Capitulo 2. Marco teórico

Antes de entrar en detalle en cuestiones técnicas hay que describir brevemente los términos que se utilizarán a lo largo de esté documento comenzando por Web e Internet continuando con Ingeniería de Software, patrón de diseño MVC, tecnología JSP / Servlet y finalmente con la definición de Base de Datos.

2.1 WEB e Internet

Como lo dice el título de la tesis esté sistema funciona bajo un contexto *Web*, pero ¿a se refiere esta palabra?, para desarrollar esté concepto nada mejor que recurrir a *wikipedia (http://es.wikipedia.org/wiki/Portada)* donde encontré que el World Wide Web, o simplemente Web, es un universo de información accesible a través de Internet, una fuente inagotable del conocimiento humano.

Su característica sobresaliente es el texto remarcado, un método para referencias cruzadas instantáneas. En la mayoría de los Sitios Web, ciertas palabras aparecen de otro color diferente al resto del documento. Por lo general, este texto es subrayado. Al dar *clic* sobre una palabra o frase, uno es transferido al sitio o página relacionada a esa frase. En algunas ocasiones hay botones, imágenes, o porciones de imágenes que pueden activarse mediante un *clic*.

Usando el Web, se tiene acceso a millones de páginas de información. La exploración en el Web se realiza por medio de un software especial denominado navegador, Browser o Explorador. La apariencia de un Sitio Web puede variar ligeramente dependiendo del explorador que se use. Así mismo, las versiones más recientes disponen de una funcionalidad mucho mayor tal como animación, realidad virtual, sonido y música.

La Web no se puede entender sin saber que es **Internet**, esta palabra se refiere a una red mundial de computadoras conocida también como red de redes, ya que es integrado por las diferentes redes de cada país del mundo, por medio del cual un usuario en cualquier computadora puede, en caso de contar con los permisos apropiados,

acceder a información de otra computadora y poder tener inclusive comunicación directa con usuarios de otras computadoras.

Fue concebido por la agencia de nombre ARPA (*Advanced Research Projects Agency*) del gobierno de los Estados Unidos en el año de 1969 y se le conocía inicialmente como ARPANET. El propósito original fue crear una red que permitiera a los investigadores en un Campus poder comunicarse a través de los sistemas de cómputo con investigadores en otras Universidades [Ceballos, 2002].

Hoy en día, el Internet es un medio de comunicación público, cooperativo y autosuficiente en términos económicos, accesible a cientos de millones de personas en el mundo entero. Físicamente, el Internet usa gran parte del total de recursos actualmente existentes en las redes de telecomunicaciones. Técnicamente, lo que distingue al Internet es el uso del protocolo de comunicación llamado TCP/IP (*Transmission Control Protocol/Internet Protocol*) [Ceballos, 2002].

Para muchos usuarios de Internet, el correo electrónico (e-mail) ha reemplazado prácticamente al servicio postal para breves mensajes por escrito. El correo electrónico es la aplicación de mayor uso en la red. También se pueden realizar conversaciones "en vivo" con otros usuarios en otras localidades usando el IRC (*Internet Relay Chat*). Más recientemente, el software y hardware para telefonía en Internet permite conversaciones de voz en línea.

Para no salirnos del tema, la clave para el desarrollo de esté y cualquier otro sistema, la ingeniería de software es fundamental y si no se realiza adecuadamente puede valer un fracaso. Pero ¿Qué es Ingeniería de software?

2.2 Ingeniería de software

“La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo operación (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software” [IEEE, 1990].

Ahora tenemos que entender el concepto de software ya que la definición anterior gira alrededor de esta palabra. “Software es el conjunto de requerimientos

operacionales, especificaciones, código, guías, manuales y documentación de mantenimiento de un sistema basado en computadora"[Pressman, 2002].

A grandes rasgos la ingeniería de software se refiere al conjunto de operaciones (análisis, diseño, desarrollo y pruebas) que se deben realizar antes, durante el desarrollo y hasta la deliberación del software.

2.3 MVC: Model View Controller (Modelo Vista Controlador)

Como se describe en el libro *"Head First Design Patterns"* [Freeman 2004] el termino Patrón de Diseño no es nada nuevo, sólo hoy en día está siendo utilizado de manera más común, ya que un patrón de diseño no es exclusivo del desarrollo de software debido a que se puede aplicar a cualquier área que involucre la tarea de diseñar. Por otro lado MVC es un patrón de diseño utilizado para proveer una solución al problema de recurrencia de funciones, es decir, se separan los componentes por dos razones principales; primero ahorrar líneas de código permitiendo recurrencia de ciertas funciones y segundo que si algún componente de la aplicación cambia no afecte a la estructura general del sistema.

Un patrón de diseño en términos computacionales se puede decir que es independiente del lenguaje de programación que se desee utilizar, pero es aplicable a diversos dominios de problema, pero semejantes desde el punto de vista de la estructura lógica de la solución, es decir, utiliza principios de diseño orientado a objetos. Teniendo esto en consideración entonces un patrón es una estructura común que tienen diferentes aplicaciones y/u objetos [Freeman 2004].

También conviene distinguir entre un patrón y una arquitectura global del sistema. Por decirlo brevemente, es la misma distancia que hay entre el diseño de un componente (o módulo) y el análisis del sistema. Es la diferencia que hay entre el aspecto micro y el macro, por ello, en ocasiones se denomina a los patrones como "micro arquitecturas". Desde el punto de vista de la arquitectura de sistemas hablamos habitualmente de arquitecturas multi-capa, distinguiendo para este sistema la siguiente distribución (véase capítulo 4):

- Capa de usuarios.- A través de esta capa el usuario establece la comunicación con el sistema, básicamente es la presentación del sistema con el usuario.
- Capa de servicios.- Aquí es donde se define la lógica aplicativa utilizada para realizar las operaciones requeridas por el usuario.
- Capa de base de datos.- Esta capa solamente se refiere al acceso de datos para generar información.

La mayor parte de los patrones más utilizados, son esquemas lógicos aplicables a dichas capas. Los Patrones de Diseño a su vez son muchos y muy variados, pero concentran muchas de las ideas de un modelo lógico a seguir para plantear una solución de acuerdo a las particularidades de cada negocio, siguiendo un orden tecnológico para su posterior desarrollo, por ejemplo, el patrón Modelo-Vista-Controlador (MVC).

Las aplicaciones Web en un inicio fueron simplemente CGIs corriendo en un servidor Web con escasos recursos, pronto la aparición de las paginas dinámicas (Servlets, JSP, etc.), hicieron que se idearan y construyeran aplicaciones Web más complejas lo cual naturalmente incremento la carga en el servidor, anexado las consultas cada vez más complejas a la base de datos, provocando que se pensara en la separación de la presentación y acceso a datos (lo cual se puede realizar mediante MVC).

El Modelo Vista Controlador (MVC), es a grandes rasgos:

1. El Modelo es todo acceso a Base de Datos, y funciones que controlan la integridad de los datos y una pequeña "lógica de negocio".
2. La Vista es la cara de la aplicación, la presentación visual de los datos, o la transformación de la misma.
3. El Controlador, a simple vista el Modelo o acceso a datos y la Vista o capa de presentación requieren un eslabón que parece perdido, pues el controlador se encarga de esta necesaria función, enlazar el acceso a datos con la presentación de los mismos.

MVC es un patrón de diseño aportado originariamente por el lenguaje SmallTalk a la Ingeniería de Software. El paradigma MVC consiste en dividir las aplicaciones en tres partes:

• Controlador: es el encargado de redirigir o asignar una aplicación (un modelo) a cada petición; el controlador debe poseer de algún modo, un "mapa" de correspondencias entre peticiones y respuestas (aplicaciones o modelo) que se les asignan.

• Modelo: seria la aplicación que responde a una petición, es la lógica de negocio a fin de cuentas.

• Vista: Una vez realizadas las operaciones necesarias el flujo vuelve al controlador y éste devuelve los resultados a una **vista** asignada.

Veamos las diferencias que supone éste modelo con los modelos convencionales

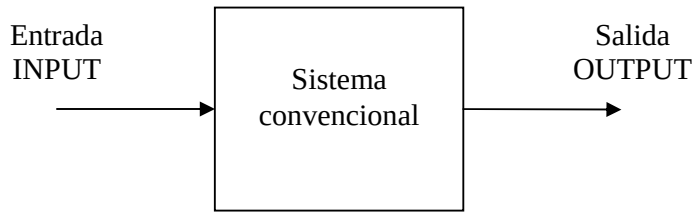

Fig.2.1 Sistema convencional

Si vemos al esquema más básico de programación (Fig.2.1), tenemos una entrada o parámetros que llegan (INPUT), se procesan y se muestra el resultado (OUTPUT).

Para el caso más específico en Web tenemos un modelo muy conocido por todos, aquel que separa la presentación del acceso a datos, en una aplicación Web en Java (JSP para este caso) se esquematizaría de la siguiente manera:

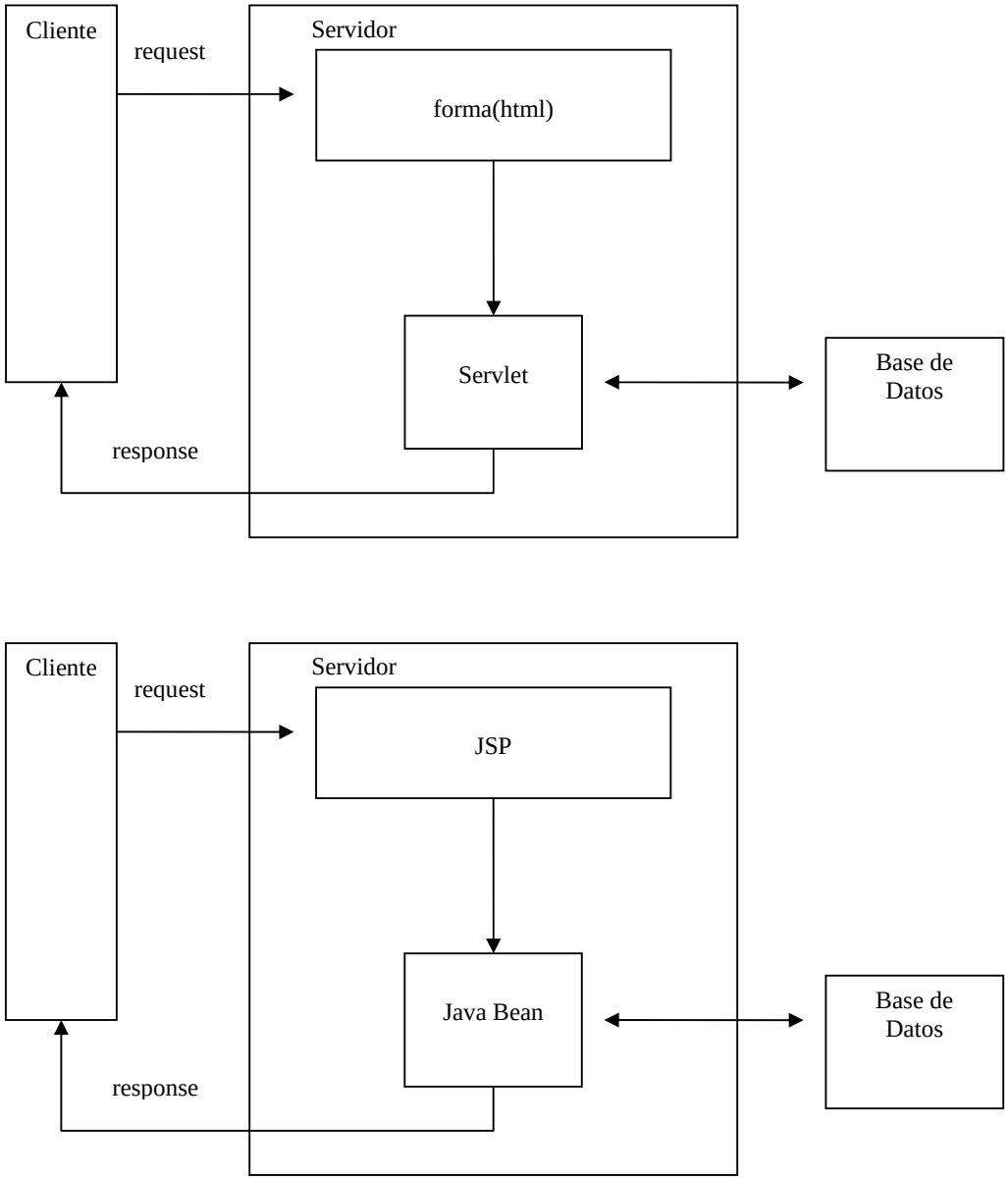

Fig.2.2 Aplicaciones Web convencionales

De acuerdo a este esquema tenemos que procesar muchas peticiones y enviar la misma cantidad de respuestas, hacen que la página soporte demasiado trabajo, para el cual no está diseñada, de igual manera al usarse *scripting** en su forma más pura, hace que el resultado sea totalmente desfavorable. La aplicación se hace cada vez más lenta, con cada petición.

** Los programas escritos mediante lenguajes interpretados se suelen llamar scripts aunque no todos los scripts están hechos en lenguajes interpretados ya que algunos realizan la fase de compilado (de una manera transparente al usuario) como es el caso de JSP.*

En el caso del patrón MVC el procesamiento se lleva a cabo entre sus tres componentes. El Controlador recibe una orden y decide quien la lleva a cabo en el modelo. Una vez que el modelo (la lógica de negocio) termina sus operaciones devuelve el flujo vuelve al controlador y éste envía el resultado a la vista o capa de presentación.

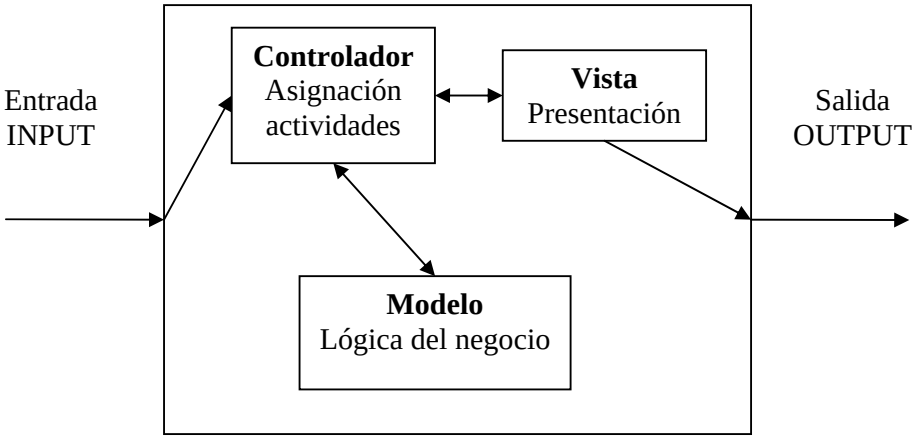

Fig.2.3 Sistema MVC

El Controlador en cierta forma debe tener un registro de la relación entre ordenes que le pueden llegar y la lógica de negocio que le corresponde (es como una operadora de teléfono que recibe una petición y une dos líneas).

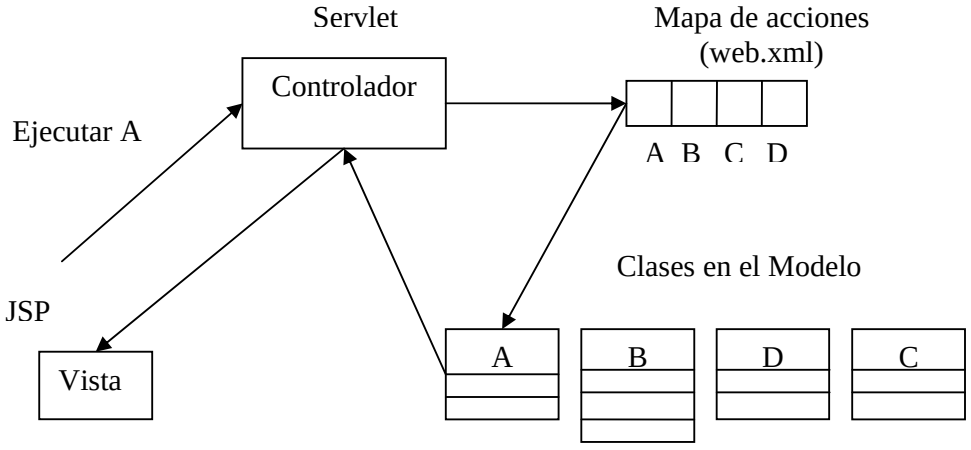

Fig.2.4 Controlador

MVC es un derivado de los modelos convencionales. Aprovecha muchas ideas expresadas y quizás dejadas a medias (Fig 2.2.). El acceso a la Base de Datos, puede seguir controlado por un *Bean* o una “clase java”. El Modelo de petición y respuesta al usuario sufre una transformación, la Vista es en este caso quien se encarga de la respuesta al usuario (JSP´s), mientras que la sobrecarga de tareas, como procesar las peticiones y generar datos de acuerdo a las mismas los realiza el Servlet, un componente medio, que actúa de Controlador entre el modelado de los datos y la Vista. El servlet llama a un *Bean* o “clase java” para realizar la carga más pesada de datos, para entregar a la Vista datos ya procesados según las peticiones del usuario.

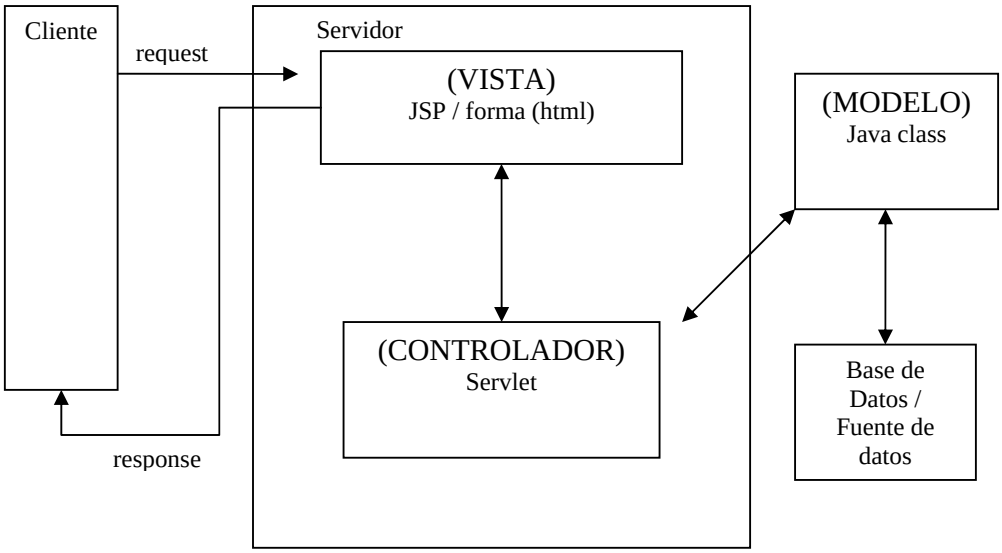

Fig.2.5 Aplicación Web bajo MVC

En este esquema ocurre una separación de la lógica, los datos y la presentación, hasta aquí el MVC sumado a los *patterns* o *frameworks* existentes; Struts, por nombrar a uno de los mas conocidos y de fácil construcción, hacen de MVC una opción más que aceptable y recomendable para el desarrollo de cualquier aplicación Web bajo el lenguaje de programación Java.

2.3.1 Ventajas.

¿Qué ventajas obtenemos de este modelo? Una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multi-lenguaje, distintos diseños de presentación, etc., sin alterar la lógica de negocio.

La separación de capas como presentación, lógica de negocio, acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácil de mantener, lo que al final resulta en un ahorro de tiempo en el desarrollo de posteriores proyectos.

2.3.2 MVC y JAVA

En el lenguaje Java disponemos de unas clases muy sencillas para implantar el modelo MVC por ejemplo: la clase *Observer*, Observable del paquete *util.* Aunque esa implementación del MVC con esas clases se podría hacer a un nivel muy simple de interacción entre unas pocas clases.

Java represento (o representa) uno de los pilares más importantes en desarrollo de aplicaciones distribuidas usando *scripting* del lado del servidor, que se conozca hasta la actualidad. Es por eso que el Modelo MVC esta orientado al lenguaje más abierto y portable que se haya construido [Freeman 2004] [Basham 2004]. En al ámbito del desarrollo Web se siguen unas pautas que tratan *más o menos* de conseguir un desarrollo estructurado de las aplicaciones, donde la verificación de sesión se centraliza y cada caso de uso se distingue claramente. En muchos desarrollos Web se diseña consciente o inconscientemente siguiendo este patrón, por tanto la adopción del modelo Struts no debiera ser un problema mayor. Además hay que tener en cuenta que Struts nos da parte del trabajo hecho, funciona correctamente, y nos da ciertos extras. Teniendo en cuenta además las aportaciones de grupos de desarrollo (taglibs nuevos, etc.), la adopción de esta plataforma puede resultar muy interesante [Hightower 2004].

2.4 STRUTS

Struts es un *framework* para aplicaciones Web java que implementa el modelo MVC. Realmente lo que provee es un conjunto de clases y TAG-LIBS que conforman el Controlador, la integración con el Modelo (o lógica de negocio) y facilitan la construcción de vistas. Naturalmente, el Modelo o lógica de negocio es la parte que se debe desarrollar. Por eso Struts es una plataforma sobre la que montamos la lógica de negocio, y esta plataforma nos permite dividir la lógica de la presentación entre otras cosas [Hightower 2004].

Supongamos que tenemos una aplicación Web que consiste en un formulario que recoge datos y los manda al servidor para su proceso. Este ejemplo tan simple necesitaría implementar unos 6 archivos (2 de ellos JSP). Suena mal pero es el precio que hay que pagar por una aplicación bien estructurada.

Utilizando Struts nunca se llega a una página de la capa de presentación directamente. Esto es, en la URL(*Universal Resource Locator*) nunca se llega a una página JSP o HTML*(Hyper Text Markup Language)* a través de su nombre. De eso se trata el MVC, la presentación está separada en otra capa.

En este entorno, se debe invocar una acción o aplicación que debe estar mapeada en Struts: una acción se corresponderá con una clase Java (heredera de la clase *Action* de Struts). El mapeo de acciones y clases se especifica en un archivo de importancia vital: struts-config.xml. Ahí se especifican todas las relaciones entre acciones y clases, formularios y clases, acciones y JSPs de presentación, que globalmente conforman el "mapa" de la aplicación.

2.5 Server Pages

2.5.1 Servlet

De acuerdo a lo descrito en los libros "Head First Servlets and JSP" y "More SERVLETS and JAVASERVER PAGES" [Basham 2004][Hall 2001] los Servlets son la respuesta de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y construyen páginas Web. Construir páginas Web mediante Servlets o CGIs es útil (y comúnmente usado) por ciertas de razones:

- **La página Web está basada en datos enviados por el usuario**. Por ejemplo, las páginas de resultados de los motores de búsqueda se generan de esta forma, y los programas que procesan pedidos desde sitios de comercio electrónico también.
- **Los datos cambian frecuentemente**. Por ejemplo, un informe sobre el tiempo o páginas de cabeceras de noticias podrían construir la página dinámicamente, quizás devolviendo una página previamente construida y luego actualizándola.

Las páginas Web que usan información desde bases de datos corporativas u otras fuentes. Por ejemplo, se puede hacer una página Web de una tienda en línea que liste los precios actuales y el número de artículos en almacén.

2.5.2 JSP

Un JSP (*Java Server Pages*) es una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente. Muchas páginas Web que están construidas con programas CGI son casi estáticas, con la parte dinámica limitada a muy pocas localizaciones. Pero muchas variaciones CGI, incluyendo los servlets, hacen que generemos la página completa mediante nuestro programa, incluso aunque la mayoría de ella sea siempre lo mismo. JSP nos permite crear dos partes de forma separada [Hall 2001] [Basham 2004].

2.5.3 Ventajas Server Pages

¿Por qué utilizar “Servlet/JSP”?

Los Servlets Java son más eficientes, fáciles de usar, más poderosos, más portables, y más baratos que el CGI tradicional y otras muchas tecnologías del tipo CGI [Hall 2001] [Hall 2003].

- **Eficiencia**. Con un CGI tradicional, se arranca un nuevo proceso para cada solicitud HTTP. Si el programa CGI hace una operación relativamente rápida, la sobrecarga del proceso de arrancada puede dominar el tiempo de ejecución. Con los Servlets, la máquina Virtual Java permanece arrancada, y cada petición es manejada por un *thread* Java de peso ligero, no un pesado proceso del sistema operativo. De forma similar, en CGI tradicional, si hay N peticiones simultáneas para el mismo programa CGI, el código de este problema se cargará N veces en memoria. Sin embargo, con los Servlets, hay N *threads* pero sólo una copia de la clase Servlet. Los Servlets también tienen más alternativas que los programas normales CGI para optimizaciones como los “cachés” de cálculos previos, mantener abiertas las conexiones de bases de datos, etc.

- **Conveniencia**. Poder utilizar un lenguaje familiar, los Servlets tienen una gran infraestructura para análisis automático y decodificación de datos de formularios HTML, leer y seleccionar cabeceras HTTP, manejar “cookies”, seguimiento de sesiones, y muchas otras utilidades.

- **Potencia**. Los Servlets Java nos permiten fácilmente hacer muchas cosas que son difíciles o imposibles con CGI normal. Por algo, los servlets pueden hablar directamente con el servidor Web. Esto simplifica las operaciones que se necesitan para buscar imágenes y otros datos almacenados en situaciones estándares. Los Servlets también pueden compartir los datos entre ellos, haciendo las cosas útiles como almacenes de conexiones a bases de datos fáciles de implementar. También pueden mantener información de solicitud en

solicitud, simplificando cosas como seguimiento de sesión y el caché de cálculos anteriores.

- **Portable**. Los Servlets están escritos en Java y siguen un API bien estandarizado. Consecuentemente, los servlets escritos, digamos en el servidor I-Planet Enterprise(linux), se pueden ejecutar sin modificarse en Apache, Microsoft IIS, o WebStar. Los Servlets están soportados directamente o mediante *plug-in* en la mayoría de los servidores Web.

- **Barato**. Hay un número de servidores Web gratuitos o muy baratos que son buenos para el uso "personal" o el uso en sitios Web de bajo nivel. Sin embargo, con la excepción de Apache, que es gratuito, la mayoría de los servidores Web comerciales son relativamente caros. Una vez que tengamos un servidor Web, no importa el costo de añadirle soporte para Servlets (si no viene pre-configurado para soportarlos) es gratuito o muy barato.

Ventajas de JSP con respecto a:

- **Active Server Pages (ASP)**. ASP es una tecnología similar de Microsoft. Las ventajas de JSP están duplicadas. Primero, la parte dinámica está escrita en Java, no en Visual Basic, otro lenguaje específico de MS, por eso es mucho más poderosa y fácil de usar. Segundo, es portable a otros sistemas operativos y servidores Web [Hall 2003].

- **Servlets**. JSP no nos da nada que no pudiéramos en principio hacer con un servlet. Pero es mucho más conveniente escribir (y modificar) HTML normal que tener que hacer un billón de sentencias “println” que generen HTML. Además, separando el formato del contenido podemos dejar a los expertos en diseño de páginas Web construir el HTML, dejando espacio para que programadores de servlets inserten el contenido dinámico.

- **Server-Side Includes (SSI)**. SSI es una tecnología ampliamente soportada que incluye piezas definidas externamente dentro de una página Web estática. JSP es mejor porque nos permite usar servlets en vez de un programa separado para generar las partes dinámicas. Además, SSI, realmente está diseñado para inclusiones sencillas, no para programas "reales" que usen formularios de datos, hagan conexiones a bases de datos, etc.

- **JavaScript**. JavaScript puede general HTML dinámicamente en el cliente. Ésta es una capacidad útil, pero sólo maneja situaciones donde la información dinámica está basada en el entorno del cliente. Con la excepción de las “cookies”, el HTTP y el envió de formularios no están disponibles con *JavaScript.* Y, como se ejecuta en el cliente, *JavaScript* no puede acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etcétera [Hall 2003].

2.6 Sistema de Gestión de Base de Datos (SGBD)

Una base de datos es una colección de datos relacionados (no independientes). Modelan y abstraen los objetos de una parte del “mundo real”.Sirven de soporte a las aplicaciones. Se debe tener claro que una Base de Datos es independiente de los programas que la utilizan. Es decir, si modificamos un programa no afecta los datos o la organización de cierta base de datos, ya que estos datos sólo se sirven de ésta para realizar una variedad de consultas, de igual manera las bases de datos se sirven de los programas para actualizar su contenido lo cual es posible a través de los SGBD.

Los SGBD son aquellos programas que permiten a los usuarios crear y mantener una base de datos. También proveen procedimientos para la definición, construcción y manipulación (búsqueda y selección) de información. Un SGBD se ocupa de organizar la información sobre memorias secundarias. Algo que debe considerarse para que una base de datos sea de utilidad es que tenga poca redundancia o repetición de datos, ya que esto provoca inconsistencias a la hora de hacer consultas. Esta es la razón principal

por la que se utilizan bases de datos, ya que si se pudiera almacenar y recuperar mágicamente toda la información que constituye el universo estaría resuelto el problema. Sin embargo, es necesario dar una estructuración adecuada a la información para evitar los problemas de redundancia y duplicidad. Esta estructuración significa identificar y definir las entidades que constituyen el universo en el que se va trabajar. Lo anterior se puede establecer antes de crear la base de datos. Primero comprendiendo las necesidades y problemas de los usuarios finales (esquema externo), continuando con el diseño de un esquema conceptual (por ejemplo un diagrama "entidad – relación"), después con la concepción de un esquema lógico (por ejemplo: relaciones 3FN) y finalmente con la implementación [Silberschatz 2003].

2.7 Sistemas de aplicación de exámenes relacionados

Continuación daré un explicación breve de un par de aplicaciones relacionadas y a mi parecer las más significativas. La primera es la que actualmente es utilizada por la UDLA-P la cual además se pretendió mejorar. La segunda aplicación es el "blackboard" aplicación utilizada por algunas instituciones del sistema ITESM.

2.7.1 Aplicación de exámenes de ubicación y admisión UDLA-P

En la UDLA-P no es nada nuevo la aplicación de exámenes vía Web. La manera que actualmente es utilizada es eficiente pero consta de un proceso largo y complejo para su realización. Dicho proceso se divide en tres partes. La primera parte es la evaluación del alumno la cual es realizada a través de un CGI donde están contenidas todas las preguntas y respuestas. Dicho CGI además de ser el encargado de presentar y evaluar dicho examen genera un archivo tipo "txt" en el cual se guardan los resultados del alumno. La segunda parte del proceso es realizada mediante una aplicación independiente del CGI y debe ser ejecutada por un administrador. Dicha aplicación consiste en abrir todos los archivos generados por el CGI descrito y decodificarlos para actualizar la base de datos de acuerdo a los resultados de los alumnos. Por último, la tercera parte consiste en otra aplicación encargada de eliminar los archivos generados por el CGI. Al igual que la anterior esta debe ser ejecutada por un administrador de manera independiente. Por todo lo anterior notamos que sólo existen dos tipos de usuarios (administrador y alumno). El alumno únicamente puede tomar el examen,

mientras que el administrador levanta el servicio, ejecuta las aplicaciones del proceso de evaluación y captura, y es el único capacitado para modificar el código del CGI en caso de que se requiera cambiar las preguntas.

2.7.2 Blackboard

Esta aplicación de aprendizaje en línea fue desarrollada en *Cornell University*. Consiste de varias funcionalidades entre ellas la aplicación de exámenes vía Web. Dicha funcionalidad es administrada totalmente por el profesor que imparta el curso. Es decir, el se encarga de inscribir a los alumnos del curso, editar las preguntas y respuestas, programar los exámenes (hora de aplicación y limite de tiempo) y evaluar las preguntas las preguntas tipo ensayo. El proceso de aplicación de examen a diferencia del anterior no requiere de la intervención de un administrador para el registro de los resultados. Pero desafortunadamente cuando varios alumnos ya han aplicado un examen éste no puede ser modificado. La retroalimentación de la calificación obtenida por el alumno es rápida y una vez obtenida la calificación se registra para evitar que tome nuevamente cierto examen.

2.8 Discusión

Para esta parte es preciso decir que el sistema desarrollado no es tan complejo como “blackboard” y sólo es comparable respecto a la funcionalidad para aplicar exámenes. Por otro lado también debe mencionarse que este sistema además de estar enfocado en la comunidad de la UDLA-P también puede llegar a implementarse en instituciones que no tengan los recursos para adquirir un sistema como es “blackboard”. Para finalizar éste capítulo se puede decir que se ha definido el marco teórico sobre el cual éste trabajo de tesis se encuentra fundamentado.