SGBD como MYSQL proveen de mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Después de hacer mención de la parte más baja del sistema (manejo de datos), ahora en el siguiente capítulo nos enfocaremos a la arquitectura de éste dando detalle de la estructuración, sin embargo, en la parte del diseño se dará el diagrama de clases que representa nuestras relaciones, entidades, tributos y triggers utilizados en el sistema, para dar mayor transparencia en los procesos y transacciones que se realizan mientras el usuario interactúa con el sistema.

4. Arquitectura del sistema

La aplicación se desarrolló en un lenguaje de programación conocido como java el cual es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. Se eligió este lenguaje debido a varias razones, en primer lugar permite el desarrollo de aplicaciones con un paradigma orientado a objetos, esto posibilita diseñar un software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones, mediante la construcción de clases y objetos, entendiéndose como objeto un paquete que contiene el "comportamiento" es decir el código y el "estado" o los datos. El objetivo básico que se pretende alcanzar al aplicar el paradigma conocido como orientado a objetos, es el de separar aquello que cambia de las cosas que permanecen inalterables. Esta separación en objetos coherentes e independientes en teoría ofrece una base más estable para el diseño de un sistema software razón por la cual se optó por este paradigma.

Otras de las razones por la que java es el lenguaje que se ha escogido para el desarrollo de la interfaz, es que permite algo que se conoce como independencia de plataforma, lo que significa que los programas escritos en este lenguaje pueden ejecutarse igualmente en cualquier tipo de hardware, ya que provee una máquina virtual que ejecuta cualquier código que haya sido escrito en dicho lenguaje, permitiendo que el mismo binario ejecutable se pueda usar en todos los sistemas compatibles con el software Java por lo menos en Windows, GNU/Linux, y Solaris.

Por último cabe mencionar que java es que es un lenguaje que cuanta con una licencia de tipo pública general o GNU por su nombre en inglés General Public License, esta licencia está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre, lo que significa que esta cualquier software que tenga esta denominación respeta la libertad de los usuarios sobre su producto adquirido y por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente, lo que es importante porque todo lo anterior se refleja en un ahorro de costos ya que no se tienen que pagar licencias, como es el caso de otras tecnologías. (GNU Operating System, 2009). Ahora vamos analizar modelos específicos que usaremos en el sistema donde daremos más detalle en el siguiente capítulo.

4.1 MVC

El patrón de arquitectura MVC o Model-View-Controller es un patrón que define la organización independiente del Model referente a los objetos de negocio, la View que es la interfaz con el usuario u otro sistema y finalmente el Controller que funge como

controlador del flujo de trabajo en la aplicación. Cabe señalar que al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- El agregar nuevas vistas según las necesidades que se presenten.
- Agregar nuevas formas de recolectar las ordenes del usuario en caso de que así sea necesario.
- Modificar los objetos bien sea para mejorar el desempeño o para migrar a otra tecnología.

Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones solo se deben hacer en un solo lugar y no en varios como sucedería si tuviésemos una mezcla de presentación e implementación de la lógica del negocio.

Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se paralice. Adicionalmente el patrón MVC propende a la especialización de cada rol del equipo, por tanto en cada liberación de una nueva versión se verán los resultados.

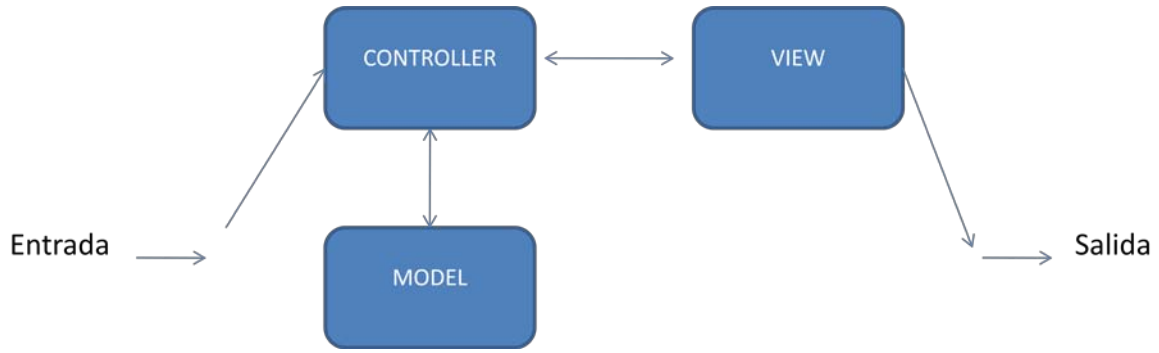

Figura 4.1: Flujo a través del modelo MVC

En la Figura 6.1 se muestra la estructura básica que se encuentra dentro de Java Server Faces, cumpliendo con un estándar más en la programación básica de cualquier aplicación, pero en este caso enfocada a una aplicación web.

En el siguiente capítulo pasaremos de la arquitectura a los complementos que se pueden integrar fácilmente gracias a este modelo de programación.

5. JFreeChart

JFreeChart es una librería para gráficos escrita en Java por lo que su compatibilidad con el sistema está asegurada, como se mencionó en el párrafo anterior esta librería facilita mostrar gráficas, entre las características principales de esta biblioteca se pueden mencionar las siguientes:

- Es un API consistente y bien documentado con soporte para un amplio rango de tipos de gráficas.

- Cuenta con un diseño flexible fácilmente extendible, y la posibilidad de ser usado en tecnologías de servidor como la que planea desarrollar.

- Ofrece soporte para varios tipos de salida, incluyendo componentes Swing, archivos de imagen como PNG y JPEG, y formatos gráficos de vectores incluyendo PDF, EPS y SVG.

- JFreeChart es open source y más especificamente, Software Libre, éste está distribuido bajo la licencia LGPL, que permite el uso en aplicaciones propietarias.

Algunos ejemplos de las graficas que se pueden generar son los siguientes