Capítulo III. Arquitectura del sistema.

Debido a las necesidades de hacer al sistema accesible vía web, se decidió seguir la arquitectura *Model View Controller*, la cual aumenta las capacidades de la arquitectura conocida como Modelo Cliente/Servidor y a su vez organiza todos los componentes que conforman al sistema según su función.

Al implementar el sistema siguiendo la arquitectura mencionada anteriormente se obtienen todas las características más utilizadas en la actualidad para el desarrollo de aplicaciones web y las ventajas de los sistemas distribuidos.

En la figura 3.1 se muestra el diagrama de bloques preliminar del sistema, en este se puede ver claramente la parte de programación como la parte de infraestructura que se utilizará para su desarrollo.

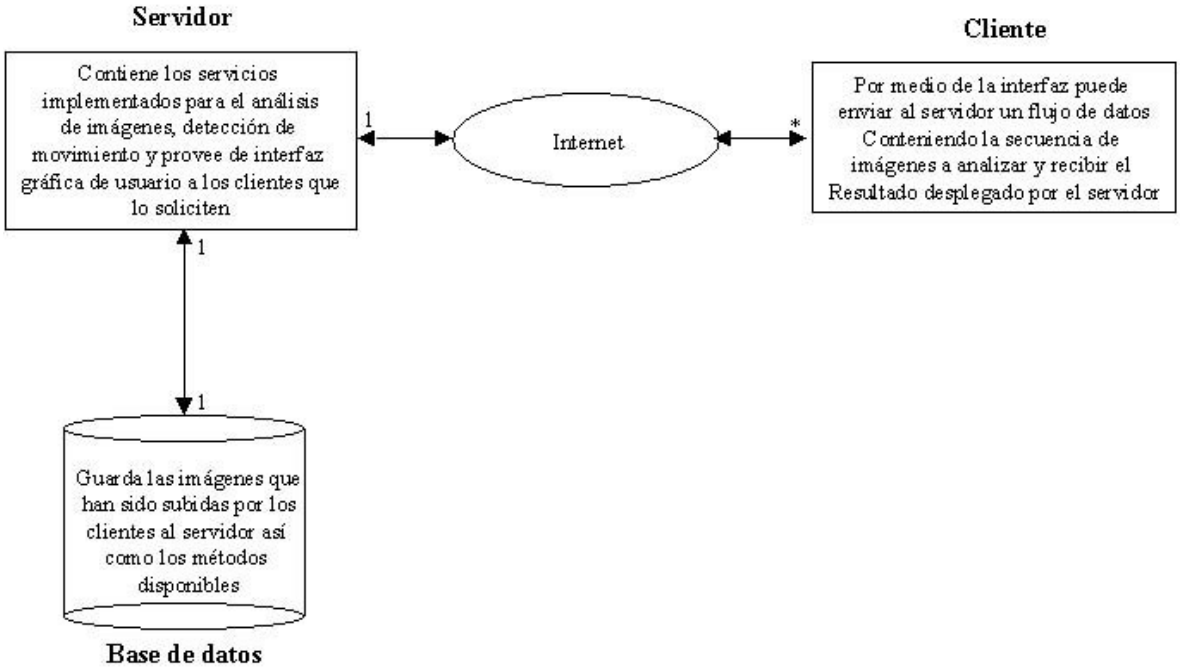

Figura 3.1 Diagrama de bloques del sistema.

3.1 Modelo Cliente/Servidor.

La arquitectura Cliente/Servidor es el modelo más utilizado tanto para aplicaciones web como para proporcionar servicios de alto nivel o paginas web dinámicas. Es un modelo para el desarrollo de aplicaciones en el que las operaciones a realizar se dividen en procesos independientes que cooperan entre sí para proporcionar un servicio o una respuesta.

Este tipo de arquitectura consta de tres componentes: al proceso el cual inicia el diálogo o la solicitud de un servicio se le denomina Cliente, generalmente son computadoras personales o estaciones de trabajo con capacidades limitadas para el procesamiento de información. El proceso que atiende a las solicitudes realizadas por los clientes se le denomina Servidor, las cuales son computadoras más poderosas que los Clientes. El último componente de esta arquitectura es el medio físico mediante el cual se comunican los Clientes con el Servidor, éstos medios son generalmente redes de área local (LAN) o redes de área amplia (WAN) [Murillo, 1997].

También en [Murillo, 1997] se resaltan las principales funciones tanto de los clientes como de los servidores. Los Clientes son los procesos que interactúan con el usuario, generalmente lo hacen de forma gráfica mediante Interfaces Gráficas de Usuario (GUI's) y son estos últimos quienes envían peticiones al Servidor y reciben la respuesta. Algunas de las funciones de los Clientes son:

- Enviar solicitudes de procesamiento a Servidores.
- Proporcionar información necesaria al Servidor para atender las peticiones.
- Manejo de Interfaces Gráficas de Usuario.

- Captura y validación de datos.

Por otra parte, los servidores son quienes computan las solicitudes hechas por los Clientes y regresan el resultado del cómputo de datos a cada Cliente. El servidor que se utilizará para el desarrollo del sistema es el Web Server Apache Tomcat 4. Algunas de las funciones del Servidor son:

- Recepción de solicitudes de Clientes.
- Computo de datos solicitados por el Cliente.
- Acceso a Bases de Datos.

Las principales características de la arquitectura Cliente/Servidor son:

- El Servidor presenta a todos los Clientes una interfaz única y bien definida.
- El Cliente no necesita saber la lógica y el funcionamiento del Servidor.
- El Cliente no depende de nada del Servidor, ni su ubicación, ni su tipo de equipo, ni su sistema operativo.
- Los cambios en el Servidor implican pocos o ningún cambio a los Clientes.
- Un Servidor puede atender a uno o más Clientes

La arquitectura cliente servidor funciona bajo la estructura mostrada en la Figura 3.2.

Figura 3.2 Arquitectura Cliente/Servidor.

3.1.1 Arquitectura *Model-View-Controller.*

Éste tipo de arquitectura aún cuando fue diseñada durante los años 70's, es la más utilizada en la actualidad para la creación de sistemas en los que se pueden generar varias vistas de los datos según la operación que se desee realizar a ellos [Wheeler, 1996].

Como se mencionó con anterioridad en éste capítulo, éste tipo de arquitectura es muy similar a la arquitectura Cliente/Servidor, con la diferencia que *Model View Controller* sigue un patrón de diseño en donde son separados los componentes de la aplicación según su función en las tres diferentes clasificaciones que lo componen.

Modelo (Model).

Es el módulo el cual conoce todos los datos necesarios para ser desplegados y las operaciones que pueden realizarse en la aplicación, sin embargo, no conoce nada acerca de las interfaces gráficas de usuario GUI's, la manera y forma en que son desplegados dichos datos.

El modelo es la parte invariable de la aplicación, es decir, la parte que nunca cambia. En el caso del paradigma orientado a objetos, el modelo está compuesto por las clases que modelan y soportan el problema a resolver.

Los datos conocidos por el modelo son manipulados por métodos que son totalmente independientes de las interfaces gráficas de usuario [Deacon, 1995].

Vista (View).

El módulo de la vista utiliza los métodos de consulta del modelo para obtener los datos necesarios para posteriormente mostrarlos al usuario final.

La vista sólo tiene la función de desplegar los datos en cualquier forma, pero no puede hacer operaciones extras a éstos datos, lo único que puede hacer es almacenarlos temporalmente para desplegarlos en el momento necesario.

Dado que la vista no tiene facultades de hacer operaciones o modificaciones a los datos, los objetos del modelo no se deben relacionar directamente con los objetos de la vista. La manera adecuada de comunicar a la vista que los datos del modelo han cambiado se realiza por medio de otro módulo que se llama controlador [Deacon, 1995].

Controlador (controller).

La función principal del controlador es verificar las operaciones que los usuarios solicitan hacer tanto para el modelo como para la vista. En aplicaciones web éstas peticiones son de tipo HTTP GET y HTTP POST.

En una interfaz gráfica de usuario, el controlador es quien recibe las acciones del usuario final ya sea por medio del mouse o del teclado. Éstas acciones son tomadas como los métodos de manipulación de datos del modelo [Deacon, 1995]

Puede darse el caso en que el controlador deba interactuar con la vista sin pasar antes por el modelo, por lo que el diagrama de bloques de la arquitectura Model View Controller es la siguiente:

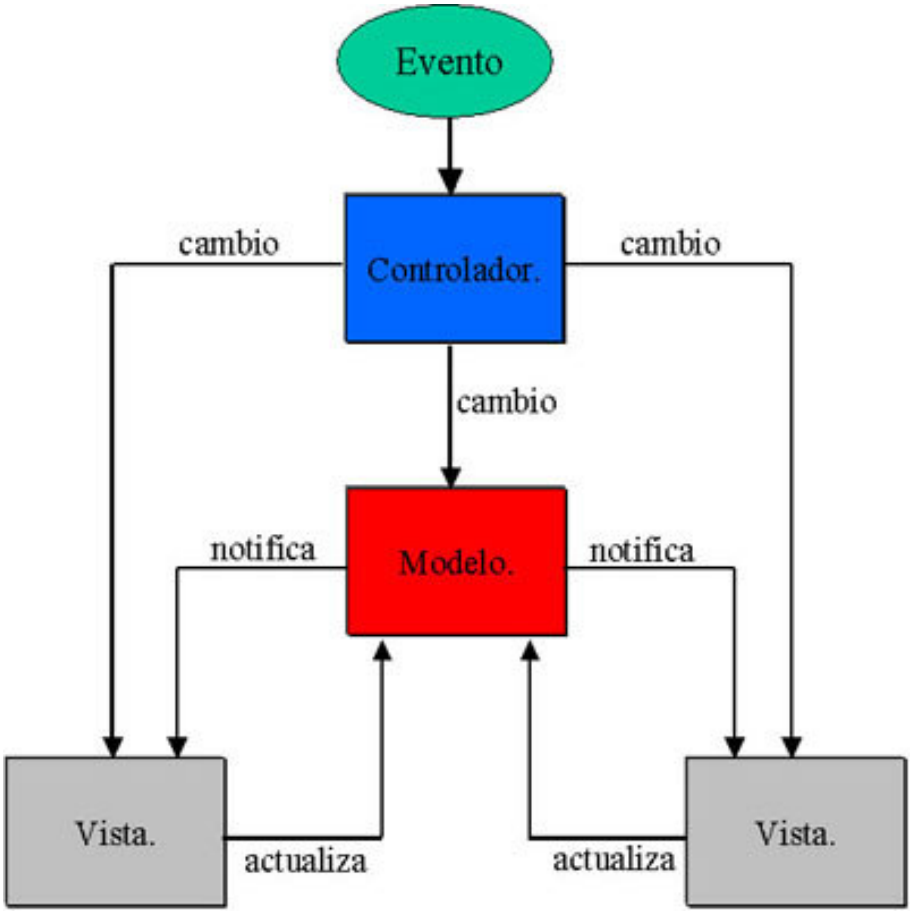

Figura 3.3 Arquitectura Model-View-Controller

3.2 Configuración del sistema.

Para el correcto funcionamiento de la aplicación y que pueda seguir el paradigma de la arquitectura Model-View-Controller es necesario hacer uso de un Servidor Web, para el caso específico de ésta aplicación, dicho Servidor Web será Apache Tomcat 4. También se necesitará un manejador de bases de datos para el correcto almacenamiento de las imágenes a procesar y los métodos disponibles para el procesamiento de imágenes, por lo que se utilizará el DBMS MySQL.

Antes de poder instalar el Servidor Web será necesario instalar un compilador de soporte, el cual será el Java Development Kit 1.4.2.

La instalación de todo el software mencionado anteriormente sólo se instalará en la máquina que servirá como Servidor, ya que todas las demás máquinas que se conecten al servidor no necesitan de éste software, sin embargo, si se necesitara el Plug-in de Java 2, éste está disponible en http://www.java.sun.com.

3.2.1 Instalación de Java 2.

El Java 2 SDK es el entorno para desarrollo de aplicaciones, applets, servlets y demás componentes usando el lenguaje de programación Java. Además, es el compilador de soporte necesario para poder instalar el Servidor Web Apache Tomcat 4 [Lemmy, 1999].

En la actualidad existen varias versiones del J2SDK disponibles, sin embargo, para el desarrollo del sistema se utilizará la versión Java 1.4.2, debido a que es una versión reciente y estable la cual se puede descargar gratuitamente de http://www.java.sun.com/j2se/1.4.2/download.html.

Java es soportado por diversas plataformas, para verificar las posibles plataformas que lo soportan se puede acceder a la página http://www.java.sun.com/j2se/1.4.2/system-configurations.html.

Una vez que se verificó si la plataforma soporta el software y habiendo escogido la versión a instalar, se deben verificar los requerimientos mínimos necesarios para poder hacer la instalación. En éste caso se instalará en una computadora PC, para la cual los requerimientos mínimos son los siguientes, procesador Intel o 100% compatible, Pentium I

a 166MHz o superior, 32 Mb de memoria RAM o superior y 70 Mb de espacio libre en disco[Sun Microsystems, 2004][Adams, 2004].

Una vez instalado el J2SDK se debe indicar al compilador por medio de las variables de entorno la ubicación de las librerías necesarias. Para hacer lo anterior se crea una variable de entorno llamada CLASSPATH como se muestra en la siguiente figura:

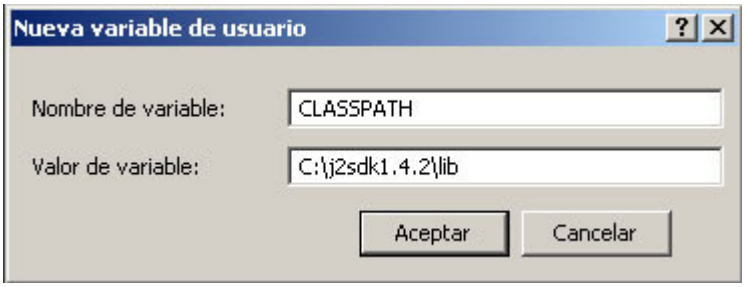

Figura 3.4 Classpath para J2SDK

De la misma manera se debe de indicar al sistema operativo la ubicación de los archivos ejecutables de Java, como se muestra en la siguiente figura:

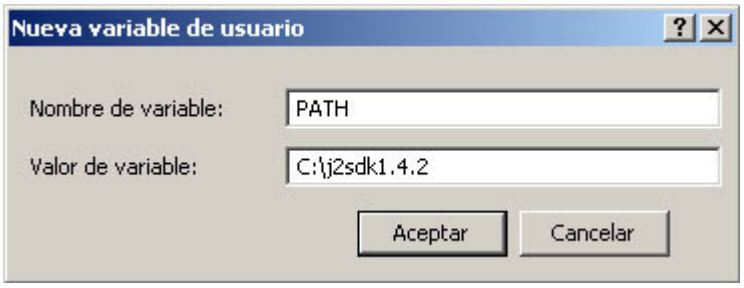

Figura 3.5 Path para J2SDK

Para finalizar, se debe reiniciar el equipo para que los cambios tengan efecto. Posteriormente, se abre una consola de MS-DOS y se da el comando *java –version* para verificar si todo está bien, de ser así, debe de desplegar un mensaje como el siguiente:

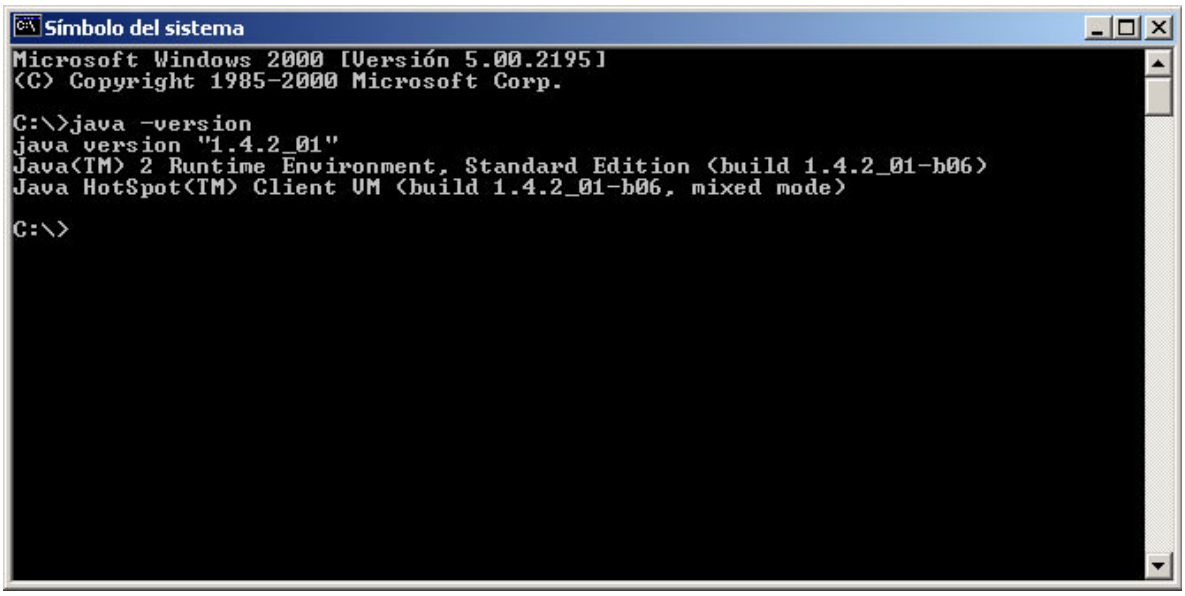

Figura 3.6 Comprobación de instalación de J2SDK

Una vez instalado correctamente el compilador de soporte para el Servidor Web, se procede a la instalación de éste.

3.2.2 Instalación de Apache Tomcat 4.

Como se mencionó con anterioridad en este capítulo es necesario el uso de un Servidor Web que sea el encargado de recibir las peticiones de los clientes, procesarlas y hacer la respuesta de la petición a cada cliente.

En la actualidad existen muchos productos que tienen esa función, sin embargo, el más utilizado es el Apache Tomcat, debido a que es gratuito, confiable, muy rápido y soporta las tecnologías Servlet y JSP, es por esta razón que Apache Tomcat será el Servidor Web que se utilizará para el desarrollo del sistema [Hall, 2000].

Existen diversas versiones de Apache Tomcat, estas versiones así como todos sus detalles pueden ser consultadas en Internet en la página http://jakarta.apache.org. Para efectuar esta tesis se utilizará la versión 4.0.4, la cual esta disponible en http://jakarta.apache.org/site/binindex.cgi

En [Adams, 2004] se presentan los pasos detallados de la instalación del Servidor Web así como las variables de entorno que se necesitan generar. Una vez descargado e instalado el software, es necesario generar dos nuevas variables de entorno, la primera sirve para darle al servidor la ubicación de los archivos de configuración, esta variable lleva el nombre de CATALINA_HOME y el valor de la carpeta en donde se instaló Tomcat, este paso se muestra a continuación:

Figura 3.7 Variable de entorno CATALINA_HOME

La segunda variable de entorno le indica al servidor la ubicación del compilador de soporte, en éste caso el J2SDK, esto se muestra en la siguiente figura:

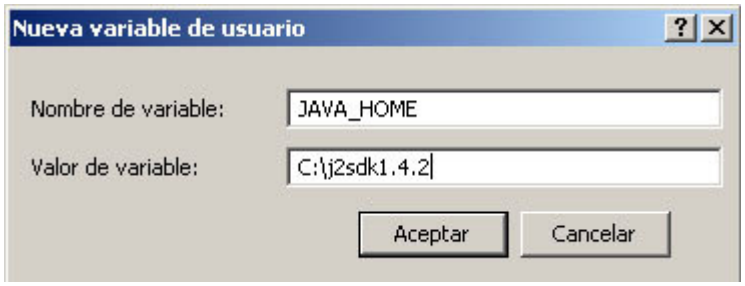

Figura 3.8 Variable de entorno JAVA_HOME

Una vez dadas de alta las dos variables de entorno, es posible levantar y bajar el servidor por medio de los scripts *startup* y *shutdown* respectivamente, los cuales se encuentran ubicados de la carpeta bin dentro del directorio de instalación de Tomcat [Adams, 2004] [The Jakarta Site – Apache Tomcat, 2004].

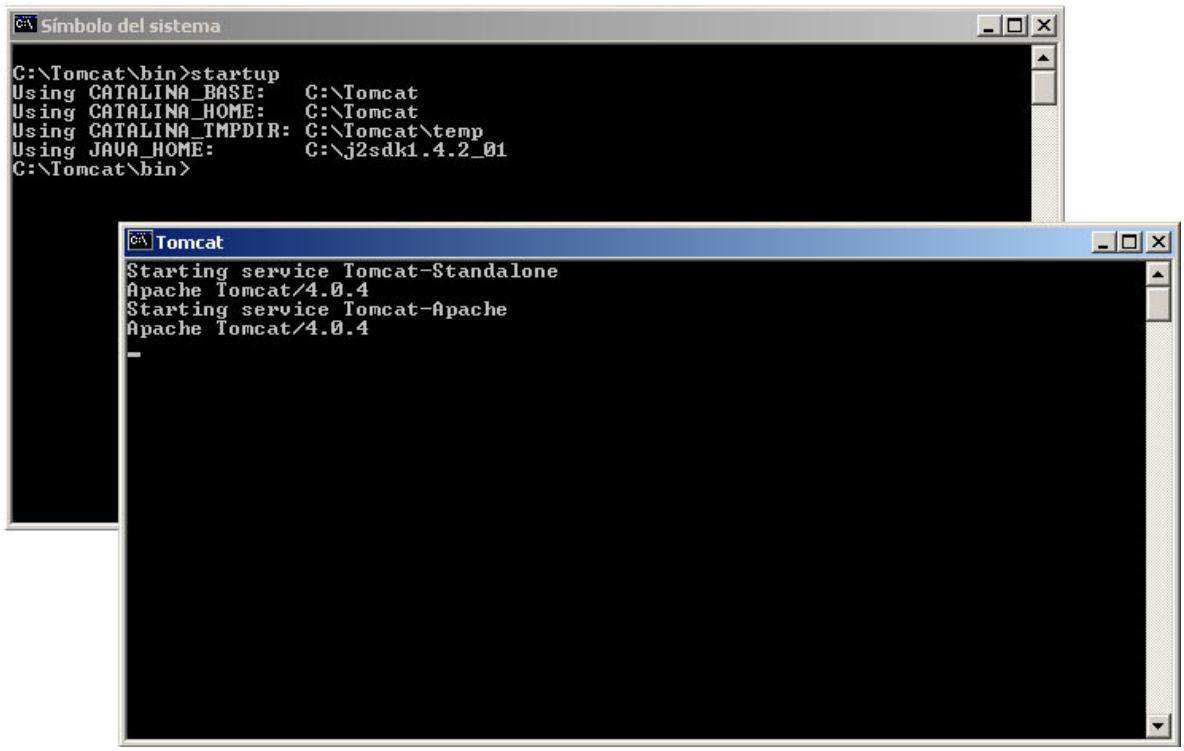

Figura 3.9 Comprobación de scripts

Si se obtuvo el resultado de la figura anterior, se puede comprobar que el servidor esta listo para recibir peticiones, verificando desde un navegador la dirección web http://localhost:8080/index.html y deberá abrir la página de bienvenida de Tomcat como se muestra a continuación:

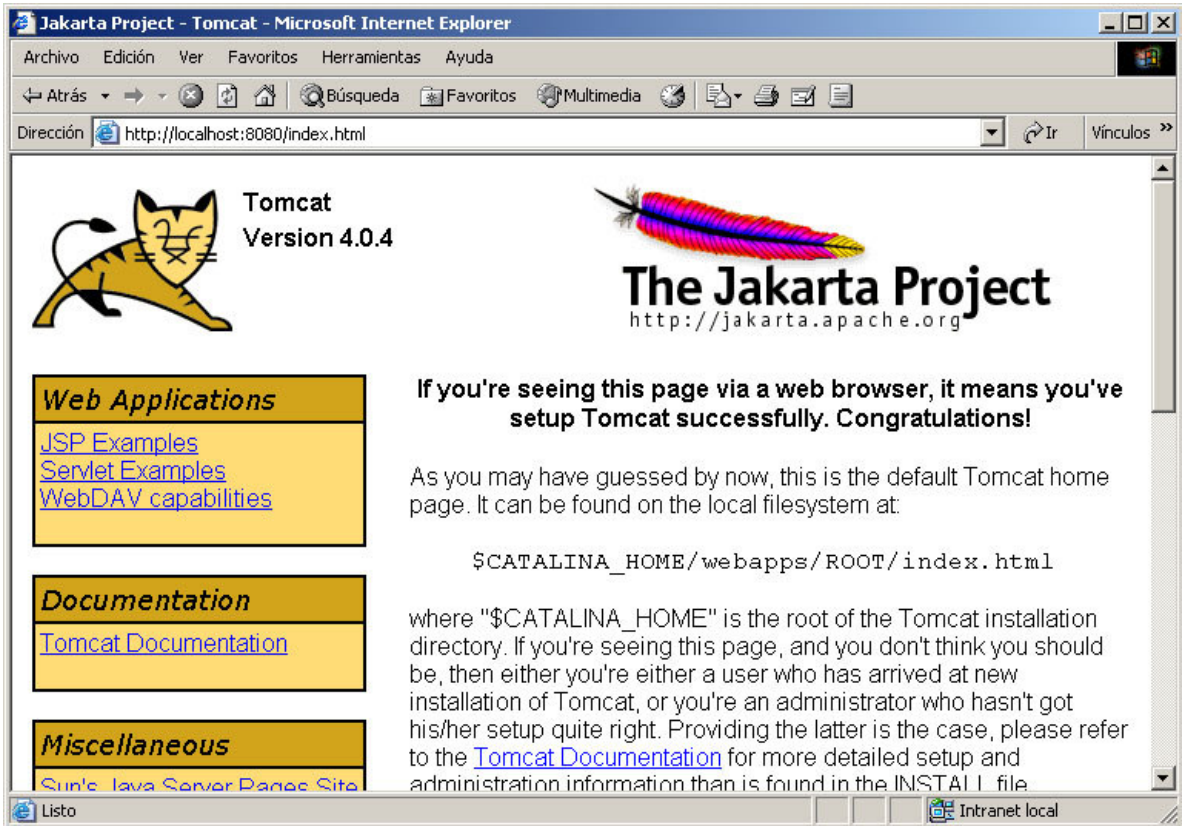

Figura 3.10 Comprobación de ejecución de Tomcat

3.2.3 Instalación de MySQL.

MySQL es un manejador de bases de datos relacionales de código abierto. Durante los últimos años, MySQL ha tomado importancia en las aplicaciones web debido a su rapidez de procesamiento, confiabilidad, flexibilidad y principalmente a que es un software gratuito. Este manejador de bases de datos utiliza el lenguaje de consultas estructurado SQL para las consultas a las bases de datos [Maslakowski, et al., 2001] [MySQL Site, 2004].

Como se mencionó con anterioridad, MySQL es de código abierto y gratuito, es decir, se puede obtener directamente de la página de Internet oficial del sitio. Existen varias

versiones disponibles, la más reciente es la versión 5, sin embargo, para la implementación del sistema se utilizará la versión estable más reciente la cual es la 4.0.16. Toda la información sobre las versiones y descargas se encuentran en http://www.mysql.com/download.

MySQL al igual que muchos otros sistemas de administración de bases de datos relacionales funcionan como un servicio o daemon, el cual es un programa que se ejecuta continuamente en segundo plano, en el caso de MySQL es *mysqld* el cual se puede verificar su ejecución en Windows por medio del administrador de tares [Maslakowski, et al., 2001].

Una vez instalado MySQL, se deba de levantar el servidor para poder ejecutar directamente desde una terminal las consultas a las bases de datos. Para levantar el servidor, se debe ir a la carpeta *bin* dentro de la carpeta de instalación y dar doble clic en el programa *winmysqladmin*. Una vez levantado el servidor de MySQL se puede ver su funcionamiento por medio de una imagen de un semáforo en la barra de tareas como se muestra en la figura 3.10 [Adams, 2004].

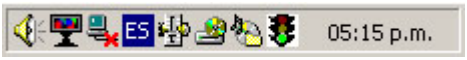

Figura 3.11 verificación de instalación de MySQL

Este punto marca el final de la instalación de MySQL en la computadora, para verificar si la instalación fue satisfactoria, se debe de abrir una terminal de MS-DOS y cambiarse al directorio *bin* de la carpeta de instalación y ejecutar el *script mysql*. Si la instalación es correcta se debe de obtener lo siguiente:

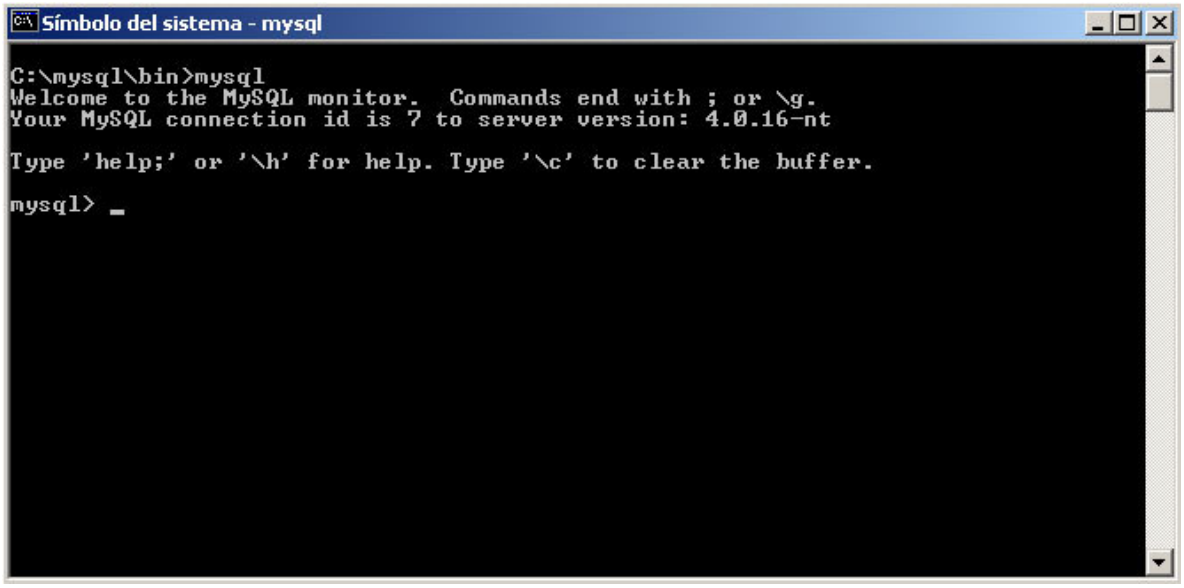

Figura 3.12 Verificación de ejecución de MySQL

Teniendo instalado MySQL, es necesario dar de alta otra variable de entorno para darle a conocer al compilador de Java la ubicación del controlador que necesita Java para comunicarse con MySQL, esta variable se declara de la siguiente manera:

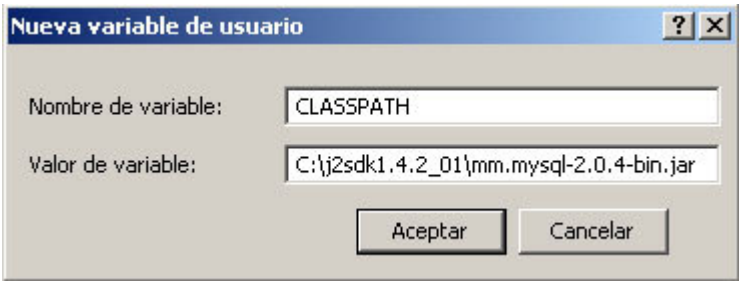

Figura 3.13 Classpath para el controlador de MySQL

Teniendo todo lo antes mencionado, ya es posible hacer la comunicación entre programas escritos en Java y las bases de datos que sean administradas por MySQL.

3.3 Conclusiones.

En éste capítulo se define la arquitectura que se seguirá para la implementación del sistema, la cual como se mencionó con anterioridad será Model View Controller debido a que las ventajas que ofrece este tipo de arquitectura son las que más se apegan a los objetivos de esta tesis.

También se trataron los puntos sobre la configuración del sistema y la instalación del software necesario para el correcto funcionamiento del servidor.

En el siguiente capitulo se tocaran los puntos de diseño del sistema, es decir, lo referente a la ingeniería de software necesaria para la correcta implementación de la aplicación.