esto permite a las aplicaciones escritas para esa plataforma ejecutarse en cualquier sistema compatible con las ventajas del lenguaje Java: independencia de plataforma, estabilidad, facilidad de desarrollo y seguridad.

Java SE

Esta plataforma provee las funcionalidades base del lenguaje de programación Java. Ésta define desde los tipos de datos básicos y objetos del lenguaje hasta las clases de alto nivel que se usan para la seguridad, acceso a base de datos, redes, desarrollo de interfaces gráficas de usuario y análisis sintáctico de documentos en formato XML[9].

Java EE

Esta plataforma está construida sobre la plataforma Java SE. Ésta provee una API y un ambiente de desarrollo y ejecución para aplicaciones a gran escala, multicapa, escalables, confiables y distribuidas.

[4] Java Stardard Edition, Java Edición Estandar
[5] Java EnterPrise Edition, Java Edición Empresarial
[6] Java Micro Edition, Java Edición para Móviles
[7] Java Virtual Machine, Máquina Virtual de Java
[8] Application Programming Interface (API), Interfaz de Programación de Aplicaciones
[9] Extensible Markup Language, Lenguaje de Marcado Extensible. Lenguaje diseñado para transportar y guardar datos.

Este tipo de aplicaciones son denominadas *aplicaciones empresariales* porque generalmente son diseñadas para resolver problemas de carácter empresarial.

Java ME

Esta plataforma provee de una API y una máquina virtual, la cual es reducida en comparación a las VM de Java SE y Java EE, para ejecutar aplicaciones Java en dispositivos tales como teléfonos celulares. La API de Java ME es un subconjunto de la API de Java SE junto con algunas librerías útiles en el desarrollo para pequeños dispositivos.

1.3.2 Aplicaciones distribuidas multicapa y Java EE

En una aplicación multicapa la funcionalidad de la aplicación es separada en partes o capas de acuerdo a su funcionalidad. Por lo general, las aplicaciones multicapa tienen una capa de cliente, una capa media y una capa de datos (a veces llamada capa de sistemas empresariales de información). La capa cliente consiste en un programa cliente que hace una petición hacia la capa media. Las funciones o métodos de negocio de la capa media atienden la petición y procesan la información almacenándola en un depósito de información.

La plataforma Java EE usa un modelo de aplicaciones distribuidas multicapa. La lógica de la aplicación se divide en componentes de acuerdo a la función, estos componentes están instalados en diferentes máquinas dependiendo de la capa en la que se encuentre de acuerdo al ambiente Java EE.

La figura 2 muestra dos aplicaciones Java EE divididas en capas, las cuales se describen a continuación:

- Capa cliente, se ejecuta en la máquina del cliente.
- Capa web, se ejecuta en el servidor Java EE.
- Capa de negocio, se ejecuta en el servidor Java EE.
- Sistema de Información Empresarial (SIE), se ejecuta en el servidor de base de datos.

Aunque una aplicación Java EE puede consistir de 3 de las 4 capas mostradas en la figura 2, las aplicaciones Java EE son generalmente consideradas como aplicaciones de tres capas porque están distribuidas en tres lugares: máquinas de los clientes, el servidor Java EE y los servidores de base de datos.

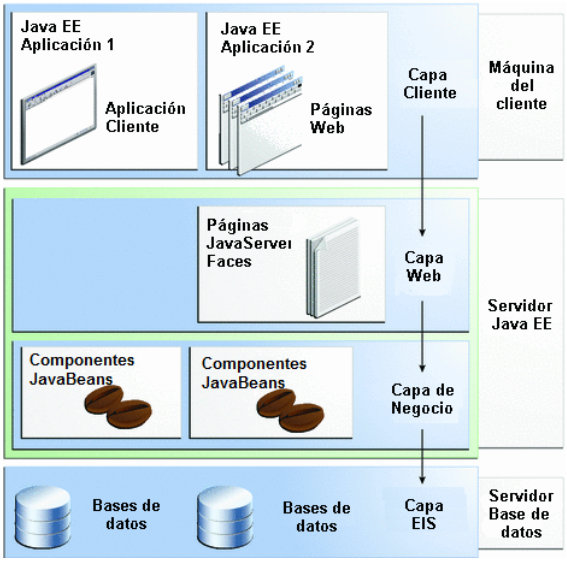

Figura 2. Capas de aplicaciones distribuidas

1.3.2.1 Componentes Java EE

Las aplicaciones Java EE están conformadas por componentes. Un componente Java EE es una unidad de software que es ensamblada en una aplicación Java EE con sus clases y archivos que se comunican con los otros componentes. La especificación Java EE define los siguientes componentes:

- Aplicaciones en el cliente y Applets son componentes que se ejecutan de lado del cliente.
- Java Servlet, JavaServer Faces Y JavaServer Pages (JSP) son componentes web que corren de lado del servidor.
- Los Enterprise JavaBeans (EJB) son componentes de negocio que corren sobre el servidor.

Los componentes son escritos en el lenguaje Java y son compilados de la misma manera que cualquier otro programa. La diferencia entre los componentes Java EE y las clases Java estándar es que los componentes Java EE son ensamblados en una aplicación Java EE, se verifica que se encuentren bien formados de acuerdo la especificación Java EE y son desempaquetados en un servidor de aplicaciones.

Clientes Java EE
Los clientes Java EE pueden ser web o aplicaciones.

Clientes web.
Un cliente web consiste de dos partes:

- Páginas web dinámicas. La páginas pueden contener varios tipos de lenguaje de marcado (HTML[10], XHTML[11], etc.) que son generadas por componentes que se ejecutan en la capa web.
- Navegador web. Presenta las páginas recibidas del servidor.

Un cliente web frecuentemente es llamado un cliente ligero. Los clientes ligeros usualmente no consultan la base de datos, no ejecutan reglas complejas de negocio ni se conectan con sistemas heredados[12]. Cuando se usa un cliente ligero, las operaciones pesadas son hechas por los EJBs que corren en el servidor, donde éstos se encargan de brindar la seguridad, rapidez, servicios y confiabilidad por medio de la comunicación con las tecnologías instaladas en el servidor.

Applets
Una página web que proviene de la capa web puede incluir un Applet embebido. Un Applet es una pequeña aplicación escrita en lenguaje Java que se ejecuta en la máquina virtual instalada en el navegador web. Sin embargo, los sistemas cliente necesitan un *plugin* y un archivo de póliza de seguridad para que el Applet se pueda ejecutar en el navegador web exitosamente.

Aplicaciones cliente
Una aplicación cliente corre sobre la máquina del cliente y brinda a los usuarios un modo para poder utilizar interfaces de usuario mucho más ricas (completas) que las proporcionadas por un lenguaje de

[10] HyperText Markup Language, Lenguaje de Marcado de Hipertexto. Lenguaje diseñado para creación de páginas web.
[11] Extensible HyperText Markup Language, Lenguaje Extensible de Marcado de Hipertexto. Lenguaje más estricto y limpio que el HTML.
[12] Sistema heredado. Es cualquier sistema que significativamente se resiste a la modificación y evolución.

marcado. Por lo general éstas tienen una interfaz gráfica creada a partir de la API Swing[13] ó AWT[14], aunque una interfaz de línea de comando también es posible.

Arquitectura de componentes JavaBeans

Las capas de servidor y cliente pueden incluir componentes basados en la arquitectura de componentes JavaBean para administrar el flujo de información entre una aplicación cliente o un Applet y los componentes que corren en el servidor, o entre los componentes del servidor y la base de datos.

Comunicación con el servidor

La figura 3 muestra la comunicación entre la capa cliente y el servidor. El cliente se puede comunicar directamente con la capa de negocio que reside en el servidor, o en el caso de de que el cliente se comunique por medio del navegador, la comunicación se da a través de páginas JSP y Servlets que corren en la capa web.

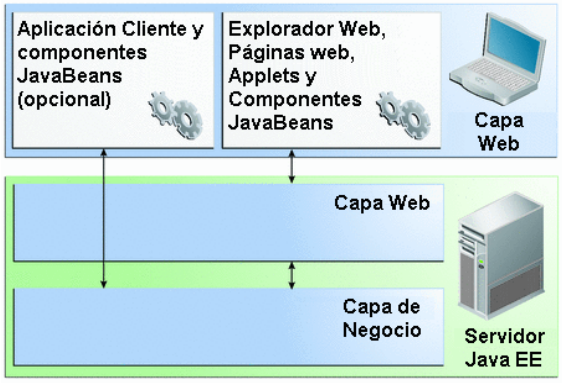

Figura 3. Comunicación de aplicaciones cliente con el servidor.

Componentes web

Los componentes web (figura 4) son Servlets o páginas creadas usando la tecnología JSP y/o la tecnología JavaServer Faces. Los Servlets son clases programadas en Java que dinámicamente procesan las peticiones y construyen las respuestas hacia la capa cliente. Las páginas JSP son documentos de texto que se ejecutan como Servlets pero que permiten un acercamiento más natural en el momento de crear contenido estático.

Las páginas HTML estáticas y Applets son integradas con los componentes web durante el ensamblado de la aplicación pero no son considerados componentes web por la especificación Java EE. Las clases adicionales o de utilidad también pueden ser integradas con los componentes web pero no son consideradas componentes web.

[13] Es una librería que implementa una serie de componentes GUI que se construyen sobre AWT.
[14] Abastract WindowToolkit. Es una librería que soporta la programación de GUI.

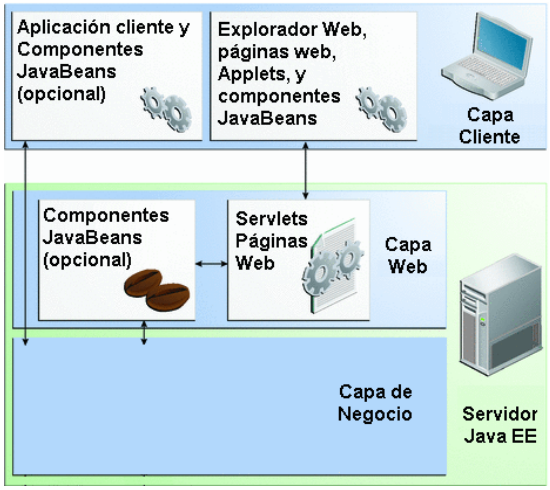

Figura 4. Componentes web.

Componentes de negocio

El código que contiene la lógica que permite resolver las necesidades de un área de conocimiento en específico, tal como: el sector financiero, bancario, etc., se encuentra en los Enterprise Beans (EJBs) que se ejecutan en la capa de negocio. La figura 5 muestra como un EJB recibe datos de programas cliente, los procesa y los envía a la capa del sistema de información para su almacenamiento. Un EJB también recupera datos almacenados, los procesa y los regresa al programa cliente.

Existen tres tipos de EJB: Beans de Sesión, Beans de Entidad y Beans de Mensajes. Un Bean de Sesión representa una conversación transitoria con el cliente. Cuando se termina la ejecución del proceso del cliente, el Bean de Sesión y sus datos se eliminan. En contraste, un Bean de Entidad representa los datos almacenados en un renglón de una tabla de base de datos. Un Bean de Mensaje combina las características de un Bean de Sesión y un oyente de mensajes JMS[15] (Java Message Service), permitiendo que un componente de negocio reciba mensaje asíncronamente.

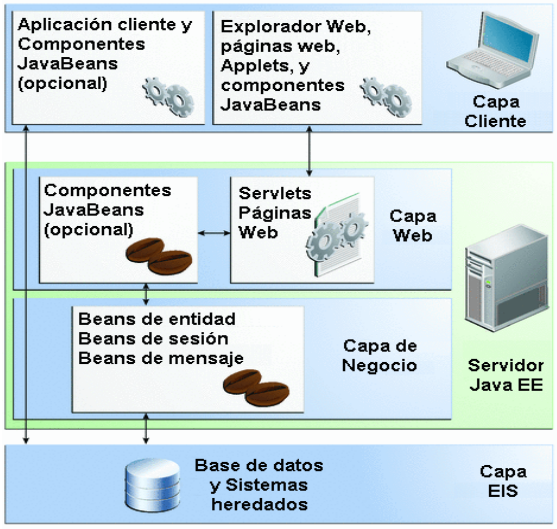

Figura 5. Componentes de negocio.

[15] Java Message Service. Es un API de java que permite a las aplicaciones crear, enviar, recibir y leer mensajes. JMS provee una forma de comunicación entre componentes de software y aplicaciones.

Capa del sistema de información.

La capa del sistema de información contiene el SIE e incluye sistemas de infraestructura como un ERP[16], un procesador mainframe de transacciones, sistemas de bases de datos y otros sistemas de información heredados.

1.4 Patrón Modelo Vista Controlador (MVC)

Pueden surgir muchos problemas cuando en las aplicaciones se mezcla el código para el acceso a datos, lógica del negocio y código para la presentación de datos. Estas aplicaciones son complicadas en el momento del mantenimiento porque las dependencias entre todos los componentes causan efectos indeseados cuando se hace algún cambio en alguna parte del código.

El patrón de diseño MVC resuelve este tipo de problemas separando el acceso a base de datos, la lógica de negocio, la presentación de datos y la interacción con el usuario.

1.4.1 Estructura

El siguiente esquema (figura 6) representa el patrón de diseño MVC:

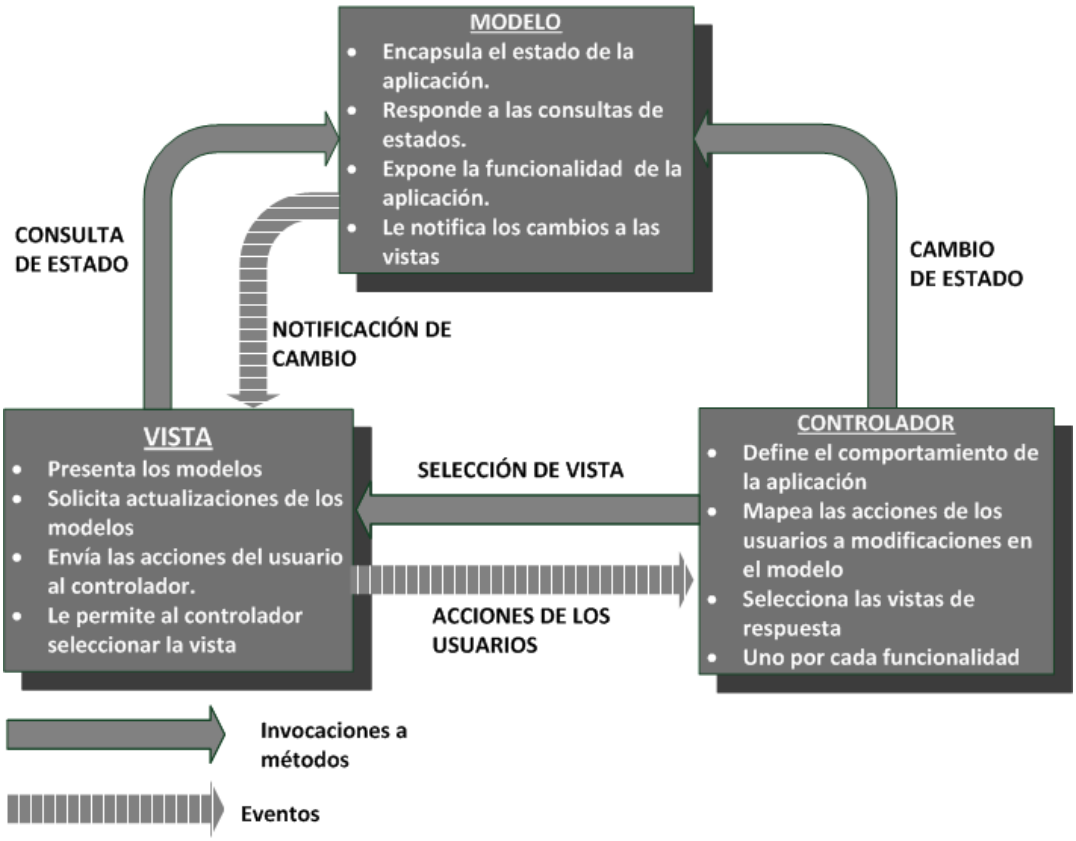

Figura 6.Esquema del patrón de diseño MVC.

[16] Enterprise Resource Planing, Planificación de Recursos Empresariales. Son sistemas enormes creados para integrar la administración de la operación y producción de una empresa en un solo sistema. Los sistemas ERP típicamente manejan la producción, logística, distribución, inventario, envíos, facturas y contabilidad de una compañía.

1.4.2 Participantes y responsabilidades

Modelo**.** El modelo representa los datos y las reglas de negocio que gobiernan el acceso y las modificaciones a esos datos. El modelo es un software que se asemeja al proceso en el mundo real, por lo tanto las técnicas de modelado del mundo real son las que se utilizan para definir el modelo.

Vista**.** La vista muestra los contenidos del modelo. Ésta accede a los datos a través del modelo y especifica como esos datos deberán presentarse. Es la responsable de mantener la consistencia en la presentación cuando el modelo cambia. Esto puede ser mejorado usando un modelo *push*, donde la vista se registra con el modelo para recibir notificaciones de cambio, o el modelo *pull*, donde la vista es responsable de llamar al modelo cuando ésta necesita presentar los datos más actuales.

Controlador**.** El controlador traduce las interacciones que tiene con la Vista en acciones que serán hechas por el Modelo. En una interfaz gráfica de usuario, las interacciones del usuario podrían ser los clics a un botón o la selección de menús, mientras que en una aplicación web, éstas aparecerían como peticiones HTTP. Las acciones hechas por el modelo incluyen activación de procesos de negocio o cambio del estado del modelo.

Basándose en las interacciones con el usuario y en las respuestas del modelo, el controlador responde seleccionando una vista apropiada.

1.4.3 MVC y Java EE

El modelo MVC es la arquitectura de diseño más recomendada para aplicaciones Java EE debido a que provee los siguientes beneficios: decremento de la duplicación de código, se tiene un control centralizado y sobre todo hace más fácil la modificación de la aplicación.

El modelo MVC se puede implementar utilizando Java EE de la siguiente manera:

Controlador

- Un Servlet recibe las peticiones del usuario, procesa la URL recibida y delega el procesamiento a los EJBs.
- El Servlet guardar el resultado del procesamiento realizado por los EJBs.
- El Servlet transfiere el control al JSP que lleva a cabo la presentación de los datos.

Modelo

- Los EJBs desempeñan el rol del modelo:
    - Algunos ejecutan lógica.
    - Otros guardan datos.

Vista

- Rol ejecutado por JSPs.
- El Servlet controlador transfiere el control al JSP.
- El JSP utiliza los datos generados por los EJBS para presentar la respuesta en HTML, XHTML o XML.

1.5 Servidores

1.5.1 Servidor web

Un servidor web es un programa (software) que exclusivamente atiende las peticiones del protocolo HTTP y responde en el mismo protocolo. Para procesar una petición, el servidor web responde con una página HTML estática o una imagen, o delega la generación de la respuesta a otros programas como scripts CGI[17], JSPs, Servlets, ASP[18] u otra tecnología de lado del servidor.

El servidor web no provee ninguna funcionalidad, tan sólo proporciona un ambiente en el cual un programa del lado del servidor puede ejecutarse y regresar las respuestas generadas.

1.5.1.1 Servidor Apache

El servidor Apache es un servidor web de código abierto desarrollado por una comunidad de desarrolladores alrededor del mundo denominada Fundación Apache de Software cuyos miembros están constantemente añadiendo nuevas funcionalidades.

Este servidor está disponible para la mayoría de los sistemas operativos actuales como Unix, Windows, Linux, Solaris, Novell, NetWare, Mac OSX, etc.

El servidor Apache soporta un variedad de servicios que son implementados por módulos que extienden la funcionalidad del *core*[19]. Algunas de las capacidades que proveen estos módulos son:

- Soporte para CGI.
- Autenticación de usuario.
- Direccionamiento (reescritura de la URL).
- Soporte para carga de módulos.
- Funcionalidades de cache.
- Dominios virtuales.
- Control de tráfico y ancho de banda.

1.5.2 Servidor de aplicaciones

Un servidor de aplicaciones es un programa (software) que proporciona acceso a la lógica del negocio a las aplicaciones cliente a través de diferentes protocolos, incluyendo el HTTP. Los servidores de aplicaciones se distinguen de los servidores web por el uso extensivo del contenido dinámico y acceso a base de datos.

1.5.2.1 Servidor Java EE

Un servidor Java EE es un servidor de aplicaciones que implementa la API de la plataforma Java EE y provee los servicios estándar Java EE. Los elementos primordiales de un servidor Java EE son: contenedor web y contenedor de EJB, aunque implícitamente también se encuentra un servidor de páginas o servidor web.

El contenedor web ofrece un ambiente donde residen los componentes web (JSPs, Servlets y páginas

[17] Common GateWay Interface. Interfaz de Entrada Común. Aplicación o programa escrito en cualquier lenguaje de programación que se ejecutan de lado del servidor que permite la construcción de páginas web dinámicas
[18] Active Server Pages. Tecnología de Microsoft parecida a los JSP para la generación dinámica de páginas web.
[19] Código base de una aplicación

Java Server Faces), además de que administra el ciclo de vida de los componentes, direcciona las peticiones hacia estos componentes, provee de herramientas para el manejo de sesiones y obtención de datos del contexto.

El contendor de EJBs ofrece un ambiente en donde residen los EJBs, los cuales contienen la lógica de negocio de la aplicación. Estos contenedores contemplan varias funcionalidades como:

- Manejo de transacciones. Apertura y cierre de transacciones asociadas a las llamadas a los métodos del EJB.
- Seguridad. Comprobación de permisos de acceso a los métodos del EJB.
- Concurrencia. Llamada simultánea a un mismo EJB desde múltiples clientes.
- Servicios de red. Comunicación entre el cliente y el EJB en máquinas distintas.
- Gestión de recursos. Gestión automática de múltiples recursos, tales como: colas de mensajes y bases de datos de aplicaciones heredadas.
- Persistencia. Sincronización entre los datos del EJB y las tablas de una base de datos.
- Gestión de mensajes. Manejo de Java Message Service (JMS).
- Escalabilidad. Posibilidad de constituir clusters[20] de servidores de aplicaciones con múltiples hosts[21] para poder dar respuesta a aumentos repentinos de carga de la aplicación con sólo añadir hosts adicionales.
- Adaptación en tiempo de despliegue. Posibilidad de modificación de todas estas características en el momento del despliegue del EJB.
- Conectividad ERP.
- Conectividad a sistemas heredados.

1.5.3 Servidor proxy en reversa

Un servidor proxy en reversa en un servidor que sirve como intermediario entre el cliente y el servidor donde está el servicio requerido. Cuando un cliente hace una petición, el servidor proxy en reversa intercepta esa petición y después direcciona la petición hacia el servidor que tiene el contenido que busca el usuario.

Los servidores proxy en reversa se usan generalmente para brindarles acceso a los usuarios a un servidor que se encuentre atrás de un firewall (figura 7). También pueden ser usados para balancear las cargas de varios servidores que se encuentren detrás de éste o proveer de memoria cache para incrementar la velocidad de respuesta. En algunos casos, los servidores proxy en reversa pueden ser usados simplemente para que distintos servidores compartan un mismo dominio en la URL.

1.5.3.1 Proxy en reversa como frente de otro servidor

Cuando se tiene un servidor donde se almacena información que debe ser asegurada, por ejemplo una base de datos de números de tarjetas de crédito, se puede configurar un servidor proxy en reversa afuera de un firewall y que sirva como frente del servidor que tiene el contenido. Cuando los clientes traten de acceder a través de Internet al servidor que tiene el contenido, éstos serán enviados al servidor proxy. La información reside en el servidor de contenido, la cual se encuentra segura detrás

[20] Cluster. Es un grupo de ordenadores unidos mediante una red, de tal forma que el conjunto es visto como un único ordenador.

[21] Host. Lugar donde reside un sitio web.

del firewall. Mientras que el servidor proxy aparece ante el cliente como si fuera el servidor de contenido.

De este modo, el proxy provee de una barrera adicional entre la base de datos y la posibilidad de un ataque.

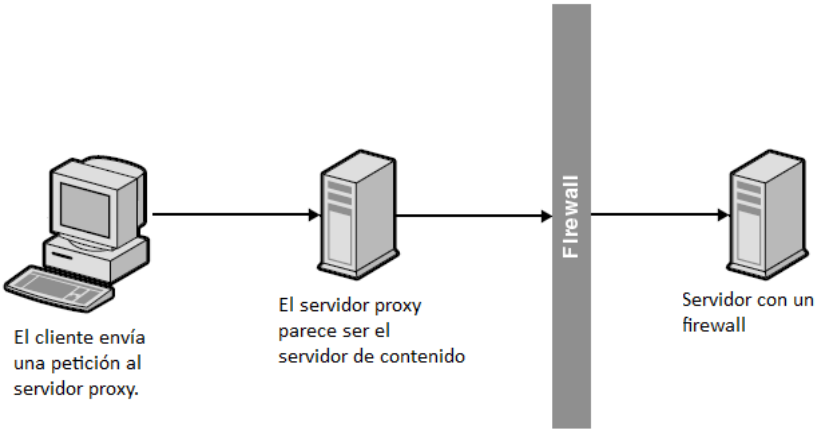

Figura 7. Esquema de servidor proxy en reversa.

1.5.3.2 Servidor proxy seguro en reversa

Cuando una o más de las conexiones entre el servidor proxy y otra máquina usan el protocolo SSL[22] para encriptar los datos, entonces se tiene un servidor proxy seguro.

Los servidores proxy seguros en reversa tienen los siguientes usos:

- Proveer de una conexión encriptada del servidor proxy hacia el servidor de contenido que se encuentra detrás del firewall.
- Permitir a los clientes conectarse de manera segura al servidor proxy, facilitando la transmisión segura de información.

Usar un servidor proxy seguro en reversa causa que cada conexión sea más lenta por el tiempo que lleva encriptar los datos.

1.5.3.3 Servidor proxy para balanceo de cargas

Es posible usar múltiples servidores proxy organizados de tal manera que permitan balancear la carga en la red en varios servidores web. Este modelo permite tomar ventaja de la memoria cache de un servidor proxy para crear un conjunto de servidores para balancear la carga en la red. Si se tiene un servidor web que recibe un alto número de peticiones por día, se pueden usar servidores proxy para repartir la carga sobre el servidor web y hacer el acceso a la red más eficiente.

Los servidores proxy actúan como direccionadores de las peticiones del cliente hacia el servidor de contenido y ponen en memoria cache los documentos solicitados. Si existe más de un servidor proxy, un servidor DNS[23] puede direccionar las peticiones aleatoriamente hacia los servidores proxy.

[22] Secure Sockets Layer. Protocolo de Capa de Conexión Segura. Es un protocolo que proporcionan comunicaciones seguras por una red.

[23] Domain Name System. Sistema de Nombre de Dominio. Es un sistema de nombres que permite traducir de nombre de dominio a una dirección IP.

El cliente usa la misma URL por cada petición, pero la petición puede ser atendida por un servidor proxy distinto. Después de un cierto periodo de estabilización donde los servidores proxy obtienen los documentos del servidor de contenido por primera vez, el número de peticiones hacia el servidor de contenido baja dramáticamente.

1.6 Oracle

La base de datos Oracle es un sistema de administración de base de datos relacional (RDBMS[24]) creado por la corporación Oracle. Es un sistema de base de datos, en donde se destaca:

- Soporte de transacciones. Las transacciones son un mecanismo estándar para manejar los cambios que se realizan al estado de un sistema distribuido. Proveen un modelo para controlar el acceso concurrente a los datos y para manejar las fallas inherentes al cómputo distribuido.
- Estabilidad. Es la habilidad de presentar un bajo número de fallas en el sistema.
- Escalabilidad. La escalabilidad es la propiedad que indica la capacidad de un sistema para crecer en tamaño sin perder calidad en los servicios ofrecidos. A partir de la versión 11g de Oracle, se anunció que su motor de base de datos tenía la habilidad de manejar cantidades de información superiores a un exabyte, donde un exabyte es 1,152,921,504,606,846,976 bytes[25].
- Soporte multiplataforma. Se refiere a la capacidad de poder ser utilizado en distintos sistemas operativos y plataformas de hardware. La versión 11g puede ser soportada por las siguientes combinaciones de sistemas operativos y procesadores:
    - Solaris. Procesador SPARC.
    - HP-UX 11i. Procesador PA-RISC.
    - AIX. Procesador POWER.
    - Red Hat Enterprise Linux. Procesador Intel/AMD.
    - Windows 2003 Server Enterprise Edition. Procesador Itanium.

1.6.1 Características de Oracle

Las características de Oracle se pueden agruparse en las siguientes secciones:

1.6.1.1 Características de programación de aplicaciones de base de datos

Todas las versiones de la base de datos Oracle incluyen diferentes lenguajes e interfaces, las cuales permiten a los programadores acceder y manipular los datos en la base de datos. Las características de programación usualmente son del interés de dos grupos: desarrolladores que crean aplicaciones que venden comercialmente y organizaciones o compañías que desarrollan aplicaciones muy particulares para sus negocios.

Los lenguajes e interfaces que soporta Oracle son:

- SQL[26]. Provee funciones básicas para la manipulación de datos, control de transacciones y consulta de datos.

[24] RDBMS. Relational Data Base Management System. Sistema de Administración de Base de Datos Relacional
[25] Oracle Database 11g, A Beginner's Guide. McGrawHill. 2009.
[26] Structured Query Language. Lenguaje de Consulta Estructurado. Es un lenguaje declarativo de acceso a base de datos relacionales que permite la elaboración de consultas con el fin de recuperar y modificar información.

- PL/SQL. Es un lenguaje procedural que extiende al SQL es comúnmente usado para la implementación de procesos almacenados[27], triggers[28], ciclos, sentencias condicionales y captura de errores.
- Java. Oracle 8i introdujo el uso de Java como un lenguaje procedural con una JVM (Java Virtual Machine, Máquina Virtual Java) en la base de datos. La JVM incluye soporte para procesos almacenados Java, métodos, triggers, EJBs, CORBA[29] y HTTP.
- Almacenamiento de objetos grandes como imágenes sin limitación de espacio.
- Programación orientada a objetos.
- Lenguajes de tercera generación. Los programadores pueden interactuar con la base de datos desde C, C++, Java, COBOL o FORTRAN con sentencias SQL embebidas en sus aplicaciones.
- Driver de bases de datos. Todas las versiones de Oracle incluyen drivers que permiten a las aplicaciones conectarse por ODBC[30] o JDBC[31]. También se permiten los proveedores OLE DB[32] y para .NET[33]
- NLS. Provee conjuntos de caracteres por diferentes idiomas.
- Extendido de la base. Se puede extender el SQL estándar para guardar y manipular imágenes, audio, video y series de tiempo de información.
- XML. Oracle añadió un tipo de dato nativo XML que se puede usar con el SQL para búsquedas.

1.6.1.2 Características de bases de datos distribuidas

Unas de las más fuertes características de la base de datos Oracle es su habilidad de ampliarse para soportar volúmenes extremadamente grandes de datos y usuarios. Oracle se caracteriza por ejecutarse en varias plataformas y en una configuración distribuida. Las bases de datos Oracle que se encuentran en diferentes plataformas pueden combinarse para actuar como una sola base distribuida.

1.6.1.3 Características de la migración de datos

Mover la información de una base a otra es un requerimiento frecuente cuando se usan bases de datos distribuidas o cuando un usuario quiere implementar múltiples copias de la misma base en diferentes lugares para reducir el tráfico o incrementar la disponibilidad de los datos. Se pueden exportar los datos y los diccionarios de datos de una base y se pueden importar en otra.

[27] Programa el cual es almacenado y ejecutado en la base de datos.
[28] Procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción, actualización o borrado de información.
[29] Common Object Request Broker Architecture. Arquitectura Común de Intermediarios en Peticiones de Objetos. Estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.
[30] Open Database Connectivity es un estándar de acceso a base de datos desarrollado por Microsoft que tiene por objetivo el acceder a la información desde cualquier aplicación, sin importar que sistema gestor de base de datos almacene los datos.
[31] Java Database Connectivity. Un API que permite la ejecución de operaciones sobre la base de datos desde el lenguaje de programación Java.
[32] Object Linking and Embedding for Databases. Enlace e Incrustación de Objetos para Base de Datos. Tecnología desarrollada por Microsoft usada para el acceso a diferentes fuentes de información de manera uniforme.
[33] .NET. Es una plataforma de Microsoft para el desarrollo y ejecución de aplicaciones.

1.6.1.4 Características de rendimiento

Oracle incluye diferentes características específicamente diseñadas para mejorar el rendimiento en ciertas situaciones. Entre las que se pueden mencionar son:

- Paralelismo. Las tareas que implementan el paralelismo incrementan la rapidez de consultas, optimización y mantenimiento de la base de datos. El paralelismo divide una sola tarea en tareas más pequeñas y a cada sub-tarea se le asigna un proceso independiente, lo cual incrementa de manera notable la rapidez de cierto tipo de operaciones.
- Índices. Proveen una rápida forma de seleccionar y obtener ciertos tipos de datos.
- Optimización de consultas.
- Vistas materializadas.
- Funciones estadísticas.
- Opción de minería de datos.

1.7 JavaScript

JavaScript es un lenguaje de programación interpretado con capacidades de orientación a objetos. Sintácticamente, el núcleo del lenguaje se asemeja a C, C++ y Java, debido a que cuenta con sentencias de programación similares como *if*, el ciclo *while* y el operador &&.

JavaScript no es un lenguaje tipado, es decir, que no se necesita especificar un tipo de dato para las variables. Los objetos en JavaScript mapean los nombres de las propiedades con valores arbitrarios. Por lo tanto éstos se asemejan más a *tablas hash* o *arrays asociativos*. El núcleo del lenguaje JavaScript soporta valores numéricos, cadenas y boléanos como tipos de datos primitivos. Además tiene soporte para objetos como arreglos, fechas y expresiones regulares.

JavaScript es comúnmente usado en exploradores web, en este contexto, el principal propósito es permitir crear código que interactúe con el usuario, que controle el navegador, que modifique el contenido que aparece en la ventana del explorador, que valide los datos capturados en un formulario, etc.

1.7.1 AJAX

El término AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como JavaScript asíncrono + XML.

AJAX no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes (como se observa en la figura 8). Las tecnologías que forman AJAX son:

- XHTML, HTML y CSS[34], para crear una presentación basada en estándares.
- DOM[35], para la interacción y manipulación dinámica de la presentación.
- XML, XSLT[36] y JSON[37], para el intercambio y la manipulación de información.

[34] CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML

[35] Document Object Model es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.

[36] Extensible Stylesheet Language, Lenguaje extendido de hojas de estilo. Hojas de estilo para documentos XML

[37] JavaScript Object Notation. Es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de

- XMLHttpRequest[38], para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

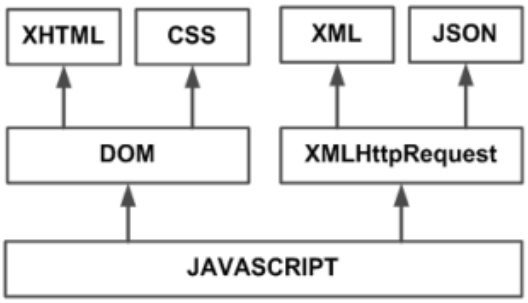

Figura 8. Esquema de las tecnologías involucradas con AJAX.

En las aplicaciones web tradicionales, las acciones del usuario en la página web (dar clic en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario. Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una sensación completa de funcionalidad al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano (de forma asíncrona). Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

En la figura 9, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el modelo propuesto por AJAX:

objetos de JavaScript.

[38] Es una interfaz de lado del explorador web empleada para realizar peticiones HTTP y HTTPS a servidores web.

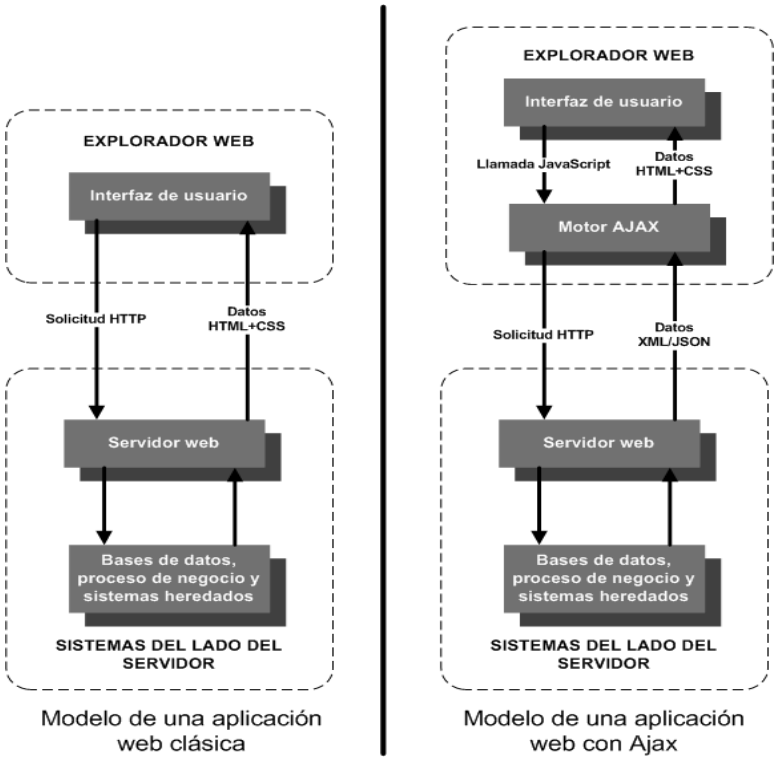

Figura 9. Comparación de modelo de una aplicación AJAX con un modelo de aplicación clásica.

1.7.2 Ext JS

Ext JS es una librería de JavaScript ligera y de alto rendimiento compatible con muchos navegadores que permite crear páginas web dinámicas a través de interfaces de usuario tipo escritorio.

Ext JS cuenta con diversos componentes (interfaces gráficas), tales como: grid de datos, árboles, gráficas, ventanas, formas y otros componentes útiles en el desarrollo de sistemas web. Todos estos componentes e interfaces tienen la capacidad de comunicarse a un servidor a través de AJAX y manejan XML y JSON para la transferencia de datos.

1.8 Metodologías de desarrollo de software

Las metodologías de desarrollo de software sirven para: conducir un proyecto bajo reglas específicas, tener control del código que desarrollamos y poder hacer más predecible y eficiente el desarrollo del proyecto. A su vez sirve para definir ¿Quién debe hacer qué?, ¿Cuándo? y ¿Cómo debe hacerlo?

1.8.1 Metodologías monumentales o tradicionales

Las metodologías monumentales (o tradicionales) indican cómo construir técnicamente el software y abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Éstas dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas[39].

[39] Pressman, Roger S. *Ingeniería del Software: un enfoque práctico.* Quinta edición. Mc. Graw-Hill, 2002.

1.8.2 Modelo lineal secuencial

También llamado ciclo de vida básico o modelo en cascada, el modelo lineal secuencial sugiere un enfoque sistemático y secuencial para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. La figura 10 muestra el modelo lineal secuencial para la ingeniería del software. El modelo lineal secuencial comprende las siguientes actividades Análisis, Diseño, Código y Prueba.

Figura 10. El modelo lineal secuencial. Pressman, Roger S. Ingeniería del software: un enfoque práctico.

El modelo lineal secuencial es el paradigma más antiguo y más extensamente utilizado en la ingeniería del software. Sin embargo, la crítica del paradigma ha puesto en duda su eficacia[40]. Entre los problemas que se encuentran algunas veces en el modelo lineal secuencial se incluyen:

1. Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo. Aunque el modelo lineal puede acoplar interacción, lo hace indirectamente. Como resultado, los cambios pueden causar confusión cuando el equipo del proyecto comienza.
2. A menudo es difícil que el cliente exponga explícitamente todos los requisitos. El modelo lineal secuencial lo requiere y tiene dificultades a la hora de acomodar la incertidumbre natural al comienzo de muchos proyectos.
3. El cliente debe tener paciencia. Una versión de trabajo del programa no estará disponible hasta que el proyecto esté muy avanzado. Un grave error puede ser desastroso si no se detecta hasta que se revisa el programa.

1.8.3 Modelo de construcción de prototipos

Un cliente, a menudo, define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficacia de un algoritmo, de la capacidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción hombre-máquina. En éstas y en otras muchas situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque.

El paradigma de construcción de prototipos (figura 11) comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un *diseño rápido*. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el

[40] Hanna, M. *Farewell to Waterfallm*, Software Magazine, Mayo 1995.

desarrollador comprenda mejor lo que se necesita hacer.

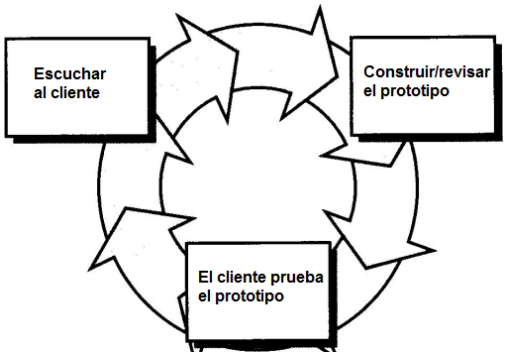

Figura 11. El paradigma de construcción de prototipos. Pressman, Roger S. Ingeniería del software: un enfoque práctico.

Sin embargo, la construcción de prototipos también puede ser problemática por las siguientes razones:

- El cliente ve lo que parece ser una versión de trabajo del software, sin saber que con la prisa de hacer que funcione no se ha tenido en cuenta la calidad del software global o la facilidad de mantenimiento a largo plazo.
- El desarrollador, a menudo, hace compromisos de implementación para hacer que el prototipo funcione rápidamente. Se puede utilizar un sistema operativo o lenguaje de programación inadecuado simplemente porque está disponible y porque es conocido.

1.8.4 Modelo incremental

El modelo incremental combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. Como se muestra en la figura 12, el modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software. Se debería tener en cuenta que el flujo del proceso de cualquier incremento puede incorporar el paradigma de construcción de prototipos.

Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial. Es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias (algunas conocidas, otras no) quedan sin extraer. El cliente utiliza el producto central (o sufre la revisión detallada). Como un resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto central a fin de cumplir mejor las necesidades del cliente y la entrega de funciones y características adicionales.

Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo. Cuando se tenga una fecha de entrega imposible de cambiar, el modelo incremental es un buen paradigma a considerar. El modelo de proceso incremental, como la construcción de prototipos y otros enfoques evolutivos, es iterativo por naturaleza. Pero a diferencia de la construcción de prototipos, el modelo incremental se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

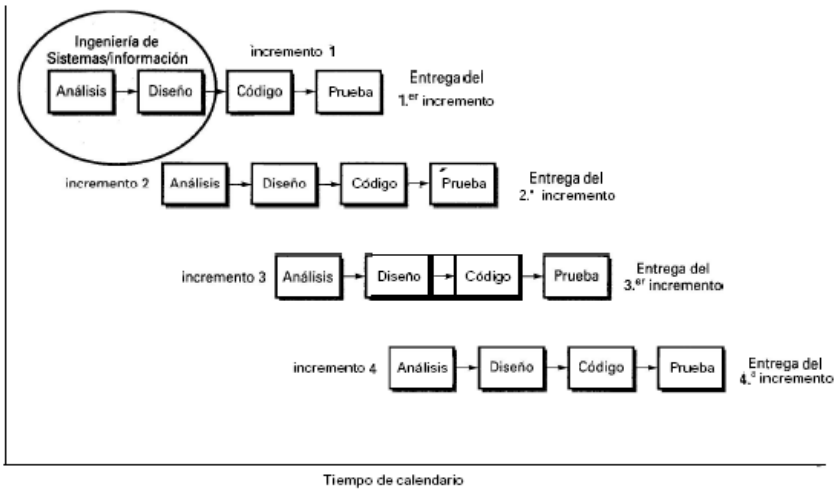

Figura 12. El modelo incremental Pressman, Roger S. Ingeniería del software: un enfoque práctico.

1.8.5 Modelo espiral

El modelo en espiral, propuesto originalmente por Boehm[41], es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

Figura 13 Modelo espiral típico. Pressman, Roger S. Ingeniería del software: un enfoque práctico.

La Figura 13 representa un modelo en espiral que contiene seis regiones de tareas:

- Comunicación con el cliente. Las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- Planificación. Las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.
- Análisis de riesgos. Las tareas requeridas para evaluar riesgos técnicos y de gestión.
- Ingeniería. Las tareas requeridas para construir una o más representaciones de la aplicación.
- Construcción y acción. Las tareas requeridas para construir, probar, instalar y

[41] Boehm, B.. *A Spiral Model for Software Developement and Enhancement.* Computer, Vol. 21. Mayo 1988, pp. 61-72.

proporcionar soporte al usuario (por ejemplo: documentación y práctica).

- Evaluación del cliente. Las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. Utiliza la construcción de prototipos como mecanismo de reducción de riesgos y permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real. El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto, si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemáticos.

1.9 Sitios web para dispositivos móviles

Los dispositivos móviles cada vez son más comunes en la vida diaria de las personas, a su vez, gracias a la tecnología actual es posible conectarse a Internet a través de dichos dispositivos. Por lo que los sitios de Internet ahora cuentan con diferentes versiones para ser visualizados en los dispositivos móviles con mayor uso.

Actualmente los sitios de Internet más populares cuentan con una versión para dispositivos móviles. Cuando se accede a un sitio de Internet a través de dichos dispositivos se puede acceder de manera directa a la versión para dispositivos móviles, otros sitios brindan la opción de elegir entre visualizar el sitio en su versión normal o móvil. Incluso existen sitios que permiten descargar una aplicación optimizada que permite contar con características específicas para el dispositivo en uso.

Las principales diferencias entre los sitios de Internet comunes y los sitios para dispositivos móviles se encuentran en la forma de visualización y la navegación, en donde la versión para móviles se encuentra optimizada para cualquier dispositivo, es decir, el sitio se adapta al tamaño de la pantalla, además toma en cuenta las limitaciones físicas y tecnológicas, tales como el límite de teclas, menor velocidad de descarga de contenido y límite del uso de tecnologías utilizadas en los sitios comunes, tales como Flash y JavaScript.

Por ejemplo, el uso, la exploración, visualización y navegación de un sitio a través de una PC o laptop es posible realizarla de manera sencilla, fácil y rápida por medio de un mouse, un monitor de tamaño considerable y un teclado de más de 100 teclas. En cambio si se desea acceder al sitio a través de un dispositivo móvil, la visualización y la navegación del sitio se vuelve más difícil debido a sus características limitadas en comparación con una computadora de escritorio o laptop, debido a que, éste cuenta con una pantalla pequeña, un dispositivo señalador sencillo y un teclado limitado a menos teclas.

De esta manera, se vuelve fundamental contar con sitios optimizados para dispositivos móviles, en los cuales se despliega el mismo contenido, con una interfaz más sencilla, pero con características que

permitan al usuario utilizar el sitio con su dispositivo móvil de preferencia.

1.9.1 Recomendaciones para el desarrollo de aplicaciones móviles

Existen recomendaciones establecidas por el grupo World Wide Web Consortium[42] (W3C), Open Mobile Alliance (OMA)[43] y MobiForge[44] para el desarrollo de sitios web para aplicaciones móviles.

Las recomendaciones que existen son las siguientes: Mobile web best practices[45] por parte de la W3C, XHTML Mobile Profile[46] y Wireless CSS[47] determinados por OMA y dotMobi Mobile Web Developers Guide realizada por mobiForge, en donde las recomendaciones de OMA y dotMobi están basadas en la recomendación de XHTML Basic 1.1[48] de la W3C.

1.9.1.1 Mejores prácticas para el desarrollo de sitios web para aplicaciones móviles (Mobile Web Best Practices)

Las mejores prácticas para el desarrollo de sitios web para dispositivos móviles han sido determinadas por el grupo de la W3C en el documento Mobile Web Best Practices 1.0[49]. En donde el principal objetivo de las mejores prácticas es establecer los estándares necesarios para mejorar la experiencia del usuario en la navegación del sitio cuando éste accede desde un dispositivo móvil.

El documento Mobile Web Best Practices 1.0, en resumen, establece lo siguiente:

1.9.1.2 Limitaciones de los dispositivos

Los dispositivos móviles frecuentemente no soportan *scripts* o *plugins*, lo que significa que el rango de contenidos que soportan es limitado. En muchos casos el usuario no tiene opción de elegir el explorador y la actualización a veces no es posible.

Otras actividades asociadas con el despliegue de páginas de Internet representan una fuerte carga computacional, por ejemplo, direccionamiento de páginas, maquetación[50] basada en tablas, uso innecesario de largas y complejas hojas de estilo y manipulación de código de maquetación inválido. Los dispositivos móviles tienen poder de procesamiento limitado lo que significa que el despliegue de la página puede tomar un largo tiempo.

Muchos dispositivos tienen memoria disponible limitada para páginas e imágenes y el exceso del límite de memoria resulta en un despliegue incompleto de la página y pueden causar otros problemas.

[42] World Wide Web Consortium http://www.w3.org/
[43] Open Mobile Alliance http://openmobilealliance.org/
[44] mobiForge http://mobiforge.com/
[45] Mobile Web Best Practices 1.0 http://www.w3.org/TR/mobile-bp/
[46] XHTML Mobile Profile http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf
[47] Wireless CSS http://www.openmobilealliance.org/technical/release_program/docs/Browsing/V2_3-20080331-A/OMA-WAP-WCSS-V1_1-20061020-A.pdf
[48] XHTML Basic 1.1 http://www.w3.org/TR/xhtml-basic/
[49] Mobile Web Best Practices 1.0. http://www.w3.org/TR/mobile-bp/
[50] Se refiere a la acción de organizar en un espacio, contenidos escritos, visuales y en algunos casos audiovisuales en medios impresos y electrónicos

1.9.1.3 Web única

Una web única significa hacer, en la medida de lo razonable, que la misma información y servicios se encuentren disponibles para los usuarios, independientemente del dispositivo que estén usando. Sin embargo, esto no significa que la misma información esté disponible exactamente en la misma representación a través de todos los dispositivos. Los factores que afectan la representación son: el contexto de uso móvil, variaciones de las capacidades de los dispositivos, problemas de ancho de banda y capacidades de la red móvil.

Es probable que los diseñadores de aplicaciones y los proveedores de servicios deseen brindar la mejor experiencia posible en el contexto en el cual su servicio tiene el mayor atractivo. Sin embargo, mientras los servicios pueden ser más apropiados en un contexto u otro, es considerada una buena práctica brindar la mejor experiencia tanto como sea posible dadas las limitaciones del dispositivo y no excluir el acceso de ninguna clase de dispositivo en particular, excepto donde esto sea necesario debido a las limitaciones del dispositivo.

1.9.1.4 Contexto de Entrega Predeterminado (CEP)

Con el fin de permitir que los proveedores de contenidos compartan una visión consistente de una experiencia móvil se ha definido el CEP, el cual ha sido determinado como la mínima especificación de entrega de contexto necesaria para una experiencia razonable del sitio web.

En resumen, el principal propósito del CEP es apoyar a las siguientes reglas:

- Si el proveedor de contenido puede obtener las características del dispositivo, entonces la información que es conocida como el contexto de entrega actual debería ser usado para variar el contenido entregado para hacerlo más adaptable para ese contexto de entrega específico o para brindar una experiencia de usuario mejorada.
- Si los detalles del contexto de entrega no pueden ser adecuadamente determinados, entonces el contenido de entrega debería adaptarse al CEP y debe cumplir con las especificaciones de las mejores prácticas.

Los requerimientos mínimos para que los usuarios de dispositivos móviles puedan tener la mejor experiencia posible al visitar un sitio web se determinan en el CEP, el cual se define a continuación:

- Ancho de pantalla: 120 píxeles como mínimo.
- Lenguaje de marcado: XHTML Basic con el tipo de contenido *application/xhtml+xml.*
- Codificación de caracteres: UTF-8.
- Formato de imágenes: JPEG y GIF.
- Máximo peso total de la página: 20 kilobytes.
- Colores: mínimo 256.
- Hoja de estilos: CSS Level 1.
- HTTP: HTTP/1.0 ó HTTP/1.1.
- No usar scripts .

1.9.1.5 Mejores prácticas

Contexto

- Debe estar disponible la misma información y los mismos servicios para los usuarios independientemente del dispositivo que estén usando.
- Explotar las capacidades del dispositivo para brindar al usuario una experiencia mejorada.
- Llevar a cabo pruebas del sitio web en dispositivos móviles y en emuladores.
- Mantener cortas las URL de inicio.

Navegación y enlaces

- Brindar mínima navegación en el encabezado de la página.
- Hacer un balance entre evitar agregar muchos enlaces hacia otras páginas y evitar que el usuario siga muchos enlaces para obtener la información que está buscando.
- Brindar mecanismos consistentes de navegación.
- Asignar teclas de acceso rápido a los enlaces que se encuentran en el menú de navegación y a los enlaces frecuentemente accedidos.
- Definir claramente el objetivo de cada enlace.
- Evitar el uso de mapas de imágenes, los cuales contienen más de un enlace en una imagen, a menos que los dispositivos de los usuarios lo soporten.
- No abrir ventanas emergentes o abrir nuevas ventanas.
- No crear páginas que se auto-refresquen periódicamente.
- No utilice marcadores para redirigir las páginas automáticamente debido a que el tiempo de carga de la página se incrementa.
- Minimizar el número de enlaces a recursos externos.

Gráficos

- Evitar el uso de imágenes de gran tamaño y peso. Una alternativa para el manejo de imágenes es especificar sus dimensiones en pixeles a través de la hoja de estilos. Otra alternativa es usar dimensiones de porcentaje o usar unidades de medida relativas tales como em y ex. Las unidades de medida relativas definen su valor real en relación con otra medida. La unidades relativas em y ex se definen como el ancho de la letra M (eme mayúscula) y el ancho de la letra x (equis minúscula) del tipo y tamaño de letra que se esté utilizando.
- Al usar imágenes de fondo, asegurarse de que el contenido sigue siendo legible en el dispositivo.

Contenido y presentación de las páginas

- Asegurar que el contenido es apto para ser usado en el contexto de una aplicación móvil.
- Limitar el contenido de la información que los usuarios obtienen, es decir, usar un lenguaje claro y conciso que responda a las necesidades del cliente.
- Evitar que el tamaño total de cada página sea el adecuado para no sobrepasar las limitaciones de memoria del dispositivo.
- En caso de que la información sea grande habrá que dividir la información en porciones de tamaño limitado.
- Limitar en lo más posible el recorrido de la página en un sólo sentido, es decir, de arriba hacia abajo, o de izquierda a derecha.

Definición de la página

- Las páginas deben tener un título corto y descriptivo.
- No usar animaciones.
- No usar tablas a menos que los dispositivos de los usuarios las soporten. En su defecto usar alternativas a una presentación tabular tal como un diseño basado en etiquetas “div”.
- En caso de usar tablas, evitar el uso de tablas anidadas.
- Evitar el uso de tablas para realizar maquetación.
- Evitar el uso de scripts y objetos embebidos.
- Brindar una equivalencia de texto para aquellos elementos que no sean texto, por ejemplo, para las imágenes agregar un nombre a través de su atributo alt=”nombre”, de esta forma cuando la imagen no se visualice en el dispositivo el usuario sabrá que debiera existir una imagen.
- Utilizar XHTML básico o XHTML Mobile Profile validado, el cual asegura una máxima eficiencia en los dispositivos móviles. El código utilizado en la página se puede validar por medio del sitio de la W3C[51].
- Utilizar hojas de estilos para controlar la maquetación y la presentación, a su vez su tamaño debe ser pequeño.
- Organizar el documento para que pueda ser leído sin hojas de estilo, en caso de que el dispositivo móvil no soporte el uso de hojas de estilo.
- Enviar contenido en un formato que sea soportado por el dispositivo móvil, tal como application/xhtml+xml. A su vez asegurarse de que el contenido está codificado en una codificación de caracteres soportado por el dispositivo móvil, el más común es UTF-8.
- Manejo de errores a través de pantallas que muestren que ha sucedido un error y que permitan navegar hacia una página correspondiente al sitio en uso.
- No confiar en la disponibilidad de cookies.
- No confiar en el soporte de estilos para las fuentes.
- Utilizar la memoria cache para reducir los tiempos de carga de la página. La información que puede almacenarse son las hojas de estilo, imágenes y páginas.

Captura de información

- Evitar que el usuario ingrese texto de manera libre, a menos que sea indispensable.
- En caso de que el usuario deba ingresar texto brindar ajustes del texto a través de las hojas estilo.
- En el caso de los formularios brindar valores preseleccionados donde sea posible.

1.9.1.6 XHTML MP 1.2

XHTML-MP[52] es una especialización de XHTML diseñada para incorporar características útiles para los dispositivos móviles. XHTML-MP 1.0 fue definido por la OMA y es una extensión de XHTML Basic 1.0.

A través del tiempo, OMA ha desarrollado XHTML-MP y ahora tiene una propuesta de versión 1.2 de su especificación. Gracias a los esfuerzos de alineación entre la W3C y la OMA, la propuesta de la W3C, XHTML Basic 1.1 y XHTML-MP 1.2 son virtualmente indistinguibles.

[51] Sitio de la W3C que brinda un servicio de validación del código de documentos web: http://validator.w3.org/
[52] Extensible Hypertext Markup Language - Mobile Profile

Dado que XHTML Basic y XHTML-MP son subconjuntos de XHTML, la transición a la producción de contenidos móviles amigables necesariamente no es difícil para los desarrolladores, herramientas estándar de desarrollo tales como exploradores de Internet de escritorio y ambientes de desarrollo integrados pueden ser usados para desarrollo de aplicaciones para móviles, de esta forma, los desarrolladores no necesitan entender un lenguaje completamente nuevo.

XHTML Basic 1.1 es un conjunto que comienza a ser el nivel de estándar de soporte en los dispositivos móviles; en el presente XHTML-MP es el dialecto más ampliamente soportado. Un explorador XHTML-MP 1.0 y cualquier explorador XHMTL (tal como un explorador de escritorio) deberán desplegar apropiadamente un sitio codificado en XHTML Basic 1.1.

1.9.1.7 Wireless CSS Profile

XHTML-MP viene con un concepto móvil amigable que consiste en usar CSS para separar la presentación del marcado, al igual que en el escritorio. El estándar Wireless CSS, administrado por OMA, es un subconjunto de CSS y también es parte de la especificación WAP 2.0[53]. Cabe aclarar que Wireless CSS no es compatible con WML[54].

Se pueden agregar Wireless CSS a un documento de la misma manera como se hace en un documento HTML. Wireless CSS soporta muchos atributos de CSS, pero no todos ellos. Cabe notar que las técnicas de estilo más avanzadas funcionarán de manera correcta a través de múltiples navegadores móviles. El mejor consejo es mantener el CSS tan simple como sea posible.

1.9.2 Arquitectura de Información de sitios web para dispositivos móviles

Está basada en el documento Mobile Web Developers Guide de MobiForge[55], en donde, en el contexto de dispositivos móviles es especialmente importante que la estructura de la información se mantenga tan simple como sea posible.

Los siguientes enfoques son los más apropiados para el desarrollo de sitios para dispositivos móviles:

1.9.2.1 Arquitectura Drill-Down

La arquitectura Dril-Down consiste en anidar contenido dentro de categorías bien definidas. Así el usuario profundiza en el detalle de la información cada vez que accede a las subcategorías de cada categoría.
Las recomendaciones de la arquitectura Drill-Down son las siguientes:

- Limitar las categorías en lo más posible. Se recomienda un límite de cinco categorías.
- Intentar limitar los enlaces de cada categoría, a su vez cada enlace debe estar asociado a una tecla de acceso rápido. Se recomienda limitar a 10 enlaces por página, (0-9), para que exista compatibilidad con cualquier dispositivo que no posea teclado.
- Priorizar los enlaces de acuerdo a su actividad o popularidad.

[53] Wireless Application Protocol http://www.wapforum.org/what/WAPWhite_Paper1.pdf
[54] Wireless Markup Language
[55] Mobile Web Developers Guide of MobiForge http://mobiforge.com/starting/story/dotmobi-mobile-web-developers-guide

La figura 14 muestra un ejemplo de arquitectura Drill-Down.

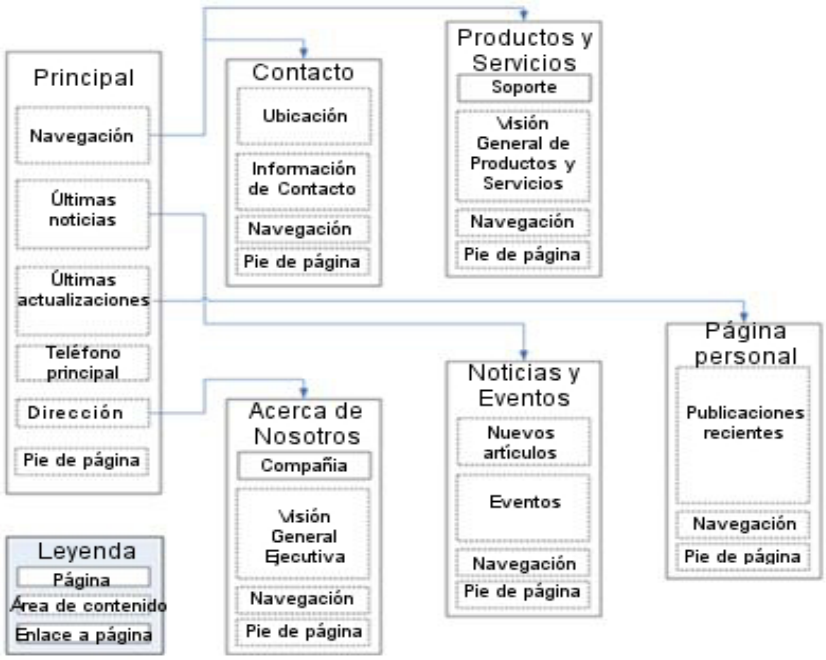

Figura 14.Ejemplo de arquitectura Drill-Down.

1.9.2.2 Limitar opciones

Para la publicación de información en el sitio se debe tomar el contenido que sea relevante para el usuario y descartar el resto. Lo anterior resulta en un sitio simple y concentrado que reduce el riesgo de que el usuario se pierda en la navegación del sitio.

1.9.2.3 Diseño para diferentes pantallas

Se debe realizar el diseño de las páginas de tal manera que pueda ser visualizado en las resoluciones de pantalla de la figura 15.

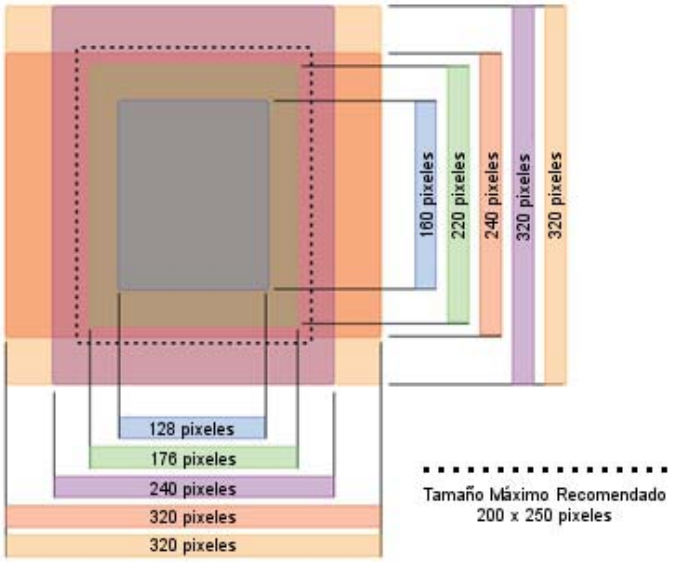

Figura 15. Resolución de pantallas para dispositivos móviles.

1.9.2.4 Diseño de página e información

El método más común para la creación de esquemas de navegación es usar una simple lista vertical de opciones (figura 16 y figura 17) que corresponden a teclas de acceso rápido del 0-9, además de que sólo se muestran los enlaces relacionados a la página en uso. El diseño de página recomendado es del tipo vertical y de una sola columna, en donde el recorrido de la página se realiza de arriba hacia abajo.

Figura 16. Resolución de pantallas para dispositivos móviles.

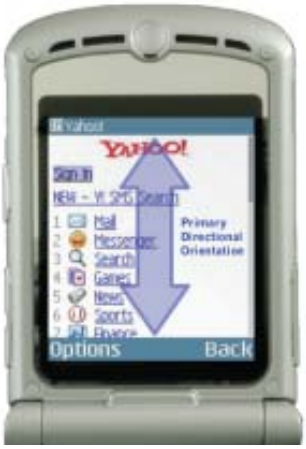

Figura 17. Resolución de pantallas para dispositivos móviles.

1.9.3 Estándares web para la publicación y navegación de contenidos para dispositivos móviles

Los estándares para la publicación y navegación de contenidos se refieren a lo siguiente: soporte de dispositivos, nombre del sitio y configuración del tipo MIME del servidor

Dichos estándares se encuentran definidos en el documento Mobile Web Developers Guide creado por la empresa MobiForge.

1.9.3.1 Soporte de dispositivos

Siguiendo las recomendaciones de MobiForge y sus estándares se debe asegurar que el sitio puede ser visualizado en la mayoría de los dispositivos, es decir, se debe asegurar que la publicación del sitio ha sido realizada de manera correcta. A su vez, para brindar un mayor soporte a los usuarios que visiten el sitio web, se puede verificar cuáles son los *user agents*[56] almacenados en la bitácora de acceso al servidor web.

[56] Agente de Usuario. Cadena de texto que forma parte de una petición HTTP, la cual contiene información del dispositivo y su explorador web.

1.9.3.2 Nombre del sitio

A continuación se muestran las formas en las cuales el usuario puede navegar en un sitio diseñado para una aplicación móvil:

- Utilizar el dominio .mobi para indicar que el sitio está diseñado para ser accedido por medio de dispositivos móviles.
- Si no se cuenta con un dominio .mobi, se puede indicar a los usuarios que ingresen de la siguiente manera: *miaplicacion.com/mobile* o *mobile.miaplicacion.com*
- Detectar el dispositivo móvil de manera automática y direccionar al usuario a la ubicación del sitio diseñado para aplicaciones móviles. En este caso el usuario ingresa la dirección de la aplicación, la cual es única, brindando al usuario la opción de acceder a la versión completa en caso de que el usuario cuente con un dispositivo móvil avanzado.

1.9.3.3 Configuración del tipo MIME[57] en el servidor

Para cada petición realizada desde un explorador de Internet hacia el servidor web, el protocolo HTTP requiere que el servidor incluya un encabezado con el tipo de contenido que identifica el formato de la respuesta (MIME), que es utilizado por el explorador para decodificarla.

Así, se vuelve fundamental configurar el servidor web para que entregue el encabezado con el tipo de contenido correcto, el cual es *application/xhtml+xml*. En caso de que esto no suceda, en el mejor de los casos causaría que la carga de la página sea lenta y en el peor de los casos que el sitio no pueda ser leído y el explorador envíe un mensaje de error.

1.9.4 Pruebas de sitios web para dispositivos móviles

Las pruebas de los sitios web pueden considerarse como una tarea pesada y larga considerando la gran cantidad de dispositivos que existen y las diferencias en cómo se despliega el contenido del sitio en dichos dispositivos.

Para evitar realizar pruebas en cada uno de los dispositivos móviles existentes y simplificar dicha tarea se deberán realizar los siguientes tipos de pruebas: escritorio, iframes, firefox, emuladores y dispositivos reales.

1.9.4.1 Pruebas de escritorio

Es recomendable realizar pruebas del lenguaje de marcado y de las hojas de estilos en un explorador de internet de escritorio antes de probarlo en un dispositivo móvil. Al realizar este tipo de pruebas se pueden encontrar errores básicos de XHTML y CSS de manera rápida y sencilla a diferencia de realizar pruebas en dispositivos móviles.

Por ejemplo el explorador Opera tiene una opción para mostrar una pequeña pantalla aproximadamente del tamaño de un dispositivo móvil, de esta manera podemos observar el tamaño de los elementos, el tamaño de la letra y la manera en que se muestra el estilo de las páginas.

[57] Multipurpose Internet Mail Extensions. Extensiones Multipropósito de Correo de Internet.

1.9.4.2 Pruebas por medio de iframes

Otra forma de probar los sitios es la de crear una página web con un iframe incrustado. Un iframe es una etiqueta HTML que permite incrustar un sitio web dentro de una página web por medio de un enlace y con medidas específicas. De esta manera, para probar un sitio web para dispositivos móviles en una página HTML se deberá agregar una etiqueta iframe con dimensiones específicas y asociadas a un dispositivo móvil.

1.9.4.3 Pruebas cambiando el agente de usuario con Firefox

Firefox permite cambiar el agente de usuario por medio de la extensión User Agent Switcher[58]. El agente de usuario es una cadena de texto enviada como parámetro cuando se realiza una petición HTTP por medio del explorador de Internet. De esta forma el explorador se identifica ante un servidor de aplicaciones como un dispositivo móvil y si el servidor cuenta con una versión para dichos dispositivos entonces se recibirá como respuesta la versión del sitio para dispositivos móviles.

1.9.4.4 Pruebas en emuladores

Las pruebas en emuladores son lo más cercano a utilizar un dispositivo móvil específico. Una de las ventajas de utilizar un emulador es que permite realizar las pruebas del sitio de manera rápida evitando los tiempos de carga en un dispositivo móvil real.
Un emulador no reemplaza las pruebas en un dispositivo móvil real, es una herramienta útil durante el desarrollo de sitios para dispositivos móviles para realizar la rápida verificación de la forma en que el sitio se muestra.

1.9.4.5 Pruebas en dispositivos móviles

Representa la experiencia real de navegar en un sitio desarrollado para aplicaciones móviles. Se recomienda hacer pruebas en los dispositivos más comunes y complementar las pruebas con las mencionadas anteriormente.

[58] User Agent Switcher. Extensión de Firefox que permite cambiar el agente de usuario que es enviado por medio de la petición HTTP. https://addons.mozilla.org/es-es/firefox/addon/user-agent-switcher/