UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“DESARROLLO EN JAVA MASTER”

Para obtener el título de

INGENIERO EN COMPUTACIÓN

Modalidad: Seminarios y cursos de actualización y capacitación profesional

PRESENTA:

MOISÉS RAMÓN HERRERA PÉREZ

DIRIGÍO EL INFORME

INGENIER0 ANTONIA NAVARRO GONZÁLEZ

FES Aragón

Nezahualcóyotl, Estado de México, 2013.

A mis padres:

A MI MADRE: (†) Por que siempre me brindo su amor y me enseño a no rendirme para seguir adelante hasta el final, en mi corazón siempre permanecerás.

A MI PADRE: Por todo lo que has hecho para que sea una persona de bien y por el conocimiento que has puesto en mis manos.

A mis hermanos:

Miguel Ángel, Julio Antonio, Araceli y José Luis por compartir grandes momentos y apoyarme en todo momento.

A Josefina y Manuel:

Por que me quieren como un hijo y me han dado consejos tan valiosos.

A Héctor, Enrique, María del Roció y Karina:

Porque me han dado el honor de quererme como su hermano.

A Martin:

Por ser ese amigo que siempre esta ahí cuando lo necesitas.

"ATÍ IRMA ESPOSA MÍA"

Por darme tu amor, apoyo, comprensión en cada instante de mi vida, por hacer que nuevamente retome el camino que me corresponde y haber llegado a mi vida y darme esa felicidad anhelada.

A MI HIJA MARÍA FERNANDA:

Por la alegría que nos das día con día y por ti quiero mostrar lo mejor que hay en mí, porque en ti mis esperanzas reviven y mi amor se fortalece para vencer barreras que hay en el camino.

A la Universidad Nacional Autónoma de México. FES ARAGÓN:

Por darme la oportunidad de obtener el grado de Licenciatura para tener una vida mejor, formándome con tus conocimientos invaluables para enfrentar este mundo cambiante y que cada día es mas exigente, para llevar tus enseñanzas en cada lugar que vaya.

A mis maestros:

Por sus conocimientos aportados en mi formación y por su dedicación en mi desarrollo profesional, pero en especial a los Maestros: Pedro Gabriel Ramírez Hernández, Miguel Ángel Sánchez Hernández y Jesús Hernández Cabrera.

A mi asesor y sinodal:

Ingeniero Antonia Navarro González por haberme dirigido en este informe y darme así la oportunidad de ser una mejor persona.

A mis sinodales:

Maestro Juan Gastaldí Pérez, Ingeniero Martín Hernández Hernández, Maestra María Angélica Feria Victoria y Maestro Sergio Hernández López ya que sus aportaciones enriquecieron mi informe y a su vez me motivaron para terminarlo y lograr mi meta que había esperado por mucho tiempo.

ÍNDICE

Introducción 8

Capítulo 1

Análisis y diseño orientado a reglas de negocios usando UML

1.1 Conceptos generales 10

1.1.1 Administración del proyecto 10

1.1.2 Modelado del negocio 11

1.2 Producto obtenido en el módulo 12

Capítulo 2

Lenguaje de programación en Java

2.1 Conceptos generales 20

2.1.1 Conceptos Básicos de Java 20

2.1.2 Programación con Java 21

2.2 Producto obtenido en el módulo 21

Capítulo 3

Lenguaje de programación avanzada

3.1 Conceptos generales 24

3.1.1 Datos Primitivos 24

3.1.2 Clases 24

3.1.3 Herencia 25

3.1.4 Referencias 25

3.1.5 Polimorfismo 25

3.1.6 Interfaces 26

3.1.7 Acceso al disco 26

3.1.8 JDBC 26

3.1.9 Swing 27

3.1.10 Interfaz de múltiples documentos (MDI) 27

3.2 Producto obtenido en el módulo 27

Capítulo 4

Java Conectividad para la empresa

4.1 Java Enterprise Edition (J2EE) 30

4.1.1 Java Database Connectivity (JDBC) 30

4.1.2 MySQL 30

4.1.3 SQL (Lenguaje de Consulta Estructurado) 30

4.1.4 DAO (Data Access Object) 31

4.2 Producto obtenido en el módulo 31

Capítulo 5

Java programación para desarrolladores

5.1 SERVLETS 35

5.1.1 Java Server Pages (JSP) 35

5.1.2 Tag Libs 36

5.1.3 Custom Tag 36

5.2 Producto obtenido en el módulo 36

Capítulo 6

J2EE desarrollo de componentes web

6.1 Struts 2 39

6.1.1 OGNL (Lenguaje de navegación de objetos y gráficos) 40

6.1.2 Etiquetas de Struts 40

6.1.3 JDBC con Struts 2 40

6.1.4 Hibernate y Struts 2 41

6.2 Producto obtenido en el módulo 42

Capítulo VII

Java Empresarial Java Beans

7.1 Java Server Faces (JSF) 45

7.1.1 ICEfaces 47

7.2 Producto obtenido en el módulo 48

Conclusiones 50

Glosario 52

Referencias 54

INTRODUCCIÓN

Este es un informe del 4to. Diplomado en Java Master el cual tuvo inicio en el año 2010 y terminó en el año 2011, en el cual describo los 7 módulos que divido en siete capítulos de acuerdo a cada módulo que lo conformaron.

En el capítulo 1, mencionó lo necesario que tenemos para analizar un proyecto de tecnologías de Información y la administración del mismo, utilizando diagramas que nos ayudan para el análisis, uno de ellos el diagrama de GANTT que es el que nos permite administrar el proyecto donde se describen todas las actividades y podemos ver sus avances, costos, personal necesario, etc.; al final se realizó un proyecto final, el cual consistió en crear un ERP, de las parroquias que se encuentran en la Diócesis del Distrito federal en especifico a la Vicaria VIII.

En el capítulo 2, comenzamos a introducirnos al lenguaje java, donde se nos imparte desde lo más básico y así arranquemos con los mismos conocimientos, aquí vemos la programación desde la consola de java para correr los programas y compilarlos, utilizamos comandos y la programación es lineal. También muestro la aplicación que realizamos que consistió en la simulación de una compra en una tienda.

En el capítulo 3, la programación que vemos ya es más avanzada y es más orientada a objetos para lo cual utilizamos un IDE que es Netbeans el cual nos proporciona una parte donde podemos programar y otra donde podemos diseñar vistas, el cual nos permite administrar nuestros proyectos, también se hace uso de las características de java como la herencia y el polimorfismo, hacemos la primera conexión a una base de datos diseñada en Oracle y accedemos a ella mediante sentencias de SQL. Se describe una aplicación que consiste en la creación de una agenda personal.

En el capítulo 4, utilizamos patrones que ya están establecidos para aplicarlo a empresas que se pueden conectar a una base de datos, se realizó una aplicación pero ahora conectada a una base de datos de Mysql la cual consistió en aplicar el patrón DAO a un sistema de control escolar donde podemos insertar, eliminar o modificar datos de los alumnos o poder ver su promedio lo que nos hace ver que este patrón se puede modificar y aplicar a otros sistemas.

En el capítulo 5, se describen los componentes necesarios para que ahora nuestras aplicaciones corran bajo internet, y realizamos las primeras aplicaciones web utilizando JSP que es aquí donde java se consolida como uno de los lenguajes más populares, ya que fue diseñado para la web.

La aplicación desarrollada es de una simulación de un carrito de compras bajo internet donde podemos comprar discos.

En el capítulo 6, se muestran otros componentes que nos ayudan para un mejor desempeño de la página web como es Struts2, que se implementan en el modelo vista controlador, el cual separa la lógica del negocio, con la presentación del usuario, también se muestra otro componente como “hibernet” que nos ayuda para hacer una conexión a la base de datos sin tener que hacer un programa para la conexión sino que mediante la configuración se puede tener acceso a ella. La aplicación fue desarrollar una página que nos permitiera que un usuario pudiera subir archivos eliminarlos y bajarlos una vez que se hubiese registrado.

En el capítulo 7, se muestran otros dos componentes como es java server faces que utiliza aplicaciones de programación de internet la cual nos da un ambiente gráfico más interactivo y el otro componente que es uno de los más actuales es ICEfaces donde en una página web se puede hacer que se refresquen sólo algunos datos y dejando los otros datos en segundo plano lo que a la vista lo hace más agradable.

Hasta aquí termina el curso del Diplomado en Java Master, pero me ha dejado un gran interés en seguir aprendiendo porque esta tecnología sigue creciendo y es necesario estar actualizado a lo que demandan los avances tecnológicos.

CAPÍTULO 1

ANÁLISIS Y DISEÑO ORIENTADO A REGLAS DE NEGOCIOS USANDO UML

1.1 Conceptos generales

En este módulo se realizó el modelado de un proyecto de Tecnologías de información en torno a su Planeación, Análisis, Diseño e implementación del mismo, utilizando reglas basadas en UML (Unified Modeling Language) que significa Modelado de lenguaje unificado, el UML nace de la necesidad de que al crear un sistema de información, que no solo el programador pudiera saber como esta diseñado si no que también cualquier otro programador entendiera qué es lo que se tiene que construir, de la misma forma que un arquitecto cuenta con planos para la construcción de un edificio.

Aprendí que para realizar este proyecto era necesario dividirlo en dos partes una parte que concierne a la parte de la administración y otra que corresponde al modelado del negocio.

1.1.1 Administración del proyecto

En esta parte se utilizó un software de administración de proyectos llamado project perteneciente a Microsoft office, que nos permite hacer un listado de las tareas que se necesitan, desde que se comienza hasta que se termina, por ejemplo desde la tarea del que toma la decisión política para realizar este proyecto hasta el momento de la entrega del proyecto liberado y el soporte que se le dará a este, permitiéndome controlar la duración desde el comienzo hasta el fin de cada tarea, podemos determinar qué actividad precede a la otra, además este control nos permite ver qué actividades comienzan al mismo tiempo que otra, así mismo podemos introducir los recursos humanos con los que contamos en cada tarea asignada e indicarnos el nombre de la persona encargada, podemos controlar cuánto nos va a costar las horas hombre, y también de los recursos materiales, nos permite hacer informes generales, de costos, de actividades actuales de asignaciones y de carga de trabajo y tener informes personalizados. Lo importante de Project es que nos permite obtener el Diagrama de GANTT, para ver los avances del proyecto de una manera gráfica, indica además cuál debe ser la ruta crítica del proyecto para darle prioridad a dichas tareas. Debido a que el programa Project nos da todas estas facilidades para controlar todos nuestros recursos fue base de estudio en este módulo enseñándonos cómo administrar un proyecto.

Obteniendo los resultados que nos proporciona el Diagrama de GANTT, podemos evaluarlo en costo, tiempo, alcance y calidad. También debemos tomar en cuenta qué tipo de madurez deseamos obtener en nuestro proyecto, por ello debemos tomar en cuenta el CMMI

(Capability and Maturity Model Integrated) ó Modelo de Capacidad y Madurez Integrado refiriéndose al software, donde el grado de madurez va de la primera etapa a la quinta etapa, donde la primera etapa es la inicial y el sistema es impredecible y poco controlado, la segunda etapa es repetible, esto es posible repetir tareas previamente realizadas con éxito, la tercera etapa que es definido ó administrado, donde tiene procesos caracterizados y bien comprendidos, la cuarta etapa, manejado o gestionado, donde el proceso es medido y controlado y la quinta etapa es la optimizada en esta busca la mejora del software utilizado. Esta es una parte muy importante porque siempre tenemos que observar en qué etapa está nuestro sistema qué deseamos desarrollar y hacia dónde queremos llegar, y que lo que siempre se quiere es llegar al optimizado, sin embargo los costos son elevados y la mayoría de los proyectos solo se quedan en la cuarta etapa.

Para poder el evaluar el sistema de software se debe apoyar en la norma ISO/IEC 9126 que posee cuatro partes, la primera parte es el modelo de calidad, la segunda parte ; son las métricas internas, la tercera parte métricas externas y la cuarta parte son la calidad en las métricas de uso, dónde sólo la primera parte es un estándar aprobado y publicado y el resto se encuentran en la parte denominada Technical Report(TR). Un cuantificador de medida de la calidad del software es el modelo de McCall que lo divide en tres partes: revisión, que se refiere a la calidad interna de la programación del software, buscando que sea fácil en su mantenimiento, si es fácil de probarlo y si puede ser modificado fácilmente; la otra es operación que se refiere a la calidad externa, si es fácil usarlo, si el programa hace lo que deseo, si lo ejecuta en forma óptima y si es exacto todo el tiempo y la tercera etapa la de transición que se refiere a la movilidad, interoperabilidad y la de reutilización del software y el hardware. La calidad de uso del software posibilita la obtención de eficacia, productividad, seguridad y satisfacción.

1.1.2 Modelado del negocio

En el modelado del sistema UML cuenta con nueve diagramas

- ***Diagramas de casos de uso.*** Se usan para modelar los procesos que sigue el negocio, aquí el Actor del negocio que puede ser alguien o algo, juega un papel importante que nos permite ver la secuencia de acciones realizadas en un negocio observable.
- ***Diagramas de secuencia.*** Es para modelar el paso de mensajes entre objetos, aquí es donde los mensajes me dicen que acción debo de realizar.
- ***Diagramas de Colaboración.*** Es para modelar interacciones entre objetos.
- ***Diagramas de Estado.*** Es para modelar el comportamiento de los sistemas, es decir nos permite ver en que estatus se encuentra.
- ***Diagramas de Actividades.*** Es para modelar el comportamiento de los casos de usos, objetos u operaciones.

- ***Diagramas de Clases.*** Es para modelar la estructura estática de los clases del sistema, que es donde nos permite ver que campos contienen las clases y que actividades realizan
- ***Diagramas de Objetos.*** Es para modelar la estructura estática de los objetos del sistema.
- ***Diagramas de Componentes.*** Es para modelar componentes
- ***Diagramas de Implementación.*** Es para modelar la distribución del sistema.

Todos estos diagramas existen para el apoyo del proyecto y sólo se utilizan los que se requieran y el que nos permita ver más claramente cómo es que se desarrollan las actividades en la forma de hacer negocios, ya que cada empresa realiza los negocios como le funcionan en la práctica, porque hay veces que lo que marca la teoría no funciona. Para hacer estos diagramas es necesario apoyarnos en la parte de la administración del proyecto donde tengamos la tarea de hacer estudios sobre cada una de los áreas con que cuenta una empresa.

1.2 Producto obtenido en el módulo

Se realizó un proyecto con base en un ERP (Enterprice Resource Planning) que significa Sistema de Gestión Empresarial, él proyecto lo apliqué en la en las Iglesias de la Diócesis del Distrito Federal pero en específico a la vicaria VIII , lo apliqué aquí por que la Iglesia católica tiene una muy buena organización no sólo dentro del Distrito federal si no en todo el mundo porque una vez realizado este proyecto su aplicación podría ser no solamente en el Distrito Federal así también en toda la república y a su vez a nivel mundial que son los alcances que nos permiten los ERP demostrando además que los ERP funcionan en cualquier sistema en que lo deseemos aplicarlo.

En la parte de la administración del proyecto utilicé la muestra de diagrama de GANTT que se nos dio en el curso del diplomado dado para este fin, las cuales se muestran en la ***tabla 1.1***.

Para realizar este proyecto primeramente se investigó como se conforma la Diócesis del Distrito Federal y mostrándolo en un diagrama a bloques, la cual consta de 8 vicarias que son las que gobiernan en cierto territorio, que se muestra en la ***Diagrama 1.1***; yo me enfoqué sólo al estudio de la vicaria VIII que abarca algunas delegaciones como Iztapalapa, Milpa Alta, Tlalpan y Xochimilco, a su vez cada vicaria esta dividida por decanatos y la vicaria VIII tiene 5 decanatos, también estos decanatos están divididas en parroquias la vicaria también fue plasmada en un diagrama de bloques para observar mejor su estructura, las cuales se muestran en la ***Diagrama 1.2*** .

1	No aplica	50	Selecciona Sistema
2	Decisión política	51	Aplicación de criterios primarios
3	Evaluación previa de medios y costos	52	Evaluación del cumplimiento de los criterios primarios
4	Definir y fijar prioridades	53	Selección de Proveedores Semifinalistas
5	Aprobación	54	Evaluación de los criterios Detallados
6	Definición y objetivo (Misión de la empresa)	55	Presentación de los finalistas a las áreas
7	Identificación de usuarios clave	56	Visitas a clientes
8	Programación y realización de entrevistas	57	Evaluación de la percepción de los sistemas
9	Identificación de requerimientos de usuario	58	Análisis de las cotizaciones
10	Definición de criterios primarios	59	Presentación de la recomendación del ganador
11	Arquitectura de la información	60	Selecciona Sistema
12	Aprobación Dirección	61	Diseño Lógico
13	Definición de estructura orgánica	62	Diseño físico
14	Estudio de viabilidad	63	Prueba de Sistemas Hardware
15	Estudio y análisis del sistema actual	64	Prueba de servicios
16	Identificación de criterios detallados	65	Prueba de nodos cliente
17	Identificación de requerimientos detallados	66	Prueba de redes cliente
18	División de áreas funcionales	67	Prueba General
19	Analizar gestión de compras	68	Construcción del Programa
20	Analizar gestión de compras	69	Capa de aplicación
21	Orden de compra	70	Capa Base de Datos
22	Informe de recepción	71	Aplicación intranet
23	Factura de proveedor	72	Aplicación Web
24	Orden de pago	73	Modelo1
25	Analizar Gestión Comercial	74	Prototipo1
26	Pedido	75	Pruebas y validación 1
27	Surtido	76	Modelo 2
28	Entrega	77	Prototipo 2
29	Facturación	78	Pruebas y validación 2
30	Recibo de Cobro	79	Pruebas en Unidades
31	Analizar Gestión de inventario	80	Pruebas de integración con todo
32	Apartado	81	Aceptación
33	Realización del Servicio	82	Entrega
34	Comprobante Servicio	83	Capacitación
35	Recibo de Cooperación	84	Soporte
36	Analizar Gestión de Inventarios		
37	Analizar gestión de inventario		
38	Entradas y salidas		
39	Apartados		
40	Saldos		
41	Analizar Gestión Financiera		
42	Impuestos		
43	Pagos		
44	Cuentas		
45	Cobros		
46	Evaluación situación financiera		
47	Evaluación del Resultado de Áreas Funcionales		
48	Evaluación de hardware y software		
49	Desarrollo de plan estratégico		

Tabla 1.1 Puntos a evaluar en el Diagrama de GANTT
Fuente: 4to. Diplomado en Desarrollo Java Master (2010)

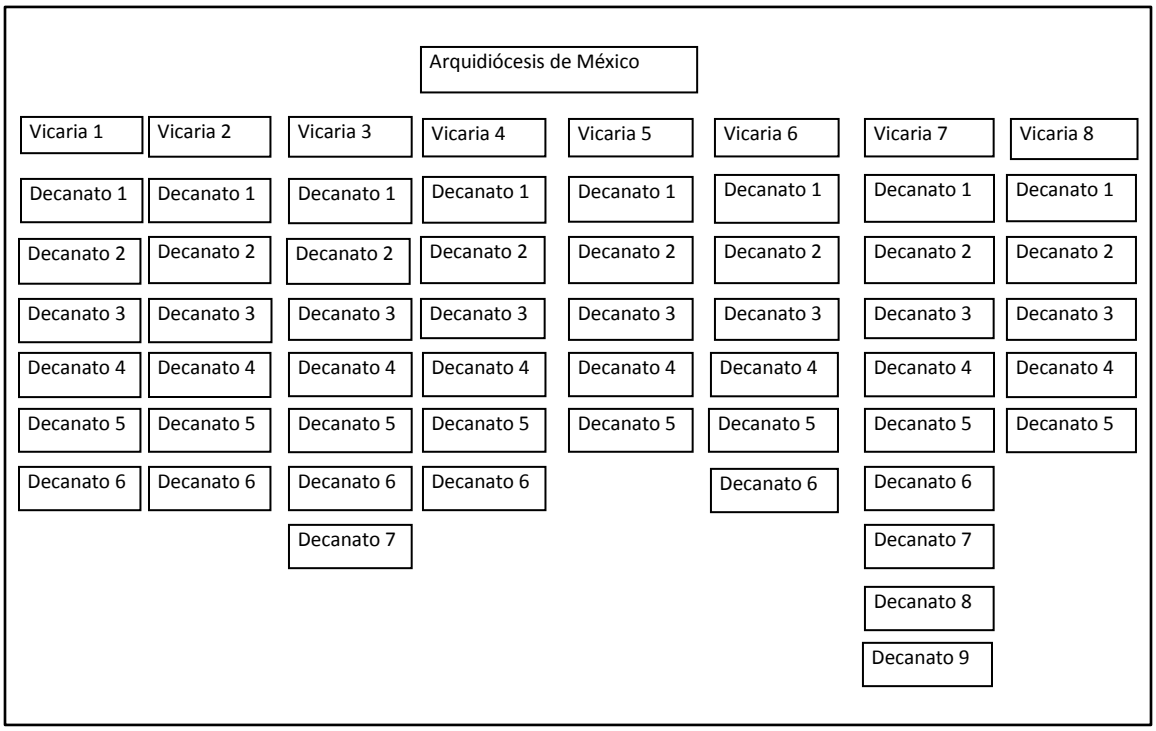


Diagrama 1.1 Diagrama de Bloques Arquidiócesis de México
Fuente: http://vicaria8.arquidiocesismexico.org.mx (2010)

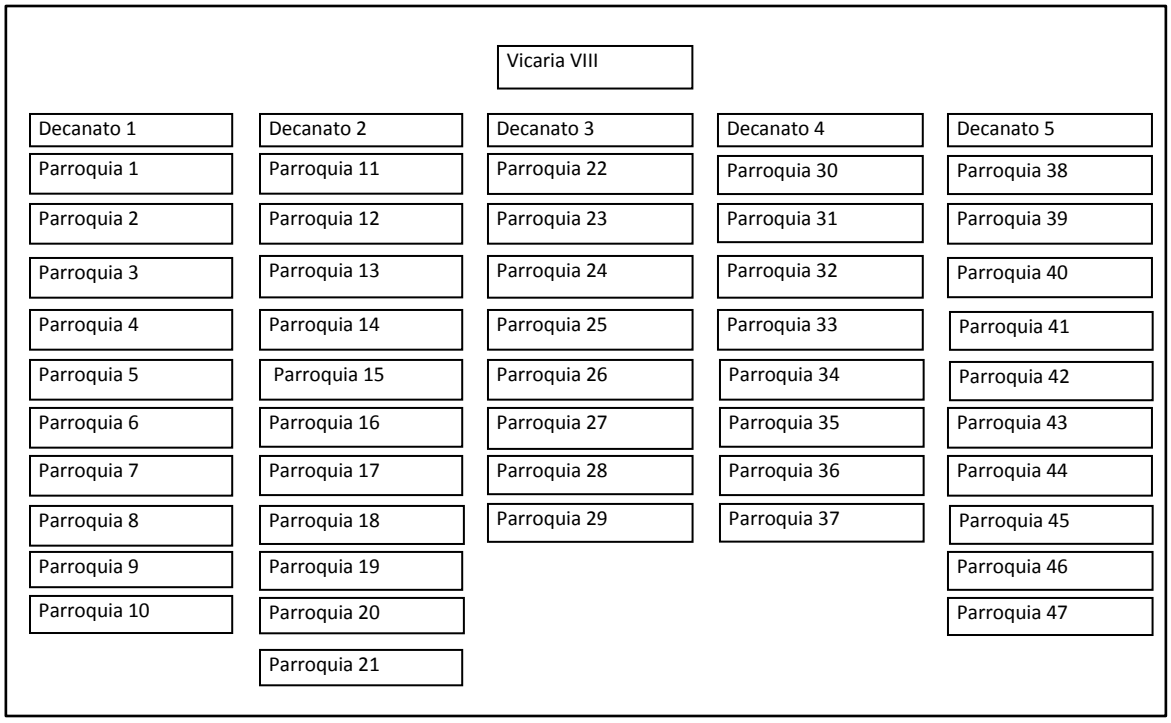


Diagrama 1.2 Diagrama de Bloques de la Vicaria VIII
Fuente: http://vicaria8.arquidiocesismexico.org.mx año (2010)

Se trabajó sólo en la parte de la parroquia que es donde se realizan todas las actividades que deseamos controlar, debido a que esta es la parte más pequeña del sistema que estudié pero es la parte principal, ya que ahí se realizan todas las operaciones como son los servicios, las ventas, las compras, la contabilidad y el inventario, sólo se realizó estudio en una parroquia debido a que todas las parroquias cuentan con las mismas actividades.

En cuanto a los ***servicios*** que se ofrecen podemos encontrar: bautizos, confirmaciones, confesiones, comuniones, matrimonios, misas, pláticas, bendiciones, permisos, expedición de documentos, amonestaciones, catecismo.

En cuanto a ***ventas***: si son ventas a mayoreo a menudeo, que producto es, libros, revistas, velas, rosarios, crucifijos, medallas, ropa para bautizo, recuerdos, y quien lo realizó.

Para las ***compras*** saber si son bienes y servicios, insumos o materia, prima; saber qué compañía la vende en dónde se localiza la compañía, el nombre del proveedor y algunas de las compras: vino para consagrar, hostias, velas, libros, revistas, imágenes, flores, arreglos decorativos, juegos pirotécnicos, etc.

En ***compra de servicios***: bandas de música, limpieza, pintores, restauradores.

En ***contabilidad*** pagos a proveedores, nomina e impuestos, cobros a clientes, transitorias y valuación.

Una vez reunida esta información se realizaron otros organigramas pero ahora identificando sus campos principales que corresponden a cada bloque de igual manera los que se heredan de los principales que nos permitirán controlar los objetos.

- ***Diagrama de uso***

Utilizando el programa de ArgoUML se realizó un diagrama de uso, un diagrama de clases y un diagrama de secuencias.

Una vez realizados los organigramas se utilizó la herramienta de Diagramas de Uso, estos describen cómo es el proceso que generalmente se emplea al hacer un movimiento por la persona que esta encargada por cada departamento, describiré brevemente el diagrama de uso *(ver diagrama **1.3**)*. En este diagrama de uso nombro como servidor a la persona que ingresa al sistema, por ejemplo puede ser el vendedor en una venta, el contador en uso de contable, el encargado del almacén en el almacén y la secretaria en servicios.

Comencemos por el caso de servicios: la secretaria selecciona el tipo de servicio, si sólo desean informes consulta en su base de datos y checa el estatus del servicio, si le piden apartar un servicio checa la disponibilidad, si la hay aparta el servicio, marca hora y fecha señalada, solicita la cooperación del servicio, si es dada la cooperación registra la cooperación e imprime el comprobante de servicio y comprobante de la cooperación.

En venta el vendedor levanta el pedido, pide informe fiscal, verifica o consulta si hay existencia de la mercancía, surte el pedido y lo descuenta del almacén, pide el pago de la venta, si lo recibe registra pago e imprime el ticket.

En contabilidad el contador selecciona pagos, después selecciona el tipo de pago en caso de proveedor libera el pago e imprime un contra recibo, en caso de nómina libera la nómina e imprime recibo de nómina, en caso de impuesto libera el pago de impuesto e imprime el monto a pagar. Si selecciona cobros imprime los recibos para ser cobrados. Si selecciona cuenta contable podrá seleccionar que tipo de cuenta requiere.

En almacén, el encargado consulta los productos existentes e imprime lista de productos. En caso de consulta de almacén de servicios consulta los servicios existentes e imprime lista de servicios.

- ***Diagrama de Clase:***

Apoyándome también de los Diagramas de clases, nombro los campos de cada clase mencionando su tipo, si es string, entero, indefinido, booleano, en el caso del servidor y describo las acciones que deberán realizarse, como nuevo servicio, consulta del servicio, o seleccionar el servicio, también se ve si la relación entre las clases es de 1 a 1 o de 1 a muchos que los represento así (1-1 o 1-0…*) como en el caso de producto hay una relación de 1 a 1 esto significa que si se consume un producto éste sólo se puede consumir solo una vez *(ver diagrama **1.4**).*

- ***Diagrama de Secuencia:***

Se realizó un diagrama de secuencia, donde detallo hacia donde van las actividades que realizo al hacer un movimiento, por ejemplo si el servidor selecciona el tipo de movimiento que es venta, para realizar la venta tiene que crear la venta, por lo cual le pide datos al cliente y lo da de alta, luego consulta existencia del producto, si lo hay surte el pedido y por consecuencia lo descuenta del inventario, pide el pago y registra el pago e imprime el ticket y se muestra en el diagrama de clases con una flecha punteada que termina el proceso y este regresa nuevamente en donde creó el pedido por si se requiere crear uno nuevo, si no es así se regresa hasta donde sea necesario *(ver diagrama **1.5**).*

Esta aplicación me permitió aprender las herramientas que están disponibles para poder realizar un proyecto y poder controlarlo de principio a fin y que dentro de estos proyectos involucran mucho personal especializado para cada tarea específica y para lograr una buena aplicación de sistemas de información es necesario conocer muy bien a la empresa tanto en sus objetivos como en sus procesos, por lo cual el análisis es un factor al que se debe asignar el suficiente tiempo para realizarlo ; debemos aprender a estandarizar todos las tareas para que así como el arquitecto que analiza unos planos de una construcción sabe qué es lo que debe de hacer para llevarlo acabo, de igual manera cuando tomemos un proyecto de una aplicación ya iniciada se pueda saber con claridad qué es lo que tenemos

que hacer con la información y de esta manera aprovechar todo lo que se ha venido construyendo en la aplicación y no que tengamos que volver a comenzar de cero debido a que no hay un historial de cómo se construyó esa aplicación.

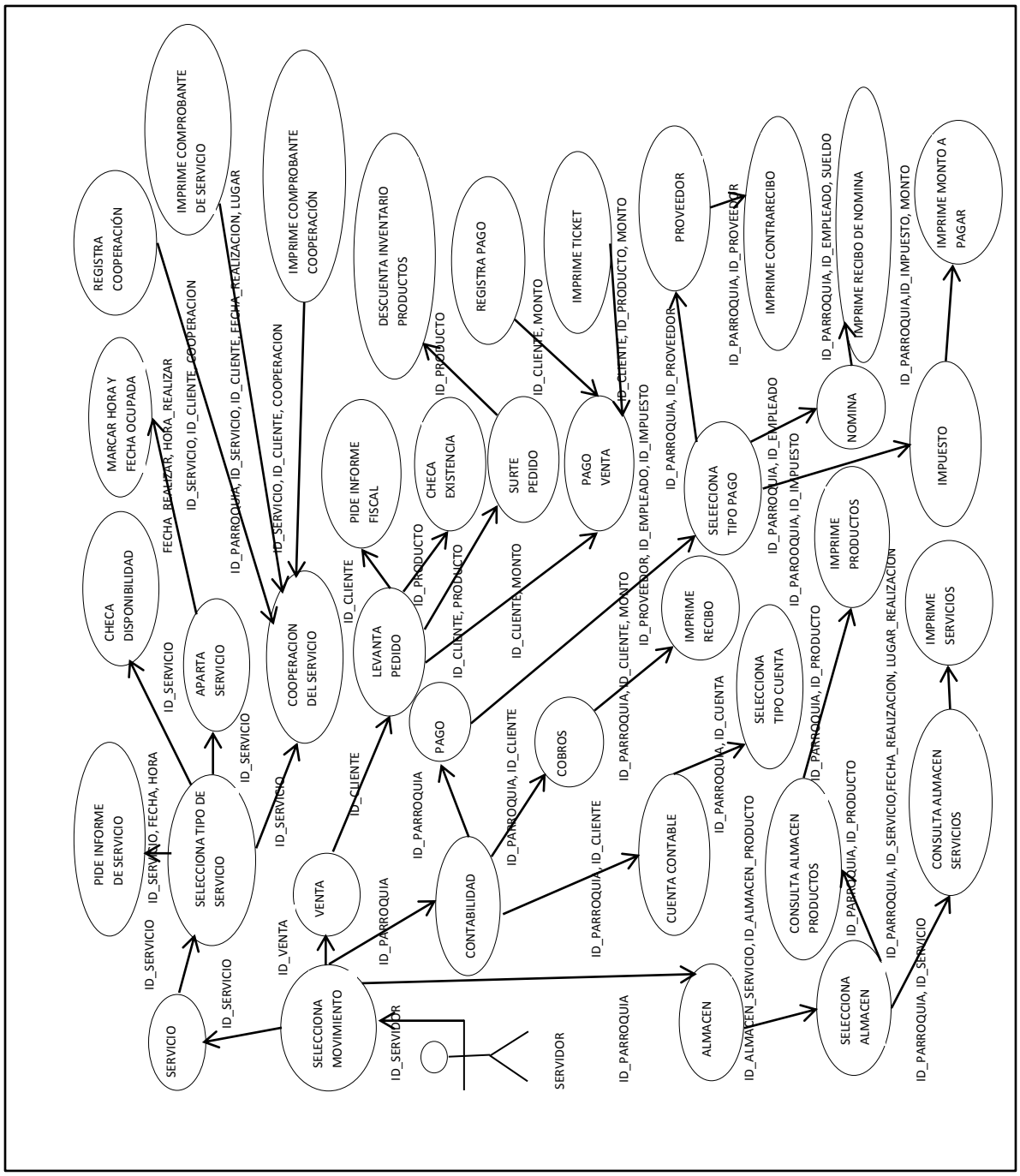


Diagrama 1.3. Diagrama de Usos de la Parroquia
Fuente: Elaboración propia (2010)

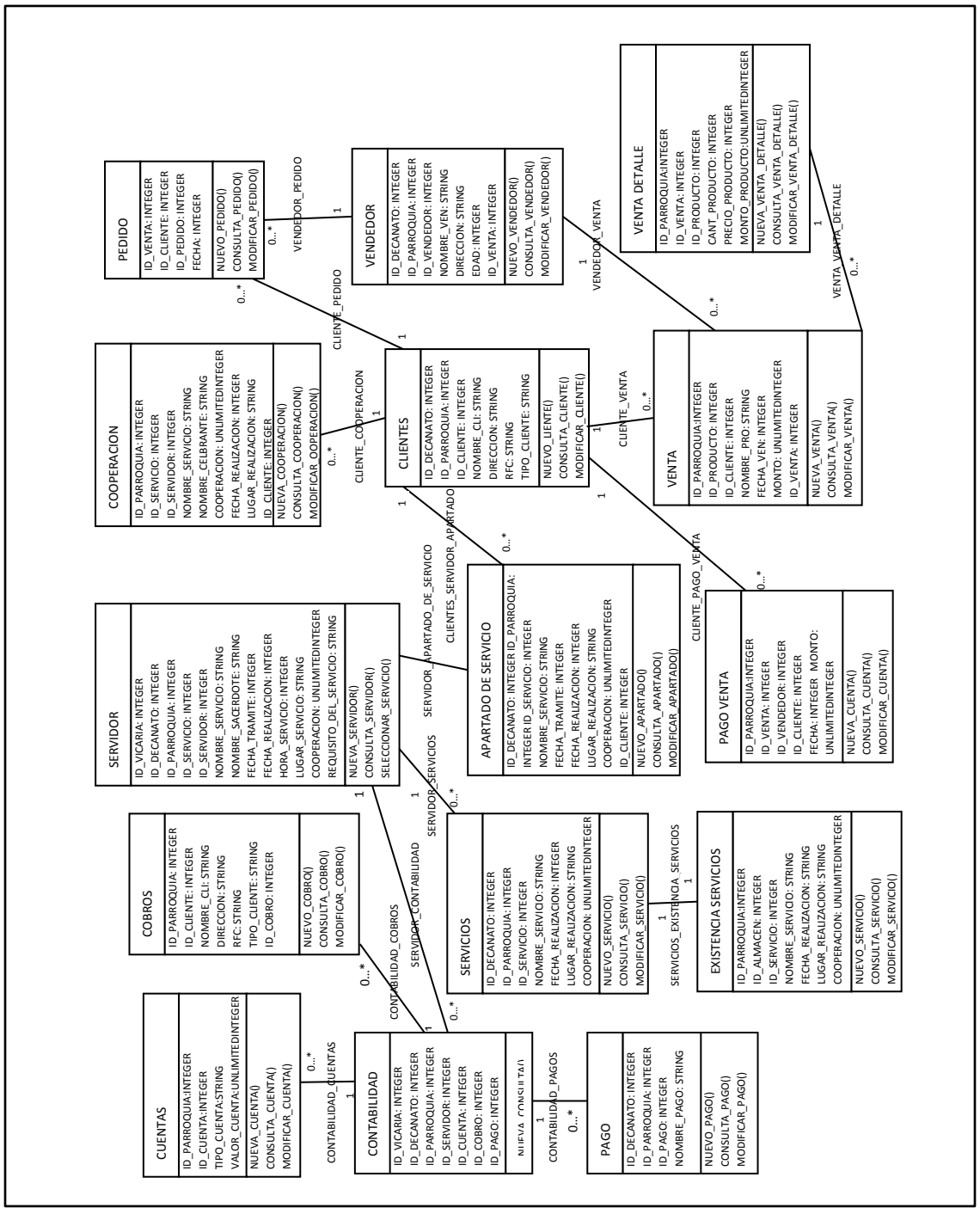


Diagrama 1.4. Diagrama de Clases de la Parroquia
Fuente: Elaboración propia año 2010

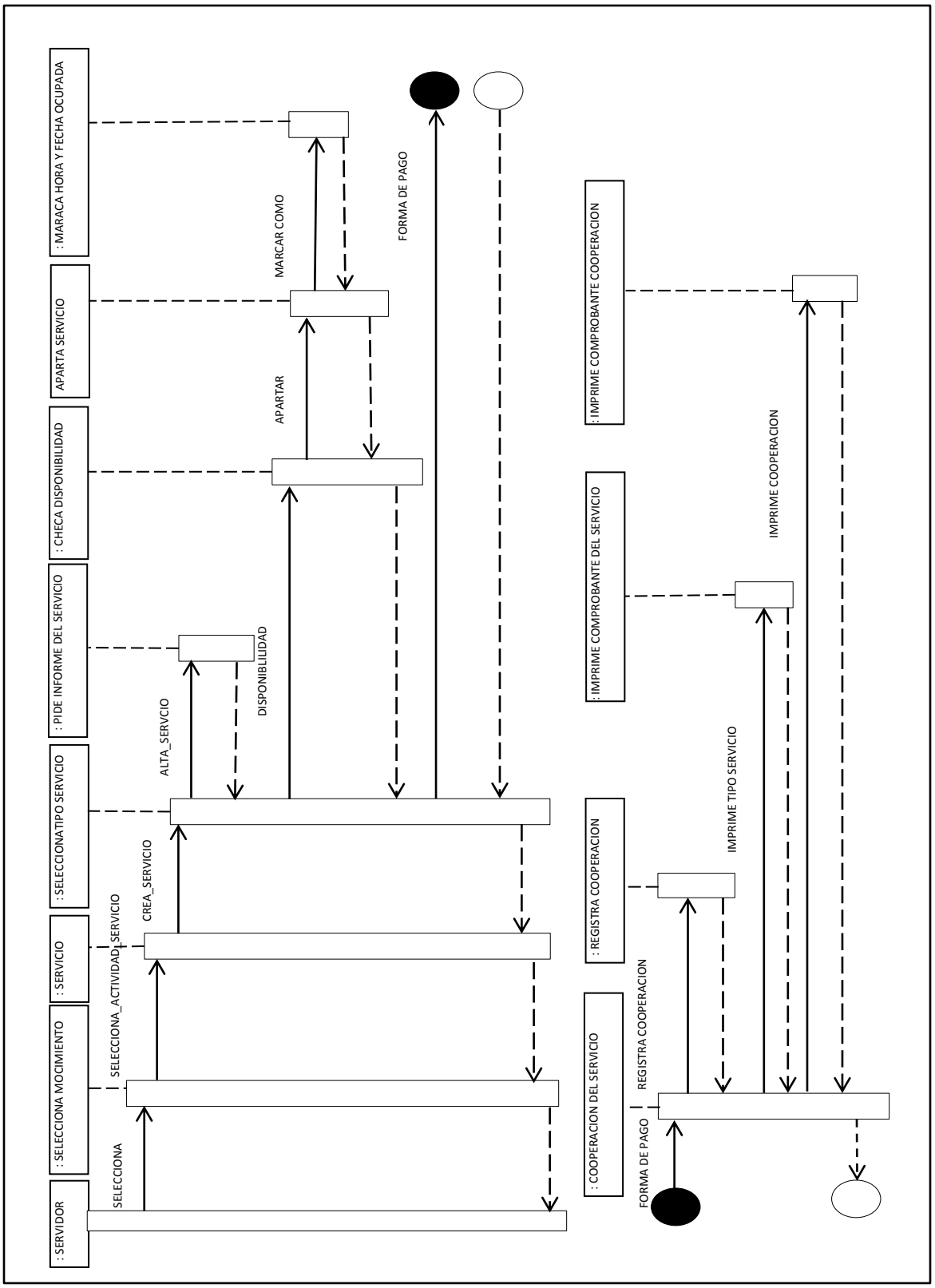


Diagrama 1.5.Diagrama de Secuencias de la Parroquia
Fuente: Elaboración propia año 2010

CAPÍTULO 2

LENGUAJE DE PROGRAMACIÓN EN JAVA

2.1 Conceptos generales

Java es un lenguaje de programación que cuenta con las siguientes características: simple, orientado a objetos, distribuido, multiproceso, dinámico, robusto, seguro, interpretado, portátil y de alto desempeño. Por tal motivo lo podemos explotar de muy diversas formas, ya que prácticamente corre en todas las plataformas, además está diseñado para que funcione en navegadores web y servicios; se pueden desarrollar aplicaciones para servidores y también para dispositivos móviles como celulares o tabletas, por sus características java es un lenguaje muy poderoso.

La plataforma de java se diferencia de otras plataformas ya que esta contiene una máquina virtual, además de la interfaz de programación de aplicaciones, conocida como API (***Java Aplication Programming Interface***).

2.1.1 Conceptos básicos de Java

Para programar se utilizó la consola de java utilizando un programa llamado Java SE Development kit 6, este nos permite introducir los comandos para poder compilar los programas y ejecutarlos, utilizando un editor de texto para realizar la estructura de los programas. En esta plataforma podemos realizar programas de aplicación y ***applets***, donde los applets son aplicaciones diseñadas para ser transmitidos por internet e interpretada por un navegador web.

Al igual que otros lenguajes de programación java también cuenta con palabras reservadas, como por ejemplo ***abstract***, ***import*** , ***implements***, ***trhows*** etc. Contiene las reglas que se utilizar para nombrar las clases, paquetes, interfaces, métodos, variables e instancias llamados identificadores, la cuales son: pueden tener cualquier longitud, no puede tener identificadores como **+**, **-** , etc.; no puede coincidir con una palabra reservada, el primer carácter solo puede ser una letra, el carácter ***$*** o el subrayado; después del primer carácter puede haber una serie de combinaciones de letras y dígitos; las letras pueden ser mayúsculas y minúsculas, acentuadas y la ***ñ***; por último debe ser referenciado de la misma forma. Sobre los tipos de datos tanto de variables como de constantes primero deben ser declarados estos, hay tipos de datos simples y los definidos por el usuario, los datos simples conocidos también como primitivos se dividen en enteros, reales, caracteres y boléanos. Para declarar una variable primero se debe de indicar el tipo de dato y después el nombre de la variable.

El lenguaje java tiene un elemento básico que son las clases aquí es donde se definen todas las propiedades de los objetos que pertenecen a ella llamadas atributos y las funciones

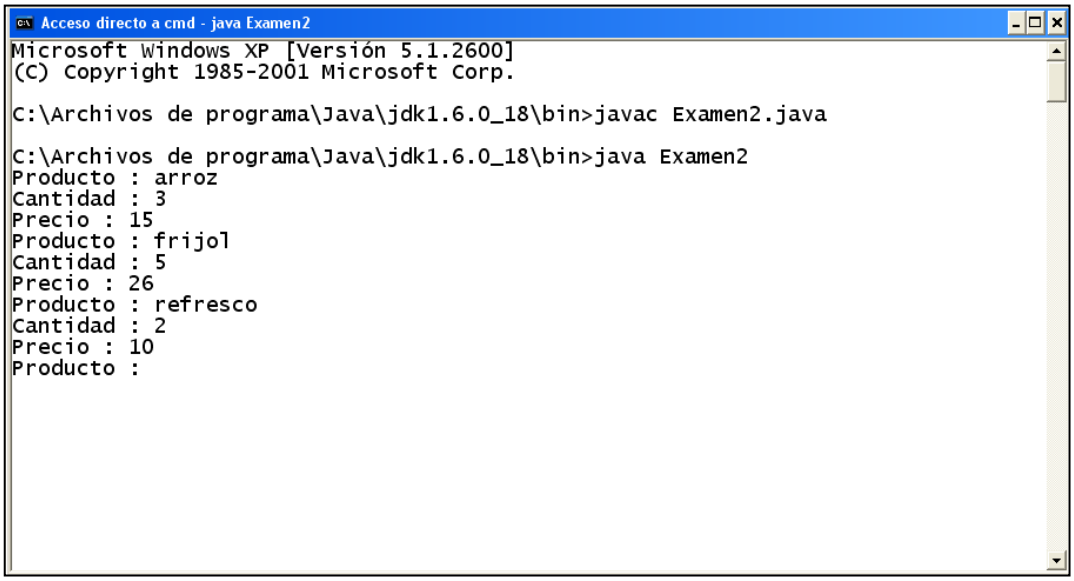
que actúan sobre ellos y se llaman métodos. En las bibliotecas de java es dónde podemos ver su potencia, ya que aquí es donde las clases hacen variadas tareas que facilitan la programación, la biblioteca más importante es la JAVA.LANG ya que esta misma se incluye automáticamente en el programa, otra que es de gran utilidad es la JAVA.UTIL debido a que implementa algunas de las estructuras más comunes de datos, tiene operaciones sobre fechas, el calendario y otras más, las bibliotecas también incluyen funciones que nos permiten detectar y hacer excepciones sobre los errores lo cual es muy útil ya que podemos controlarlos y hacer que el sistema no colapse ante uno de estos errores.

2.1.2 Programando con java.

La forma de trabajar en el curso fue realizando programas para mostrar algunas de las cosas que se pueden hacer con java, como copiar caracteres que están contenidos en un arreglo utilizando operadores lógicos como ***while*** y ***dowhile***; a cómo utilizar el operador, if, ***ifelse***, ***switch***; utilizamos los tipos de datos e hicimos conversiones por ejemplo de enteros a dobles, ver los valores de los caracteres que cuando ésta escrito en mayúsculas tiene un valor y cuando ésta escrito es minúsculas tiene otro valor, también cómo programar en bloques, cómo capturar los errores, crear bloques dentro del programa; utilizamos la biblioteca ***JAVA.IO*** que nos permite extraer información del sistema de la misma manera que podemos introducir datos desde el teclado, con esta biblioteca podemos copiar archivos, escanear archivos concatenar archivos, copiar líneas; de la misma manera podemos hacerlo a través de internet y poder extraer información del código fuente de una página web; con la biblioteca ***JAVA.UTIL*** podemos también programar tareas y colocarle un sonido "***beep***" para que nos recuerde que comenzó o terminó, utilizado en una forma de programación que se le conoce como hilos, en la que se pueden tener varios procesos a la vez, por ejemplo algunos necesitan de otros, esto quiere decir que unos son productores y otros consumidores, para poder comenzar un proceso consumidor necesita que el productor termine para poder utilizar lo que produce, utilizando hilos ya no es necesario que termine todo el proceso para poder utilizar los datos si no que puede utilizarlos en cuanto son liberados y esto nos ayuda a que los tiempos del procesamiento sean más cortos. Con la biblioteca ***JAVA.AWT*** y la ***JAVA.SWING***, podemos programar con un ambiente más vistoso utilizando frames y colocando botones de mando, el swing es una forma gráfica de programar. Por último mencionaré a los Applets que son aplicaciones hacia el internet, para esto utilizamos la biblioteca ***JAVA.APPLET*** y ***JAVA.AWT***, se crea la estructura del programa en un editor, y se crea otro programa pero ahora en HTML, en este se llama al programa creado en java y ya hemos creado un applet.

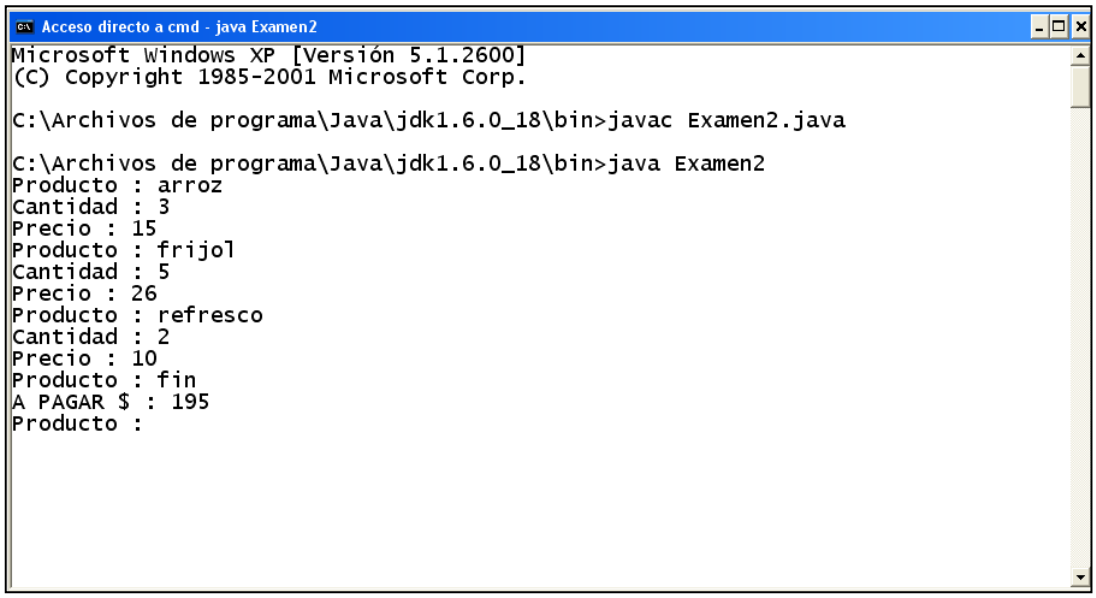
2.2 Producto obtenido en el módulo

Se realizó un programa que simulaba la venta de productos de un establecimiento como una tienda, donde debían capturar el producto la cantidad y el costo; y una vez capturado el primer producto pidiera el siguiente que debía de ingresar como se muestra en la ***vista 2.1***.



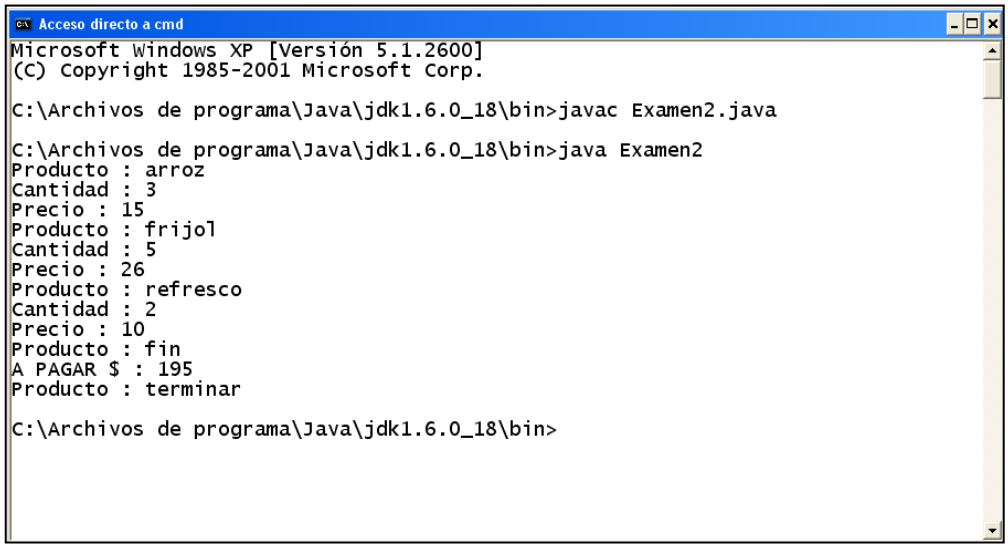
Vista 2.1 Captura de los datos de los productos
Fuente: Elaboración propia año 2010

Si no había más que ingresar en la compra se debía de teclear, fin y que nos diera el monto a pagar mostrado en la ***vista 2.2***.



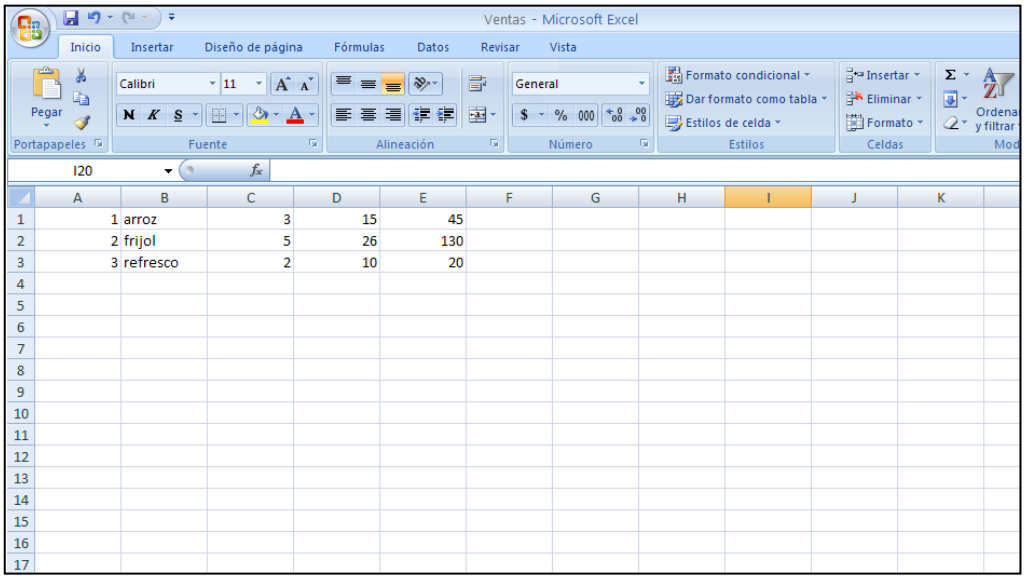
Vista 2.2 Al teclear la palabra fin aparece el total a pagar
Fuente: Elaboración propia año 2010

En caso de que no utilizáramos el sistema tecleáramos terminar en vez del nombre del producto y que nos sacara del sistema como se muestra en la ***vista 2.3***.



Vista 2.3 Al teclear la palabra terminar sale del sistema
Fuente: Elaboración propia año 2010

Simultáneamente que se realizaba el proceso de captura el programa guarda los datos en un archivo llamado ventas para que pudiéramos manipular la información, éste era un archivo separado por comas de Excel mostrado en la ***vista 2.4***, ya que estos nos permiten separar los datos, para que de esta manera pudiéramos sacar reportes como cuántos artículos se vendieron, el nombre del producto y el número de la venta.



Vista 2.4 Llenado de la hoja de Excel desde la captura del producto
Fuente: Elaboración propia año 2010

Este proyecto fue realizado para ejecutarse sólo en la consola, también lo podemos realizar para ambiente gráfico o como un applet.

CAPíTULO 3

LENGUAJE DE PROGRAMACIÓN AVANZADA

3.1 Conceptos generales

En este módulo aprovecharemos la propiedad que tiene java en la orientación a objetos, utilizamos un IDE (Entorno de desarrollo integrado) que es NetBeans, que nos permite desarrollar aplicaciones, para comenzar describiré la estructura del código fuente mediante este ejemplo:

```
public class NombreClase{

    public static void main(String[] args ){

        sentencias del programa

    }

}
```

Primeramente debemos escribir el nombre de la clase, después señalaremos que es una parte principal que es la que nos permitirá ejecutar el programa y por último tenemos el cuerpo del método que esta marcado por llaves.

3.1.1 Datos Primitivos

En java se utilizan 8 datos de tipo primitivo que son enteros como ***int***, short, ***long***, ***byte***, de coma flotante como ***float***, ***double***, de carácter ***char***, y un booleano de verdadero y falso. Otro punto importante es que cuando declaramos nuestras variables siempre hay que inicializarlas.

3.1.2 Clases

El diseño orientado a objetos se centra en los datos y las interfaces de los objetos, en las clases es dónde se crean los objetos y cuando construimos un objeto a partir de una clase se dice que hemos creado un ejemplar, todo esto se hace mediante la ayuda de la biblioteca que ofrece java, dentro de java se maneja el término encapsulación que no es otra cosa que combinar los datos y el comportamiento en un único paquete ocultando la implementación de los datos a la vista del usuario del objeto. A los datos del objeto se les llama campos del ejemplar, y a los procedimientos se les llama métodos. Dentro de las características de los objetos tenemos las siguientes, el comportamiento, el estado, y la identidad.

Para crear objetos utilizamos un constructor que es un método especial y el fin es darle valores iniciales, los constructores deben tener el mismo nombre que la clase, para poder quedarse con los objetos que se construyen basta con almacenar el descriptor del objeto en una variable. Java permite agrupar las clases en una colección llamada paquete. Los modificadores de acceso a las clases son ***public*** de acceso a todas las clases y subclases que lo referencien, ***private*** de acceso sólo en los métodos de la clase y ***protected*** de acceso por métodos de una superclase y por los métodos de cualquier clase que se deriven de esa superclase y por las otras clases en el mismo paquete. En los métodos de los parámetros, se da, por llamada, por valor, donde sólo el método obtiene una copia del valor de todos los parámetros y la llamada por referencia, dónde significa que se tiene la ubicación de los valores de los parámetros; java emplea el llamado por valor. Existen dos tipos de clases de parámetros, los tipos primitivos y los de referencia a objetos, es necesario saber que los métodos no pueden modificar parámetros de tipo primitivo, los métodos pueden cambiar el estado de los parámetros de tipo objeto que los métodos no pueden hacer que un parámetro de tipo objeto pase a referencia a un nuevo objeto. Debemos saber también que existen campos estáticos y que sólo hay uno de estos por clase, estos métodos no operan sobre objetos sólo pueden acceder sobre los campos estáticos de su clase, estos se utilizan en dos situaciones, cuando sólo un método únicamente necesita acceso a campos ***static*** y cuando los métodos no necesitan acceder al estado del objeto.

3.1.3 Herencia

La herencia es una de las propiedades de la programación orientada a objetos, donde las clases se crean absorbiendo los campos y los métodos de una clase existente; las ya existentes se les conoce como superclase y la nueva como subclase. Como las segundas deben valerse por si mismas no es conveniente utilizar el método de acceso ***protected*** ya que estos los hace dependientes y frágiles ya que un pequeño cambio en las primeras afectaría a la subclase, se recomienda que se cambie el acceso ***protected*** por ***private*** y se generen los métodos que nos permitan cambiar sus valores o que se utilice solo cuando se proporcione un servicio.

3.1.4 Referencias

Para asignar referencias hay cuatro formas; la primera es que la asignación de una referencia a una variable de tipo superclase es simple y directa, la segunda que una referencia a una variable de tipo subclase es simple y directa, la tercera que una asignación de la referencia de un objeto a una variable de tipo subclase es un objeto de la superclase y la cuarta es que cuando se asigna un objeto de superclase a una variable de tipo subclase, para evitar error en la compilación se debe convertir en el mismo tipo de manera explícita.

3.1.5 Polimorfismo

Otra de las propiedades en la programación orientada a objetos es el polimorfismo, siendo la capacidad de los objetos al responder a un mismo mensaje, haciendo que los objetos de clases que formen la misma jerarquía respondan como si fueran todos objetos de

sus superclases, cuando instanciamos objetos formamos lo que se llaman clases abstractas, estas al ser creadas son genéricas ya que no están completas, las encargadas de completar las partes faltantes son las subclases; para crearla se utiliza la palabra reservada ***abstract***, generalmente tienen uno o más métodos abstractos, estos actúan como reservas de espacio para los métodos que se implementarán; cuando se hereda de una clase abstracta se tiene dos opciones, se deja sin definir algunos de los métodos abstractos, entonces la subclase hay que marcarla como abstracta también, o se definen todos los métodos y ya no se declara abstracta.

3.1.6 Interfaces

La interfaz es el conjunto de requisitos que deben satisfacer las clases que se ajusten a esta, todos los métodos de esta son ***public***, no tienen campos y los métodos nunca se implementan, es en las clases donde se deben implementar y para crearlas se utiliza la palabra reservada interfaz. Para crear ejemplares se pueden declarar variables de este tipo siempre y cuando se implementen en la interfaz base. Para extender una interfaz se deben construir jerarquías de clase.

3.1.7 Acceso al disco

Las clases para acceder a la información del disco, como lectura y escritura, archivos de texto, datos primitivos, objetos de archivo que esta contenida en el paquete ***JAVA.UTIL***, como por ejemplo utilizar la clase ***Pattern*** para crear patrones y la clase ***Matche***r para la búsqueda de coincidencias, el paquete también lo podemos aplicar en las colecciones que nos permiten hacer operaciones como añadir, eliminar, añadir y obtener un objeto en la colección, tambien podemos iterar a través de una colección, con la clase ***hashtable*** que es una colección de objetos basada en claves para identificar objetos la ventaja es que puede ser cualquier tipo de objeto, aquí para poder recorrer el hashtable se necesita de otra clase que es ***Enumeration***.

3.1.8 JBDC

Se utilizó el API de conectividad de base de datos desde (JDBC),utilizando un lenguaje estructurado de consulta (SQL). Utilizamos el programa de base de datos Oracle que es compatible con JDBC.

Para poder conectarnos con la base de datos es necesario cargar el controlador de JDBC para poder conectarse con el DBMS, utilizando el método ***Class.forName();*** cargado éste se utiliza el método ***DriverManager.getConnection()*** de la clase driver manager, que recibe la base de datos a conectar, y verificaría nuestro password y usuario para poder acceder, después se puede crear instrucciones en SQL para consultar al DBMS utilizando el método ***Connection.createStatement(),*** el cual nos va a devolver un objeto ***ResultSet,*** este contiene el método next() que nos permite recorrer las filas devueltas, también contiene el método ***getString***, y el ***getInt*** en el cual le indicamos la columna a devolver, se puede indicar por un entero o por el nombre de la columna después sólo debemos de cerrar la conexión.

3.1.9 Swing

Es un GUI donde dibuja sobre ventanas en blanco, pero donde la interfaz del usuario es rico y cómodo, tiene pocas dependencias de la plataforma subyacente, ofrece una experiencia de usuario coherente entre plataformas. Para usar swing se utiliza la biblioteca JAVA.AWT y JAVAX.SWING, ésta contiene una clase llamada JFrame donde permite sacar ventanas, a la principal que es la más importante, se le llama marco, en estas pueden ir barras de menú, dentro del marco se colocan laminas que sirven para dibujar componentes, en estas GUI se manejan fuentes de eventos como botones, barras de desplazamiento, marcos etc.; se utilizan oyentes de eventos, que es cualquier evento que implemente la interfaz. Para organizar los componentes de Swing éste dispone de gestores de presentación ya que no dispone de un diseñador de formularios. Swing también cuenta con cuadros de dialogo que pueden ser modales cuando se necesita cierta información del usuario para seguir adelante y no modales cuando permite introducir información al usuario en el cuadro de diálogo como en el resto de la aplicación.

3.1.10 Interfaz de múltiples documentos (MDI)

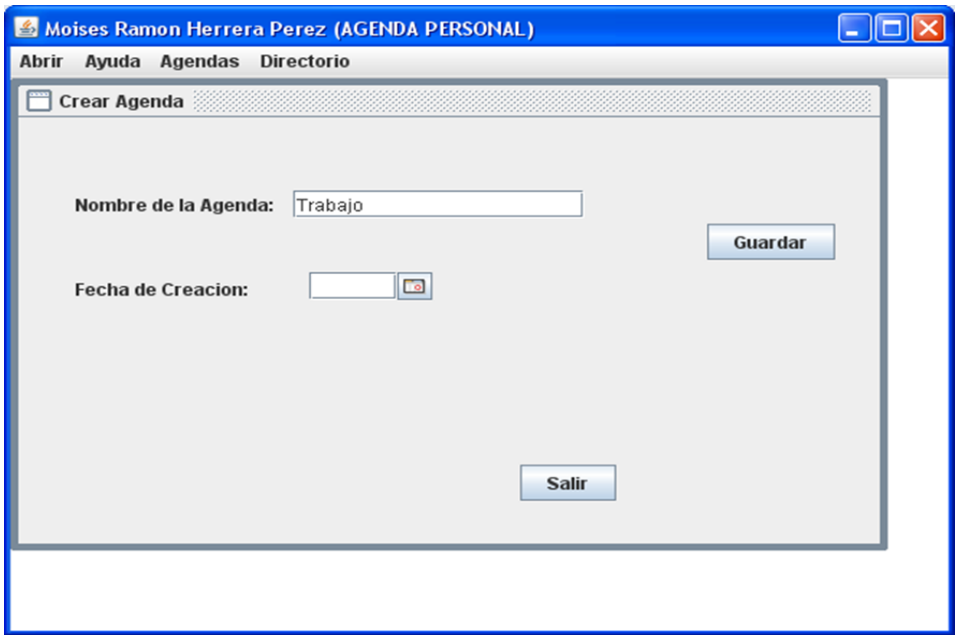
Son las aplicaciones que presentan múltiples ventanas contenidas en un único marco de grandes dimensiones, MDI reduce la complejidad de la ventana, éste sí cuenta con una parte de diseño, lo cual nos facilita más la programación que en swing, los marcos internos cuentan con asideros para trasladar los marcos, se puede modificar el tamaño de las ventanas arrastrando las esquinas. También se pueden crear cuadros de dialogo.

3.2 Producto Obtenido en el módulo

Nuestra aplicación fue crear una agenda personal en la cual se pudieran generar estas, consultar y también poder eliminarlas, dentro de esta misma crear directorios, consultarlos, actualizarlos y eliminarlos.

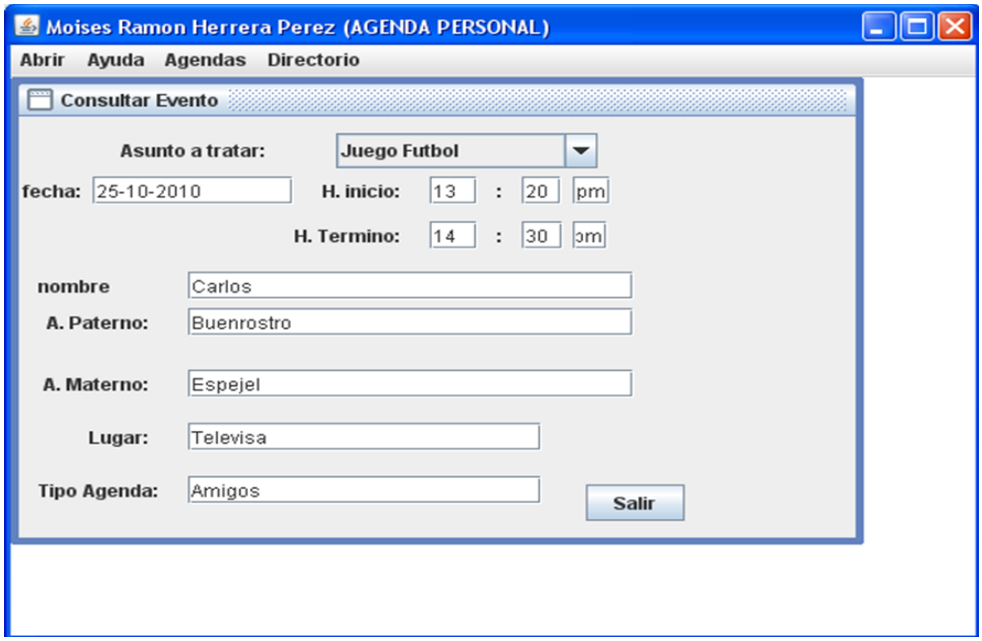
Para guardar los datos podía ser en archivos o en una base de datos, como Mysql y Oracle. Empleando Access para guardar los datos ya que JavaBeans la maneja como si fuera una base de datos aunque esta no sea en si, para lograr esto generé las tablas correspondientes una para contactos, eventos, agendas y directorios.

Una vez creadas estas tablas se obtiene un paquete donde se generen las clases en java que correspondían a cada una de las tablas, además de una clase para la conexión a Access, otras para hacer consultas y poder crear, guardar, actualizar y eliminar. Para mostrar las vistas se genera otro paquete donde se crearán las aplicaciones MDI estas tienen una parte de diseño y otra de programación, que es donde controlaremos la ventanas en el diseño al abrir un menú y dirigirnos a cierta actividad. Las siguientes imágenes muestran las vistas generadas en el proyecto de la agenda.



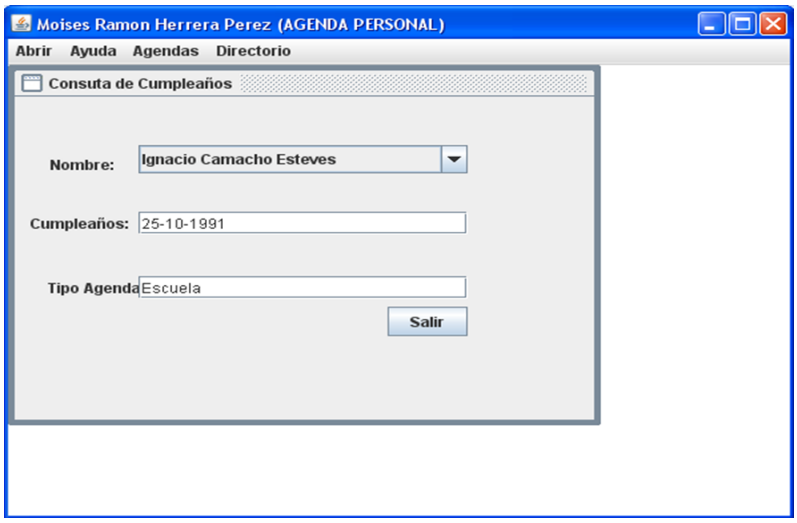
Vista 3.1 *Crear una Agenda*
Fuente: Elaboración propia año 2010

En la creación de la agenda debemos señalar el nombre, en este caso se pide la fecha de la creación de la misma, mostrada en la vista ***3.1***.



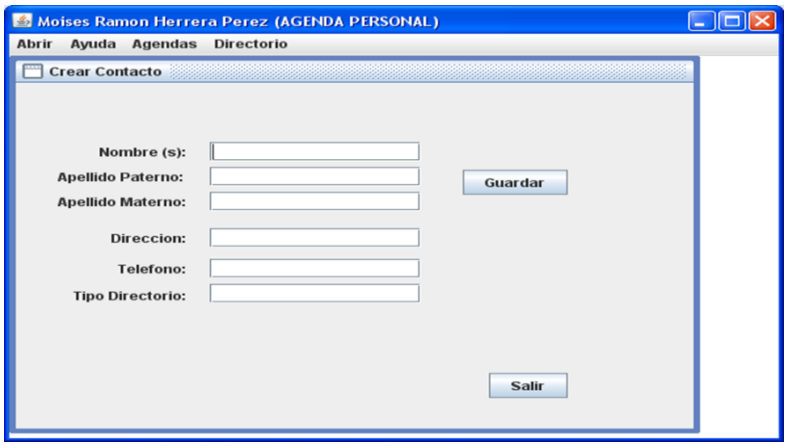
Vista 3.2 *Consulta de un Evento*
Fuente: Elaboración propia año 2010

Al consultar un evento elegimos el tipo de evento en el cual nos muestra con que persona hay que dirigirse, la hora, la fecha y el lugar donde se realizara, indicándonos a que agenda pertenece como se muestra en la vista ***3.2***.



Vista 3.3 *Consulta de Cumpleaños*
Fuente: Elaboración propia año 2010

Al consultar un cumpleaños se elige por el nombre y al seleccionarlo nos muestra el día de nacimiento y a que agenda pertenece mostrada en la vista ***3.3***.



Vista 3.4 *Crear un Contacto.*
Fuente: Elaboración propia año 2010

Cuando creamos un contacto nos pide sus datos como nombre, apellido paterno, apellido materno, dirección, teléfono y el tipo de directorio como se muestra en la vista ***3.4***. Este módulo me permitió comprender la forma en como se pueden realizar aplicaciones ya con objetos, en un ambiente de diseño gráfico el cual para nuestras aplicaciones nos da una mejor presentación y entendimiento de lo que se desea realizarlas con la facilidad que nos proporciona java para poder hacer conexiones a una base de datos nos abre una puerta a una infinidad de aplicaciones tales como crear una aplicación para una ama de casa por ejemplo de un recetario; hasta para una empresa de tener el control de la información que maneja.

CAPíTULO 4

JAVA CONECTIVIDAD PARA LA EMPRESA

4.1 Java Enterprise Edition(J2EE)

Aquí podemos tener una aplicación de usuario simple, donde uno utiliza el servidor y los demás esperan a que se desocupe; pueden haber usuarios múltiples donde estos a la misma vez pueden usar el servidor; es aquí donde nace la aplicación empresarial cuando utilizando la lógica de negocio y la aplicación le agregamos algo más como soporte a multiusuarios, persistencia a la información, autentificación al ingresar y otras aplicaciones. De aquí que podémos decir que una aplicación empresarial es una tecnología multinivel distribuidos orientado a negocios para proveer servicios que utiliza API's para su desarrollo, entre estas se encuentran los Servlets, JSP, JSF,JDBC.

4.1.1 Java Database Connectivity (JDBC)

Es una API que nos permite a los programadores invocar sentencia SQL desde métodos en lenguaje de programación java. Está formada por dos partes una interfaz a nivel aplicación y una interfaz proveedora del servicio. JDBC es un traductor que convierte los mensajes propietarios de bajo nivel del DBMS a mensajes de bajo nivel. Como lo mencioné en el capitulo III en JDBC, se tiene que cargar el driver; en este modulo se hizo una conexión a una base de datos diseñada en SQL, por lo cual utilizamos el driver para SQL y el procedimiento para tener acceso a la base de datos fue a través de SQL. Aquí se utilizan la API JAVA.SQL y JAVAX.SQL

4.1.2 MySQL

Es un administrador de bases de datos relacionales, es software libre y es muy popular, tiene un diseño multihilo que soporta una carga de forma eficiente, dispone de API's para diversos lenguajes de programación, es portable gracias a su código, soporta hasta 32 índices por tabla, tiene administración de usuario, soporta varios tipos de datos y además soporta SQL estándar parcialmente.

4.1.3 SQL (Structured Query Language) Lenguaje de Consulta estructurado

Esta compuesto por comandos, clausulas, operadores y funciones de agregado. Este se divide en la parte que nos permite crear que es DDL ***(Data Definition Language)*** y la que nos permite manipular las bases de datos DML ***(Data Manipulation Language).***

También utiliza procedimientos almacenados ***(Stored Procedure)*** que es una subrutina que se encuentra almacenada físicamente en el DBMS, están escritos en lenguaje nativo del DBMS como en este caso ***PL/SQL***, su uso principal es encapsular procesos grandes que realizan muchas operaciones sobre los datos. Consiste en manipulación de mucha información dando un solo resultado, estos son realizados por seguridad, por rendimiento del sistema o para bajar costos de desarrollo.

4.1.4 DAO (Data Access Object)

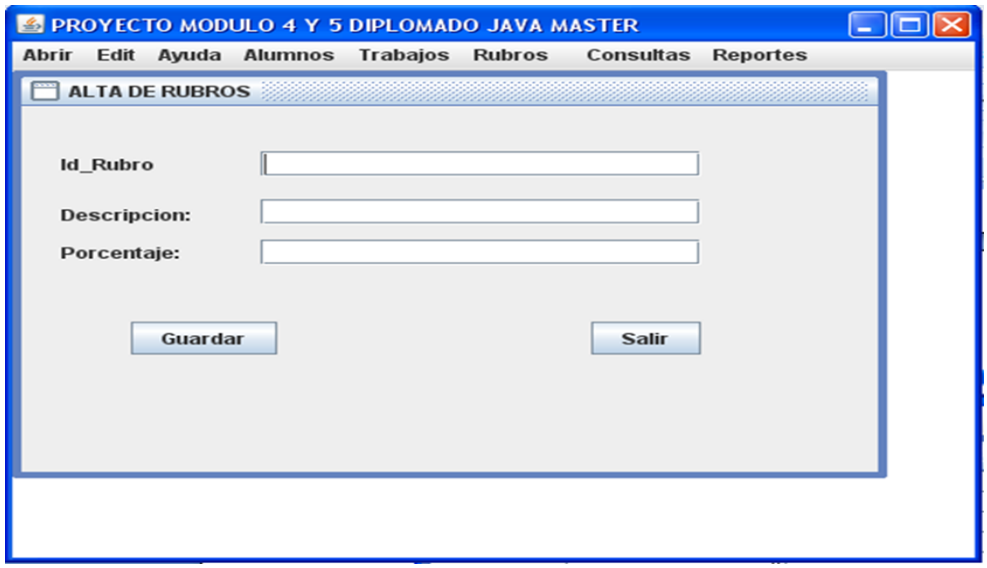
Es un patrón para dar una solución estándar a un problema común de programación, utiliza una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios, es un proyecto o estructura de implementación que logra una finalidad determinada, tiene un lenguaje de alto nivel, tiene una manera más práctica de describir ciertos aspectos de la organización de este, tiene conexiones entre componentes del mismo, es la forma de un diagrama de objeto o de un modelo de objeto.

El objetivo de este patrón es encapsular la fuente de datos, ocultando los métodos para su acceso, separando la Capa del DBMS y que su acceso sea por medio de objetos, no directamente.

4.2 Producto obtenido del módulo

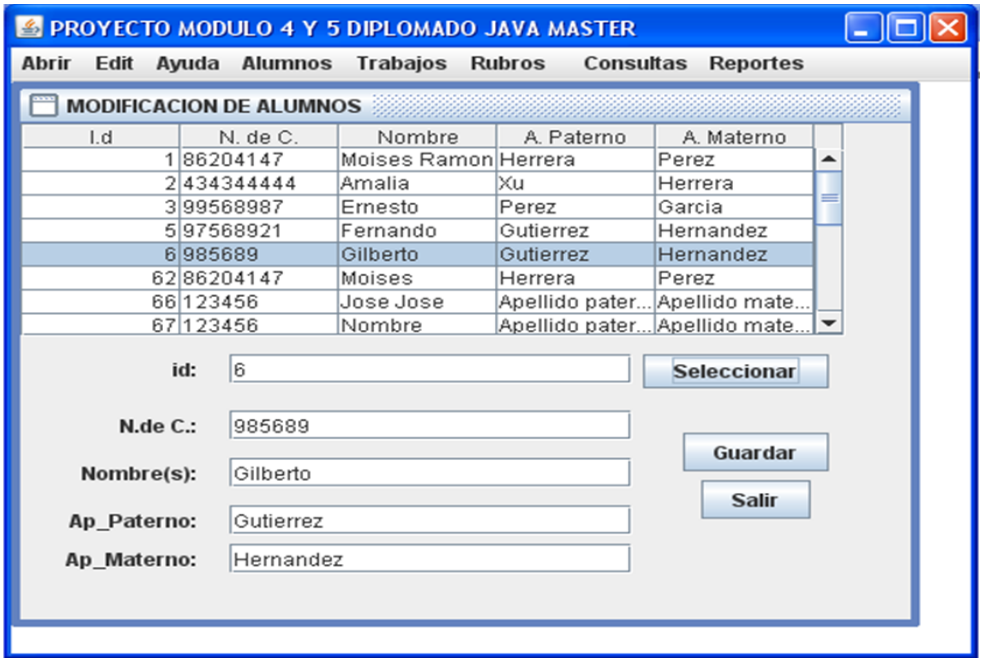
Este proyecto consistió en la utilización de un patrón llamado ***DAO***, el cual tiene una aplicación de tipo empresarial, este fue aplicado en un control escolar donde podíamos hacer altas, bajas, reportes, consultas de tareas por alumno con calificaciones, del promedio final por alumno, realizar búsquedas de alumno por apellido o número de cuenta y consultas de promedio por rubro. Aquí utilizamos la base de datos que nos proporcionaron en el diplomado que contenía las tablas ***alumno***, ***cat_rubros***, ***registros***, ***trabajos***, aquí lo único que añadimos a esta base de datos fueron los procedimientos almacenados que utilizamos para hacer algunas operaciones como para el promedio, este proyecto se fue desarrollando durante el módulo, para crear las vistas utilizamos también la aplicación MDI, en esta ocasión utilizamos tablas que nos ayudaron para mostrar los datos, para poder controlar los datos de las vistas se creó una clase de Control de ***tabla*** y para mostrar los comboBox se crearon clases de Modelo ***ComboBox***, el patrón DAO maneja una interfaz abstracta, en ella se crearon métodos para insertar, eliminar, actualizar y consultar, tanto al crear la interfaz como las que tengan contacto con la interfaz en su nombre clase deben de tener al final la terminación DAO, por ejemplo ***InterfazDAO*** ó ***AlumnoDAO*** para que el programa de Java reconozca que se está utilizando un patrón y puedan utilizarse sus propiedades, también se crearon en un paquete las clases que contienen las bases de datos, una clase para conexión a la base de datos.

A continuación muestro algunas vistas del proyecto; en la ***vista 4.1*** para dar de alta un rubro necesitamos introducir un “ID” de rubro, su descripción y el porcentaje del rubro y al darle guardar se registrará en la base de datos.



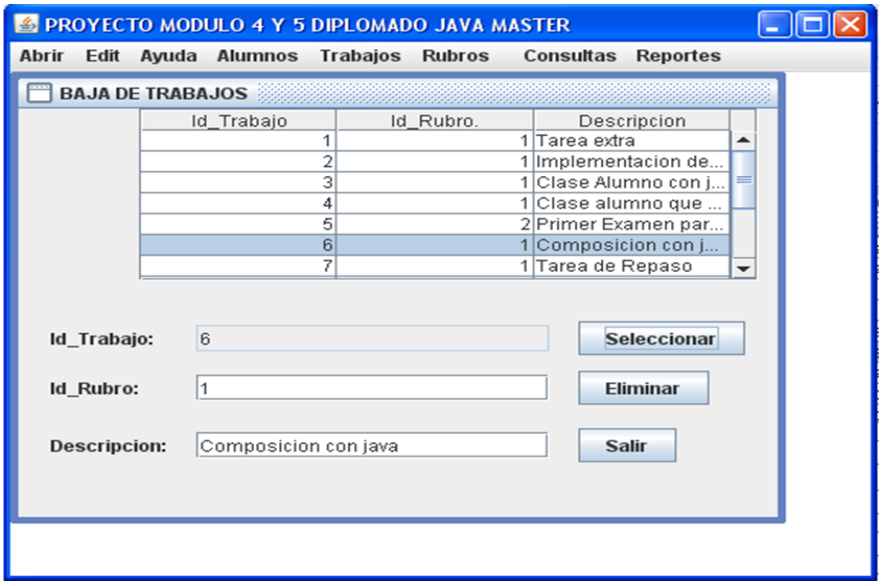
Vista 4.1 *Alta de Rubros*
Fuente: Elaboración propia año 2010

Para modificar los datos de los alumnos seleccionamos dentro de la tabla el alumno de nuestro interés y se oprime el botón seleccionar nos mostrará sus datos y podremos modificar sus campos oprimiendo en guardar se registraran estos cambios en la base de datos como lo muestra la ***vista 4.2.***



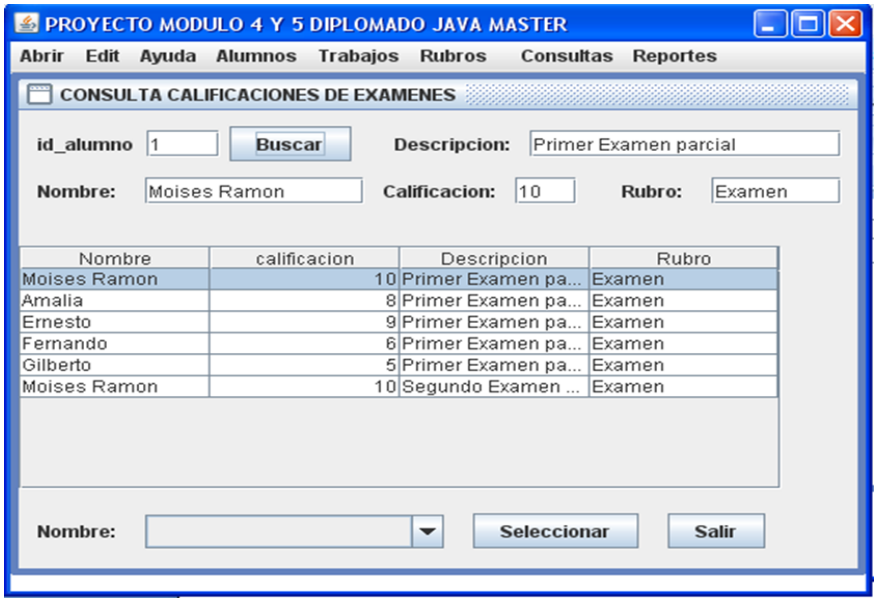
Vista 4.2 *Modificación de Alumnos.*
Fuente: Elaboración propia año 2010

Para dar de baja el trabajo de nuestro interés se selecciona y se oprime el botón seleccionar mostrará los datos del trabajo y que estemos seguros que es el que deseamos, y oprimiendo el botón eliminar se borra el registro de la base de datos como se muestra en la ***vista 4.3***.



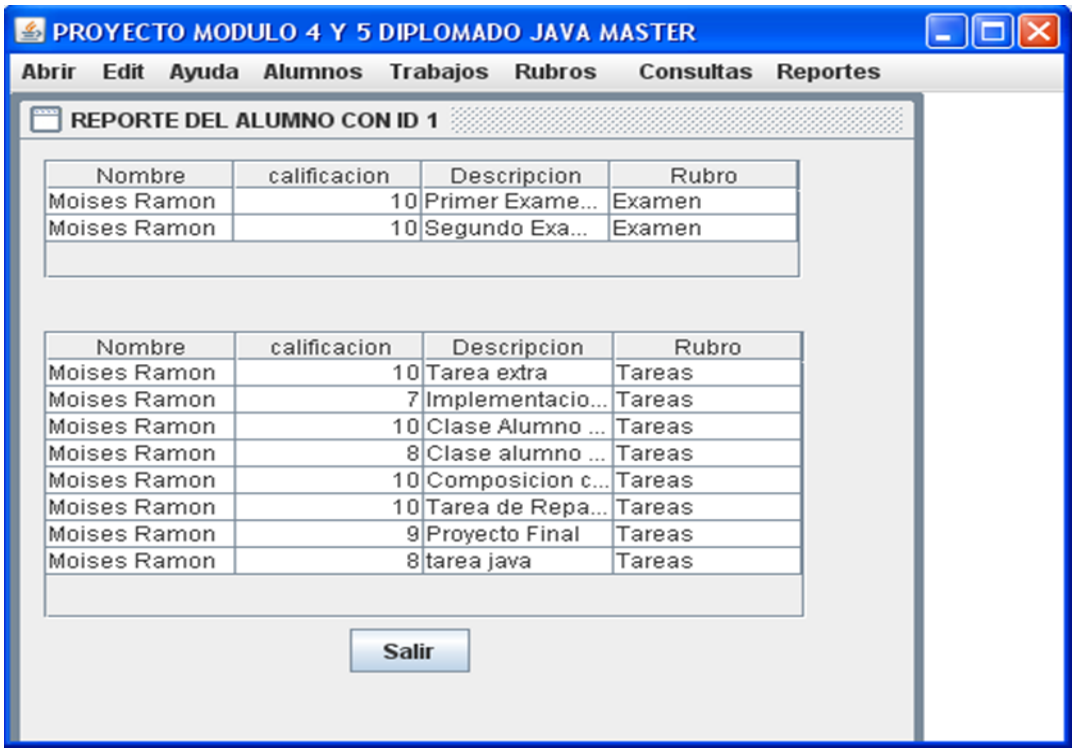
Vista 4.3 *Baja de Trabajos*
Fuente: Elaboración propia año 2010

Para consultar las calificaciones de exámenes se elige de la tabla el alumno de nuestro interés y se oprime el botón seleccionar y nos mostrara la calificación del alumno a que rubro corresponde dándonos su descripción y el nombre del alumno, también se puede buscar con su "id_alumno" oprimiendo el botón buscar como se muestra en la ***vista 4.4***.



Vista 4.4 *Consulta Calificaciones de Exámenes*
Fuente: Elaboración propia año 2010

En la ***vista 4.5*** nos muestra el reporte del alumno con "id" igual a 1 en el que nos muestra sus calificaciones de examen y de tareas dándonos la descripción de cada una de ellas.



Vista 4.5 Reporte del Alumno.
Fuente: Elaboración propia año 2010

El conocimiento obtenido de cómo funciona una base de datos y poder manipularlas pero mediante un patrón que se llama DAO así como lo vimos en la aplicación anterior esta se puede aplicar a instituciones escolares y modificarlos según las necesidades, claro que ésta es solo una parte, pero nos permite ver qué es lo que podemos llegar a hacer con estos patrones que manejan datos, ya que las bases de datos realizan los movimientos básicos de insertar, eliminar, modificar. En este módulo se hicieron aplicaciones pero para que tuvieran conexión a una base de datos externa, en este caso en un servidor de la escuela.

En esta aplicación que se conecta directamente a una base de datos remota, donde desde la pc en que se encuentre el programador puede crear las vistas que deseamos y mostrar los datos que requerimos consultar, en este caso tenemos también la autorización para realizar modificaciones, aquí entonces el programador puede trabajar desde el lugar donde se encuentre ya sea en las oficinas o en su casa. Los patrones DAO nos permiten una facilidad para el acceso a la información a la base de datos ya que ahora en estos días el acceso a la información es muy importante, el aprovechar este patrón en una institución brinda grandes beneficios tanto a los clientes como a los trabajadores.

CAPíTULO 5

JAVA PROGRAMACIÓN PARA DESARROLLADORES

5.1 SERVLETS

Es un programa que se ejecuta en el servidor Web, éste acepta peticiones de un cliente y procesa la información, también puede realizar tareas como comunicarse con otro servlet para ayudarle a su trabajo o para facilitar el acceso a la base de datos.

5.1.1 Java Server Pages (JSP)

Es una tecnología que permite combinar código HTML estático con código generado dinámicamente en un mismo fichero, solamente escribimos el HTML de la forma normal y encerramos el código de las partes dinámicas en etiquetas especiales. Es para hacer aplicaciones que corran en la WEB.

Dentro de los JSP existen tres tipos de elementos que son las Directivas, que permiten especificar la dirección acerca de la página que permanece constante para las peticiones, las acciones que permiten ejecutar determinadas acciones sobre información que se requiere en el momento de la petición del JSP y los scripts que permiten insertar código java que serán ejecutadas.

En los scripts se utiliza de la siguiente forma ***<%= expresión %>*** o en código ***<% code%>***; donde podemos realizar cualquier procedimiento como si estuviéramos programando en java.

Las directivas nos permiten re-direccionar a otras páginas, como puede ser una de error. Podemos decir si las variables a utilizar sólo se utilizarán en la página, en la aplicación y durante toda la sesión. Con estas podemos importar archivos.

Las acciones pueden ser estándares o personalizados, entre las estándares están ***<jsp: include>*** nos permiten incluir recursos estáticos o dinámicos, ejecuta y el resultado se incluye en la JSP original, ***<jsp: forward >*** es para redirigir la petición hacia otro recurso, se le puede dar varios resultados a una sola petición***;<jsp: param>*** sirve para incrustar valores en las acciones se utiliza generalmente con forward; ***<jsp: useBean>*** se usan para manipular JavaBean, estos son usados como objetos que se encuentran de lado del servidor y es posible acceder sus valores por los métodos set y get; ***<jsp: getProperty>*** nos permite obtener el valor de un atributo contenido en un JavaBean; ***<jsp: setProperty>*** nos permite establecer un valor nuevo para un atributo de un JavaBean; ***<jsp: plugin>*** permite general código HTML para asegurar que el navegador utiliza el ***plug-in*** de java.

5.1.2 Tag Libs

Estas abstraen el código mediante ***markup*** que se asemeja al HTML, mueven la lógica en clases Java que vinculan mediante ***tags***. La lógica puede manipularse sin tocar la JSP, nos permiten crear capas de presentación escrita en HTML o XML. Estos son de fácil utilización tanto para programadores y no programadores. La JSP que utiliza ***custom tag*** debe de incluir la directiva ***taglib***, dentro de las ***tags*** podemos utilizar operadores como ***if*** para evaluar una expresión, out para desplegar el valor de una literal, ***Choose*** que es una expresión equivalente a utilizar ***Switch***, la expresión ***forEach*** para hacer un recorrido enumerado además cuenta con la función SQL que nos permite hacer una conexión a una base de datos ,etc.

5.1.3 Custom Tags

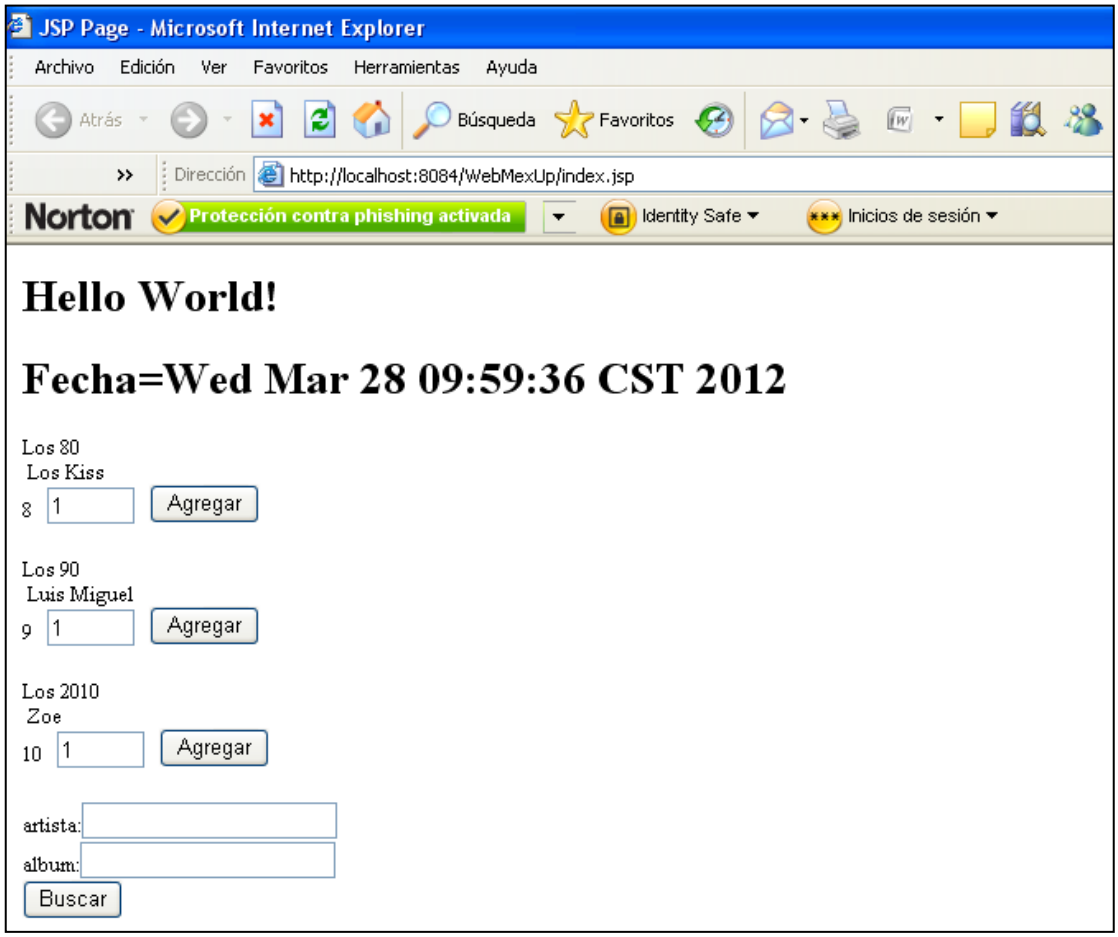
Estas tags son implementadas para personalizarlas o sea que nosotros las podemos crear a nuestras necesidades ya que la mayoría de las preestablecidas no se adecuan a lo que se necesita, estos archivos tienen la extensión ***tld***, siempre están vinculada a las clases java, estas implementan las interfaces ***javax.serlet.jsp.tagext.Tag*** que están dentro de la librería de java, al implementarla en una clase con ***extends Tag***, ***simpleTag*** o ***supportTag***, las cuales por abstracción nos crean métodos como ***doTag***, ***doStartTag***, en el cual podemos escribir código a nuestras necesidades.

5.2 Producto obtenido en el módulo

Explicaré algunos de los ejemplos que se realizaron en este módulo, en un ejemplo que usa el patrón DAO para hacer una tienda de discos con carrito de compras. Utilizamos una base de datos de Mysql que nos fué proporcionada durante el módulo. En esta tienda podíamos consultar el artista, el álbum, el precio y la existencia como se muestra en la ***vista 5.1***, al implementar la ***interfazDAO*** por abstracción se crean los constructores ***insert***, ***update***, ***delete***, ***findAll*** y ***findByPrimaryKe***y; después se crean las clases DAO que van a implementar la interfazDAO para que en ellas se construyan los métodos que nos va a permitir consultar la base de datos, en otro paquete se crean las clases de los datos persistentes y una clase que se le llamó ***carritoCompras*** para el cálculo de los artículos a comprar. Con todo lo anterior la aplicación esta completa, aquí sólo esta hecha para que la aplicación funcione sólo en la salida consola; ahora vamos a llevar esta aplicación a una aplicación web creando un JSP, que se construirán con SERVLETS que son aplicaciones que corren en un servidor web, para poder lograr esto se necesita tener instalado un servidor web en la plataforma de netbeans, el servidor que elegimos para realizar las aplicaciones en el curso de diplomado fue el ***Apache-tomcat*** que es software libre; en netbeans creamos la aplicación web y en la librería añadimos el proyecto anterior de la tienda de discos, JSP crea carpetas que contienen los programas una es ***webpages***, ***source pages***, ***Test pages***, ***libraries***, ***test libraries*** y ***configuratión files***.

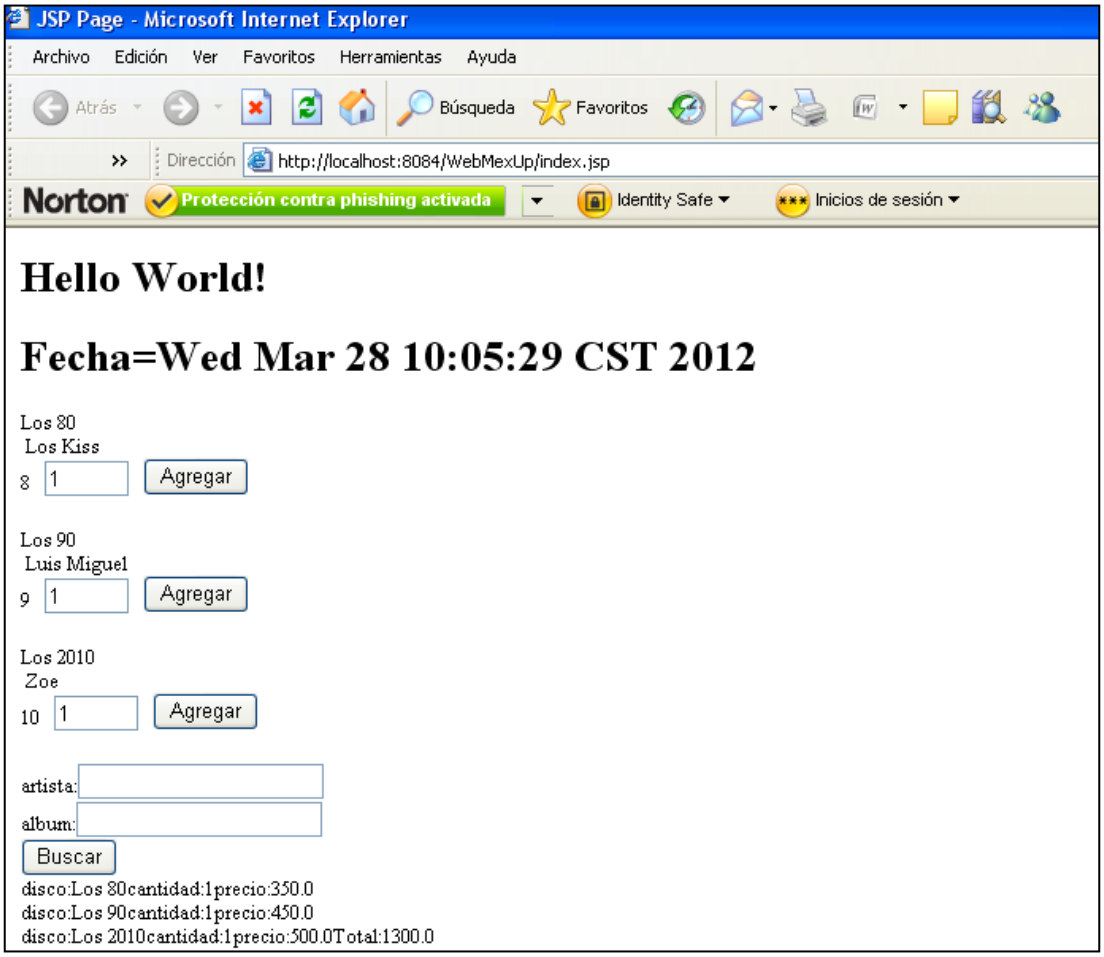
Ahora trabajamos en la vista para mostrar la información, nos ayudamos de una etiqueta personalizada, mostrando en la interfaz del usuario el artista, el álbum, el año, además de una caja de texto donde podíamos indicar la cantidad de discos a comprar y un botón para agregarlo, también otras cajas de texto donde nos permitía introducir el nombre del artista o el álbum y buscarlo; al ir agregando los discos a comprar por medio de etiquetas de textos nos mostraba la cantidad de discos, el precio y el total de la compra, hasta aquí concluye la aplicación, ahora muestro algunas imágenes

En la ***vista 5.1*** muestra la página principal que nos dá el nombre de los artistas, el álbum y el año y en la caja de texto nos da por default la cantidad 1 para agregar si deseamos agregar más se cambia la cantidad, también nos da la posibilidad de buscar por artista o por álbum.



Vista 5.1 *Página principal del carrito de compras.*
Fuente: 4to. Diplomado en Desarrollo en Java Master (2010)

Una vez que hemos agregado el álbum que deseamos, nos da la lista del nombre del álbum la cantidad y el precio de cada disco a comprar y al final nos da el total de la compra como se muestra en la ***vista 5.2***.



Vista 5.2 *Selección de artículos a comprar y su precio parcial y total*
Fuente: 4to. Diplomado en Desarrollo en Java Master (2010)

Se puede incluir una aplicación DAO en las librerías de la página web y mandarla llamar para mostrar los objetos que contiene la aplicación web, y con el uso de las etiquetas para mostrarlos en la página web me prepara para poder crear aplicaciones web con bases de datos; de esta forma se puede trabajar en equipo donde cada integrante puede realizar una parte del proyecto y al final sólo integrarla con una facilidad para obtener el producto final.

Esta aplicación es muy actual ya que la mayoría de las páginas web cuentan con una tienda virtual, ya que cada día las compras por internet van en aumento, también se pueden obtener información adicional sobre los clientes mediante las compras realizadas, como que marcas compra, que cantidad, de que lugar, con que frecuencia y de la misma manera el tenerle atención cuando hay un nuevo producto de su agrado enviándole la información necesaria, ésta se pueden comparar con otros clientes y hacer estadísticas por ejemplo donde es mas pedido su producto, y si es necesario acercar mas sucursales donde sus productos sean requeridos, cuando se maneja bien la información que recibimos se puede lograr grandes beneficios para la empresa.

CAPíTULO 6

J2EE DESARROLLO DE COMPONENTES WEB

6.1 Struts 2

Es un marco de trabajo de aplicaciones web (Framework) para la plataforma J2EE, esta aplicación surge debido a que ha mejorado la velocidad del internet, a la conectividad entre plataformas, al ocupar protocolos de transferencia de hipertexto y al avance de las tecnologías cliente servidor, para comunicarse vía web utiliza servlets. Para realizar una aplicación web se hace la conversión de parámetros de petición a Java, se validan los datos, se hacen llamadas a la lógica de negocios, se llama a la capa de datos, se hace la capa de presentación y por último la internacionalización.

Utilizaremos el patrón MVC(Modelo-Vista-Controlador), la finalidad de este es poder separar su lógica de interfaz usuario de la lógica de negocio, logrando con esto reutilizar la lógica de negocio, si se realizan cambios en la interfaz. El ***modelo*** se centra con las funcionalidades relacionadas con los datos; la ***vista*** responde al aspecto visual gráfico para interactuar creando la interfaz del usuario; y el ***controlador*** son las acciones que son llevadas entre vista y modelo.

Al realizar una petición en Struts 2, antes de llegar a la acción, se encuentran interceptores que la analizaran dentro del contexto donde se encuentran los datos almacenados, si es valida la petición se pasa a la acción y se manda el resultado y éste otra vez pasa a los interceptores antes de llegar al cliente.

Para poder utilizarlo es necesario bajar de internet los archivos JAR de ***Struts 2-2.014 all*** de allí solo hay que tomar ***commons-logging***; ***fremaker***, ***ognl***, ***struts2-core*** y ***xwork*** y los añadimos a la librería; para utilizar anotaciones debemos de hacer lo siguiente declarar el descriptor de despliegue(web.xml) en donde se encuentran las clases con acciones y debemos implementar la interfaz com.opensymphony.xwork2.Action o que nuestras clases terminen con la palabra ***Action***. Podemos también construirlos directamente con la aplicación de propia de Netbeans pero es necesario bajar de internet sus complementos y cargarlos en la aplicación, haciendo esto podemos utilizar la clase ***Action Support*** que implementa interfaces y nos permite validar datos o ir a localizar mensajes mediante el método ***validate().*** Este nos permite implementar transferencias profundas, una de ellas es crear objetos para que sus datos se trasladen directamente al objeto, por ejemplo si creamos un objeto usuario este puede trasladar sus datos a la salida de una página web.

La transferencia de datos nos permite internacionalizar nuestra página web debido a que podemos crear mensajes en el idioma que se necesite, también nos permite filtrar datos para manipular sólo los que deseamos, también nos permite realizar las proyecciones, de las

peticiones si son de tipo ***request***, ***aplicación*** o ***sesión*** esto es muy importante porque es el alcance en que los datos estarán disponibles.

6.1.1 OGNL(Lenguaje de Navegación de Objetos y Gráficos)

Es una tecnología que utiliza Struts2, es la encargada de unir el input con el output del HTTP basado en cadenas en el marco de trabajo y el procesamiento interno, éste se divide en dos cosas: Lenguaje de expresiones que es cuando convertimos un campo de formulario a nuestra propiedad en java y la conversión de tipo que nos ofrece convertidores de tipo implícitos desde sencillos a complejas como colecciones.

6.1.2 Etiquetas de Struts

Para hacer las peticiones se cuenta con etiquetas, que pueden ser de datos, de control de flujo, de interfaz de usuario y mixtas. Los atributos de las entradas pueden pasar una cadena y las que son diferentes, estás ultimas hacen referencia a ***ValueStack***. Algunas de las etiquetas de datos pueden ser ***property*** que es una vinculación con la propiedad a renderizar; set,con ella podemos renombrar a la propiedad; ***push*** ésta empuja las propiedades a ***ValueStack*** para utilizarlas; ***Bean,*** nos permite crear una instancia de un objeto para poder trabajar en él , su duración es sólo donde existe la etiqueta; ***Action*** con ella podemos invocar una acción desde la capa de vista; ***Iterator*** para iterar con colecciones: De control de flujo, ***if*** y ***else*** para ocupar la lógica de control; URL para crear ***url*** y poder pasarle parámetros; Include nos permite incluir un JSP, ***Servlet*** u otro recurso. De interfaz de usuario sólo mencionaré algunas como, ***select*** que es para trabajar con conjunto de colecciones; ***checkBoxlist,*** crea una lista de ***checkboxes*** relacionados con el mismo atributo; ***comboBox,*** combina caja de texto y ***select***;" ***CheckBox***" crea un ***CheckBox HTML***; ***Doubleselect***, crea dos elementos ***select*** donde el segundo modifica sus valores dependiendo del primero; ***updownselect*** crea un ***select***, con una serie de botones, para que el usuario ordene la lista; ***inputTrasnferedselect*** creará un ***select*** con una serie de botones para eliminar ordenar e ingresar nuevos valores y ***optgroup*** hace un grupo de elementos que manejará un ***select***.

6.1.3 JDBC con Strut 2

El gran alcance que tienen las etiquetas de struts2 nos permiten trabajar con bases de datos, de la misma manera que nos conectamos a una base de datos de Oracle y Mysql se hace aquí, utilizando los paquetes donde tenemos las clases y de ahí manipularla y obtener las respuestas de las peticiones y transferirlas a la capa de vista.

Muchas veces necesitamos que en la parte de la vista se conserven algunos datos cuando hacemos un movimiento, es decir refrescar solamente unos datos, por ejemplo si queremos que nos marque la hora pero que no se quede estática sino que siga cambiando en la página, se logra añadiéndole más archivos JAR a la aplicación como son "***dojo***" que para habilitarla será necesario incluir la ***taglib*** que nos dirija a ***dojo*** en la página a utilizar dicho código, esta forma de presentar la información nos permite tener un entorno más agradable en la interacción.

Una forma de utilizar JDBC es crear un programa con strut 2, en la cual debe de contener paquetes de clases y que en uno de ellos nos permita la conexión a la base de datos, y en otra que nos permita hacer las consultas en otro paquete tendremos las clases que nos van a permitir controlar las tablas con las que están construidas y de las cuales instanciaremos los objetos, en el paquete de default estará un archivo struts.xml donde podemos implementar las acciones para controlar las clases de los paquetes, de la parte web debemos crear un archivo ***jsp*** que iniciará la página, y podemos crea otras páginas como formularios y salidas según la acción que le indiquemos, esta parte es importante porque debemos saber de qué tipo de datos son los de la petición, también se generan por default un archivo web.xml y un context.xml que también los podemos personalizar según nuestra aplicación.

6.1.4 Hibernate y Struts2

Es un marco de trabajo de fuente abierta para integrar un modelo de dominio orientado a objetos con una base de datos relacional. El diseño de la aplicación esta conformada por una lógica de negocio, objetos de dominio java***, Hibernate*** y por último JDBC.

Se necesitan cuatro componentes en Hibernate el primero son los objetos de java persistente, que son las clases que tienen una representación en la abstracción del dominio, el segundo es el archivo de configuración de hibernate para su conexión a la base de datos que incluye toda la información de la base de datos y las clases persistentes, la tercera es el archivo de Mapeo Hibernate que asocia las columnas y las relaciona entre clases, el cuarto es el API ***Hibernate***, para tener acceso a todas las clases y que podamos controlar las interfaces en el caso de las clase se cuenta con ***SessionFactory*** que es donde se encuentran los metadatos de la base de datos y la información de las clase java para realizar operaciones persistentes y sesión que es la entrada para toda operación de persistencia ya que tiene métodos para guardar y cargar objetos, de la parte de interfaz tiene "***Transaction***" que es una interfaz que hará una operación en la base de datos, "***Query***" nos sirve para recuperar objetos que utiliza un lenguaje de SQL llamado ***HQL***, y "***Criteria***" su aplicación es más orientada a objetos que "***query***" utiliza SQL dinámicas en tiempo de ejecución.

Para crear una aplicación en hibernate debemos de incluir las librerías Struts2 y hibernate, después debemos configurar el archivo hibernate.cfg.xml que se crea por default que es el que nos permitirá tener la conexión a la base de datos y dar acceso a la información, una vez hecho esto y que ya tenemos conexión a la base abrimos un paquete y creamos los archivos ***POJO*** con el hechicero de Netbeans este utiliza la ingeniería inversa y crea las clases persistentes por nosotros, esta es una de las ventajas de hibernate, en otro paquete creamos una clase que nos permita controlar nuestras peticiones, después en el paquete donde están los archivos POJO se crea el archivo ***hibernateUtil*** con el hechicero de Netbeans y se hace una nueva configuración en vez de anotación, también en ese paquete crea otra clase, pero al nombrar la clase deberá tener la terminación UI que la utilizaremos en la capa de vista para transferir los datos por medio de una etiqueta de interfaz de usuario(UI) donde programaremos las transacciones que se requieran, se crea también un archivo struts.xml

en el paquete que se genera por default; por último en la carpeta ***web-pages*** se creará un archivo de ***inicio.jsp*** y en otra carpeta dentro de web page se creara otra carpeta con los archivos ***jsp*** que pueden ser formularios y salidas, lar carpetas ***META-INF*** y ***web-inf*** se crean por default. Toda esta serie de configuraciones es lo que forman una aplicación en hibernet.

6.2 Producto obtenido en el módulo

El proyecto es elaborar un ***Java Server Pages*** (JSP) donde sólo deba subir imágenes pero en formato jpg, pero para poder acceder a este recurso debo primero registrarme, una vez hecho esto ya puedo subir archivos y ver el nombre de los archivos que he subido, también tener la opción para descargarlos y eliminarlos, debe haber una cuenta para un administrador que tenga acceso a los archivos de los usuarios.

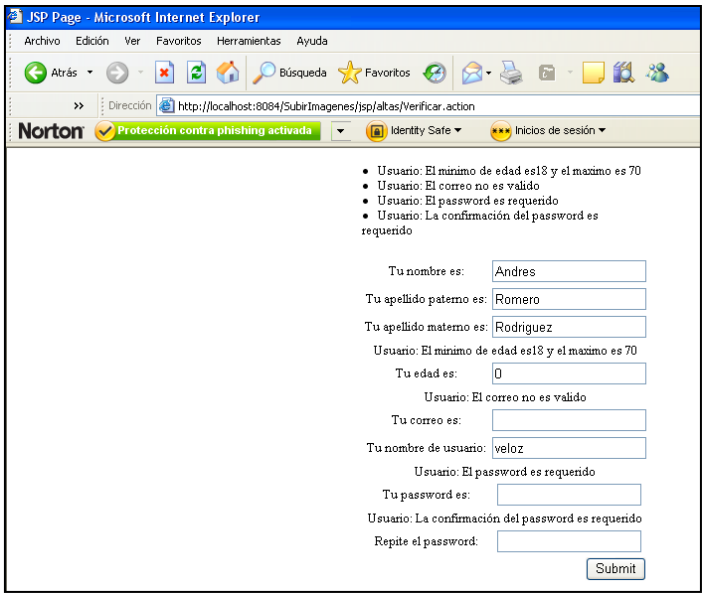
Para realizar este proyecto utilicé Hibernet y Struts2, Mysql para generar la base de datos. En la aplicación del JSP al registrarnos el programa debería de ser capaz de registrarlo en la base de datos y a la véz generar una carpeta con su nombre de usuario para guardar sus archivos, y en la página donde se registraba debían de ir validados todos los campos en el caso del correo que utilizara el @ , en el usuario que no fuera repetido y si cumplía con todos los requisitos, se registraría en la base de datos, en cuanto a que ya se registró y desea subir imágenes el archivo a subir lo valide para que sólo pueda subir archivos que son de imagen con extensión jpg, y si cumple con esto registrarlo tanto en la base de datos como colocarlo en la carpeta del usuario, y de la misma forma si desea eliminarlo.

Este proyecto es una aplicación que utilizamos cotidianamente y que nos permiten prestaciones como en este caso el de subir archivos, lo cual al realizarlo nos da una visión muy completa siendo ésta una base que se utiliza en la mayoría de los sitios web donde se pide que para usar sus aplicaciones es necesario registrarse.

Esta aplicación nos permite tomar la información de nuestros usuarios y mejorar el servicio que se les brinda, ya que mediante el formulario recibimos información que es de gran importancia para la empresa, en este caso se podría saber cuanto espacio se le podría disponer para guardar sus archivos y en caso que necesitarán más ofrecerles mas espacio ya sea gratuitamente o con un costo extra, también se pudiera saber si el servicio es para una persona o una empresa.

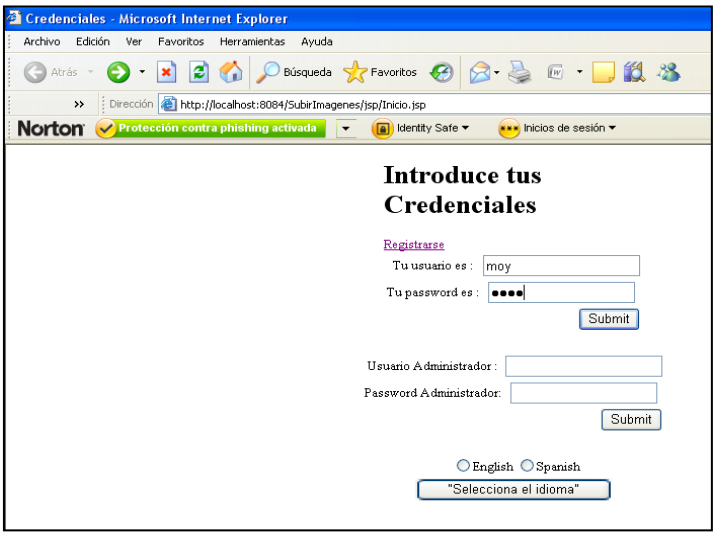
La aplicación nos da una idea clara de lo que ahora son las paginas web que es necesario registrarse para obtener los beneficios, y que las empresas la utilizan para obtener información nuestra y saber que tan populares son en el mercado de acuerdo a los seguidores que están registrados, y en cuanto a sus prestaciones pueden ser: tener acceso a juegos, videos, sorteos, información, exclusivas, compartir mensajes y videoconferencias que son utilizadas para atraer a los usuarios, las mas populares son las de las redes sociales y cada día van incrementado sus seguidores, estos últimos aprovechan mejor la información que es entregada mediante el registro, por lo tanto esta aplicación es muy importante para todo programador ya que esta parte es requerida dentro de una página web.

A continuación muestro algunas imágenes del proyecto:



Vista 6.1 *Verificación de Datos*
Fuente: Elaboración propia año2011

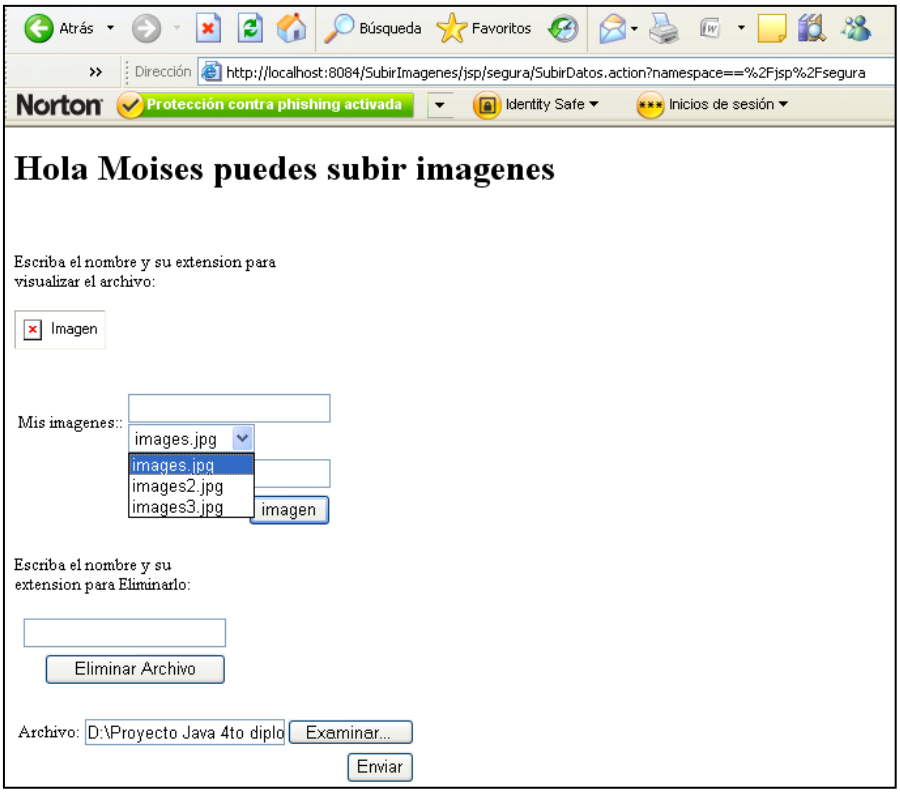
En la ***vista 6.1*** se muestra que si al registrarse uno omite campos este validará los campos antes de registrarlo en la base de datos y si no se cumple con ello mandará mensajes hasta que sean correctamente llenados.



Vista 6.2 *Página principal de acceso al recurso*
Fuente: Elaboración propia año 2011

En la ***vista 6.2*** muestro la página principal para acceso a las aplicaciones; en la cual los campos serán validados con los usuarios de la base de datos para acceder al sitio.

Una vez validado y siendo un usuario registrado se da acceso a los recursos para poder almacenar imágenes, dándonos la bienvenida y mostrando listado de las imágenes que tenemos, ahora ya podemos subir, borrar o descargar imágenes, como se muestra en la ***vista 6.3.***



Vista 6.3 *Página de Bienvenida*
Fuente: Elaboración propia año 2011

En este módulo apreciamos las bondades que nos proporciona Netbeans con Struts 2 donde aquí, lo que se utiliza son ***taglib*** que contienen etiquetas especiales que nos permiten mostrar las datos de una forma más agradable e interactiva, nos muestra más a fondo cómo es que se configuran las JSP para dirigirse a una página según la acción, si esta tiene éxito o nó y que dichas configuraciones al mismo tiempo nos dan más seguridad ya que muchas veces cuando se quiere tener acceso a determinada página algunos sólo escriben la dirección URL en que esta contenida la información y sin registrarse aun cuando son páginas que se necesita utilizar usuario y contraseña, para evitar esto en java se configura para que cuando se quiera acceder a cierta página, deben ocurrir unas acciones para realizar otras evitando así que se pueda tener acceso directo a una página.

CAPíTULO 7

JAVA EMPRESARIAL JAVA BEANS

7.1 Java Server Faces (JSF)

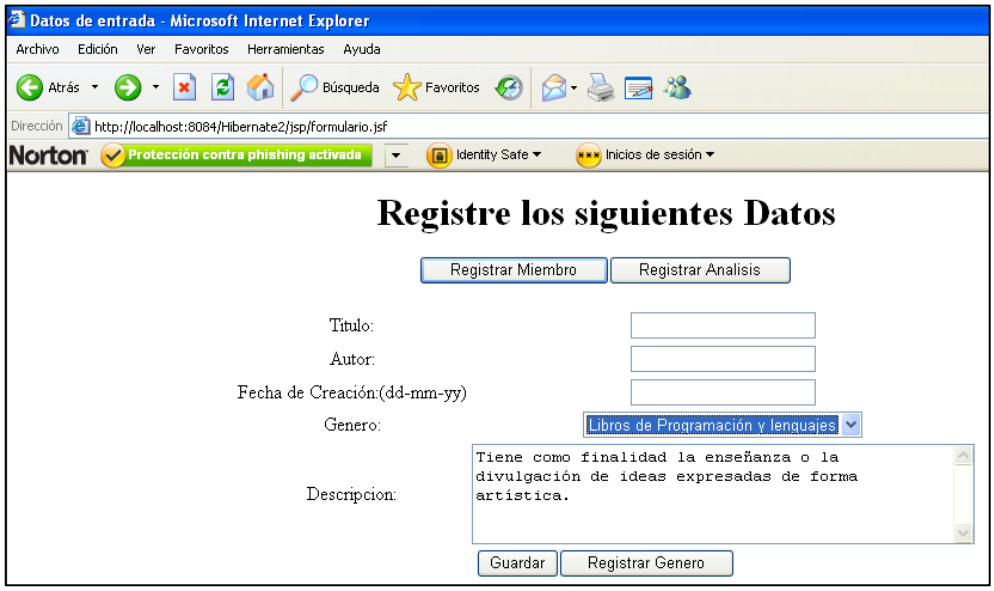
Esta cuenta con los siguientes elementos como API's para controlar componentes gráficos, controlar eventos, validar entradas de datos, definir la navegación en distintas páginas de la aplicación, da soporte a la internacionalización, tiene interfaces accesibles, contiene una librería personalizada para expresar una interfaz JSF en una página JSP.

JSF nos da una facilidad de empleo, hay una separación de la lógica de la aplicación con la presentación, tiene una facilidad en conectar la capa de la presentación con el código de la aplicación, nos permite que cada desarrollador dedique una parte en el proceso de desarrollo y que posteriormente se unan las piezas.

JSF utiliza componentes tales como JavaBeans que tienen métodos y eventos que están organizados dentro de la vista, éste es un árbol de componentes normalmente mostrados como una página; también utiliza el componente de interfaz ***renderer*** que muestra los ***componetes*** de interfaz del usuario y traduce la entrada de usuario a valores de los componentes, el componente ***validator*** asegura que el valor introducido por el usuario sea aceptable, ***backing beans*** son JavaBeans que coleccionan los valores de los componentes gráficos e implementan métodos que capturan eventos, el componente ***converter*** convierte el valor de una cadena para mostrar un solo componente, los eventos y captura de eventos son generados por la interfaz del usuario, los mensajes son información que se muestra en cualquier parte de la aplicación y pueden ser para mostrar errores y el componente ***navigation*** que nos sirve para movernos de una página a otra.

En JSF se puede tener un pool de conexiones (agrupamiento de conexiones) que es una colección de conexiones abiertas a una base de datos para realizar múltiples consultas. Esto nos permite ahorrar tiempo en la conexión a la base de datos ya que lo que hace el pool es que las deja listas para su uso y no que tenga que iniciar una conexión cada vez que un usuario se conecte.

Un ejemplo que desarrollamos en el módulo fue el de la creación de una página web en la cual nos permitiría poder registrar un análisis, el libro y el género y registrarse como miembro en una base de datos desarrollada en Mysql, donde las tablas correspondían a libro, miembro, análisis y género como se muestra en la ***vista 7.1***.



Vista 7.1 *Página de registro de un libro.*
Fuente: 4to. Diplomado Desarrollo en Java Master (2011)

Vista 7.2 *Página de registro de análisis.*
Fuente: 4to. Diplomado Desarrollo en Java Master (2011)

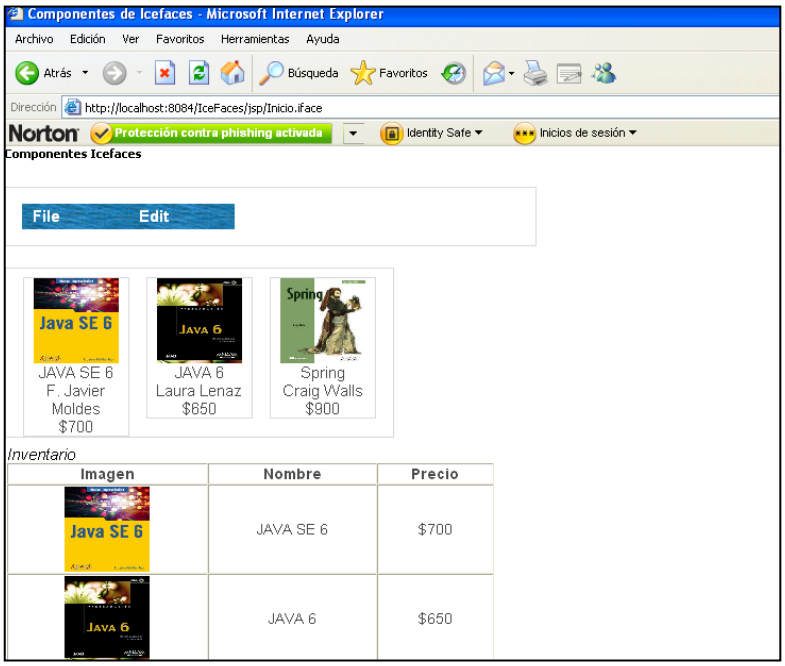
Al acceder a registrar análisis nos direcciona al formulario donde podíamos registrar el análisis de un libro que ya está registrado en la base de datos como se muestra en la ***vista 7.2***.

Si decidimos registrarnos como miembro se nos pedirán los datos personales, como son: nombre dirección, estado, código postal; si decidimos registrar un nuevo género se nos pedirá el nombre y su descripción, como nos hemos dado cuenta esta aplicación es aplicable a una librería en la que podemos dar nuestros puntos de vista sobre un libro y si tenemos preferencias por otras tendencias literarias podemos mostrarlos a los demás para que los conozcan a la vez que podemos pasar a ser miembros de una asociación en la que la lectura es lo principal.

7.1.1 ICEFACES

Es un framework para construir aplicaciones con Ajax (***Asyncronous Javascript And Xml***) que es una técnica de desarrollo web para diseñar aplicaciones interactivas tipo RIA (***Rich Internet Aplication***), donde ICEfaces aísla completamente al desarrollador de Ajax. No se necesitan etiquetas especiales. ICEfaces nos permite poder tomar sólo algunos de los datos y procesarlos, manteniendo los restantes en segundo plano lo que nos permite una vista mas agradable y una mejor interactividad.

Ajax es una técnica usada en diversos sistemas operativos y navegadores web debido a que está basada en estándares abiertos como JavaScript y ***Document Objet Model***(DOM) es aquí donde se pueden manipular los objetos; es una interfaz de programación de aplicaciones(API), donde podemos cambiar y añadir dinámicamente el contenido en el documento. Mostraré una imagen de una aplicación realizada en el diplomado construída con ICEfaces que es la siguiente:



Vista 7.3 Aplicación construida con ICEfaces.
Fuente: 4to. Diplomado Desarrollo en Java Master (2011)

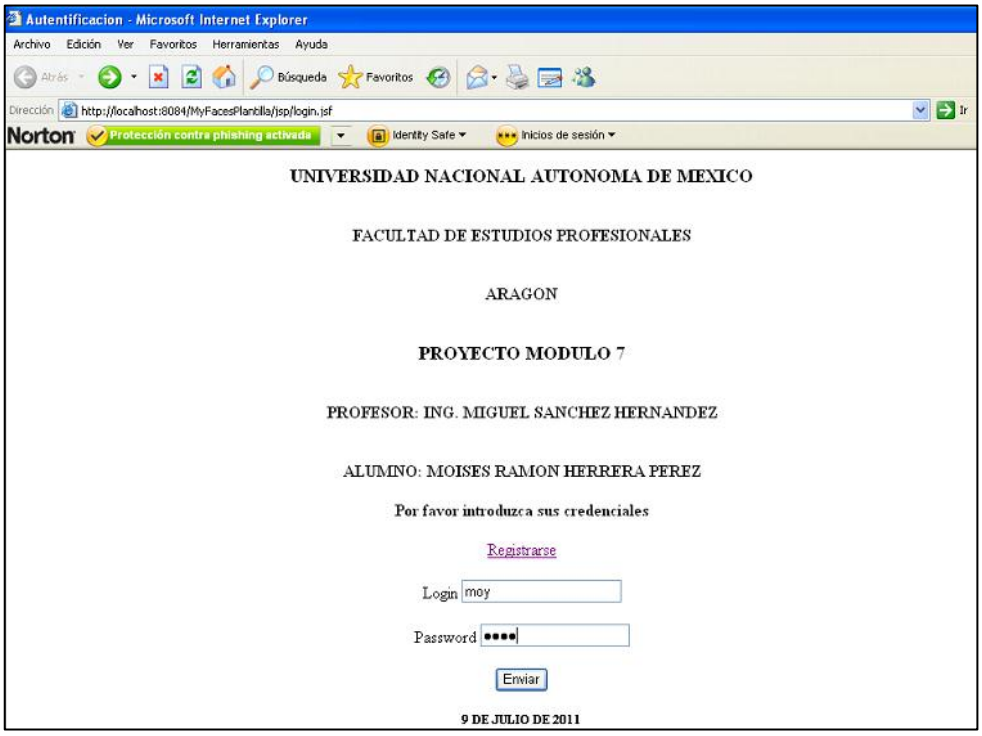
Esta aplicación es la simulación de un carrito de compras en internet que tiene tres libros y si deseamos adquirir uno de ellos nosotros podemos arrastrar la imagen hacia la tabla que los contiene y automáticamente se añade a nuestra lista de compras mostrándonos el nombre y el costo del libro como se muestra en la ***vista 7.3***.

ICEfaces desarrolla su programación hacia las vista del usuario donde se busca que su entorno sea agradable y de fácil interacción.

7.2 Producto obtenido en el módulo

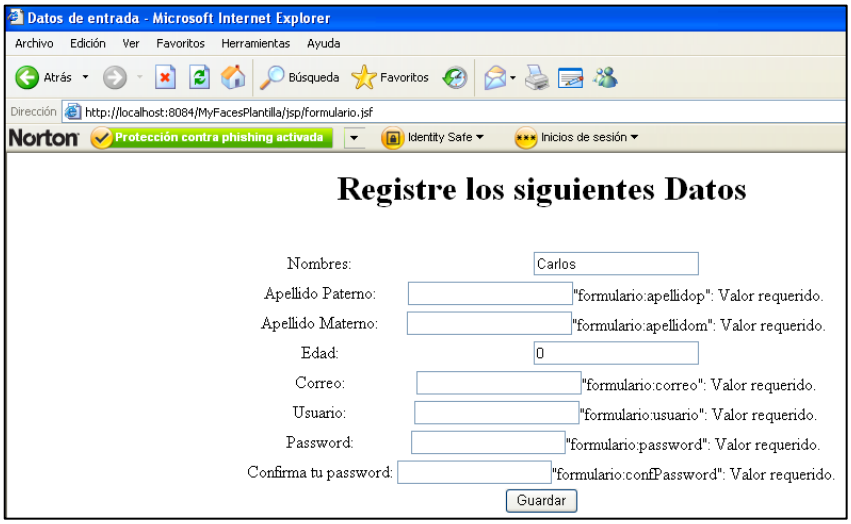
En este módulo se desarrollo un proyecto similar al del módulo 6 solo que ahora utilizando ***Java Server Faces*** (JSF), en la cual se creó una base de datos en ***Mysql*** con las tablas de archivos y usuario, en la que se deberían de poder subir a la web archivos con extensión jpg, dónde deberían registrarse para poder ser usuarios de esa aplicación y en ese momento se creará una carpeta que se le asignara al darse de alta en la base de datos, dónde al ya ser usuario y entrar a la página web deberían de mostrarse sus archivos y tener la posibilidad de subir mas archivos, de eliminarlos y poder descargarlos.

En la ***vista 7.4*** se muestra la página principal para poder acceder a sus recursos que brinda mediante un registro en el que debe crear un usuario y una contraseña.



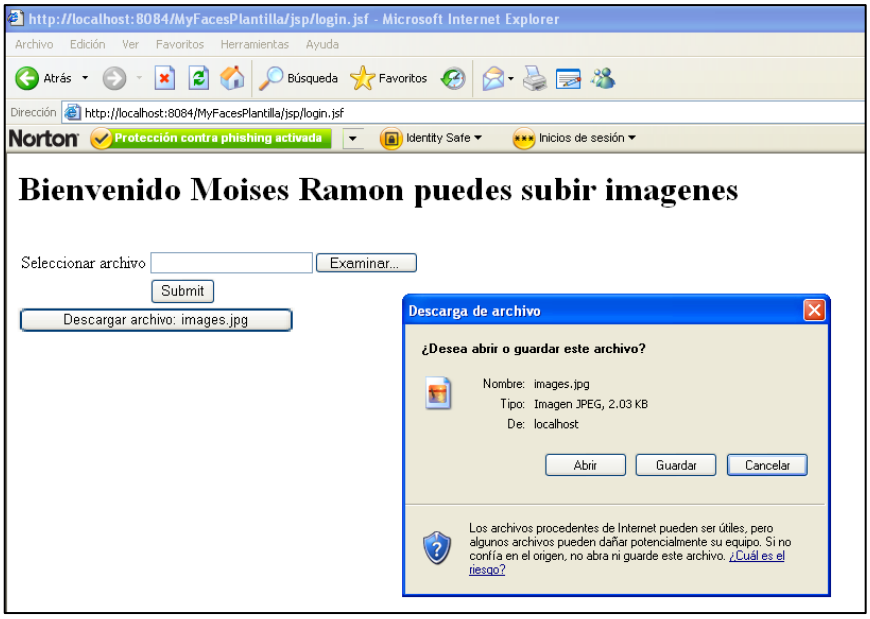
Vista 7.4 *Página de acceso al recurso*
Fuente: Elaboración propia año: 2011

En este registro, si el formulario no es llenado correctamente este los validará y mandará un mensaje como lo muestra en la ***vista 7.5*** este, si es correcto lo registrará en la base de datos y creará una carpeta donde podrá almacenar sus archivos de imágenes jpg.



Vista 7.5 *Validación de los datos del formulario de registro*
Fuente: Elaboración propia año 2011

En la ***vista 7.6*** se muestra que una vez subido un archivo éste esta disponible para poder bajarlo de la web, también si el archivo a subir no tiene la extensión permitida mandará un mensaje donde nos avisa que el formato no es valido.



Vista 7.6 *Página de Bienvenida al recurso prestado*
Fuente: Elaboración propia año 2011

CONCLUSIONES

Que se realizaron aplicaciones que se usan en la vida real y existiendo una interacción con una base de datos que complementa nuestros conocimientos y mejora la proyección de las aplicaciones.

Que la información que hay en una empresa al ser controlada puede tener diversos usos, ya que está es algo muy valioso que se debe proteger y también saber recopilarla.

Que al analizar el porqué de las cosas en la programación podemos realmente crear aplicaciones y no sólo transcribir los programas.

Que el lenguaje de programación java nos permite crear aplicaciones diversas con ambientes gráficos donde la interacción es agradable y entendible y que su mayor potencia está en el desarrollo web donde cada vez evoluciona para facilitar más la programación y separa la lógica del negocio de las vistas de interacción del usuario.

Que la finalidad de la enseñanza es que todo lo aprendido se pueda aplicar a cualquier tipo de empresa para mejorar su desempeño, su productividad, así como servicios.

Que con los conocimientos obtenidos me complementa como profesional ya que con los conocimientos de hardware que cuento ahora integro a ellos la otra parte que es la del software y así se me abren muchas posibilidades de desarrollo ya que el diplomado fue muy completo y puedo combinarlo.

Ahora cuento con la posibilidad de poder montar un servidor de bases de datos con soporte en lenguaje java y poder diseñar páginas web y ofrecer un servicio a las necesidades de las empresas, otra posibilidad es crear aplicaciones que no tengan la necesidad de utilizar el servidor que se puedan ofrecer a pequeños negocios diseñándolas a sus necesidades y orientándola con los objetivos de la empresa para que tengan su propia personalidad.

Que el área de trabajo que he considerado son escuelas, pequeñas empresas, negocios que aun no cuentan con esta plataforma para llevar sus negocios a la web, lo cual hace que sus negocios no sean vistos por muchos usuarios que necesitan sus servicios.

Que el servicio que puedo ofrecer a las empresas también es dar el soporte y mantenimiento de su base de datos, brindando este servicio desde la oficina donde esté instalado y no necesariamente estar en el sitio de la empresa.

Que en el futuro pretendo montar un servidor donde pueda ofrecer estos servicios y tener clientes no solo de donde resido sino también en los estados y crecer cada día mas brindando mejores servicios y compitiendo con otros servidores.

Que este informe muestra como es necesaria la planeación y la investigación para crear un buen sistema de información; y que ya existen herramientas que nos permiten hacer un seguimiento de un proyecto ya iniciado, el cual al retomarlo podemos lograr obtener el producto final para el que fue diseñado en un principio.

Que en la parte de la creación de aplicaciones con MDI tiene una gran potencia ya que no sólo podemos tener aplicaciones para una red local, si no también para conectarse mediante la red global al hacer conexión con un servidor remoto.

Que para tener éxito en la programación para empresas es necesario conocer muy bien la lógica del negocio y las necesidades del usuario, ya que cada empresa tiene sus objetivos a seguir y la forma de hacer los negocios y no debemos imponer nuestra forma de trabajar si no al contrario fusionarnos a ella, para esto se necesita un profundo conocimiento de todas sus actividades y formas de desempeño en cada puesto.

GLOSARIO

Capítulo 1

UML	Modelado de Lenguaje Unificado
GANTT	Diagrama Creado por Henry Laurence Gantt
CMMI	Modelo de capacidad y madurez
ERP	Sistema de Gestión empresarial
DIOCESIS	Jurisdicción eclesiástica
DECANATO	Agrupación de varias parroquias
VICARIA	jurisdicción de un vicario
VICARIO	Persona que ejerce funciones de otra en todo, en parte por delegación

Capítulo 2

API	Interfaz de programación de aplicaciones
APPLET	Aplicación que se ejecuta en el contexto de otro programa

Capítulo 3

IDE	Entorno de desarrollo Integrado
JDBC	Java conectividad de base de datos
DBMS	Sistema de administración de base de datos
GUI	Interface grafica de usuario
MDI	Interfaz de Múltiples documentos

Capítulo 4

SQL	Lenguaje de consulta Estructurado
DDL	Lenguaje de definición de datos
DML	Lenguaje de Manipulación de datos

DAO	Acceso de datos de objeto

Capítulo 5

JSP	Java Server Pages

Capítulo 6

MVC	Modelo Vista controlador
JAR	Archivo Java
OGNL	Lenguaje de Navegación de Objetos gráficos

Capítulo 7

JSF	Java Server Faces
AJAX	Javascript Asincrono y Xml
RIA	Aplicación que enriquece a Internet
DOM	Modelo Documento Objeto

REFERENCIAS

Capítulo 1

Pedro Gabriel, Ramírez Hernández. (2010).Diplomado Java Master. Administración de proyectos.

s/a http://vicaria8.arquidiocesismexico.org.mx consultado en junio a Julio de 2010. Disponible

Capítulo 2

Pedro Gabriel, Ramírez Hernández. (2010) Lenguaje de Programación.

Capítulo3

Sánchez Hernández, Miguel Ángel. (2010). Programación Avanzada con Java.

Capítulo 4

Hernández Cabrera, Jesús. (2010). J2EE Conectividad Empresarial.

Capítulo 5

Hernández Cabrera, Jesús. (2011). J2EE Desarrollo Web.

Capítulo 6

Sánchez Hernández, Miguel Ángel. (2011). Struts 2.

Capítulo 7

Sánchez Hernández, Miguel Ángel. (2011). Java Server Faces.

s/a http://es.wikipedia.org/wiki/Document_Object_Model consultado en Julio 2011. Disponible