Sistema para el control de clientes, inmuebles y empleados (CCIE)

Autor: Arturo Huerta Díaz

Tesis presentada para obtener el título de:
Ing. en Sistemas Computacionales

Nombre del asesor:
Misael Madrigal Torres

UVAQ M.R.

UNIVERSIDAD VASCO DE QUIROGA

Escuela de Ingeniería en Sistemas Computacionales

"SISTEMA PARA EL CONTROL DE CLIENTES, INMUEBLES Y EMPLEADOS (CCIE)"

TESIS

Que para obtener el título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Presenta:
Arturo Huerta Díaz

Asesor:

Ing. Misael Madrigal Torres

Acuerdo
16PSU0049F

Clave
LIC000808

Morelia, Mich., Junio de 2008

UVAQ M.R.

UNIVERSIDAD VASCO DE QUIROGA

Escuela de Ingeniería en Sistemas Computacionales

"SISTEMA PARA EL CONTROL DE CLIENTES, INMUEBLES Y EMPLEADOS (CCIE)"

TESIS

Que para obtener el título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Presenta:

Arturo Huerta Díaz

Acuerdo	Clave
16PSU0049F	LIC000808

Morelia, Mich., Junio de 2008

UNIVERSIDAD VASCO DE QUIROGA

Escuela de Ingeniería en Sistemas Computacionales

"SISTEMA PARA EL CONTROL DE CLIENTES, INMUEBLES Y EMPLEADOS (CCIE)"

TESIS

Que para obtener el título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Presenta:

Arturo Huerta Díaz

Asesor:

Ing. Misael Madrigal Torres

Acuerdo	Clave
16PSU0049F	LIC000808

Morelia, Mich., Junio de 2008

Sumario

Índice de figuras

Introducción

Objetivos Específicos

El presente estudio de investigación pretende evitar el papeleo en la empresa, llevar el control de los clientes, inmuebles y empleados de una mejor manera, poder programar citas dentro del sistema y que te avise con tiempo anticipado para que no se olvide; así como también evitar conflictos de las personas que laboran en la empresa, además el personal puede consular los diferentes tipos de inmuebles de manera más rápida y eficiente.

Preguntas de Investigación.

1. ¿Qué necesidades tiene la empresa?
2. ¿Por qué la empresa preferiría un sistema en lugar de lo que ya viene realizando?
3. ¿Qué entorno grafico preferiría la empresa?
4. ¿Necesito implementar una base de datos?
5. ¿Qué paradigma de programación tengo que utilizar?

HIPÓTESIS 1: "La empresa maneja información cada vez mayor y es más difícil llevar el control de los clientes, inmuebles y empleados, generando conflictos de diversos tipos y por lo tanto pérdida de clientes y dinero.
La mejor solución para esto es desarrollar un sistema que lleve el control y manejo de misma".

HIPÓTESIS 2: "En la actualidad la mayoría de las empresas y negocios no tan grandes manejan gran cantidad de información: productos, proveedores, inventarios etc. por lo cual implementan sistemas dando como resultado una manera más eficiente y rápida de controlar la información, con estos antecedentes y debido al desarrollo tecnológico la empresa está obligada a implementarlos ya que si la competencia los está utilizando sería un error quedarse en el pasado, sus competidores tendrían ventaja, por estas razones preferirían un sistema".

HIPÓTESIS 3: "La mayoría de la gente que ha utilizado un sistema en algún negocio me ha mencionado que el sistema cuando no tiene un entorno grafico amigable se les complica el uso.

El entorno grafico juega un papel muy importante dentro del sistema ya que es por donde va navegar el usuario por lo cual el que implementare en el sistema va a ser muy sencillo de manejar, y puesto que va ser utilizado por gente que no tiene muchos conocimientos en computación deberá bastar con que hagan clic en lo que quieren realizar y no van a tener que escribir comandos o cosas que realmente no es necesario que sepan, esto hará que se desplacen con mayor rapidez dentro del sistema y que no tome mucho tiempo en aprender a utilizarlo".

HIPÓTESIS 4: "Si es necesario implementar una base de datos ya en ella se guarda gran cantidad de información y es la parte fundamental de la empresa ya que en ella va estar guardada la información vital con la cual trabaja la empresa.

Ventajas de las bases de dato:

a) *Independencia de datos y tratamiento.*
   - Cambio en datos no implica cambio en programas y viceversa (Menor coste de mantenimiento).

b) Coherencia de resultados.
   - Reduce redundancia: Acciones lógicamente únicas, se evita inconsistencia.

c) Mejora en la disponibilidad de datos
   - No hay dueño de datos (No igual a ser públicos).
   - Ni aplicaciones ni usuarios.
   - Guardamos descripción (Idea de catálogos).

d) Cumplimiento de ciertas normas.
   - *Restricciones de seguridad.*
   - Accesos (Usuarios a datos).
   - Operaciones (Operaciones sobre datos)".

HIPÓTESIS 5: "Una pregunta muy importante es saber en qué lenguaje tengo que programar el sistema ya que se divide en varias partes: base de datos, interfaz grafica, conexión entre los dos. Por lo cual necesito un lenguaje en el que lo pueda hacer de manera más fácil. Por lo tanto el lenguaje que voy a utilizar tiene que tener las siguientes características:

- Que pueda desarrollar la interfaz grafica fácilmente y que pueda utilizar todas las diferentes herramientas que un programa profesional tiene (botones, menús, tips, etc.)
- Orientado a objetos; esto quiere decir que cualquier cosa del mundo puede ser modelada como un objeto así un auto es un objeto, un avión etc. Esto hace que la programación sea más fácil.
- Distribuido; implica que varias computadoras trabajan juntas en la red.
- Que se pueda conectar a una base de datos".

Tipo de estudio

Después de realizar un análisis de los tipos de estudio existentes, se llegó a la conclusión de que la investigación que se está llevando a cabo es un estudio de tipo Descriptivo y Explicativo pues tal investigación cumple con las características de estos tipos.

Con el objeto de crear un sistema eficiente al hacer una revisión de la literatura me di cuenta que hay mucha información acerca de este tema (entorno grafico y conexión con base de datos), hay varios lenguajes o programas que pueden hacerlo, esto es una gran ventaja ya que facilitara la realización del sistema.

Puesto que el tipo de estudio es Descriptivo y Explicativo, Para poder realizar todo esto tengo que hacer una investigación en la empresa sobre su manejo y funcionamiento para poder ver cuál es la problemática y encontrar la causa de los conflictos y así poder realizar preguntas que puedan llevar a resolver los conflictos para poder realizar una predicción de lo que pueda pasar con el sistema.

Otro aspecto del estudio Descriptivo es que se cuenta con un amplio conocimiento de los temas tratados para resolver los diversos conflictos que se puedan ir generando dentro de esta investigación.

Justificación

Las grandes y pequeñas empresas en la actualidad se hacen más competitivas dentro de su ramo y cada vez adoptan más estrategias a fin de garantizar el éxito. Estas organizaciones están adoptando herramientas de optimización, basadas en las nuevas tecnologías, a fin de alcanzar el éxito a corto, mediano y largo plazo con el propósito de establecerse metas que permitan el alcance de los Planes Estratégicos del Negocio, enfocados al cumplimiento de la Visión, Misión, Valores etc., elementos que conjugados comprometen tanto a empleados como supervisores a la identificación con la organización, a través de un sentimiento de compromiso para alcanzar los objetivos de la misma. Por todas estas razones y para competir con las demás empresas que ya utilizan estas tecnologías la empresa JAJOR SA. De CV. Desea beneficiarse de estas tecnologías por lo cual requiere de un sistema informático, ya que mediante su utilización se podrá optimizar los procesos que en ella se realizan en beneficio de esta y a su Plan Estratégico de Negocio.

Por lo tanto, repercutirá en la calidad de los servicios de la empresa, mediante la implementación de este sistema se evitaran los diversos conflictos que en ella se generan, el seguimiento de los clientes será llevado de mejor manera, la atención del encargado de contestar el teléfono será más eficiente y rápida, la empresa se dará cuenta cuantas ventas lleva en el mes o en determinada fecha cuanto le toco al vendedor y cuanto a ellos, con esto podrá planear mejor su plan de ventas, todo esto a fin de mejorar la calidad, el control de la gestión, la satisfacción y la respuesta a los clientes en forma oportuna y eficiente para el beneficio de la empresa y mantener un nivel de satisfacción y equilibrio.

Por otra parte, en cuanto a su alcance, esta investigación abrirá nuevos caminos para empresas que presenten situaciones similares a la que aquí se plantea, sirviendo como marco de referencia a estas y la posible expansión del sistema.

Por último, profesionalmente pondrá en manifiesto los conocimientos adquiridos durante la carrera y permitirá sentar las bases para otros estudios que surjan partiendo de la problemática aquí especificada.

Marco Teórico

Existencia de teorías

Para el desarrollo de este proyecto de tesis se basara en tres partes una que es la base de datos, el enlace o puente entre la interfaz y la base de datos y por último la interfaz grafica del usuario.

A continuación se explica cada uno de ellos.

Algunos conceptos de bases de datos:
Desde el punto de vista de la informática, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.
Base de Datos: es la *colección de datos aparentes usados por el sistema* de aplicaciones de una determinada empresa.
Base de Datos: es un conjunto de información relacionada que se encuentra agrupada o estructurada. Un archivo por sí mismo no constituye una base de datos, sino más bien la forma en que está organizada la información es la que da origen a la base de datos.
Base de Datos: colección de datos organizada para dar servicio a muchas aplicaciones al mismo tiempo al combinar los datos de manera que aparezcan *estar en una sola ubicación*[1]

Requerimientos de las bases de datos:
El *análisis de requerimientos para una base de datos incorpora las mismas* tareas que el análisis de requerimientos del software. Es necesario un contacto estrecho con el cliente; es esencial la identificación de las funciones e interfaces; se requiere la especificación del flujo, estructura y asociatividad de la información y debe desarrollarse un documento formal de los requerimientos.

[1] M Kroenke, David (2003), Procesamiento de bases de datos octava edición, México, Prentice Hall, 25-26.

Requerimientos administrativos: se requiere mucho más para el desarrollo de sistemas de bases de datos que únicamente seleccionar un modelo lógico de base de datos. La bases de datos es una disciplina organizacional, un método, más que una herramienta o una tecnología. Requiere de un cambio conceptual y organizacional.

Interfaz de acceso a bases de datos

Proporciona un acceso uniforme a una gran variedad de bases de datos relacionales. También proporciona una base común para la construcción de herramientas y utilidades de alto nivel.

Es una parte muy importante ya que es el enlace entre la interfaz grafica del usuario y la base de datos; todas las peticiones que se le hacen a la base de datos pasan por este puente o enlace

Simplemente hace posible estas tres cosas:

- Establece una conexión con la base de datos.
- Envía sentencias SQL (explicare en el capítulo 4 y 5)
- Procesa los resultados.

El driver con la base de datos

Lo primero que necesitamos para conectarnos con una base de datos es un Driver (o Connector) con ella. Ese Driver es la clase o conjunto de clases que, de alguna forma, sabe cómo hablar con la base de datos. Desgraciadamente (y hasta cierto punto es lógico), java no viene con todos los Drivers de todas las posibles bases de datos del mercado. Debemos ir a internet y obtener el Driver, normalmente en la página de nuestra base de datos

Interfaz gráfica de usuario, en informática, tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú.

Las selecciones pueden activarse bien a través del teclado o con el ratón. [2]

Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con la computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no está sujeto a los detalles de la visualización ni a la entrada a través del ratón o del teclado. También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios.

- [2] Donald Norman (2002), The design of everyday things, Estados Unidos, Basic Books, PP 5.

Capítulo I. Antecedentes

1.1 Antecedentes de la empresa.

La empresa Inmobiliaria JAJOR S.A. de C.V. comienza sus operaciones en la ciudad de Morelia Michoacán en el año de 2003, teniendo principalmente dos perspectivas una de ellas está dirigida a proyecto y construcción, y la otra a la promoción de inmuebles, pero las dos tienen una misma finalidad y es brindar un buen servicio a la sociedad, actuando con honestidad y responsabilidad.
Tiene como meta principal ser una empresa reconocida en la promoción y construcción de inmuebles, ofreciendo a la sociedad no una casa sino un hogar.

Al cabo de varios años la empresa ha adquirido más prestigio por lo cual los desarrolladores tienen más confianza en sus servicios y cada vez son más los inmuebles que tienen disponibles, además la cartera de clientes ha aumentado considerablemente. La empresa venía manejando de manera clásica la información de los clientes e inmuebles, es decir escritos en libretas, y para seguir avanzando en su búsqueda de aumentar la calidad en sus servicios, así como aprovechar las ventajas que hoy en día nos ofrecen las tecnologías, la empresa encuentra en esta, la solución a los conflictos que tienen y de esta manera llevar un mejor control y manejo de la inmobiliaria, los principales conflictos que tiene la empresa son los siguientes:

Conflictos con los clientes:

Los empleados realizan guardias en los inmuebles en venta, escriben el nombre de los clientes en cuadernillos de reportes, mismos que no resultan prácticos ya que se pierde tiempo al momento de consultar la información y en ocasiones esta se pierde.

Conflictos con los inmuebles:

El encargado de la empresa sabe donde están todos los inmuebles, pero debido a que son muchos y a veces no tiene tiempo de llevar a los empleados a conocer los nuevos inmuebles que tienen en venta no saben que nuevos inmuebles tienen o la ubicación de otros.

La empresa hizo un catalogo impreso de los inmuebles, pero no están todos, además de que cuando esta el encardado de recibir las llamadas y le preguntan por determinado inmueble tiene que buscar en el catalogo, lo cual implica pérdida de tiempo y al no encontrar la información se puede perder una venta.

Conflictos internos:

Puede ser que dos vendedores estén atendiendo al mismo cliente y no se dan cuenta, entonces si uno de los dos le vende un inmueble se crea un conflicto por la comisión y esto crea roces entre ellos ya que el cliente puede ser que ya haya sido atendido desde hace tiempo por uno de los dos pero el que lo atendió después le vendió.

Muchas veces hacen citas y no se acuerdan de ellas o hacen citas el mismo día y hora.

No llevan un control de sus ventas, originando con esto un conflicto cuando quieren saber cuántos inmuebles vendieron en determinado tiempo o cuanto recibieron de comisión sus vendedores y cuanto la inmobiliaria.

Se puede dar el caso de que un empleado ya no trabaje en la empresa y deje de dar seguimiento a los clientes, en consecuencia, los clientes molestos hacen reclamaciones a la empresa por no mantenerlos informados acerca de la compra de inmuebles, créditos personales o cualquier asunto relacionado con ellos.

1.2 Solución a los conflictos

Por todas estas razones la empresa requiere de un sistema que lleve el control de los clientes, inmuebles y empleados.

El sistema está dirigido al control y manejo de la inmobiliaria, con un enfoque a servir mejor al cliente; pretende evitar estos conflictos y llevar un mejor control y manejo de la misma, el nombre del sistema será ***"Sistema para el control de clientes inmuebles y empleados (CCIE)"***, para el control el sistema utiliza dos tipos de usuarios el administrador y el empleado, uno que contrala todo el sistema y el otro que solo puede consultar información de los inmuebles, guardar y revisar información sobre sus clientes.

Para evitar los conflictos con los clientes los empleados pueden subir al sistema toda la información de sus clientes (nombre, teléfonos etc.) también pueden llevar un historial de todo lo que van realizando con ellos, que casas le gusto que tipo de crédito tiene, como va su crédito etc. Cuando buscan sus clientes en el sistema lo van a poder hacer por tipo de crédito o por nombre.

El encargado de la empresa va poder guardar la información de los inmuebles que tienen, esto les va a facilitar difundirlos entre sus empleados ya que en cualquier tiempo libre lo pueden hacer en un corto periodo de tiempo, también pueden guardar una foto del inmueble solo una ya que como dije el programa está dirigido a los que laboran en la empresa y solo les va a servir como referencia del inmueble.

Si un empleado quiere guardar un cliente pero ya está registrado, el sistema le va a mandar un mensaje diciendo que ya está el cliente registrado y quien lo registro además de que va a mandar un mensaje al usuario de tipo administrador para que se dé cuenta que ese empleado tiene registrado un cliente con el mismo nombre ya que pueden haber empleados que no sean honestos y aunque el sistema les mencione que ya está registrado puede ser que no digan nada y lo sigan atendiendo como si nada.

Así ya con esto se evitaran los conflictos de que dos empleados están atendiendo al mismo cliente. Se puede dar el caso de que el cliente tenga un homónimo por lo cual el sistema aunque les manda el mensaje que ya está registrado si permite guardarlo.

El sistema va tener implementado un recordatorio de citas, el cual les va a recordar las citas del día, pueden especificar cada cuanto tiempo quieren que les este recordando, el tiempo va de cada cinco minutos hasta una hora, así como también si ya tiene una cita en la hora y día especificados les manda un mensaje para que se den cuenta y la cambien.

El sistema va tener la opción de llevar a cabo un movimiento que representa una venta, registrando el inmueble que se vendió, quien lo compro y quien lo vendió, una vez realizado puede consultar por fechas que inmuebles se vendieron quien lo compro cuanto le toco de comisión al que lo vendió y cuanto a la inmobiliaria.

El usuario de tipo administrador puede consultar los clientes de todos los empleados únicamente consultar, no puede hacer modificaciones esto para evitar conflictos cuando el empleado ya no esté trabajando en la empresa o este mal atendiendo al cliente así el administrador se puede dar cuenta de que es lo que ha pasado con ese cliente y poder orientarlo de una mejor manera, ya que es todo un proceso el estar atendiendo a un cliente.

Se espera que a través de este sistema queden resueltos estos conflictos. Para que todo funcione bien y se lleve un mejor control, la empresa tiene por obligación decirle a sus empleados que por regla tienen que guardar toda la información y el seguimiento que les están haciendo a los clientes porque de nada va servir el sistema si no es alimentado adecuadamente, claro con una capacitación previa sobre el manejo del sistema.

Capítulo II. Lenguaje de programación

El lenguaje de programación que debo utilizar tiene que tener las siguientes características:

- Orientado a objetos ya que esto facilita más la programación.
- Facilidad para crear la interfaz grafica del usuario.
- Conexión con la base de datos.
- Que funcione en red.
- Que sea seguro.

por lo tanto voy a utilizar Java versión 5.0 porque es una herramienta muy poderosa y cumple con todas las características que necesito ya que con otros lenguajes es más complicado, Java tiene implementadas clases para manejar la interfaz y debido a que el diseño del sistema está basado en la arquitectura MVC (modelo vista controlador y que explicare a fondo en el siguiente capítulo) me da todas las ventajas para hacerlo así, a continuación voy a explicar más a fondo lo que es la programación orientada a objetos sus diferentes características ya que es la parte medular en cuanto a la estructura del programa, y como Java cumple con todas las características mencionadas.

2.1 Programación Orientada a Objetos (POO)

Una parte muy importante es que el lenguaje sea orientado a objetos ya que al programar hace que esté mejor estructurado y los diferentes módulos sean más fáciles de modificar.

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos.

2.1.1 Motivación

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Que es una serie de normas de realizar las cosas de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.

La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador. Aunque podamos hacer los programas de formas distintas, no todas ellas son correctas, lo difícil no es programar orientado a objetos sino programar bien. Programar bien es importante porque así nos podemos aprovechar de todas las ventajas de la POO.

2.1.2 Cómo se piensa en objetos

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real. Por ejemplo vamos a pensar en un coche para tratar de hacer un modelo en un esquema de POO. Diríamos que el coche es el elemento principal que tiene una serie de características, como podrían ser el color, el modelo o la marca. Además tiene una serie de funcionalidades asociadas, como pueden ser ponerse en marcha, parar o aparcar.

Pues en un esquema POO el coche sería el objeto, las propiedades serían las características como el color o el modelo y los métodos serían las funcionalidades asociadas como ponerse en marcha o parar.

Por poner otro ejemplo vamos a ver cómo hacer un modelo en un esquema POO de una fracción, es decir, esa estructura matemática que tiene un numerador y un denominador que divide al numerador, por ejemplo 3/2.

La fracción será el objeto y tendrá dos propiedades, el numerador y el denominador. Luego podría tener varios métodos como simplificarse, sumarse con otra fracción o número, restarse con otra fracción, etc.

Estos objetos se podrán utilizar en los programas, por ejemplo en un programa de matemáticas harás uso de objetos fracción y en un programa que gestione un taller de coches utilizarás objetos coche. Los programas Orientados a objetos utilizan muchos objetos para realizar las acciones que se desean realizar y ellos mismos también son objetos. Es decir, el taller de coches será un objeto que utilizará objetos coche, herramienta, mecánico, recambios, etc.

2.1.3 Clases en POO

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase. En los ejemplos anteriores en realidad hablábamos de las clases coche o fracción porque sólo estuvimos definiendo de forma genérica sus características y comportamiento.

2.1.4 Propiedades en clases

Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son variables donde almacenamos datos relacionados con los objetos.

2.1.5 Métodos en las clases

Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son funciones que están asociadas a un objeto.

2.1.6 Objetos en POO

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar). Por ejemplo, un objeto de la clase fracción es 3/5.

El concepto o definición de fracción sería la clase, pero cuando ya estamos hablando de una fracción en concreto 4/7, 8/1000 o cualquier otra, la llamamos objeto.

Para crear un objeto se tiene que escribir una instrucción especial que puede ser distinta dependiendo el lenguaje de programación que se emplee, pero será algo parecido a esto.

miCoche = new Coche()

Con la palabra new especificamos que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis podríamos colocar parámetros con los que inicializar el objeto de la clase coche.

2.1.7 Estados en objetos

Cuando tenemos un objeto sus propiedades toman valores. Por ejemplo, cuando tenemos un coche la propiedad color tomará un valor en concreto, como rojo o gris metalizado. El valor concreto de una propiedad de un objeto se llama estado.

Para acceder a un estado de un objeto para ver su valor o cambiarlo se utiliza el operador punto.

miCoche.color = rojo

El objeto es miCoche, luego colocamos el operador punto y por último el nombre de la propiedad a la que deseamos acceder. En este ejemplo estamos cambiando el valor del estado de la propiedad del objeto a rojo con una simple asignación.

2.1.8 Mensajes en objetos

Un mensaje en un objeto es la acción de efectuar una llamada a un método. Por ejemplo, cuando le decimos a un objeto coche que se ponga en marcha estamos pasándole el mensaje "ponte en marcha".

Para mandar mensajes a los objetos utilizamos el operador punto, seguido del método que deseamos invocar.

miCoche.ponerseEnMarcha()

En este ejemplo pasamos el mensaje ponerseEnMarcha(). Hay que colocar paréntesis igual que cualquier llamada a una función, dentro irían los parámetros.

Hay mucho todavía que conocer de la POO ya que sólo hemos hecho referencia a las cosas más básicas. También existen mecanismos como la herencia y el polimorfismo que son unas de las posibilidades más potentes de la POO.

La herencia sirve para crear objetos que incorporen propiedades y métodos de otros objetos. Así podremos construir unos objetos a partir de otros sin tener que reescribirlo todo.

El polimorfismo sirve para que no tengamos que preocuparnos sobre lo que estamos trabajando, y abstraernos para definir un código que sea compatible con objetos de varios tipos.

2.2 Java

He podido leer más de cinco versiones distintas sobre el origen, concepción y desarrollo de Java, desde la que dice que este fue un proyecto que rebotó durante mucho tiempo por distintos departamentos de Sun sin que nadie le prestara ninguna atención.

Hasta que finalmente encontró su nicho de mercado en la aldea global que es Internet; hasta la más difundida, que justifica a Java como lenguaje de pequeños electrodomésticos y que a continuación voy a mencionar.

Sun Microsystems, líder en servidores para Internet, uno de cuyos lemas desde hace mucho tiempo es "the network is the computer" (lo que quiere dar a entender que la verdadera computadora es la red en su conjunto y no cada máquina individual), es quien ha desarrollado el lenguaje Java, en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes máquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma máquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

Hace algunos años, Sun Microsystems decidió intentar introducirse en el mercado de la electrónica de consumo y desarrollar programas para pequeños dispositivos electrónicos. Tras unos comienzos dudosos, Sun decidió crear una filial, denominada FirstPerson Inc., para dar margen de maniobra al equipo responsable del proyecto.

El mercado inicialmente previsto para los programas de FirstPerson eran los equipos domésticos: microondas, tostadoras y, fundamentalmente, televisión interactiva. Este mercado, dada la falta de pericia de los usuarios para el manejo de estos dispositivos, requería unas interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban en el momento.

Otros requisitos importantes a tener en cuenta eran la fiabilidad del código y la facilidad de desarrollo. James Gosling, el miembro del equipo con más experiencia en lenguajes de programación, decidió que las ventajas aportadas por la eficiencia de C++ no compensaban el gran costo de pruebas y depuración.

Gosling había estado trabajando en su tiempo libre en un lenguaje de programación que él había llamado Oak, el cual, aun partiendo de la sintaxis de C++, intentaba remediar las deficiencias que iba observando.

Los lenguajes, como C o C++, deben ser compilados para un chip, y si se cambia el chip, todo el software debe compilarse de nuevo. Esto encarece mucho los desarrollos y el problema es especialmente acusado en el campo de la electrónica de consumo. La aparición de un chip más barato y, generalmente, más eficiente, conduce inmediatamente a los fabricantes a incluirlo en las nuevas series de sus cadenas de producción, por pequeña que sea la diferencia en precio ya que, multiplicada por la tirada masiva de los aparatos, supone un ahorro considerable. Por tanto, Gosling decidió mejorar las características de Oak y utilizarlo.

El primer proyecto en que se aplicó este lenguaje recibió el nombre de proyecto Green y consistía en un sistema de control completo de los aparatos electrónicos y el entorno de un hogar. Para ello se construyó una computadora experimental denominada *7 (Star Seven). El sistema presentaba una interfaz basada en la representación de la casa de forma animada y el control se llevaba a cabo mediante una pantalla sensible al tacto. En el sistema aparecía Duke, la actual mascota de Java.

Posteriormente se aplicó a otro proyecto denominado VOD (Video On Demand, Video a la Carta) en el que se empleaba como interfaz para la televisión interactiva. Ninguno de estos proyectos se convirtió nunca en un sistema comercial, pero fueron desarrollados enteramente en un Java primitivo y fueron como su bautismo de fuego.

Una vez que en Sun se dieron cuenta de que a corto plazo la televisión interactiva no iba a ser un gran éxito, urgieron a FirstPerson a desarrollar con rapidez nuevas estrategias que produjeran beneficios. No lo consiguieron y FirstPerson cerró en la primavera de 1994.

Pese a lo que parecía ya un olvido definitivo, Bill Joy, cofundador de Sun y uno de los desarrolladores principales del Unix de Berkeley, juzgó que Internet podría llegar a ser el campo de juego adecuado para disputar a Microsoft supremacía casi absoluta en el terreno del software, y vio en Oak el instrumento idóneo para llevar a cabo estos planes. Tras un cambio de nombre y modificaciones de diseño, el lenguaje Java fue presentado en sociedad en agosto de 1995.

2.2.1 Porque utilizar Java

Lo mejor será hacer caso omiso de las historias que pretenden dar carta de naturaleza a la clarividencia industrial de sus protagonistas; porque la cuestión es si independientemente de su origen y entorno comercial, Java ofrece soluciones a nuestras expectativas.

No es arriesgado afirmar que Java supone un significativo avance en el mundo de los entornos software, y esto viene avalado por tres elementos claves que diferencian a este lenguaje desde un punto de vista tecnológico:

- Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
- Proporciona un conjunto de clases potente y flexible.
- Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web (aplicaciones denominadas applets).

Actualmente, la plataforma Java ha atraído a cerca de 4 millones de desarrolladores de software, se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, computadoras y redes de cualquier tecnología de programación.

De hecho, su versatilidad y eficiencia, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes, de manera que hoy en día, más de 2.500 millones de dispositivos utilizan la tecnología Java.

- Más de 700 millones de computadoras
- 708 millones de teléfonos móviles y otros dispositivos de mano
- 1000 millones de tarjetas inteligentes
- Además de sintonizadores, impresoras, web cams, juegos, sistemas de navegación para automóviles, terminales de lotería, dispositivos médicos, cajeros de pago en aparcamientos, etc.

Hoy en día, puede encontrar la tecnología Java en redes y dispositivos que comprenden desde Internet y supercomputadoras científicas hasta portátiles y teléfonos móviles; desde simuladores de mercado en Wall Street hasta juegos de uso doméstico y tarjetas de crédito: Java está todas partes. El lenguaje de programación Java ha sido totalmente mejorado, ampliado y probado por una comunidad activa de unos cuatro millones de desarrolladores de software.

La tecnología Java, una tecnología madura, extremadamente eficaz y sorprendentemente versátil, se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Crear programas para que funcionen en un navegador Web y en servicios Web
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo costo y prácticamente cualquier dispositivo digital.

2.2.2 Características de Java

Orientado a objetos

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Cualquier objeto del mundo se puede modelar y así sus características y métodos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

En Java el concepto de objeto resulta sencillo y fácil de ampliar. Además se conservan elementos "no objetos", como números, caracteres y otros tipos de datos simples.

Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (Runtime).

Robusto

Java fue diseñado para crear software altamente fiable. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución, lo que hace que se detecten errores lo antes posible, normalmente en el ciclo de desarrollo. Algunas de estas verificaciones que hacen que Java sea un lenguaje robusto son:

- Verificación del código de byte.
- Gestión de excepciones y errores.
- Comprobación de punteros y de límites de vectores.

Se aprecia una clara diferencia con C++ quién no realiza ninguna de estas verificaciones.

Seguro

Dada la naturaleza distribuida de Java, donde los applets se bajan desde cualquier punto de la red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su computadora programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

En Java no se permite los accesos ilegales a memoria, algo que sí se permitía en C++. Esto es algo muy importante puesto que este tipo de problema puede ocasionar la propagación de virus y otras clases de programas dañinos por la red.

El código Java pasa muchos tests antes de ejecutarse en una máquina. El código se pasa a través de un verificador de código de byte que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal, código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto.

Algunos de los conocimientos que podemos obtener de los códigos de byte si pasan la verificación sin generar ningún mensaje de error son:

- El código no produce desbordamiento de operandos en la pila.
- El tipo de los parámetros de todos los códigos de operación es conocido y correcto.
- No ha ocurrido ninguna conversión ilegal de datos, tal como convertir enteros en punteros.
- El acceso a los campos de un objeto se sabe si es legal mediante las palabras reservadas public, private y protected.
- No hay ningún intento de violar las reglas de acceso y seguridad establecidas.

Por todo esto, y por no permitir mediante Java la utilización de métodos de un programa sin los privilegios del núcleo (kernel) del sistema operativo y la obligación de autentificación por clave pública para la realización de modificaciones, se considera Java un lenguaje seguro.

Indiferente a la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variados, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples plataformas de hardware y software. El resto de problemas los soluciona el intérprete de Java.

Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Recolección automática de basura (Garbage colection)

Java modifica completamente la gestión de la memoria que se hace en C/C++. En C/C++ se utilizan punteros, reservas de memoria (con las ordenes malloc, new, free, delete...) y otra serie de elementos que dan lugar a graves errores en tiempo de ejecución difícilmente depurables.

Java tiene operadores nuevos para reservar memoria para los objetos, pero no existe ninguna función explícita para liberarla.

La recolección de basura (objetos inservibles) es una parte integral de Java durante la ejecución de sus programas. Una vez que se ha almacenado un objeto en el tiempo de ejecución, el sistema hace un seguimiento del estado del objeto, y en el momento en que se detecta que no se va a volver a utilizar ese objeto, el sistema vacía ese espacio de memoria para un uso futuro.

Esta gestión de la memoria dinámica hace que la programación en Java sea más fácil.

Multitarea

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Dinámico

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la red.

Para la obtención de un mayor provecho de la tecnología orientada a objetos, Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Esta característica ya es contemplada por Smalltalk, aunque no C++, que enlaza todos los módulos cuando se compila.

Simple

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C, Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- Aritmética de punteros
- Registros (struct)
- Definición de tipos (typedef)
- Macros (#define)
- Necesidad de liberar memoria (free)

Aunque, en realidad, lo que hace es eliminar las palabras reservadas (struct, typedef), ya que las clases son algo parecido.

2.2.3 Ventajas

- Primero: No debes volver a escribir el código si quieres ejecutar el programa en otra máquina. Un solo código funciona para todos los navegadores compatibles con Java o donde se tenga una Máquina Virtual de Java (Macs, PCs, u otros).
- Segundo: Java es un lenguaje de programación orientado a objetos, y tiene todos los beneficios que ofrece esta metodología de programación.

- Tercero: Un navegador compatible con Java deberá ejecutar cualquier programa hecho en Java, esto ahorra a los usuarios tener que estar insertando "plug-ins" y demás programas que a veces nos quitan tiempo y espacio en disco.
- Cuarto: Java puede hacer: Cálculos matemáticos, procesadores de palabras, bases de datos, aplicaciones gráficas, animaciones, sonido, hojas de cálculo, etc.
- Quinto: Si lo que me interesa son las páginas Web, ya no tienen que ser estáticas, se le pueden poner toda clase de elementos multimedia y permiten un alto nivel de interactividad, sin tener que gastar en paquetes carísimos de multimedia.

2.2.4 Desventajas

- La velocidad; los programas hechos en Java no tienden a ser muy rápidos, supuestamente se está trabajando en mejorar esto. Como los programas de Java son interpretados nunca alcanzan la velocidad de un verdadero ejecutable.
- Java es un lenguaje de programación. Esta es otra gran limitante, por más que digan que es orientado a objetos y que es muy fácil de aprender sigue siendo un lenguaje de programación y por lo tanto aprenderlo no es cosa fácil. Especialmente para los no programadores.

Pero en general Java posee muchas ventajas y se pueden hacer cosas muy interesantes con esto, a pesar de que Java es relativamente nuevo, posee mucha fuerza y es tema de moda en cualquier medio computacional por todas estas razones es porque lo voy a utilizar en mi sistema.

Capítulo III. Modelo arquitectónico

El sistema va ser implementado en la arquitectura MVC (Modelo Vista Controlador) lo cual significa que la interfaz grafica, la conexión y la base de datos están separadas es decir el código para conectarse a la base de datos está en unidades diferentes.

En su forma más simple, la arquitectura es la estructura jerárquica de los componentes del programa (módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. Sin embargo, en un sentido más amplio, los "componentes" se pueden generalizar para presentar los elementos principales del sistema y sus interacciones.

El diseño arquitectónico define la relación entre los elementos estructurales principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos.

Los patrones expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos; así como ayudan a especificar la estructura fundamental de una aplicación. Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas Java. Todos tienen un Frame que contiene todos los elementos, un controlador de eventos, cálculos y la presentación del resultado. Ante esta perspectiva, hacer un cambio aquí no es nada trivial.

3.1 Definición de las partes

El modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los controladores o de las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de *mantener enlaces entre el modelo* y *sus vistas*, y *notificar a las vistas cuando* cambia el modelo.

La vista es el objeto que maneja la presentación visual de los datos *representados por el modelo. Genera una representación visual del modelo* y muestra los datos al usuario. Interactúa con el modelo a través de una referencia al propio modelo.

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del modelo o por alteraciones de la vista. Interactúa con el modelo a través de una referencia al propio modelo.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón o enlace)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la *notificación de la acción solicitada por el usuario. El controlador gestiona* el evento que llega, frecuentemente a través de un gestor de eventos.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta *indirección entre el modelo y la vista, permitiendo al modelo notificar a* los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. *La interfaz de usuario espera nuevas interacciones del usuario*, comenzando el ciclo nuevamente.

Algunos de sus principales beneficios son:

- *Menor acoplamiento.*
  a) Desacopla las vistas de los modelos.
  b) Desacopla los modelos de la forma en que se muestran e ingresan los datos.
- Mayor cohesión.
  a) Cada elemento del patrón está altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).
- *Las vistas proveen mayor flexibilidad y agilidad.*
  a) Se puede crear múltiples vistas de un modelo.
  b) Las vistas pueden anidarse.
  c) Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.
  d) Se puede sincronizar las vistas.
  e) Las vistas pueden concentrarse en diferentes aspectos del modelo.

- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales
  a) Una vista para cada dispositivo que puede variar según sus capacidades.
  b) Una vista para la Web y otra para aplicaciones de escritorio.
- Más claridad de diseño.
- Facilita el mantenimiento.
- Mayor escalabilidad.

3.2 MVC aplicado al sistema

En el caso del sistema desarrollado, este está divido en 16 módulos que son los siguientes:

- Modulo de acceso al sistema.
- Modulo de alta de cliente.
- Modulo de programar citas.
- Modulo de historial del cliente.
- Modulo de baja de cliente.
- Modulo de nota al usuario.
- Modulo para ver clientes de otros usuarios.
- Modulo de alta de inmuebles.
- Modulo de búsqueda de inmuebles.
- Modulo para eliminar un inmueble.
- Modulo de alta de usuario.
- Modulo para revisar los pendientes.
- Modulo para registrar una venta.
- Modulo para revisar las ventas por fecha.
- Modulo para eliminar un usuario.
- Modulo para cambiar la contraseña.

Hay tres clases principales las cuales controlan toda la interfaz, la de administrador, la de vendedores y otra la cual provee de varios métodos para la creación de la interfaz del usuario como son botones, iconos, métodos, todo lo que se repite y que esta demás programarlo en varias clases; así es más fácil llamar a un solo método que escribirlo varias veces.

Todas las demás opciones del sistema están por separado, varias clases que en conjunto forman un Modulo (alta de clientes, citas, inmuebles etc.) Esto hace más fácil la modificación de cada modulo y la construcción de más en caso de requerirse.

Entonces lo que hacen las clases principales nada más es llamar a esas clases que ya tienen la interfaz y los métodos implementados cada vez que el usuario las solicite, esto hace que el sistema este mejor estructurado y que esas clases solo se llamen cuando el usuario las requiera ahorrando memoria y espacio.

El sistema tiene una clase para actualizar o consultar la base de datos. En ella se encuentran todos los métodos de acceso a esta; lo que hacen las otras clases es llamar a estos métodos pasándoles como argumentos lo que quieren consultar y estos a su vez hacen las consultas a la base de datos.

También se encuentran otras clases llamadas tipo record. Su función es, primero llenar los combo box haciendo una consulta a la base de datos mostrando de manera más eficiente los datos que aparecen ya que evitan hacer consultas más extensas, y segundo al seleccionar algún elemento del combo box tienen métodos implementados para seleccionar el id y el contenido en la base de datos, en el caso de este sistema serian nombres de clientes, inmuebles o empleados así no es necesario tener un método para llenar el combo box y otro para obtener lo que selecciono el usuario.

En este caso las clases que contienen la interfaz del usuario son la vista, la clase que se conecta con la base de datos representa el controlador y la base de datos el modelo.

Capítulo IV. Base de datos

La parte más importante o critica del sistema es la base de datos (el modelo) ya que en ella se guarda la información que es vital para la empresa, es lo primero que se tiene que diseñar ya que lleva toda la estructura y la manera en la que se va a manejar el sistema, a partir de esto se van a desarrollar los diferentes módulos del sistema en otras palabras, el modelo manda, es por esto que tiene que estar bien estructurada.

La base de datos es de tipo relacional y será implementada con MySQL versión 5.0 y el lenguaje con el que me voy a conectar con la base de datos es SQL a continuación voy a explicar los diferentes conceptos y las razones por las cuales voy a utilizarlos.

4.1 Base de datos

Una base de datos o banco de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

El primer paso para crear una base de datos, es planificar el tipo de información que se quiere almacenar en la misma, teniendo en cuenta dos aspectos: la información disponible y la información que necesitamos.

La planificación de la estructura de la base de datos, en particular de las tablas, es vital para la gestión efectiva de la misma. El diseño de la estructura de una tabla consiste en una descripción de cada uno de los campos que componen el registro y los valores o datos que contendrá cada uno de esos campos.

Los campos son los distintos tipos de datos que componen la tabla, por ejemplo: nombre, apellido, domicilio. La definición de un campo requiere: el nombre del campo, el tipo de campo, el ancho del campo, etc.

Los registros constituyen la información que va contenida en los campos de la tabla, por ejemplo: el nombre del paciente, el apellido del paciente y la dirección de este. Generalmente los diferente tipos de campos que se pueden almacenar son los siguientes: Texto (caracteres), Numérico (números), Fecha/Hora, Lógico (informaciones lógicas si/no, verdadero/falso, etc., imágenes.

En resumen, el principal aspecto a tener en cuenta durante el diseño de una tabla es determinar claramente los campos necesarios, definirlos en forma adecuada con un nombre especificando su tipo y su longitud.

4.2 Bases de datos relacionales

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Una relación es definida como un conjunto de registros donde todos tienen los mismos atributos. Esto es usualmente representado por una tabla., la cual está organizada en filas y columnas. En una base de datos relacional, todos los datos son almacenados en una columna, la que debe estar en el mismo dominio.

En la práctica esto significa que los valores almacenados en una columna deben ser de mismo tipo de dato y de acuerdo a las mismas restricciones.

El modelo relacional especifica que los registros de una relación deben tener órdenes no específicas y que los registros, no deben imponerse a los atributos.

La operación de selección relacional es equivalente a las consulta SQL SELECT (consulta de selección SQL), posiblemente con una cláusula WHERE (donde), que limita los resultados. En el modelo relacional los atributos deben estar relacionados, explícitamente a un nombre en todas las operaciones, mientras en el estándar SQL permite tanto a columnas sin nombre en conjuntos de resultados, como el asterisco taquigráfico (*) como notación de consultas.

El estándar SQL requiere que las columnas. Tengan un orden definido. Los datos almacenados en una computadora deben tener un orden, tal como la memoria de una computadora es lineal. También, cuando los datos son devueltos, deben estar en un orden tal que los datos son devueltos debido a que son transferidos por protocolos que también son lineales. Es de notar, sin embargo, que en SQL el orden de las columnas y las filas devueltas en un juego de conjunto de resultado nunca es garantizado, a no ser que explícitamente sea especificado por el usuario.

Las bases de datos relacionales pasan por un proceso al que se le conoce como normalización de una base de datos, la cual es entendida como el proceso necesario para que una base de datos sea utilizada de manera óptima.

4.3 SQL

El Lenguaje de Consulta Estructurado (SQL [/esecuele/ en español, /sicuèl/ en inglés] Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla. Es un lenguaje de cuarta generación (4GL).

Los orígenes del SQL están ligados a los de las bases de datos relacionales. En 1970 E. F. Codd propone el modelo relacional y asociado a este un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basándose en estas ideas, los laboratorios de IBM definen el lenguaje SEQUEL (Structured English QUEry Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo éste una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

Sin embargo este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado SQL-92 o SQL2.

En la actualidad el SQL es el estándar de facto de la inmensa mayoría de los SGBD comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio.

Como su nombre indica, el SQL nos permite realizar consultas a la base de datos. Pero el nombre se queda corto ya que SQL además realiza funciones de definición, control y gestión de la base de datos. Las sentencias SQL se clasifican según su finalidad dando origen a tres sublenguajes:

El **DDL** (Data Description Language), lenguaje de definición de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas. (Es el que más varia de un sistema a otro).

El **DCL** (Data Control Language), lenguaje de control de datos, contiene elementos útiles para trabajar en un entorno multiusuario, en el que es importante la protección de los datos, la seguridad de las tablas y el establecimiento de restricciones en el acceso, así como elementos para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.

El **DML** (Data Manipulation Language), lenguaje de manipulación de datos, nos permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

4.3.1 Características generales

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

Una sentencia SQL es como una frase (escrita en inglés) con la que decimos lo que queremos obtener y de donde obtenerlo.

Todas las sentencias empiezan con un verbo (palabra reservada que indica la acción a realizar), seguido del resto de cláusulas, algunas obligatorias y otras opcionales que completan la frase. Todas las sentencias siguen una sintaxis para que se puedan ejecutar correctamente.

4.3.2 Optimización

Como ya se dijo arriba, y como suele ser común en los lenguajes de acceso a bases de datos de alto nivel, el SQL es un lenguaje declarativo. Esto significa, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución. El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a cabo una optimización antes de la ejecución de la misma. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos, dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.

4.4 MySQL

Surgió alrededor de la década del 90, Michael Windenis comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel (ISAM). Tras unas primeras pruebas, llegó a la conclusión de que mSQL no era lo bastante flexible ni rápido para lo que necesitaba, por lo que tuvo que desarrollar nuevas funciones. Esto resulto en una interfaz SQL a su base de datos, totalmente compatible a mSQL.

El origen del nombre MySQL no se sabe con certeza de donde proviene, por un lado se dice que en sus librerías han llevado el prefijo "my" durante los diez últimos años, por otra parte, la hija de uno de los desarrolladores se llama My. Así que no está claramente definido cuál de estas dos causas han dado lugar al nombre de este conocido gestor de bases de datos.

MySQL, el sistema de gestión de bases de datos SQL Open Source más popular, lo distribuía y soportaba MySQL AB. MySQL AB es una compañía comercial, fundada por los desarrolladores de MySQL. Es una compañía Open Source de segunda generación que une los valores y metodología Open Source con un exitoso modelo de negocio y actualmente fue comprada por la compañía informática estadounidense Sun Microsystems.

MySQL es Open Source (código abierto) significa que es posible para cualquiera usar y modificar el software. Cualquiera puede bajar el software MySQL desde internet y usarlo sin pagar nada. Si lo desea, puede estudiar el código fuente y cambiarlo para adaptarlo a sus necesidades. El software MySQL usa la licencia GPL (GNU General Public License), para definir lo que puede y no puede hacer con el software en diferentes situaciones. Si no se encuentra cómodo con la GPL o necesita añadir código MySQL en una aplicación comercial, puede comprar una licencia comercial. El servidor de base de datos MySQL es muy rápido, fiable y fácil de usar.

El servidor MySQL también tiene una serie de características prácticas desarrolladas en cooperación con los usuarios.

MySQL Server se desarrolló originalmente para tratar grandes bases de datos mucho más rápido que soluciones existentes y ha sido usado con éxito en entornos de producción de alto rendimiento durante varios años como Cisco, Motorola y la NASA. MySQL Server ofrece hoy en día una gran cantidad de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos.

4.4.1 Principales características de MySQL

La siguiente lista describe algunas de las características más importantes del software de base de datos MySQL.

Interioridades y portabilidad

- Escrito en C y en C++
- Probado con un amplio rango de compiladores diferentes

- Funciona en diferentes plataformas.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-hilo mediante hilos del kernel. Pueden usarse fácilmente multiple CPUs si están disponibles.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en hilos.
- Joins (relaciones entre tablas) muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

Tipos de columnas

- Diversos tipos de columnas: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, y tipos espaciales OpenGIS.
- Registros de longitud fija y longitud variable.

Sentencias y funciones

- Soporte completo para operadores y funciones en las cláusulas de consultas SELECT y WHERE.

- Soporte completo para las cláusulas SQL GROUP BY y ORDER BY. Soporte de funciones de agrupación (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), y GROUP_CONCAT()).
- Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN cumpliendo estándares de sintaxis SQL y ODBC.
- Soporte para alias en tablas y columnas como lo requiere el estándar SQL.
- Es posible devolver el número de filas que serían afectadas usando una bandera al conectar con el servidor.
- El comando específico de MySQL SHOW puede usarse para obtener información acerca de la base de datos, el motor de base de datos, tablas e índices. El comando EXPLAIN puede usarse para determinar cómo el optimizador resuelve una consulta.
- Los nombres de funciones no colisionan con los nombres de tabla o columna. Por ejemplo, ABS es un nombre válido de columna. La única restricción es que para una llamada a una función, no se permiten espacios entre el nombre de función y el '(' a continuación.
- Puede mezclar tablas de distintas bases de datos en la misma consulta.

Seguridad

- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Escalabilidad y límites

- Soporte a grandes bases de datos. Usamos MySQL Server con bases de datos que contienen 50 millones de registros
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas.

El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

Conectividad

- Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows de la familia NT (NT,2000, XP, o 2003), los clientes pueden usar named pipes para la conexión. En sistemas Unix, los clientes pueden conectar usando archivos socket Unix.
- En MySQL 5.0, los servidores Windows soportan conexiones con memoria compartida si se inicializan con la opción memoria-compartida. Los clientes pueden conectar a través de memoria compartida usando la opción --protocol=memory.
- La interfaz para el conector ODBC (MyODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity). Por ejemplo, puede usar MS Access para conectar al servidor MySQL. Los clientes pueden ejecutarse en Windows o Unix. El código fuente de MyODBC está disponible. Todas las funciones para ODBC 2.5 están soportadas, así como muchas otras.

La interfaz para el conector J MySQL proporciona soporte para clientes Java que usen conexiones JDBC. Estos clientes pueden ejecutarse en Windows o Unix. El código fuente para el conector J está disponible.

4.4.2 Localización

- El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas.
- Soporte completo para distintos conjuntos de caracteres, incluyendo latin1 (ISO-8859-1), german, big5, ujis, y más. Por ejemplo, los caracteres escandinavos 'â', 'ä' y 'ö' están permitidos en nombres de tablas y columnas.

- Todos los datos se guardan en el conjunto de caracteres elegido. Todas las comparaciones para columnas normales de cadenas de caracteres no distinguen entre minúsculas y mayúsculas.
- La ordenación se realiza acorde al conjunto de caracteres elegido (usando colación Sueca por defecto). Es posible cambiarla cuando arranca el servidor MySQL. Soporta diferentes conjuntos de caracteres que deben ser especificados en tiempo de compilación y de ejecución.

Por estas razones y por las características voy a implementar la base de datos con MySQL.

4.5 Normalización

La base de datos tiene que estar muy bien estructurada por lo cual tiene que pasar por el proceso de la normalización pero que es la normalización a continuación lo voy a explicar.

La normalización es el proceso de reducir la redundancia de los datos en una base relacional. Los datos redundantes se refieren a los datos que han sido almacenados en más de una locación en la base de datos. Los datos no deben ser redundantes, lo cual significa que la redundancia de los datos se tiene que mantener al mínimo por varias razones.

Es innecesario almacenar la dirección de un empleado en más de una tabla. Porque con la repetición de datos solo hará que se use más espacio.

La normalización es la aplicación de poner unas simples reglas llamadas Primera, Segunda, y Tercera forma normal para asignar atributos a las entidades en el ERD (Diagramas de Entidad Relación). También hay niveles adicionales de normalización mas allá de la tercera forma normal como Boyce-Codd, Cuarto y quinto nivel de forma normal aunque generalmente solo se llega hasta la tercera forma normal.

La normalización es un proceso fundamental para el modelado y diseño de una base de datos relacional. Su propósito es el eliminar la redundancia de datos y evitar anomalías en la actualización de datos que pueden ocurrir en una base de datos sin normalizar.

Una base de datos que no es normalizada puede incluir datos que son contenidos en una o más tablas sin una razón aparente. Esto no es un optimo diseño para la seguridad, espacio en disco, velocidad para recibir datos, eficiencia en las actualizaciones de la base de datos, y lo más importante, la integridad de los datos.

Una base de datos antes de la normalización es una que no ha sido dividida en unidades lógicas más pequeñas, más tablas manejables.

En la normalización se considera lo siguiente:

- ¿Qué datos deben ser almacenados en la base de datos?
- ¿Cómo el usuario accesara la tabla?
- ¿Qué privilegios requiere el usuario?
- ¿Cómo debe ser agrupada la información en la base datos?
- ¿Qué tipos de datos son más comúnmente accesados?
- ¿Cómo son relacionados los datos en la base de datos?
- ¿Qué medidas deben ser tomadas para garantizar exactitud?

4.5.1 Ventajas de la Normalización

La normalización proporciona numerosos beneficios para el diseño de la base de datos, el diseño de una aplicación, y la implementación en la producción de la base de datos.

Algunos de los beneficios son:

- Se ganara organización
- La cantidad de datos redundantes será reducida
- La integridad de los datos se mantendrá más fácilmente dentro de la base de datos
- El proceso de diseño de aplicaciones será más flexible

- La seguridad será más fácil de manejar.

Con el proceso de normalización se gana organización, haciendo el trabajo de todos más fácil desde el usuario final quien accesa las tablas hasta el administrador quien es responsable del control y el mantenimiento de cada objeto en la base de datos.

La redundancia de datos es reducida la cual simplifica las estructuras de la base de datos y conserva espacio en disco.

Dado que la repetición de datos es minimizada o completamente eliminada la posibilidad de que haya datos inconsistentes es muy reducida.

Si la base de datos ha sido normalizada y dividida en pequeñas tablas se proporciona más flexibilidad mientras existan estructuras que se modifican. Es mucho más fácil el modificar una tabla pequeña con poca cantidad de información o datos que el modificar una tabla muy grande que tiene todos los datos.

4.5.2 Desventajas de la normalización

La desventaja de la normalización es que produce muchas tablas con un pequeño número de columnas. Estas columnas entonces tienen que ser relacionadas usando llaves primarias o externas para unirlas y después poderlas usar. Por ejemplo, una interrogante puede requerir la recuperación de datos de unas tablas múltiples normalizadas. Esto puede dar como resultado una unión difícil de tablas. Las tablas requeridas por la interrogante pueden ser probablemente una antes de la descomposición.

La descomposición de tablas tiene dos impactos principales la primera es el desempeño. Todas las uniones requeridas para la unión de registros son un proceso lento e incrementa el trabajo de la computadora.

El segundo impacto reta a los desarrolladores a codificar interrogantes, que regresan la información deseada, sin experimentar el impacto de la base de datos relacional que insiste en regresar una línea para cada posible combinación de valores relacionados si las tablas no están unidas apropiadamente por el desarrollador. Las filas adicionales que son regresadas por causa de las tablas que no han sido propiamente unidas son extrañas incoherencias. Esta colección de datos extraños es llamada producto cartesiano.

Una forma normal es un medio de medir niveles o profundidad, en la cual la base de datos ha sido normalizada. El nivel de normalización es determinado por la forma normal. A continuación se alistan las formas normales.

1.- La Primer forma normal
2.- La Segunda forma normal
3.- La Tercera forma normal
4.- El Boyce-Codd forma normal
5.- La Cuarta forma normal
6.- La Quinta forma normal

Las tres formas normales más comunes son las primeras tres, cada forma es un paso antes de otra forma. En pocas palabras si uno está en la segunda forma normal debió haber pasado forzosamente por la primera forma normal, de tal forma de que no se puedes empezar la segunda forma normal después de haber terminado la primera.

Generalmente este proceso solo llega hasta la tercera forma y después de la tercera forma tiende a ser conceptualmente difícil y solo es aplicable a un número limitado de situaciones ya que la tercera forma es el estándar y los demás solo deben de ser aplicados en circunstancias especificas.

4.5.3 Primer forma normal

El objetivo de la primer forma normal es de dividir la base de datos en unidades lógicas llamadas entidades, o tablas. Cuando cada entidad ha sido diseñada, una llave primaria es asignada a ella.

La entidad tiene un UID (llave) y todos los atributos deben de ser de un valor único. Un atributo repetidor o multivalor es un atributo o grupo de atributos que tendrán multivalores por un suceso del UID. Se tendrá que repetir el atributo mucha veces en cada estructura de la entidad en orden de capturar todas los posibles sucesos del evento para ser modelado.

La solución es la de mover atributos que se repiten a su propia entidad y crear una relación entre dos entidades descompuestas. Trabajando con tablas en diseño, se tendrá que mover las columnas repetidas a su propia tabla a lo largo con una copia de la entidad original UID como una llave externa. La entidad, asegura que cada atributo de la entidad es un valor único sin repetir atributos que estén permitidos.

4.5.4 Segunda forma normal

El objetivo es de tomar los registros que son solo parcialmente dependientes de la llave primaria y meterlos en otras tablas.

La entidad se encuentre en la primera forma normal y si una entidad tiene un UID compuesto, todos los no atributos UID deberán ser dependientes en todo de los atributos UID –no solo uno o algunos de ellos.

4.5.5 Tercer forma normal

El objetivo de esta forma es remover registros en una tabla que no es dependiente de una llave primaria. La entidad está en la segunda forma normal y un atributo UID no puede depender de otro atributo UID. Todos los no atributos UID deben depender directamente sobre todos los UID y ningún otro. Poniéndolo de otra manera los atributos no pueden ser atributos de ellos mismos. Si los atributos tienen atributos, ellos son entidades reales.

4.5.6 Boyce-Codd forma normal

Es en efecto cuando una entidad está dentro de la tercer forma normal, y cualquier atributo o combinación de atributos sobre cualquier atributo es funcionalmente dependiente es una llave candidata. Un atributo o combinación de atributos sobre cualquier atributo es funcionalmente dependiente es también *llamado determinante.*

La dependencia funcional puede ser derivada como sigue. Si la columna 2 es *funcionalmente dependiente de la columna 1, esto quiere decir que cada valor* en la columna 1 es asociado con uno y solo un valor en la columna 2.

Lo contrario no es verdad, sin embargo. Un valor particular para la columna 2 puede tener múltiples valores correspondientes en la columna 1.

Si hay solo un determinante sobre el cual otro atributo depende y es una llave candidato, la tercer forma normal y Boyce-Codd forma normal son idénticas. La diferencia entre la tercera y Boyce-Codd son que Boyce-Codd requiere todos los atributos que tienen otros atributos con dependencia funcionales sobre ellos, para ser una llave candidata y la tercer forma normal no lo hace. Boyce-Codd, *con estas pequeñas diferencias, es en efecto una versión más fuerte que* la tercer forma normal.

4.5.7 Cuarta forma normal

Es *en efecto cuando una entidad se encuentra en Boyce-Codd forma normal*, esta no tiene multivalores dependiente y tampoco ninguno de las siguientes condiciones:

1.- El atributo dependiente no es una sobre posición del atributo sobre el cual este depende.

2.- El determinante en unión o combinación con el atributo dependiente incluye la entidad entera.

Una dependencia de multivalores es una situación en la cual dos o más atributos son dependientes en un determinante y cada atributo dependiente tiene unos valores específicos asignados. Los valores en estos atributos dependientes son independientes uno de otro.

4.5.8 Quinta forma normal

También llamada *proyecto-unión forma normal* es cuando la cuarta forma normal esta efectuándose y la entidad no está unida con dependencias que no están relacionadas con la entidad poseedora de las llaves candidatas.

Cuando nosotros descomponemos entidades mientras aplicamos las reglas de normalización, deberíamos de poder reconstruir la entidad original mediante el hacer uniones entre dos entidades resultantes sin perder ningún registró y sin generar ninguno extra y sin generar líneas incorrectas. A esto se le llama unión-perdida menor

Una entidad en la quinta forma normal es cuando no se puede descomponer en muchas entidades pequeñas, las cuales tienen diferentes llaves que el original sin perder ningún registro. Si un registro fue perdido después de la descomposición, una unión-perdida menor no sería posible. Si todas las entidades descompuestas usan una de las llaves que tiene la entidad original *como llave, la unión de esas entidades daría como resultado una unión-*peridida*menor*. Con respecto a la entidad original, se consideraría todavía como la quinta forma normal sin descomponer.

Para el caso del sistema solo voy a llegar hasta la tercera forma normal

Capitulo V. Conexión con la base de datos

Una parte muy importante es saber cómo se va conectar el sistema con la base de datos siguiendo con lo que es java, cuenta con una herramienta muy poderosa para conectarse con la base de datos su nombre es JDBC ahora voy a explicar cómo funciona.

5.1 JDBC

Java Database Connectivity (JDBC) es una interfaz de acceso a bases de datos estándar SQL que proporciona un acceso uniforme a una gran variedad de bases de datos relacionales. JDBC también proporciona una base común para la construcción de herramientas y utilidades de alto nivel.

5.2 Conectividad de Bases de Datos de Java (JDBC)

JDBC es la API para la ejecución de sentencias SQL. (Como punto de interés JDBC es una marca registrada y no un acrónimo, no obstante a menudo es conocido como "Java Database Connectivity"). Consiste en un conjunto de clases e interfaces escritas en el lenguaje de programación Java. JDBC suministra una API estándar para los desarrolladores y hace posible escribir aplicaciones de base de datos usando una API puro Java.

Usando JDBC es fácil enviar sentencias SQL virtualmente a cualquier sistema de base de datos. En otras palabras, con la API JDBC, no es necesario escribir un programa que acceda a una base de datos Sybase, otro para acceder a Oracle y otro para acceder a Informix. Un único programa escrito usando la API JDBC y el programa será capaz de enviar sentencias SQL a la base de datos apropiada. Y, con una aplicación escrita en el lenguaje de programación Java, tampoco es necesario escribir diferentes aplicaciones para ejecutar en diferentes plataformas. La combinación de Java y JDBC permite al programador escribir una sola vez y ejecutarlo en cualquier entorno.

Java, siendo robusto, seguro, fácil de usar, fácil de entender, y descargable automáticamente desde la red, es un lenguaje base excelente para aplicaciones de base de datos.

JDBC expande las posibilidades de Java. Por ejemplo, con Java y JDBC, es posible publicar una página web que contenga un applet que usa información obtenida de una base de datos remota. O una empresa puede usar JDBC para conectar a todos sus empleados (incluso si usan un conglomerado de máquinas Windows, Macintosh y UNIX) a una base de datos interna vía intranet. Con cada vez más y más programadores desarrollando en lenguaje Java, la necesidad de acceso fácil a base de datos desde Java continúa creciendo.

Simplemente JDBC hace posible estas tres cosas:

- Establece una conexión con la base de datos.
- Envía sentencias SQL
- Procesa los resultados.

5.3 Drivers JDBC

Para usar JDBC con un sistema gestor de base de datos en particular, es necesario disponer del driver JDBC apropiado que haga de intermediario entre ésta y JDBC. Dependiendo de varios factores, este driver puede estar escrito en Java puro, o ser una mezcla de Java y métodos nativos JNI (Java Native Interface).

5.3.1 Tipos de drivers JDBC

Los drivers que son susceptibles de clasificarse en una de estas cuatro categorías.

1. Puente JDBC-ODBC más driver ODBC: El producto de JavaSoft suministra acceso vía drivers ODBC. Nótese que el código binario ODBC, y en muchos casos el código cliente de base de datos, debe cargarse en cada máquina cliente que use este driver.

Como resultado, este tipo de driver es el más apropiado en una red corporativa donde las instalaciones clientes no son un problema mayor, o para una aplicación en el servidor escrito en Java en una arquitectura en tres-niveles.

2. Driver Java parcialmente Nativo. Este tipo de driver convierte llamadas JDBC en llamadas del API cliente para Oracle, Sybase, Informix, DB2 y otros DBMS. Nótese que como el driver puente, este estilo de driver requiere que cierto código binario sea cargado en cada máquina cliente.
3. Driver Java nativo JDBC-Net. Este driver traduce llamadas JDBC al protocolo de red independiente del DBMS que después es traducido en el protocolo DBMS por el servidor. Este middleware en el servidor de red es capaz de conectar a los clientes puros Java a muchas bases de datos diferentes. El protocolo específico usado dependerá del vendedor. En general esta es la alternativa más flexible.
4. Driver puro Java y protocolo-nativo. Este tipo de driver convierte llamadas JDBC en el protocolo de la red usado por DBMS directamente. Esto permite llamadas directas desde la máquina cliente al servidor DBMS y es la solución más práctica para accesos en intranets. Dado que muchos de estos protocolos son propietarios, los fabricantes de bases de datos serán los principales suministradores.

5.4 Conexión

5.4.1 Vista Preliminar

Un objeto **Connection** representa una conexión con una base de datos. Una sesión de conexión incluye las sentencias SQL que se ejecutan y los resultados que son devueltos después de la conexión. Una única aplicación puede tener una o más conexiones con una única base de datos, o puede tener varias conexiones con varias bases de datos diferentes.

5.4.2 Apertura de una conexión

La forma estándar de establecer una conexión a la base de datos es mediante la llamada al método **DriverManager.getConnection()**. Este método toma una cadena que contiene una URL. La clase DriverManager, referida como la capa de gestión JDBC, intenta localizar un driver que pueda conectar con la base de datos representada por la URL. La clase DriverManager mantiene una lista de clases Driver registradas y cuando se llama al método getConnection(), se checa con cada driver de la lista hasta que encuentra uno que pueda conectar con la base de datos especificada en la URL. El método connect() de Driver usa esta URL para establecer la conexión.

Un usuario puede evitar la capa de gestión de JDBC y llamar a los métodos de Driver directamente. Esto puede ser útil en el caso raro que dos drivers puedan conectar con la base de datos y el usuario quiera seleccionar uno explícitamente. Normalmente, de cualquier modo, es mucho más fácil dejar que la clase DriverManager maneje la apertura de la conexión.

5.4.3 Envío de Sentencias SQL

Una vez que la conexión se haya establecido, se usa para pasar sentencias SQL a la base de datos subyacente. JDBC no pone ninguna restricción sobre los tipos de sentencias que pueden enviarse: esto da un alto grado de flexibilidad, permitiendo el uso de sentencias específicas de la base de datos o incluso sentencias no SQL. Se requiere de cualquier modo, que el usuario sea responsable de asegurarse que la base de datos subyacente sea capaz de procesar las sentencias SQL que le están siendo enviadas y soportar las consecuencias si no es así. Por ejemplo, una aplicación que intenta enviar una llamada a un procedimiento almacenado a una DBMS que no soporta procedimientos almacenados no tendrá éxito y generará una excepción. JDBC requiere que un driver cumpla al menos ANSI SQL-2 Entry Level para ser designado JDBC COMPLIANT. Esto significa que los usuarios pueden contar al menos con este nivel de funcionalidad.

JDBC suministra tres clases para el envío de sentencias SQL y tres métodos en la interfaz Connection para crear instancias de estas tres clases. Estas clases y métodos son los siguientes:

1. **Statement** creada por el método createStatement(). Un objeto Statement se usa para enviar sentencias SQL simples
2. **PreparedStatement** creada por el método prepareStatement(). Un objeto PreparedStatement se usa para sentencias SQL que toman uno o más parámetros como argumentos de entrada (parámetros IN). PreparedStatement tiene un grupo de métodos que fijan los valores de los parámetros IN, los cuales son enviados a la base de datos cuando se procesa la sentencia SQL. Instancias de PreparedStatement extienden Statement y por tanto heredan los métodos de Statement. Un objeto PreparedStatement es potencialmente más eficiente que un objeto Statement porque este ha sido precompilado y almacenado para su uso futuro.
3. **CallableStatement** creado por el método prepareCall(). Los objetos CallableStatement se usan para ejecutar procedimientos almacenados SQL un grupo de sentencias SQL que son llamados mediante un nombre, algo parecido a una función. Un objeto CallableStatement hereda métodos para el manejo de los parámetros IN de PreparedStatement, y añade métodos para el manejo de los parámetros OUT e INOUT.

Una de las áreas de dificultad es que aunque muchas SGBD (sistema de gestión de base de datos) usan un formato estándar de SQL para la funcionalidad básica, estos no conforman la sintaxis más recientemente definida o semántica para funcionalidades más avanzadas. Por ejemplo, no todas las bases de datos soportan procedimientos almacenados o joins de salida, y aquellas que lo hacen no son consistentes con otras. Es de esperar que la porción de SQL que es verdaderamente estándar se expandirá para incluir más y más funcionalidad. Entretanto, de cualquier modo, la API de JDBC debe soportar SQL tal como es.

Una forma en que la API JDBC trata este problema es permitir que cualquier cadena de búsqueda se pase al driver subyacente del SGBD. Esto quiere decir que una aplicación es libre de usar la sentencia SQL tal como quiera, pero se corre el riesgo de recibir un error en el SGBD. De hecho una consulta de una aplicación incluso no tiene porque ser SQL, o puede ser una derivación especializada de SQL diseñada para especificas SGBD (para consultas a imágenes o documentos por ejemplo).

Una segunda forma en que JDBC trata este problema es proveer cláusulas de escape al estilo de ODBC.

La sintaxis de escape provee una sintaxis JDBC estándar para varias de las áreas más comunes de divergencia SQL. Por ejemplo, hay escapes para literales de fecha o procedimientos almacenados.

Para aplicaciones complejas, JDBC trata la conformidad SQL de una tercera manera. Y es proveer información descriptiva sobre el SGBD por medio de la interfaz DatabaseMetaData por la que las aplicaciones pueden adaptarse a los requerimientos y posibilidades de cada SGBD:

Como la API JDBC se usará como base para el desarrollo de herramientas de acceso y APIs de alto nivel, direccionará el problema de la conformidad a cualquiera de estas. La designación "JDBC COMPLIANT" se creó para situar un nivel estándar de funcionalidad JDBC en la que los usuarios puedan confiar. Para poder usar esta designación, un driver debe soportar al menos el nivel de entrada ANSI SQL-2 Entry Level. Los desarrolladores de drivers pueden cerciorarse que sus drivers cumplan estas especificaciones mediante la suite de pruebas disponible en la API JDBC.

La designación "JDBC COMPLIANT" indica que la implementación JDBC de un vendedor ha pasado las pruebas de conformidad suministrados por JavaSoft. Estas pruebas de conformidad chequan la existencia de todas las clases y métodos definidos en la API JDBC, y que la checan tanto como es posible que la funcionalidad SQL Entry Level esté disponible.

Tales pruebas no son exhaustivas, por supuesto, y JavaSoft no está distribuyendo implementaciones de vendedores, pero esta definición de conformidad tiene algún grado de seguridad en una implementación JDBC. Con la mayor aceptación de JDBC por parte de vendedores de bases de datos, de servicios de Internet y desarrolladores, JDBC se está convirtiendo en el estándar de acceso a bases de datos.

Capítulo VI. Interfaz grafica del usuario

En el contexto del proceso de interacción humano-computadora, la interfaz gráfica de usuario, es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

6.1 Interfaz gráfica de usuario

La interfaz gráfica de usuario (en inglés Graphical User Interface, GUI) es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora. Sus principales funciones son:

- Manipulación de archivos y directorios
- Herramientas de desarrollo de aplicaciones
- Comunicación con otros sistemas
- Información de estado
- Configuración de la propia interfaz y entorno
- Intercambio de datos entre aplicaciones
- Control de acceso
- Sistema de ayuda interactivo.

Nos encontramos con dos tipos de interfaz de usuario:

- Interfaces alfanuméricas (intérpretes de mandatos).
- Interfaces gráficas de usuario (GUI, Graphics User Interfaces), las que permiten comunicarse con la computadora de una forma muy rápida e intuitiva.

Y pueden ser de hardware o de software:

- En el primer caso se trata de un conjunto de dispositivos que permiten la interacción humano-computadora, de modo que permiten ingresar y tomar datos de la computadora.

- También están las interfaces de software que son programas o parte de ellos que permiten expresar nuestros deseos a la computadora.

Antes de la aparición de la tarjeta gráfica moderna, los programas basaban su interfaz en la representación de la información mediante caracteres alfanuméricos y algunos caracteres especiales que permitían dibujar cajas de colores para diferenciar las distintas áreas del programa (representación de tablas, resultados de operaciones, áreas de edición, etc.)

La historia reciente de la informática está indisolublemente unida a las interfaces gráficas, puesto que los sistemas operativos gráficos que han ocasionado grandes consecuencias en la industria del software y del hardware.

Las interfaces gráficas surgen de la necesidad de hacer las computadoras más accesibles para el uso de los usuarios comunes. La mayoría de las computadoras domésticos o requerían conocimientos de BASIC (el 95% de ellos mostraban el intérprete BASIC al encenderse) u ofrecían una interfaz de línea de comandos (como los sistemas operativos CP/M o los diferentes OS del Apple II), lo que requería conocimientos por encima de la media si se deseaba hacer algo más que usarlo como consola de videojuegos.

Esta limitación fue salvada gracias al desarrollo de los entornos gráficos, que permitieron que las personas pudieran acceder a una computadora sin tener que pasar por el tortuoso proceso de tener que aprender a manejar un entorno bajo línea de comandos.

La aparición del primer Mac de Apple, la versión 1.0 de Windows y los primeros modelos de ratón origino que muchos programas comenzaran a usar interfaces gráficas, originando que cada programa utilizara su propio sistema de representación y modo de interactuar con el usuario. Muchas veces el cambio de interfaz consistía en portar el anterior aspecto del programa en texto a una nueva basada en gráficos que resultaba más atractiva para el usuario pero en ocasiones poco útiles.

Además, por cada programa que se utilizara había que aprender a moverse en un nuevo entorno con lo que ello acarreaba en complejidad de uso.

Con la creciente popularidad de Windows 3.0, muchos programas comenzaron a ofrecer una versión gráfica para Windows. El sistema operativo gráfico tiene la ventaja de poseer un GUI unificado, facilitando no solo el uso de estos programas, sino también su desarrollo. Con Windows 95, casi todas las aplicaciones terminaron corriendo bajo este sistema operativo. Sistemas como Linux desarrollaron también entornos de ventanas, provocando que hoy día prácticamente todos los sistemas operativos posean su propio GUI común a todas las aplicaciones que corran bajo ese sistema.

Para contar con una buena interfaz hacia el usuario es necesario tener clara la misión y la visión del sistema, balanceando las necesidades de los que ofrecen la información como de los que la consultan. Es importante también, determinar el contenido y la funcionalidad, especificar la organización, la navegación, las secciones y los sistemas de búsqueda.

Los usuarios quieren encontrar la información fácil y rápido. Una mala planeación de la arquitectura de nuestra interfaz puede crear usuarios confundidos, frustrados y enojados.

Cada usuario tiene diferentes necesidades, es importante soportar diferentes formas para encontrar información. Algunos usuarios saben exactamente qué es lo que buscan, quieren encontrarlo y terminar tan pronto sea posible. Otros usuarios no saben exactamente lo que buscan.

En un sistema bien diseñado, los usuarios pueden tener tanto búsquedas precisas como exploraciones que les ayuden a encontrar su información. Hay que contemplar esto ya que la satisfacción del usuario no sólo se logra con buena tecnología y gráficas atractivas.

Haciendo un estudio de la inmobiliaria, sus necesidades y la manera de manejar la información el sistema va a estar diseñado de la siguiente manera:

El sistema maneja una ventana para ingresar en la cual se introduce el nombre de usuario y contraseña. Cuenta con dos tipos de usuario el administrador y el empleado; el administrador puede consultar, modificar y eliminar la información que se maneja y el empleado solo puede consultar información y lo único que puede eliminar son sus clientes.

Al ingresar al sistema se inicia una sesión en la que aparecen los clientes del usuario y citas o notas, que otros usuarios le envíen; si el usuario es de tipo administrador aparece la ventana con todas las opciones, si es tipo empleado aparecen las opciones de consulta, en los dos casos como el sistema va dirigido a una inmobiliaria lo que manejan más en su labor de trabajo es una agenda para apuntar nombres de clientes, direcciones de inmuebles, por lo tanto el sistema tiene una navegación tipo agenda dividido por categorías en donde se encuentra una pestaña principal y sus diferentes opciones, así se puede ir navegando seleccionando la pestaña que quiera.

La pestaña que se muestra primero es la de pendientes en la cual se encuentran las citas, notas o pendientes; esto porque es lo primero que revisan al llegar a la oficina. Se pueden visualizar las citas que tiene programadas y las notas o recados les dejaron en el sistema, en base a esto van a trabajar todo el día.

El sistema cuenta con una barra de botones. Cuando el usuario ya esté familiarizado con el sistema se le va hacer más fácil darle clic al botón de la opción que desea ingresar o en su defecto también puede navegar con el teclado dentro de esta barra obteniendo con esto una navegación más rápida. Los botones cuentan con un tip (o globo) que al poner el cursor sobre el botón aparece un breve mensaje que indica, diferentes componentes tienen esta misma opción por si el usuario no sabe o no se acuerda para que sirve.

En la parte superior aparece una barra de menús dividida por categorías en la cual también tiene todas las diferentes opciones del sistema, tiene unos pequeños iconos que son iguales a la barra de botones para que el usuario los relacione y los pueda aprender más rápido.

Cuenta con teclas rápidas alt + una letra la cual se encuentra subrayada; estos menús también cuentan con un tip para ver para que sirve; y aparecen las teclas rápidas al final de la leyenda.

Entonces para llegar a una opción tiene 3 rutas por medio de la barra de menús, la barra de botones o por las pestañas esto para facilitarle al usuario la navegación, lo que le parezca más fácil.

Para hacer todo esto voy a utilizar Swing de Java.

6.2 Swing

Las Internet Foundation Classes (IFC) eran una biblioteca gráfica para el lenguaje de programación Java desarrollada originalmente por Netscape y que se publicó en 1996.

Desde sus inicios el entorno Java ya contaba con una biblioteca de componentes gráficos conocida como AWT. Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo. Entonces una aplicación Java corriendo en Windows usaría el botón estándar de Windows y una aplicación corriendo en UNIX usaría el botón estándar de Motif. En la práctica esta tecnología no funcionó de forma óptima:

- Al depender fuertemente de los componentes nativos del sistema operativo el programador AWT estaba confinado a un mínimo denominador común entre ellos. Es decir que sólo se disponen en AWT de las funcionalidades comunes en todos los sistemas operativos.
- El comportamiento de los controles varía mucho de sistema a sistema y se vuelve muy difícil construir aplicaciones portables. Fue por esto que el eslogan de Java "Escríbalo una vez, ejecútelo en todos lados" fue parodiado como "Escríbalo una vez, pruébelo en todos lados".

En cambio, los componentes de IFC eran mostrados y controlados directamente por código Java independiente de la plataforma. De dichos componentes se dice con frecuencia que son componentes ligeros, dado que no requieren reservar recursos nativos del sistema de ventanas del sistema operativo. Además al estar enteramente desarrollado en Java aumenta su *portabilidad asegurando un comportamiento idéntico en diferentes plataformas.*

Además de los componentes ligeros suministrados originalmente por la IFC, *Swing introdujo un mecanismo que permitía que el aspecto de cada* componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma. También contiene un conjunto de herramientas que nos permiten crear una interfaz atractiva para los usuarios.

Swing es una biblioteca gráfica para Java que forma parte de las Java *Foundation Classes (JFC). Incluye componentes para interfaz gráfica de* usuario tales como cajas de texto, botones, desplegables y tablas.

6.2.1 Arquitectura

Swing es una plataforma independiente, modelo vista controlador GUI framework para Java. Sigue un simple modelo de programación por hilos, y posee las siguientes principales características:

- Independencia de plataforma: Swing es una plataforma independiente *en ambos términos de su expresión (Java) y de su implementación (no-*nativa interpretación universal de componentes).
- Extensibilidad: Swing es una arquitectura altamente particionada que permite la utilización de diferentes plugins en específicos interfaces de diferentes frameworks: Los usuarios pueden proveer sus propias implementaciones modificadas para sobrescribir las implementaciones por defecto.

En general, los usuarios de Swing pueden extender el framework para: extender clases existentes (framework); proveyendo alternativas de implementación para elementos esenciales.

- Orientado a componentes: Swing es un framework basado en componentes. La diferencia entre objetos y componentes es un punto bastante sutil: *concisamente, un componente es un objeto de buena* conducta con un patrón conocido y especificado característico del comportamiento.
- Adaptable: Dado el modelo de representación programático del framework de Swing, el control permite representar diferentes 'look and feel' (desde MacOS look and feel hasta Windows XP look and feel). Más allá, los usuarios pueden proveer su propia implementación look and feel, que permitirá cambios uniformes en el look and feel existente en las *aplicaciones Swing sin efectuar ningún cambio al código de aplicación.*
- Lightweight UI: La magia de la flexibilidad de configuración de Swing, es también debido al hecho de que no utiliza los controles del GUI del OS nativo del host para la representación, pero usa algo de sus controles programados, con el uso de los APIs 2D de Java.

De lo expresado anteriormente, se puede sacar como conclusión los diferentes aspectos:

1. Tiene todas las características de AWT.
2. Un conjunto de componentes de mayor nivel.
3. Diseño en Java, no depende del código nativo.

6.2.2 Ventajas

- El diseño en Java puro posee menos limitaciones de plataforma.
- El desarrollo de componentes Swing es más activo.
- Los componentes de Swing soportan más características.

6.2.3 Desventajas

- *La mayoría de los navegadores no incluyen clases Swing, por lo que es* necesario utilizar un plugin Java.

- Los componentes Swing generalmente son más lentos y presentan más problemas debido a que están hechos en Java puro, y suelen presentar problemas relacionados con vídeo en varias plataformas.
- No siempre tienen el mismo aspecto que en el sistema donde fueron diseñados.

Capítulo VII. Esquema cliente servidor

El sistema va a estar instalado en varias maquinas por lo cual una va servir como servidor y las demás van a estar en red conectadas al sistema. Está diseñado para una arquitectura cliente/servidor el protocolo de comunicación en la red va ser el TCP/IP a continuación explicare los diferentes términos y las razones por las cuales utilizarlos.

7.1 Arquitectura cliente servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores estos últimos responden a la demanda del cliente que la produjo.

Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es Internet.

Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

7.2 Elementos de la arquitectura cliente/servidor

En esta aproximación, y con el objetivo de definir y delimitar el modelo de referencia de una arquitectura cliente/servidor, debemos identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos:

- Presentación/captación de información.
- Procesos.
- Almacenamiento de la información.
- Cliente.

- Comunicaciones.
- Servidores.

Una microcomputadora conectada a una red, que le permite acceder y gestionar una serie de recurso el cual se perfila como un puesto de trabajo universal. Nos referimos a una microcomputadora conectada al sistema de información y en el que se realiza una parte mayoritaria de los procesos.

Debemos destacar que la estación de trabajo basada en una *microcomputadora conectada a una red, favorece la flexibilidad y el dinamismo* en las organizaciones. Entre otras razones, porque permite modificar la ubicación de los puestos de trabajo, dadas las ventajas de la red.

Arquitectura

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Debemos señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura cliente/servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

Cliente

Es el que *inicia un requerimiento de servicio*. El *requerimiento inicial* puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

Los servidores o back-end

Es una máquina que suministra una serie de servicios como bases de datos, archivos, comunicaciones). Los servidores, según la especialización y los requerimientos de los servicios que debe suministrar pueden ser:

- Mainframes
- Minicomputadoras
- Especializados (dispositivos de red, imagen, etc.)

Una característica a considerar es que los diferentes servicios, según el caso, pueden ser suministrados por un único servidor o por varios servidores especializados.

Las comunicaciones

En sus dos vertientes:

- Infraestructura de redes
- Infraestructura de comunicaciones

Infraestructura de redes

Componentes hardware y software que garantizan la conexión física y la transferencia de datos entre los distintos equipos de la red.

Infraestructura de comunicaciones

Componentes hardware y software que permiten la comunicación y su gestión, entre los clientes y los servidores. La arquitectura cliente/servidor es el resultado de la integración de dos culturas.

Por un lado, la del Mainframe que aporta capacidad de almacenamiento, integridad y acceso a la información y, por el otro, la de la computadora que aporta facilidad de uso (cultura de PC), bajo costo, presentación atractiva (aspecto lúdico) y una amplia oferta en productos y aplicaciones.

7.3 Características del modelo cliente/servidor

En el modelo cliente/servidor podemos encontrar las siguientes características:

1. El cliente y el servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
2. Las funciones de cliente y servidor pueden estar en plataformas separadas, o en la misma plataforma.
3. Un servidor da servicio a múltiples clientes en forma concurrente.
4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
5. La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
6. Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.
7. También es importante hacer notar que las funciones cliente/servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

8. Además se constituye como el nexo de unión más adecuado para reconciliar los sistemas de información basados en mainframes o *minicomputadoras, con aquellos otros sustentados en entornos* informáticos pequeños y estaciones de trabajo.
9. Designa un modelo de construcción de sistemas informáticos de carácter distribuido.
10. Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede *acceder a los recursos de este host central y otros sistemas de la* organización ponen a su servicio.

En conclusión, cliente/servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, *en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar* algunas de las posibilidades.

7.4 Ventajas de la arquitectura cliente-servidor

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores *por separado.*

7.5 Red

Una red es un sistema donde los elementos que lo componen (por lo general computadoras) son autónomas y están conectadas entre sí por medios físicos y/o lógicos y que pueden comunicarse para compartir recursos. Independientemente a esto, definir el concepto de red implica diferenciar entre el *concepto de red física* y *red de comunicación*.

Respecto a la estructura física, los modos de conexión física, los flujos de datos, *etc*; *una red la constituyen dos o más computadoras que comparten* determinados recursos, sea hardware (impresoras, sistemas de almacenamiento) o sea software (aplicaciones, archivos, datos). Desde una perspectiva más comunicativa, podemos decir que existe una red cuando se encuentran involucrados un componente humano que comunica, un componente tecnológico (computadoras, televisión, telecomunicaciones) y un componente administrativo (institución o instituciones que mantienen los servicios). En fin, una red, más que varias computadoras conectadas, la *constituyen varias personas que solicitan, proporcionan e intercambian* experiencias e informaciones a través de sistemas de comunicación.

7.5.1 Estructura de las redes

Las redes tienen tres niveles de componentes: software de aplicaciones, software de red y hardware de red.

- El software de aplicaciones, son los programas que se comunican con los usuarios de la red y permiten compartir.
- *El software de red, son los programas que establecen protocolos para* que las computadoras se comuniquen entre sí. Dichos protocolos se aplican enviando y recibiendo grupos de datos formateados denominados paquetes.

- El hardware de red, formado por los componentes materiales que unen las computadoras. Dos componentes importantes son los medios de transmisión que transportan las señales de las computadoras (típicamente cables o fibras ópticas) y el adaptador de red, que permite acceder al medio material que conecta a las computadoras, recibir paquetes desde el software de red y transmitir instrucciones y peticiones a otras computadoras.

En resumen, las redes están formadas por conexiones entre grupos de computadoras y dispositivos asociados que permiten a los usuarios la transferencia electrónica de información. En estas estructuras, las diferentes computadoras se denominan estaciones de trabajo y se comunican entre sí a través de un cable o línea telefónica conectada a los servidores.

Dichos servidores son computadoras como las estaciones de trabajo pero con funciones administrativas y están dedicados en exclusiva a supervisar y controlar el acceso a la red y a los recursos compartidos. Además de las computadoras, los cables o la línea telefónica, existe en la red el módem para permitir la transferencia de información convirtiendo las señales digitales a analógicas y viceversa, también existen en esta estructura los llamados hubs y switches con la función de llevar a cabo la conectividad.

7.5.2 Protocolo de comunicaciones

Para que dos o más computadoras puedan conectarse a través de una red y ser capaces de intercambiar datos de una forma ordenada, deben seguir un protocolo de comunicaciones que sea aceptado por todos ellos. El protocolo define las reglas que se deben seguir en la comunicación, por ejemplo, enseñar a los niños a decir por favor y gracias es una forma de indicarles un protocolo de educación, y si alguna vez se olvidan de dar las gracias por algo, seguro que reciben una reprimenda de sus mayores.

Hay muchos protocolos disponibles para ser utilizados; por ejemplo, el protocolo HTTP define como se van a comunicar los servidores y navegadores web y el protocolo SMTP define la forma de transferencia del correo electrónico. Estos protocolos, son protocolos de aplicación que actúan al nivel de superficie, pero también hay otros protocolos de bajo nivel que actúan por debajo del nivel de aplicación y que son más complicados, aunque, afortunadamente, como programadores Java, no será necesario tener excesivo conocimiento de los protocolos de bajo nivel; nada que vaya más allá del conocimiento de su existencia.

7.5.3 Capas de red

Las redes están separadas lógicamente en capas, o niveles, o layers; desde el nivel de aplicación en la parte más alta hasta el nivel físico en la parte más baja. Los detalles técnicos de la división en capas o niveles de la red se escapa de la misión de este proyecto y, además, no es un conocimiento imprescindible para desarrollar programas Java que se comuniquen a través de la red, ya que la gente de JavaSoft se ha encargado de ocultar toda la dificultad que involucra el manejo de los protocolos de redes de bajo nivel y las capas de más bajo nivel del modelo de comunicaciones.

La única capa interesante para el usuario y el programador es el nivel de aplicación, que es el que se encarga de obtener los datos en una máquina desde esta capa y soltarlos en otra máquina en esta misma capa, los pasos intermedios y los saltos de capas que se hayan producido por el camino, no resultan de interés, ya que su uso está oculto en el lenguaje Java.

7.6 Protocolo TCP/IP

El protocolo de internet (IP) y el protocolo de transmisión (TCP), fueron desarrollados inicialmente en 1973 por el informático estadounidense Vinton Cerf como parte de un proyecto dirigido por el ingeniero norteamericano Robert Kahn y patrocinado por la Agencia de Programas Avanzados de Investigación (ARPA, siglas en inglés) del Departamento Estadounidense de Defensa.

Internet comenzó siendo una red informática de ARPA (llamada ARPAnet) que conectaba redes de computadoras de varias universidades y laboratorios en investigación en Estados Unidos. World Wibe Web se desarrolló en 1989 por el informático británico Timothy Berners-Lee para el Consejo Europeo de Investigación Nuclear (CERN, siglas en francés).

TCP/IP es el protocolo común utilizado por todas las computadoras conectados a Internet, de manera que éstos puedan comunicarse entre sí. Hay que tener en cuenta que en Internet se encuentran conectadas computadoras de clases muy diferentes y con hardware y software incompatibles en muchos casos, además de todos los medios y formas posibles de conexión. Aquí se encuentra una de las grandes ventajas del TCP/IP, pues este protocolo se encargará de que la comunicación entre todos sea posible. TCP/IP es compatible con cualquier sistema operativo y con cualquier tipo de hardware.

7.6.1 Características de TCP/IP

Ya que dentro de un sistema TCP/IP los datos transmitidos se dividen en pequeños paquetes, éstos resaltan una serie de características.

- La tarea de IP es llevar los datos a granel (los paquetes) de un sitio a otro. Las computadoras que encuentran las vías para llevar los datos de una red a otra (denominadas enrutadores) utilizan IP para trasladar los datos. En resumen IP mueve los paquetes de datos a granel, mientras TCP se encarga del flujo y asegura que los datos estén correctos.
- Las líneas de comunicación se pueden compartir entre varios usuarios. Cualquier tipo de paquete puede transmitirse al mismo tiempo, y se ordenará y combinará cuando llegue a su destino. Compare esto con la manera en que se transmite una conversación telefónica. Una vez que establece una conexión, se reservan algunos circuitos para usted, que no puede emplear en otra llamada, aun si deja esperando a su interlocutor por veinte minutos.
- Los datos no tienen que enviarse directamente entre dos computadoras.

Cada paquete pasa de computadora en computadora hasta llegar a su destino. Éste, claro está, es el secreto de cómo se pueden enviar datos y mensajes entre dos computadoras aunque no estén conectadas directamente entre sí. Lo que realmente sorprende es que sólo se necesitan algunos segundos para enviar un archivo de buen tamaño de una máquina a otra, aunque estén separadas por miles de kilómetros y pese a que los datos tienen que pasar por múltiples computadoras. Una de las razones de la rapidez es que, cuando algo anda mal, sólo es necesario volver a transmitir un paquete, no todo el mensaje.

- Los paquetes no necesitan seguir la misma trayectoria. La red puede llevar cada paquete de un lugar a otro y usar la conexión más idónea que esté disponible en ese instante. No todos los paquetes de los mensajes tienen que viajar, necesariamente, por la misma ruta, ni necesariamente tienen que llegar todos al mismo tiempo.
- La flexibilidad del sistema lo hace muy confiable. Si un enlace se pierde, el sistema usa otro. Cuando usted envía un mensaje, el TCP divide los datos en paquetes, ordena éstos en secuencia, agrega cierta información para control de errores y después los lanza hacia fuera, y los distribuye. En el otro extremo, el TCP recibe los paquetes, verifica si hay errores y los vuelve a combinar para convertirlos en los datos originales. De haber error en algún punto, el programa TCP destino envía un mensaje solicitando que se vuelvan a enviar determinados paquetes.

7.6.2 Cómo funciona TCP/IP

IP a diferencia del protocolo X.25, que está orientado a conexión, es sin conexión. Está basado en la idea de los datagramas interred, los cuales son transportados transparentemente, pero no siempre con seguridad, desde la fuente de origen hasta el destinatario, quizás recorriendo varias redes mientras viaja.

El protocolo IP trabaja de la siguiente manera; la capa de transporte toma los mensajes y los divide en datagramas, de hasta 64K octetos cada uno. Cada datagrama se transmite a través de la red interred, posiblemente fragmentándose en unidades más pequeñas, durante su recorrido normal. Al final, cuando todas las piezas llegan a la máquina destinataria, la capa de transporte los reensambla para así reconstruir el mensaje original.

Un datagrama IP consta de una parte de cabecera y una parte de texto. La cabecera tiene una parte fija de 20 octetos y una parte opcional de longitud variable. El campo Versión indica a qué versión del protocolo pertenece cada uno de los datagramas. Mediante la inclusión de la versión en cada datagrama, no se excluye la posibilidad de modificar los protocolos mientras la red se encuentre en operación.

El campo Opciones se utiliza para fines de seguridad, encaminamiento fuente, informe de errores, depuración, sellado de tiempo, así como otro tipo de información. Esto, básicamente, proporciona un escape para permitir que las versiones subsiguientes de los protocolos incluyan información que actualmente no está presente en el diseño original. También, para permitir que los experimentadores trabajen con nuevas ideas y para evitar, la asignación de bits de cabecera a información que muy rara vez se necesita.

Debido a que la longitud de la cabecera no es constante, un campo de la cabecera, IHL, permite que se indique la longitud que tiene la cabecera en palabras de 32 bits. El valor mínimo es de 5. Tamaño 4 bits.

El campo Tipo de servicio le permite al servidor indicarle a la subred el tipo de servicio que desea. Es posible tener varias combinaciones con respecto a la seguridad y la velocidad. Para voz digitalizada, por ejemplo, es más importante la entrega rápida que corregir errores de transmisión. En tanto que, para la transferencia de archivos, resulta más importante tener la transmisión fiable que entrega rápida. También, es posible tener algunas otras combinaciones, desde un tráfico rutinario, hasta una anulación instantánea. Tamaño 8 bits.

La Longitud total incluye todo lo que se encuentra en el datagrama tanto la cabecera como los datos. La máxima longitud es de 65 536 octetos (bytes). Tamaño 16 bits.

El campo Identificación se necesita para permitir que el servidor destinatario determine a qué datagrama pertenece el fragmento recién llegado. Todos los fragmentos de un datagrama contienen el mismo valor de identificación. Tamaño 16 bits.

Enseguida viene un bit que no se utiliza, y después dos campos de 1 bit. Las letras DF *quieren decir no fragmentar. Esta es una orden para que* las pasarelas no fragmenten el datagrama, porque el extremo destinatario es incapaz de poner las partes juntas nuevamente. Por ejemplo, supóngase que se tiene un datagrama que se carga en un micro pequeño para su ejecución; podría marcarse con DF porque la ROM de micro espera el programa completo en un datagrama. Si el datagrama no puede pasarse a través de una red, se deberá encaminar sobre otra red, o bien, desecharse.

Las letras MF significan más fragmentos. Todos los fragmentos, con excepción *del último, deberán tener ese bit puesto. Se utiliza como una verificación doble* contra el campo de Longitud total, con objeto de tener seguridad de que no faltan fragmentos y que el datagrama entero se reensamble por completo.

El desplazamiento de fragmento indica el lugar del datagrama actual al cual pertenece este fragmento. En un datagrama, todos los fragmentos, con excepción del último, deberán ser un múltiplo de 8 octetos, que es la unidad elemental de fragmentación. Dado que se proporcionan 13 bits, hay un máximo de 8192 fragmentos por datagrama, dando así una longitud máxima de *datagrama de 65 536 octetos, que coinciden con el campo Longitud total.* Tamaño 16 bits.

El campo Tiempo de vida es un contador que se utiliza para limitar el tiempo de vida de los paquetes. Cuando se llega a cero, el paquete se destruye. La unidad de tiempo es el segundo, permitiéndose un tiempo de vida máximo de 255 segundos. Tamaño 8 bits.

Cuando la capa de red ha terminado de ensamblar un datagrama completo, necesitará saber qué hacer con él. El campo Protocolo indica, a qué proceso de transporte pertenece el datagrama. El TCP es efectivamente una posibilidad, pero en realidad hay muchas más.

7.6.3 Protocolo

El número utilizado en este campo sirve para indicar a qué protocolo pertenece el datagrama que se encuentra a continuación de la cabecera IP, de manera que pueda ser tratado correctamente cuando llegue a su destino. Tamaño: 8 bits.

El código de redundancia de la cabecera es necesario para verificar que los datos contenidos en la cabecera IP son correctos. Por razones de eficiencia este campo no puede utilizarse para comprobar los datos incluidos a continuación, sino que estos datos de usuario se comprobarán posteriormente a partir del código de redundancia de la cabecera siguiente, y que corresponde al nivel de transporte. Este campo debe calcularse de nuevo cuando cambia alguna opción de la cabecera, como puede ser el tiempo de vida. Tamaño: 16 bits.

La Dirección de origen contiene la dirección del host que envía el paquete. Tamaño: 32 bit.

La dirección de destino. Esta dirección es la del host que recibirá la información. Los routers o gateways intermedios deben conocerla para dirigir correctamente el paquete. Tamaño: 32 bit.

7.6.4 En que se utiliza TCP/IP

Muchas grandes redes han sido implementadas con estos protocolos, incluyendo DARPA Internet "Defense Advanced Research Projects Agency Internet", en español, Red de la Agencia de Investigación de Proyectos Avanzados de Defensa. De igual forma, una gran variedad de universidades, agencias gubernamentales y empresas, están conectadas mediante los protocolos TCP/IP. Cualquier máquina de la red puede comunicarse con otra distinta y esta conectividad permite enlazar redes físicamente independientes en una red virtual llamada Internet. Las máquinas en Internet son denominadas "hosts" o nodos.

TCP/IP proporciona la base para muchos servicios útiles, incluyendo correo electrónico, transferencia de archivos y login remoto. El correo electrónico está diseñado para transmitir archivos de texto pequeños. Las utilidades de transferencia sirven para transferir archivos muy grandes que contengan programas o datos. También pueden proporcionar chequeos de seguridad controlando las transferencias.

Conclusiones

La elaboración de este proyecto tiene mucha importancia y conveniencia para la empresa ya que se evitarían los conflictos que tienen, será un paso más para el crecimiento de la misma y para mí la oportunidad de ampliar mis conocimientos sobre lo que es Java, base de datos, interfaz grafica y JDBC que me va ayudar mucho en mi desarrollo profesional.

La fácil disponibilidad que poseen las computadoras y las tecnologías de información en general, han creado una revolución informática en la sociedad y de forma particular en los negocios. El manejo de información generada por computadora difiere en forma significativa del manejo de datos producidos manualmente.

Los sistemas informáticos son una herramienta muy poderosa que al proveer datos y análisis oportunos, nos lleva a tener una gran ventaja competitiva y que cada vez se está usando más en las actividades organizacionales, cada vez existen más aplicaciones en diversos campos. El éxito de su implementación depende mucho de la relación adecuada que exista entre los desarrolladores y los usuarios finales en el momento del análisis, para que finalmente sea una herramienta útil y ajustada a la realidad de la empresa, y ya que en este caso yo estoy involucrado en ella y se el funcionamiento de la misma, me facilita mas desarrollar un buen sistema y que bien implementado nos dará la capacidad de seleccionar una solución adecuada, en menor tiempo, además de la satisfacción de clientes y trabajadores.

Las empresas grandes invierten hasta millones de dólares, pero también existen oportunidades para las empresas pequeñas para adquirir uno al alcance de su presupuesto como es el caso de esta empresa. Es importante aprovechar la ventaja competitiva que esta tecnología ofrece.

Trabajo Futuro

Los siguientes puntos se pueden considerar para realizar mejoras a este trabajo de tesis:

- Si la empresa quisiera seguir beneficiándose de la tecnología una opción para dejar hacer las cosas de manera clásica sería la de desarrollar un programa para una PDA en la cual los empleados pudieran guardar los datos del cliente, esto sería muy útil cuando están haciendo guardia en algún inmueble, el sistema se encargaría de mandar la información vía Internet y actualizaría la base de datos en la oficina en tiempo real.
- Otra muy buena opción sería que desde un dispositivo PDA o desde Internet se pudieran dejar notas al los diferentes usuarios del sistema en la oficina, esto podría reducir gastos en mensajes a celular o llamadas.

Bibliografía

- Steven Haines, Stephen Potts, Java 2 Primer Plus, Sams Diciembre 2002.
- Head First Java, 2nd Edition, Kathy Sierra and Bert Bates 2005
- Java Concurrency in Practice by Brian Goetz, Tim Peierls, Joshua Bloch, and Joseph Bowbeer 2006.
- Java In A Nutshell, 5th Edition by David Flanagan 2005.
- Java Persistence with Hibernate by Christian Bauer and Gavin King 2006
- Gregory D. Speegle, JDBC practical guide for Java programmers, Morgan Kaufmann Publishers 2002.
- JDBC(TM) API Tutorial and Reference (3rd Edition) (The Java Series) by Maydene Fisher, Jon Ellis, and Jonathan Bruce 2003.
- JDBC: Database Programming with J2ee by Art Taylor 2002.
- MySQL and Java Developer's Guide by Mark Matthews, Jim Cole, and Joseph D. Gradecki 2003.
- MySQL 5.0 Certification Study Guide (MySQL Press) by Paul DuBois, Stefan Hinz, and Carsten Pedersen 2005.
- Learning SQL (Learning) by Alan Beaulieu 2005.
- David M. Kroenke, procesamiento de bases de datos octava edición, Prentice Hall 2003.
- Database Systems: Design, Implementation, and Management, Seventh Edition by Peter Rob and Carlos Coronel 2006.
- Luis Joyanes Aguila, Ignacio Zahonero Martínez, Programación en Java 2 algoritmos, estructura de datos y programación orientada a objetos, Mc Graw Hill 2002.
- Architectural Model as Machine: A new view of models from antiquity to the present day by Albert Smith 2004
- Andrew S. Tanenbaum, Redes de computadoras cuarta edición, Prentice Hall 2003.
- Donald Norman, The design of everyday things, Basic Books 2002.
- http://java.sun.com/j2se/1.5.0/docs/api/
- http://java.sun.com/docs/books/tutorial/

Anexo I. Ventanas del sistema

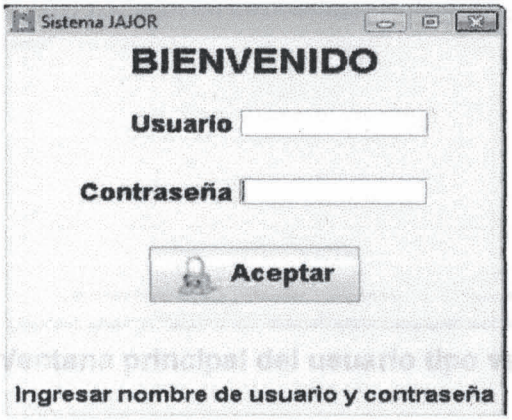

Figura 1.1 Ventana de acceso al sistema.

En esta ventana el usuario tiene que ingresar el nombre de usuario y contraseña, dependiendo del tipo de usuario abre la ventana de Administrador o la de Vendedores.

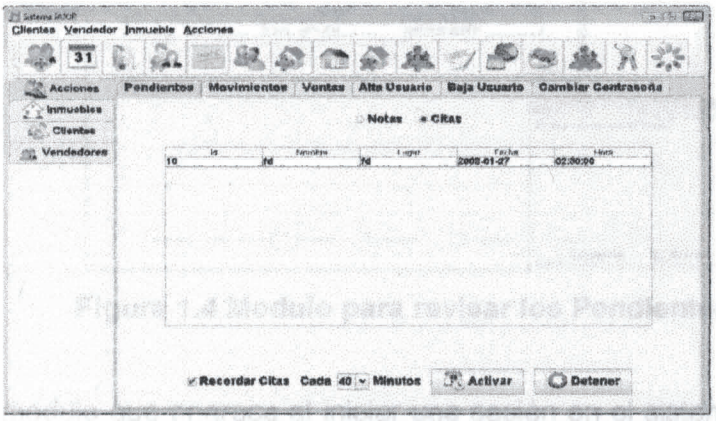

Figura 1.2 Ventana de principal del usuario tipo administrador

En esta ventana se muestran todas las opciones del sistema con el cual el usuario tipo administrador puede llevar todo el control.

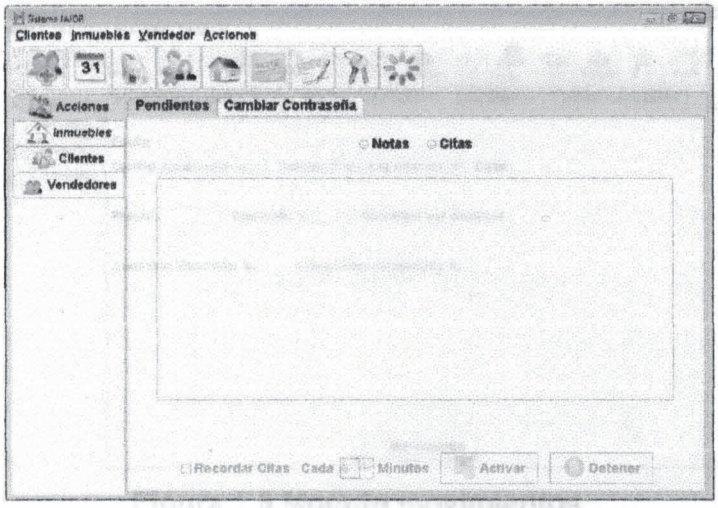

Figura 1.3 Ventana principal del usuario tipo vendedor

En esta ventana se muestran las opciones que necesita el usuario tipo vendedor (consultas de inmuebles, clientes etc.).

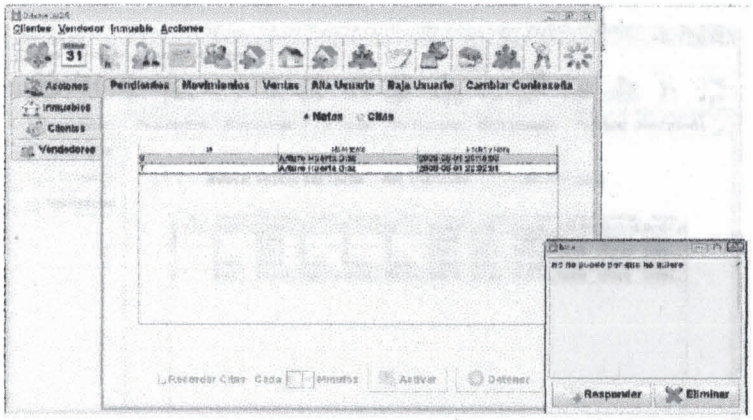

Figura 1.4 Modulo para revisar los Pendientes

Es el primer modulo que aparece al iniciar una sesión en el sistema ya sea de tipo administrador o vendedor, en ella se muestran las citas, notas o pendientes que el usuario tiene, también puede responder las notas y activar la opción para recodar las citas programadas para ese día.

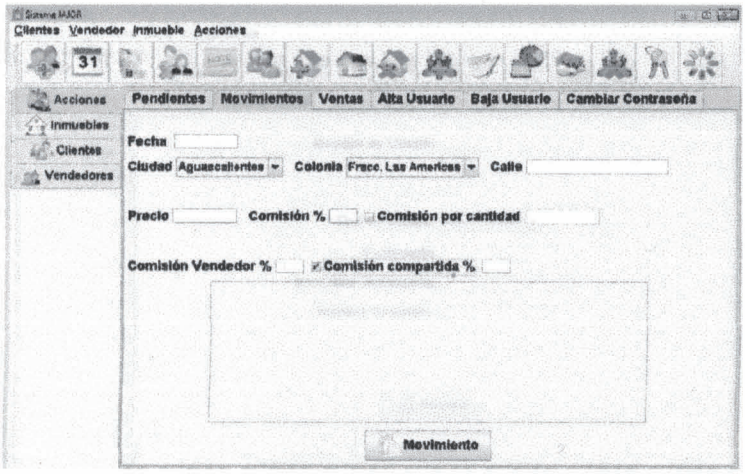

Figura 1.5 Modulo movimientos

En este modulo se registran las ventas de los inmuebles, que inmueble se vendió, quien la vendió, en cuanto se vendió, así como también la comisión de la empresa y la del vendedor.

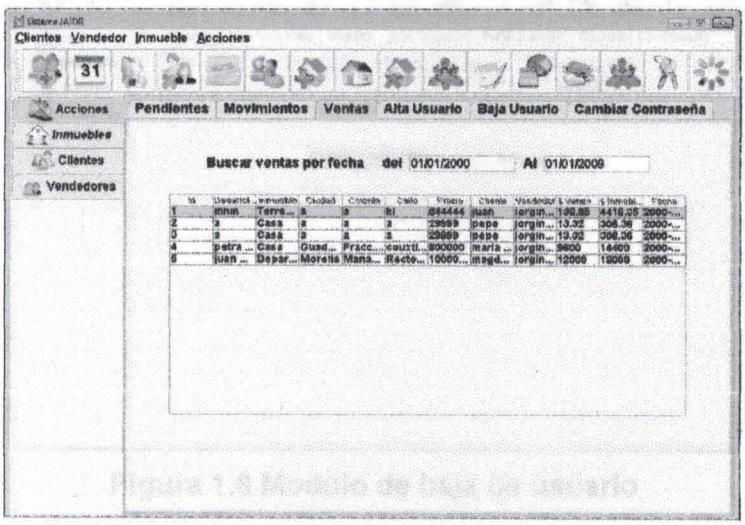

Figura 1.6 Modulo de ventas

En este modulo se buscan las ventas por rango de fechas. Si el administrador quiere ver las ventas de determinada fecha lo puede hacer a travez de esta opción.

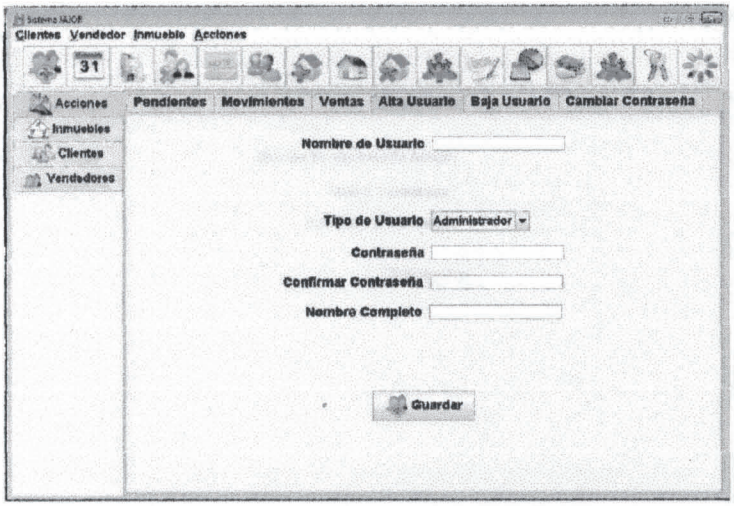

Figura 1.7 Modulo de alta de usuario

En este modulo el administrador da de alta los usuarios en el sistema.

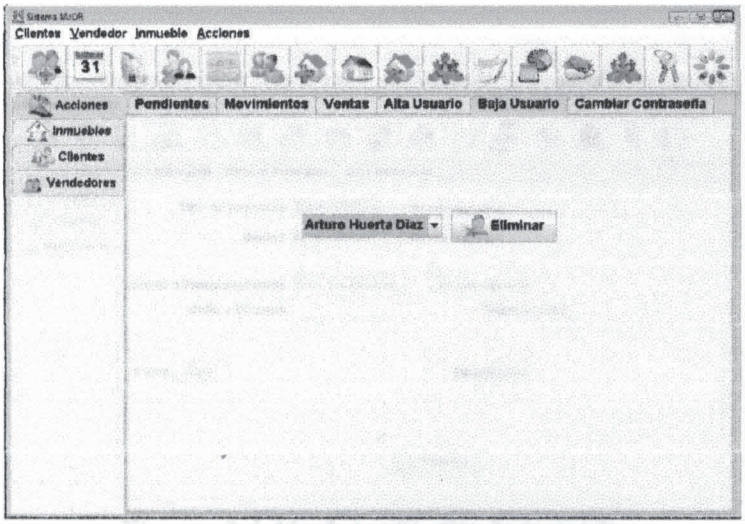

Figura 1.8 Modulo de baja de usuario

En este modulo el administrador da de baja a los usuarios del sistema.

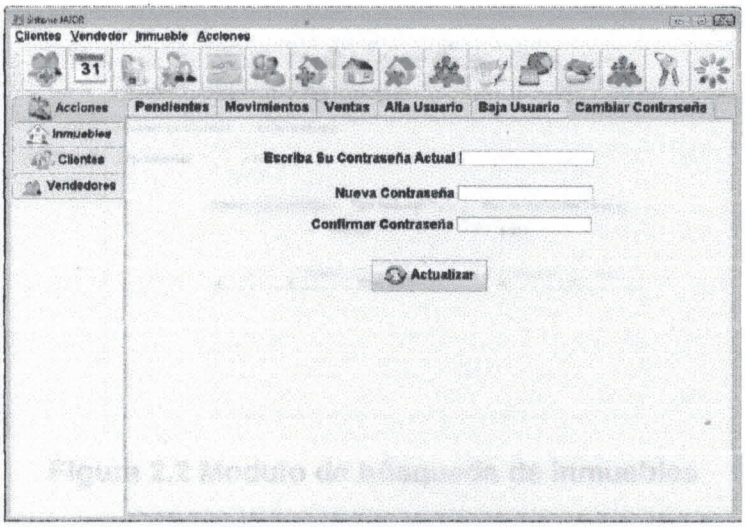

Figura 1.9 Modulo de cambiar contraseña

En este modulo los usuarios ya sea de tipo administrador o empleado pueden cambiar su contraseña.

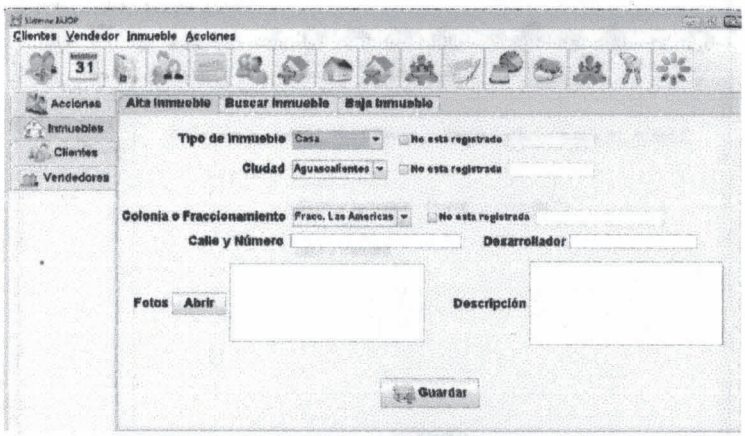

Figura 2.1 Modulo de alta inmueble

En este modulo el administrador puede dar alta un inmueble.

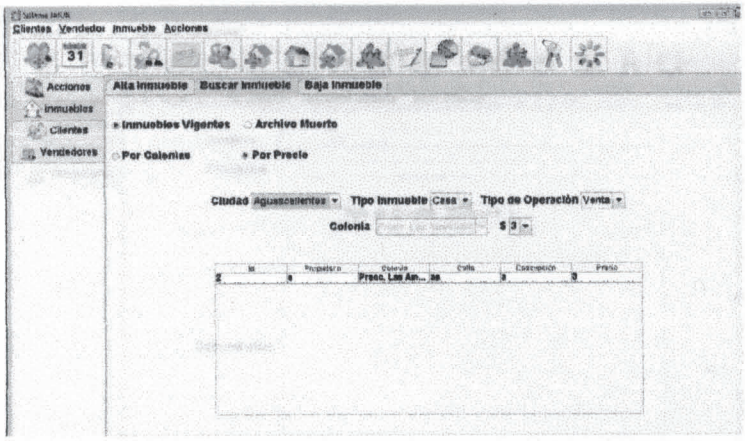

Figura 2.2 Modulo de búsqueda de inmuebles

En este modulo los usuarios ya sea tipo administrador o empleado pueden realizar búsquedas de los inmuebles que estén vigentes o los que ya se han vendido.

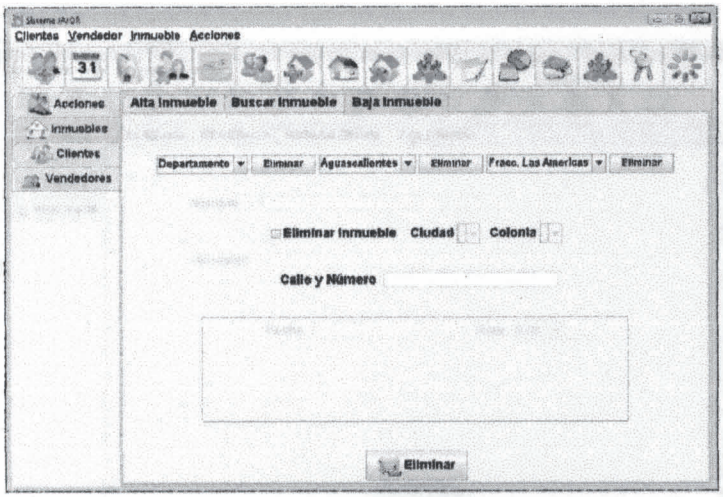

Figura 2.3 Modulo de baja de inmueble

En este modulo el administrador puede dar de baja un inmueble eliminándolo totalmente de la base de datos.

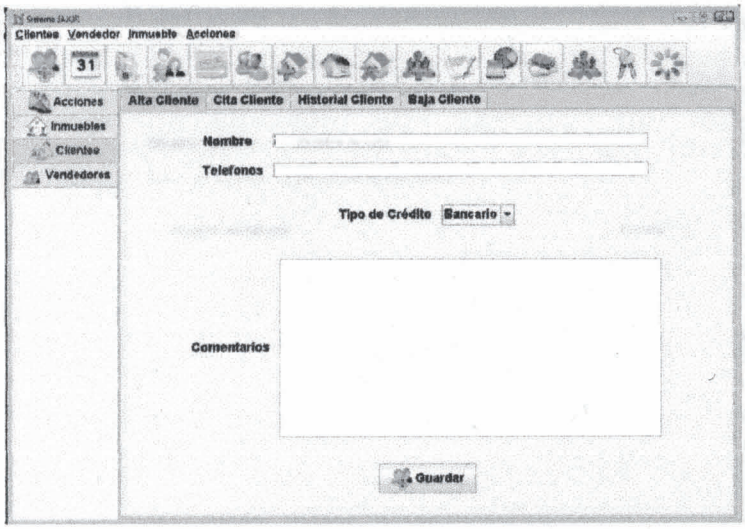

Figura 2.4 Modulo de alta cliente

En este modulo todos usuarios pueden dar de alta un cliente así como también van creando el historial del mismo.

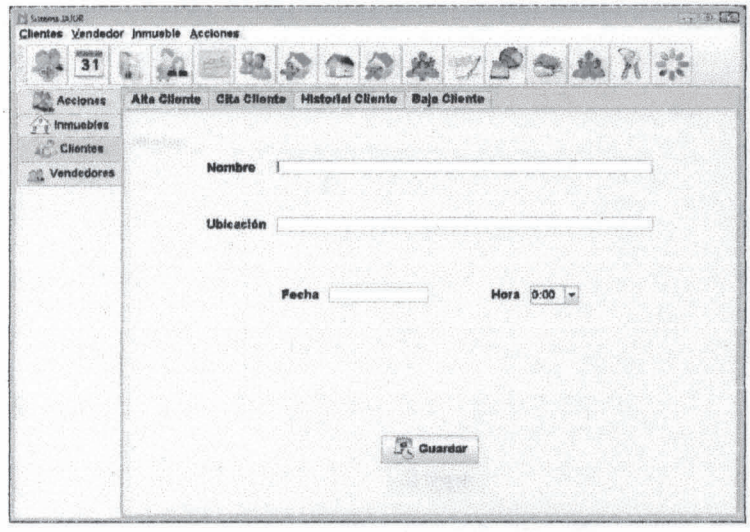

Figura 2.5 Modulo de cita cliente

En este modulo todos los usuarios pueden programar una cita.

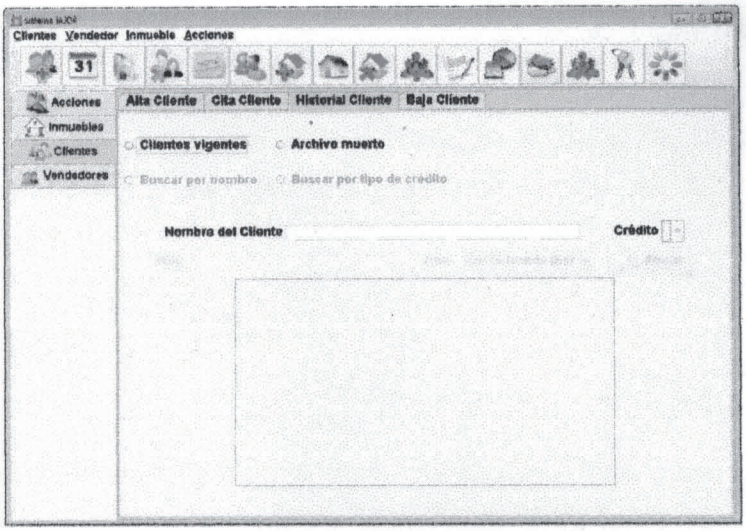

Figura 2.6 Modulo de historial del cliente

En este modulo todos los usuarios pueden ver la información de sus clientes y pueden agregar información al historial del mismo.

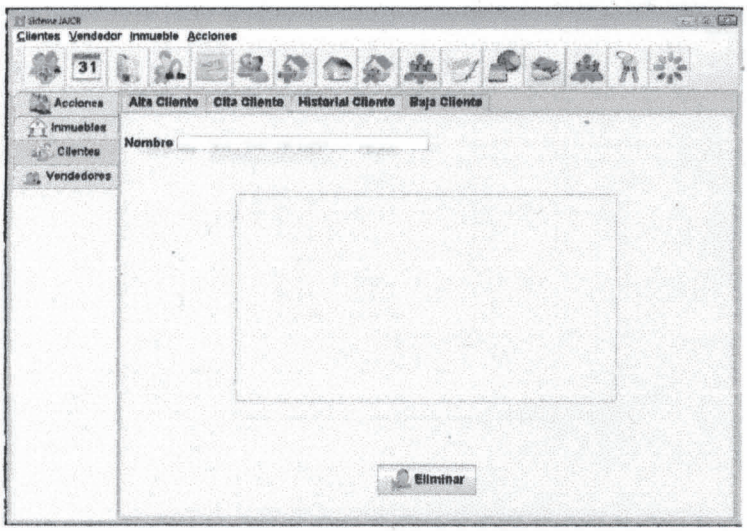

Figura 2.7 Modulo baja cliente

En este modulo todos los usuarios pueden dar de baja a sus clientes.

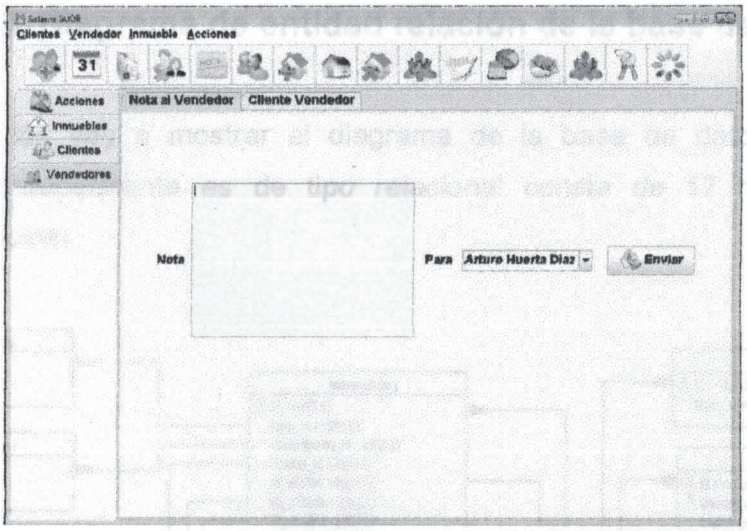

Figura 2.8 Modulo nota al vendedor

En este modulo todos los usuarios pueden mandar una nota a los otros usuarios (las notas recibidas se muestran en el modulo de pendientes).

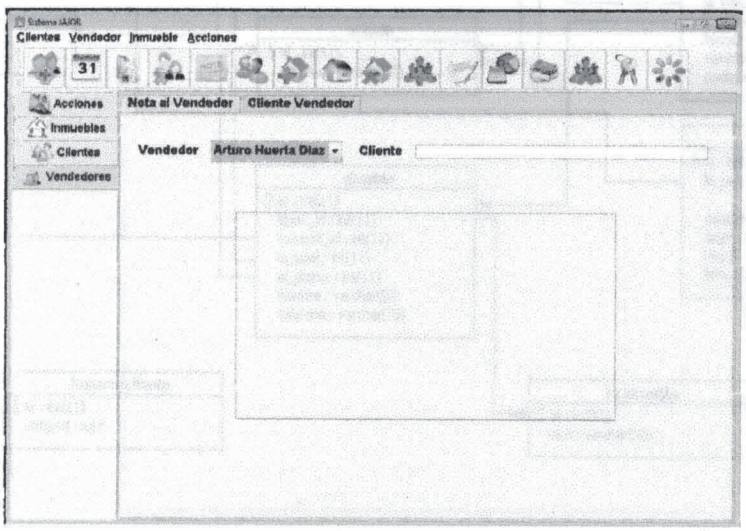

Figura 2.9 Modulo cliente vendedor

En este modulo el usuario de tipo administrador puede ver el historial de los clientes de todos los vendedores pero no lo puede modificar.

Anexo II. Diagrama de entidad relación de la base de datos.

A continuación voy a mostrar el diagrama de la base de datos como ya mencione anteriormente es de tipo relacional consta de 17 tablas y 18 relaciones (joins).

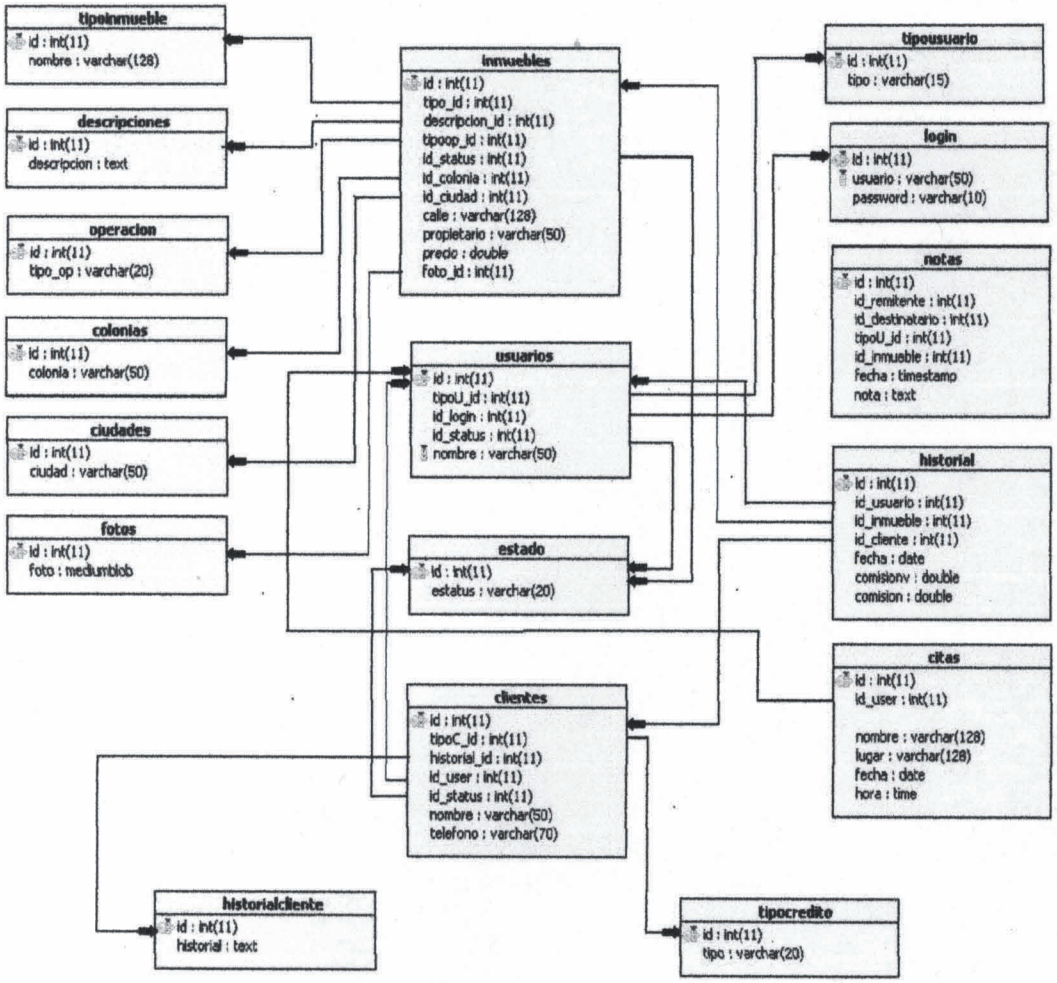

Figura 3.1 Diagrama entidad relación de la base de datos.