UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“SISTEMA WEB PARA CONTROL DE VENTAS”

TRABAJO ESCRITO

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL, QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

RAFAEL SANTIAGO SALGADO

ASESOR:

ING. RODOLFO VÁZQUEZ MORALES

MÉXICO, 2006

A mis padres, Jerónimo y Jovita,
que les debo la vida misma.

A mi esposa Analí,
por darme dicha y felicidad.

A mis hermanos,
por su apoyo incondicional.

A mis amigos y compañeros
que siempre estuvieron cuando los necesité.

A mis profesores,
que me asesoraron para ser posible el presente.

Introducción

Con el paso del tiempo las tecnologías avanzan a pasos vertiginosos, con ello los ingenieros en Computación requieren actualizarse constantemente. El estudio de nuevas técnicas para la arquitectura y desarrollo de aplicaciones se vuelve un requerimiento indispensable para todo desarrollador de sistemas.

El Diplomado de Desarrollo de Aplicaciones en Web constituyó un curso adecuado de actualización y capacitación al alumno ya que se presentaron temas de actualidad y practicidad. Otorgado en su mayoría por docentes con experiencia en el ramo profesional, enriqueciendo los temas con sus experiencias y demostrando dominio total de la información expuesta.

El presente documento muestra las temáticas expuestas durante el Diplomado de Desarrollo de Aplicaciones en Web, y presenta la documentación sobre la solución de un sistema de software de un problema real. Se aplican las técnicas y conocimientos adquiridos durante el curso.

Los contenidos del presente, son mostrados resumidamente para proporcionar información correspondiente a los temas de interés.

El capítulo I presenta información expuesta por los docentes especializados en cada área; en primera instancia, los fundamentos para el desarrollo orientado al Web proporcionan las bases para conocer el funcionamiento de Internet. El análisis y diseño con UML son un punto de partida para analizar y modelar la aplicación a resolver. Se presenta una introducción al lenguaje Java que es la tecnología usada para soportar la aplicación. El módulo capa de usuario exhibe recomendaciones para el buen diseño del front-end. El módulo de la capa de datos describe las técnicas para el desarrollo y creación de la base de datos. El último módulo, Reglas de negocio, explica las reglas que deben seguir las aplicaciones para modelar el comportamiento del negocio.

El capítulo 2 muestra el análisis de la problemática a resolver, usando técnicas de UML, se exponen algunos diagramas para modelar la aplicación.

El capítulo 3 presenta el diseño de la aplicación, revisa el funcionamiento del

patrón Model View Controller y algunas técnicas recomendadas para diseñar un sitio Web.

Finalmente el Capítulo 4 muestra las secciones referentes a la implantación del sistema y puesta en producción del mismo.

CAPÍTULO 1

DIPLOMADO DE SISTEMAS WEB

CAPÍTULO 1. DIPLOMADO DE SISTEMAS WEB

La necesidad de actualización para poder ofrecer soluciones adecuadas y efectivas, aprovechando el potencial de la tecnología informática, me motivó a cursar el Diplomado de Desarrollo de Sistemas Web, el cual se impartió en la Facultad de Estudios Superiores Aragón de la Universidad Nacional Autónoma de México. Se dividió en 6 módulos, los cuales presento para conocer el alcance y objetivos.

1.1 Módulo I. Fundamentos para el desarrollo orientado al WEB

Los objetivos del primer módulo fueron conocer el ambiente sobre el cual una aplicación orientada al Web presta sus servicios. Para ello fue necesario revisar algunos conceptos básicos como:

- **Internet:** para el desarrollo de una aplicación orientada al Web, debe tenerse en claro la infraestructura y la tecnología existente para poderlas explotar adecuadamente, y obtener así el mayor provecho obteniendo como resultado una aplicación funcional.

  Internet es la red más grande del mundo; se encuentra constituida por la interconexión de centenares de miles de redes heterogéneas. Las cuales se comunican mediante el protocolo TCP/IP.

- **Protocolo TCP/IP:** un protocolo es un conjunto de reglas o instrucciones que se definen para cumplir un objetivo. Internet utiliza varios protocolos, entre los que destacan el Protocolo TCP (*Transport Control Protocol* o Protocolo de Control de Transporte), y el IP (*Internet Protocol* o Protocolo de Internet). Se trata de una serie de reglas para transportar los datos descompuestos en paquetes de una computadora a otra, asegurándose de que todos los paquetes lleguen a su destino volviéndose a ensamblar y formar nuevamente los datos.

- **Servicios de Internet:** sobre la infraestructura de transporte de datos que proporciona TCP/IP se han construido otros protocolos más específicos que ofrecen servicios como los siguientes:

1. World Wide Web: es el sistema de información más completo y actual; se compone tanto de elementos multimedia como de hipertexto. Gracias al hipertexto, desde una página Web se puede acceder a cualquier otra página Web almacenada en un servidor HTTP[1] situado en cualquier parte del mundo.
2. Correo electrónico: el *e-mail* permite mantener correspondencia con otros usuarios en cualquier parte del mundo, es similar al correo convencional, pero tiene la ventaja que es más rápido y sencillo de usar con respecto al correo tradicional; permite enviar y recibir mensajes de texto o multimedia y permite añadir archivos al correo.
3. El servicio de transferencia de archivos (FTP)**:** el FTP[2] un servidor conectado a Internet que permite la transferencia de archivos.

Los proyectos que se desarrollan orientados a Internet, usan tecnología cliente servidor y están constituidos por páginas web (dinámicas y estáticas) que mostrarán al cliente la información necesaria para completar una solicitud. Las páginas web serán almacenadas en un servidor capaz de procesar contenido dinámico (contenedor web). Para poder comunicarse con el servidor Web y permitir alojar la aplicación, se puede usar el programa SSH; mismo que permite transportar la estructura de directorios de la aplicación Web a dicho servidor.

1.1.1 Introducción al desarrollo de tres capas

Se mostraron los conceptos sobre la tecnología de desarrollo de una solución de software para poder aplicarlos posteriormente durante el diseño del sistema.

Se observó que, actualmente, la tecnología del diseño de aplicaciones no sólo basadas en el Web suelen desarrollarse en capas.

"Una capa es un concepto abstracto que define un grupo de tecnologías que proporcionan uno o más servicios a sus clientes"[3]. En la arquitectura de tres capas, los servicios suelen dividirse en tres componentes: la capa de

[1] *Hipertext Transfer Protocol* ó Protocolo de Transferencia de Hipertexto.
[2] File Transfer Protocol, o Protocolo de Transferencia de Archivos.
[3] Keogh Jim.

presentación, la capa de reglas de negocio y la capa de datos.

- **Capa de presentación:** está compuesta por la interfaz que interactúa con el usuario. Pide datos de entrada y convierte su respuesta en peticiones que se reenvían a un componente que procesa la petición y devuelve el resultado al programa cliente. La capa de presentación también se conoce como servicio de usuario, front-end, cliente, aplicación, o aplicación cliente.
- **Capa de reglas de negocio:** o capa de negocio, se encuentran las reglas y lógica de procedimientos necesarios para realizar las operaciones del sistema, esto es, las actividades operacionales, controles de consistencia, validaciones, cálculos, etc., es la responsable de recibir la entrada del nivel de presentación, interactuar con los servicios de datos para ejecutar las operaciones de negocios para los que la aplicación fue diseñada a automatizar (por ejemplo, la preparación de impuestos por ingresos, el procesamiento de ordenes etc.), enviar el resultado procesado al nivel de presentación.
- **Capa de datos:** contiene los componentes necesarios para comunicarse con el Sistema de Gestión de Base de Datos. Estos componentes operan los métodos necesarios para poder almacenar, recuperar y manipular la información guardada en la base de datos. Los servicios que proporciona ésta capa son: almacenar, recuperar, mantener y dar integridad a los datos.

El diagrama representativo al desarrollo en tres capas es el siguiente:

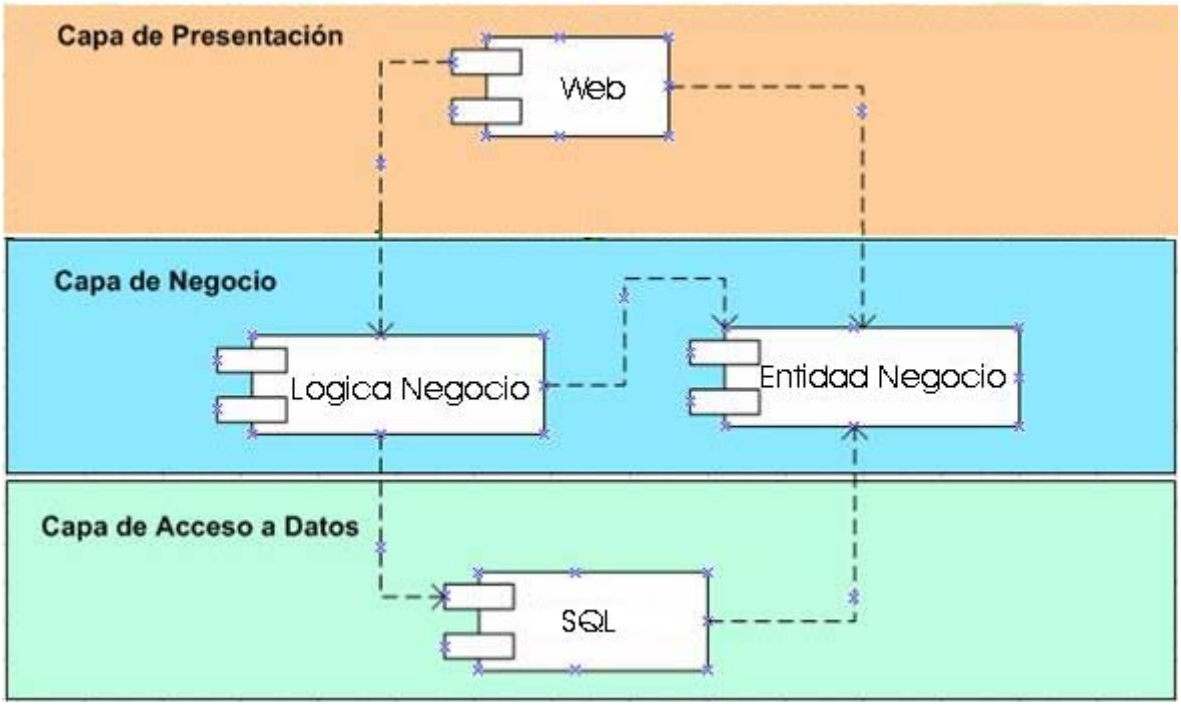


Figura No. 1 Diagrama de componentes en tres capas

El desarrollo en tres capas presenta ventajas significativas; se explota al máximo la funcionalidad de cada capa, puesto que tienen tareas específicas a desempeñar. Otra ventaja es la facilidad y rapidez para poder realizar una modificación de código. De tal manera, la aplicación a desarrollar se verá beneficiada de la tecnología en tres capas: La capa de presentación estará constituida en esencia de formularios html, información e imágenes adecuadas, pretendiendo una navegación sencilla al usuario. En la capa de reglas de negocio se pretenderá realizar todas las validaciones y cálculos de manera exhaustiva para que la lógica corresponda a la forma de operación del negocio, permitiendo de esta manera que la información entregada a la capa de datos sea confiable y sólida. En la última capa, la de datos, se pretende tratar a los datos en una forma concisa mediante el uso del lenguaje SQL, permitiendo aumentar la portabilidad del sistema.

1.1.2 Elementos de Software para un proyecto de desarrollo en WWW

La opción de Software planteada en el diplomado para diseñar, desarrollar e implantar una aplicación orientada al Web fue la siguiente.

- **Java[4]:** La solución de una aplicación basada en el Web requiere de un kit de desarrollo para llevar a cabo la programación y obtener uno o varios archivos ejecutables. Existen varios lenguajes de *script* para el desarrollo de una aplicación orientada al web, entre los mas populares se encuentran los siguientes: php, perl, asp, java, etc.

  Java es un lenguaje potente y robusto, además de ser muy popular, ha demostrado ser una gran opción para desarrollar un sistema de software.

  Java es una plataforma de software constituida de cuatro partes: El lenguaje de programación, la máquina virtual de Java (que permite la portabilidad en ejecución), el API Java (Una biblioteca estándar para el lenguaje) y El entorno de ejecución (JRE).

- **Servidor de Aplicaciones:** es aquel software capaz de resolver las solicitudes WEB del cliente, y para alojar una solución de software se auxilia por lo regular de uno o más lenguajes de programación y de un sistema gestor de base de datos. Un ejemplo de servidor Web es Apache Tomcat (Jakarta Tomcat) que funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta de Apache Software Foundation. Tomcat implementa las especificaciones de los servlets[5] y de JavaServer Pages (JSP[6]) de Sun Microsystems. Tomcat funciona con cualquier servidor web con soporte para servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets del Tomcat a menudo se presenta en combinación con el servidor Web Apache.

- **Base de datos:** MySQL es un servidor de bases de datos multiusuario, concretamente, rápido en entornos Web. Soporta el lenguaje de bases de datos estandarizado SQL. MySQL es una implementación cliente/servidor que permite administrar la concurrencia de accesos en forma óptima. Sus principales virtudes son: gran velocidad, robustez y facilidad de uso. MySQL

[4] *http://Java.sun.com*
[5] *Un servlet es un programa Java que provee la funcionalidad de generar dinámicamente contenidos Web.*
[6] *Los JSP son código java embebido dentro de plantillas HTML.*

soporta lenguajes de programación como: C, C++, Eiffel, Java, Perl, PHP, Python y TCL.

Una aplicación basada en la Web requiere que esté alojada en una computadora configurada como servidor, mismo que debe cumplir los requerimientos para la funcionalidad que ofrece el sistema; entre ellos se debe tener instalado el kit de desarrollo Java, se requiere también el servidor de aplicaciones Tomcat y el software gestor de base de datos MySQL.

1.2 Módulo II. Análisis y diseño con UML

El objetivo de éste módulo fue mostrar las herramientas que se usan frecuentemente durante el proceso de análisis de requerimientos y utilizarlas para realizar el diseño o modelo de un sistema de software orientado a objetos.

1.2.1 Proceso de análisis de Requerimientos

Para llevar a cabo el análisis del sistema, debe tenerse pleno conocimiento del problema que se va a resolver, lo cual permite establecer los requerimientos básicos y especificaciones formales del sistema.

Para el análisis del problema se recomienda usar los siguientes modelos:

- **Modelo de casos:** permiten establecer las principales funcionalidades que contendrá el sistema. Entendiéndose por transacción a cualquier interacción que el sistema tendrá con los agentes externos a él. Cada transacción que se modela recibe el nombre de Caso. Cada caso requiere que se especifique el nombre y la secuencia de pasos necesarios para poder llevarla a cabo. La secuencia de pasos debe describirse para un Caso ideal, es decir, para una transacción que no contemple situaciones de error. Se le denomina Actor a cada agente externo del sistema que interactúe directamente con él, pudiendo ser una persona u otro sistema.
- **Modelo de Interfaz:** Consta principalmente de la definición del conjunto de interfaces principales con las que el usuario final interactuará. Dichas interfaces participarán en la ejecución de un caso de uso en el sistema. Por interfaz se refiere a pantallas, reportes o llamadas a otros sistemas.

- **Modelo del Dominio del Problema:** Su objetivo es identificar los objetos de información y las relaciones que guardan entre sí. Para cada modelo de Casos se obtiene un diagrama de clases.

Se debe comprender perfectamente las necesidades de la problemática a resolver, esto se hace generalmente mediante entrevistas y encuestas a una o varias personas involucradas y que forman parte de la organización. Esto permitirá descubrir las transacciones que habrán de modelarse obteniendo de esta manera un caso por cada transacción (Modelo de Casos). Durante las entrevistas se deberá registrar la información obtenida, la cual deberá revisarse para obtener las clases de los objetos del Modelo del Dominio, los sustantivos que ayudan a la revisión por ejemplo: proveedor, pedido, factura, cliente, etc. Es necesario obtener la opinión del cliente para el diseño de la interfaz, pues mientras más amigable sea ésta, mayor aceptación tendrá por parte de los nuevos usuarios, debe trabajarse en conjunto con el desarrollador comunicándole las ideas tanto del cliente como las del analista.

1.2.2 UML

El Lenguaje de Modelado Unificado, prescribe una serie de notaciones y diagramas estándar para modelar sistemas orientados a objetos. UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. UML ofrece nueve diagramas que se usan para modelar sistemas:

- Diagramas de Casos de Uso para modelar los procesos
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos
- Diagramas de Colaboración para modelar interacciones entre objetos
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema

- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema
- Diagramas de Componentes para modelar componentes
- Diagramas de Implementación para modelar la distribución del sistema.

A continuación se describen los modelos más usados durante el análisis orientado a objetos:

1. Modelo de Casos de Uso: El diagrama de casos de uso proporciona el punto de entrada para analizar los requisitos del sistema, y el problema que se desea solucionar. Cada caso de uso se modela para cada proceso que el sistema debe llevar a cabo. Los procesos se describen dentro del caso de uso por una descripción textual o una secuencia de pasos ejecutados que cumplen de forma ideal el objetivo del sistema. Las extensiones son casos alternos en los cuales describe un comportamiento opcional del sistema. El modelo de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan con el sistema. El diagrama de un actor corresponde a un “muñeco” de palo. Los casos de uso representan el comportamiento del sistema y los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. El diagrama de un caso de uso corresponde a una elipse. La interacción entre un caso de uso y un actor se representa mediante una flecha. El objetivo es construir un diagrama de casos de uso para cada tipo de escenario diferente en el sistema.

Ejemplo de caso de uso para retirar efectivo de un Cajero Automático.

Caso de Uso: Realizar retiro de efectivo

Actores: Cliente, Cajero

Descripción del Caso de Uso:

1. Cliente introduce la tarjeta
2. Cajero solicita autentificación
3. Cliente se identifica
4. Cajero muestra servicios
5. Cliente solicita realizar una operación de retiro de efectivo por una cantidad específica.
6. El cajero le proporciona el efectivo solicitado

7. El cliente termina sesión, retira su tarjeta y se va.

Extensiones:

3. Identificación incorrecta

3a. El cajero muestra el error y vuelve a pedir autentificación

5. El cajero no puede proporcionar la cantidad solicitada

5a. El cajero muestra motivo del error, y vuelve a pedir la cantidad a retirar

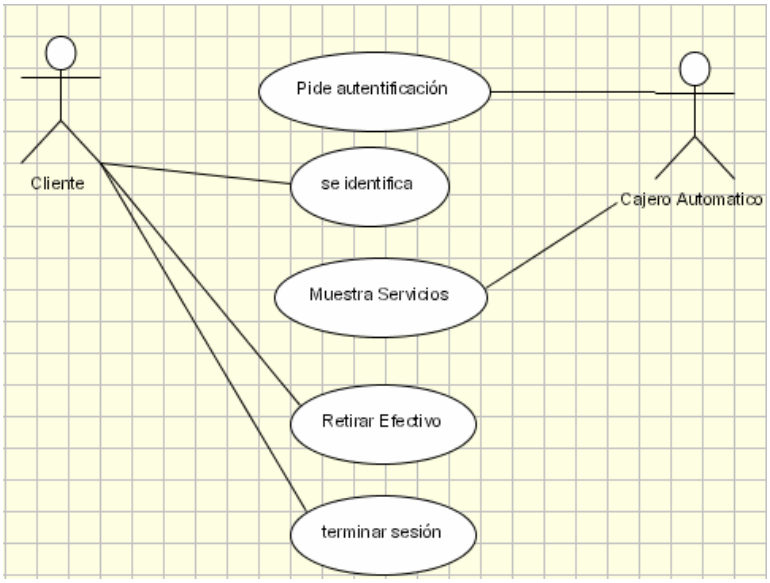


Figura 2. Diagrama de Caso de Uso

2. Diagrama de secuencia: muestra un patrón de interacción entre objetos en un sistema. Se modela para cada caso de uso. El diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y sus mensajes pasados entre objetos. Continuando con el escenario anterior, se muestra un ejemplo:

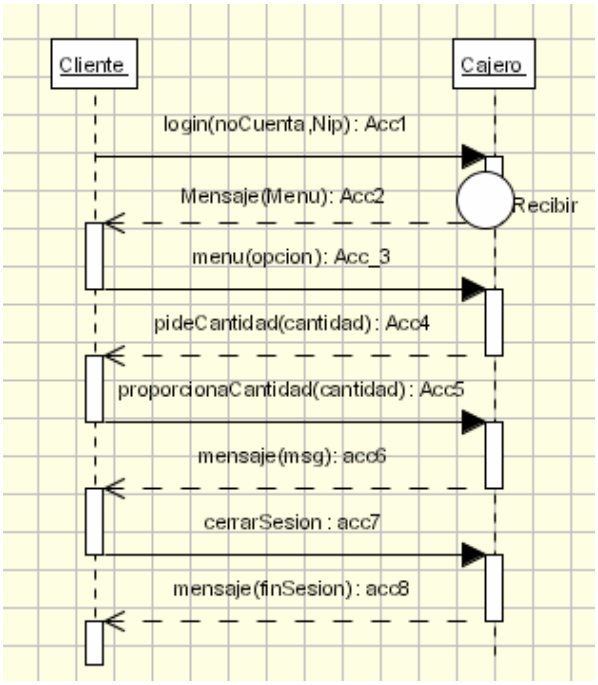


Figura 3. Diagrama de secuencia

3. Diagrama de Colaboración: se centra en estudiar todos los efectos de un objeto dado durante un escenario. Los objetos se conectan por medio de enlaces, cada enlace representa una instancia de una asociación entre las clases implicadas. El enlace muestra los mensajes enviados entre los objetos, el tipo de mensaje (sincrónico, asincrónico, simple, blanking y time-out) y la visibilidad de un objeto con respecto a los otros. Siguiendo con el ejemplo del cajero automático, éste sería su Diagrama de Colaboración.

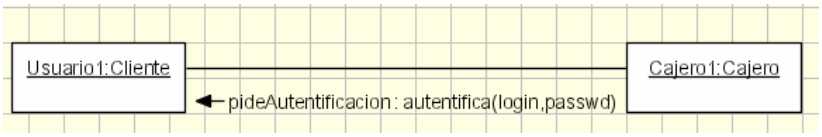


Figura 4. Diagrama de Colaboración

En el diagrama anterior muestra dos instancias de objetos, uno que pertenece a la clase Cliente cuyo nombre es Usuario1 y la otra instancia se llama Cajero1 que pertenece al objeto Cajero, muestra también el mensaje que se pasa a través de ellos, mediante un método llamado autentifica que pide dos parámetros para poder "instanciar" al Usuario1.

4 Diagrama de Estados: El diagrama de estados modela la secuencia de estado

que un objeto de la clase atraviesa durante su vida en respuesta a los estímulos recibidos junto con sus propias respuestas y acciones. Por ejemplo, un comportamiento de un objeto se modela en términos de en qué estado se encuentra inicialmente y a que estado cambia cuando recibe un evento en particular. También modela qué acciones realiza un objeto en un estado concreto. Este sería el diagrama de estados del ejemplo del cajero.

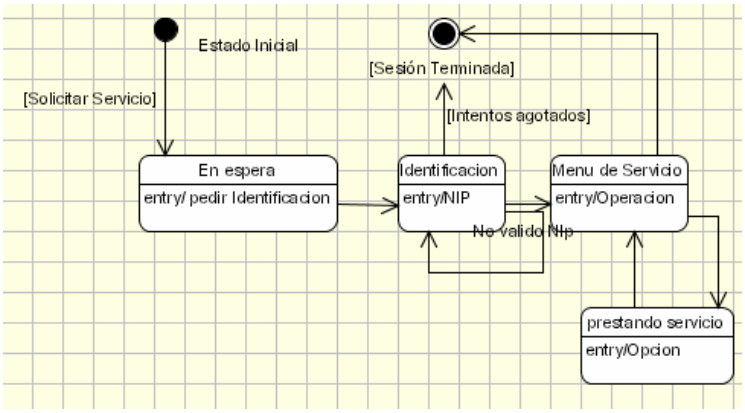


Figura 5. Diagrama de estados

Una ventaja importante de UML es que su notación estándar permite a diseñadores diferentes modelando sistemas diferentes pueden sobradamente entender cada uno de los diseños de los otros, de allí la importancia de su uso.

1.3 Módulo III. Introducción a la programación orientada a objetos con Java

La importancia de éste módulo, fué conocer en forma general el funcionamiento y la estructura de la plataforma de software Java, para adquirir las bases de un buen desarrollo de software basado en este lenguaje.

1.3.1 Introducción a Java

Java fué creado en 1991 por *Sun Microsystems* con el objetivo de desarrollar software para comunicación entre aparatos electrónicos de consumo como equipos de música, video juegos, televisores, etc. Java tuvo mayor auge en 1994, las primeras aplicaciones basadas sobre el *world wide web (WWW)* estaban basadas en applets. Java es un lenguaje de programación de alto nivel con el que se pueden escribir tanto programas convencionales como para Internet.

El código fuente se compila generando un archivo con extensión (*class*), este archivo está compuesto de *byte-code*, éste es un código intermedio de muy bajo

nivel, pero sin alcanzar las instrucciones máquina propias de la plataforma.

El byte-code está compuesto por un 80% de la aplicación. Para ejecutarse en cualquier plataforma necesita el *run-time* que es dependiente de la plataforma que se usa, le añade el 20% restante de las instrucciones faltantes para su ejecución.

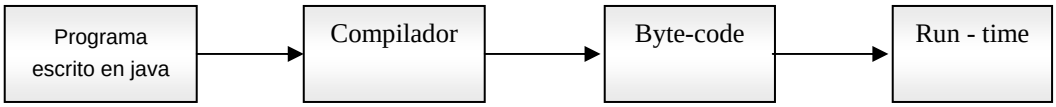


Figura 6. Esquema de generación de un programa Java

El código fuente se compila mediante la instrucción: *javac nombrePrograma.java*, si el programa no contiene errores, se generará un archivo con extensión *class*, en este caso nombrePrograma.class, éste archivo es libre de plataforma, puede llevarse a cualquier otro sistema operativo sin importar la arquitectura del procesador, para poder ejecutarlo se necesita el run-time o la máquina virtual de java que debe estar previamente instalado en el sistema. Para ejecutarlo se realiza mediante la instrucción: *java nombrePrograma*.

Java es un lenguaje de programación que se ejecuta en cualquier plataforma de hardware, una plataforma es un ambiente de software y hardware sobre la cual se ejecuta un programa. La plataforma de Java tiene dos componentes:

- Máquina Virtual de Java (JVM): es el programa alojado en el sistema e interpreta los códigos de bits.
- La API: es una gran colección de componentes de software listos, proveen funcionalidad variada.

1.3.2 Tipos de Aplicaciones que se desarrollan en Java

Con java se pueden construir diferentes tipos de aplicaciones:

- **Aplicaciones *StandAlone*:** son programas básicos de java, se ejecutan localmente con la ayuda del Java Runtime Environment (JRE).
- **Applets**: son mini aplicaciones de java, son programas que deben incluirse en páginas Web para ser observadas por otra aplicación.
- **Servlets:** son programas de servidor, diseñados para reemplazar la programación CGI. Un servlet recibe los datos y los procesa en el servidor, la

salida más habitual es una página Web.

- **Java Server Pages (JSP)**: es un programa servidor cuyo diseño y funcionalidad es similar a la de un servlet, la diferencia radica en la forma que se escriben. Los JSP se escribe con código HTML entrelazado.
- **Java Beans:** es un componente que permite agrupar funcionalidades para formar parte de una aplicación. Requiere ser integrado con otros componentes para que éste sea funcional.
- **Enterprise Java Bean (EJB):** es un componente que permite agrupar funcionalidades para formar parte de una aplicación. Un EJB requiere de un ambiente de ejecución, requiere del *EJB Container* que es parte de una aplicación de servidor.
- **Java Server Faces (JSF):** especificación aportada por Sun Microsystems como JSR 127, provee del lado del servidor un framework de componente de interfaz de usuario para generar aplicaciones web con una rica interfaz de usuario. Esta basado el en Modelo – Vista – Controlador que ofrece una clara separación entre la lógica de negocio y la presentación.

Java es una excelente opción para el desarrollo de aplicaciones, ya que es multiplataforma, es decir, puede ejecutarse en cualquier sistema operativo como Windows, Solaris, Silicon Graphics, Unix, OS2. etc., además es portable y dinámico. Java ofrece una buena diversidad para programar diferentes tipos de aplicaciones dependiendo de la problemática que se tenga. Cuenta con la posibilidad de desarrollo de aplicaciones cliente-servidor así como para aplicaciones de uso local. Para soluciones de software basadas en el Web se usan más frecuentemente los servlets, las jsp's y los java beans; se usan para implementar la capa de negocios y la de datos, en algunas ocasiones las jsp's dotan de dinamismo a la capa de usuario. Los applets de igual forma se implementan sobre el Web solo que éstas no están dotadas de la tecnología cliente-servidor.

Sin duda cada programador tiene su estilo para programar una aplicación, si en el desarrollo de un proyecto cada desarrollador se encarga de resolver un módulo, resultaría problemático la integración de la solución del proyecto o que otro

desarrollador tenga que modificar un módulo de otro. Se recomienda seguir las recomendaciones que se encuentran en la documentación de java, algunas de ellas son las siguientes:

- Utilizar nombres significativos para los identificadores y respetar la siguiente notación de mayúsculas y minúsculas:
    - Las clases: Clase, MiClase
    - Las interfases: Interfaz, MiInterfaz
    - Los métodos: metodo(), miMetodo()
    - Métodos de acceso: getAtributo(), setAtributo()
    - Las variables: variable1, miVariable
    - Las constantes: CONSTANTE, MI_CONSTANTE
    - Los paquetes: java.paquete.subpaquete
- Algunas otras recomendaciones son:
    - Incluir comentarios describiendo la funcionalidad
    - Uso adecuado de interlineado, para mejorar legibilidad
    - Escribir una operación por línea
    - No usar líneas de más de 80 caracteres

Nota: Para mayor información ver la documentación de Java

El desarrollo de una solución de software se realiza mediante un lenguaje de programación, Java interviene en el desarrollo de sistemas dotando de una gran funcionalidad y robustez a la solución. Con Java la implementación de una clase se toma como plantilla y son la base para crear objetos. Un objeto o agrupamiento forma parte de un componente, a su vez el agrupamiento de componentes forman sistemas.

Java se integra perfectamente al ambiente Web, mediante los Servlets, las JSPs, los Java Beans, los Enterprice Java Beans y los Struts. No debe olvidarse que el diseño de una aplicación en java tiene que contemplarse la reutilización de código mediante un correcto análisis y adaptación de las clases para formar objetos.

1.4 Módulo IV. Capa de usuario

La capa de usuario también es conocida como interfaz, mediante ésta, el usuario de acuerdo a los niveles de seguridad, interactúa con la base de datos y el sistema. Para el desarrollo de un proyecto basado en el Web, la interfaz se presenta como un conjunto de documentos en formato HTML, por lo que se requiere un navegador.

Las recomendaciones que se revisaron para el diseño de la interfaz son:

- Identificar los elementos con que se constituirá o formará (texto, imágenes, videos, ligas, etc.)
- Definir el aspecto visual y contenido informativo
- Definir los recursos de interacción o de comunicación del usuario con el sistema
- Definir la estructura de la información y la organización de contenidos

1.4.1 El diseño

El diseño de la interfaz se basa en los procesos y tareas de los usuarios, se debe ofrecer únicamente lo necesario para que el usuario trabaje. Por lo que se deben contemplar los siguientes puntos a la hora de proporcionar acceso a los usuarios:

- Consulta básica
- Consulta total
- Consulta y operaciones básicas o de área
- Realización de operaciones totales
- Consulta, operaciones y generación de reporte de área
- Acceso total

Una herramienta que ayuda a identificar los procesos y operaciones que conforman la interfaz es el mapa de navegación, mediante el cual se puede verificar el nivel de acceso permitido al usuario. Sirve también para ayudar a desarrollar las ventanas con las que el usuario interaccionará suponiendo la navegación deseada. El mapa de navegación es un diagrama a bloques que representa los procesos por los cuales navega un usuario.

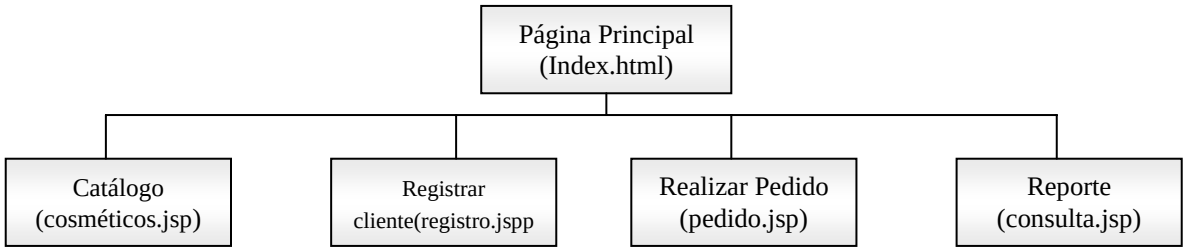


Figura 7. Mapa de navegación

Al diseñar la interfaz de usuario se deben seguir los siguientes pasos:

- Utilizar un estándar o normatividad de la organización.
- Bosquejar un diseño, el cual fuerza a que se piense a cerca de los elementos de las pantallas y el orden que éstos tendrán.
- Implementar un prototipo y presentarlo a los usuarios y determinar la congruencia con los requerimientos.

Durante la construcción de la interfaz se recomienda que los atributos de la página se realice usando hojas de estilos en cascada (CSS), pues con ésta técnica de formato de páginas Web, las modificaciones de color y tamaño se aplicarán a todos los elementos marcados con CSS.

El diseño debe realizarse pensando en el usuario final, pues éste es el que determine si usa o no el sistema, por lo que se recomienda considerar los siguientes puntos:

- Poseer un diseño amigable, lo cual se refiere a lo vistoso y la forma fácil de usarse.
- Lógica en los procesos; el sistema debe estar firmemente afianzado a los procesos establecidos en el análisis, esto determina la correcta funcionalidad
- Basarse en un formato estándar para que todas las pantallas tengan armonía visual
- Implementar la ayuda del sistema

El objetivo es construir una jerarquía de menús y páginas que parezcan naturales y bien estructuradas al usuario. Por lo general la estructura la proporcionan los procesos. La estructura de las páginas debe ser clara, funcional y con una continuidad gráfica entre los diferentes componentes y subsecciones del sistema.

HTML (*HyperText Markup Language*) es un lenguaje de marcas muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con *enlaces* (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas. La presentación final del hipertexto se realiza en un navegador como Internet Explorer, Netscape. Moisaic, etc.

Aunado al potencial de HTML se puede integrar de manera simple, diversas herramientas para cubrir todas las necesidades, un ejemplo de ello es JavaScript, el cual tienen gran semejanza en cuanto a la sintaxis de Java, pero no poseen la potencia de éste último, van incrustados en los documentos HMTL y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones,

mostrar mensajes, crear animaciones, comprobar campos etc.

1.5 Módulo V. Capa de datos

El objetivo de éste módulo fue conocer las bases necesarias para poder modelar y diseñar el esquema de la base de datos, además de explotar la información almacenada en la base de datos de la mejor manera posible.

1.5.1 SGBD

El Sistema Gestor de Base de Datos es un software encargado de la gestión de la base de datos. Una base de datos puede definirse como un conjunto integrado de datos que modelan un universo dado, por lo general son datos del mundo real. Es susceptible de ser explotada, contienen datos históricos que se usan para tomar decisiones.

Para lograr lo anterior es necesario realizar el modelado de datos, que es un mapa conceptual para implementar de manera lógica una base de datos en un SGBD. El mapa contiene el esquema y estado de la base de datos. El esquema de la base de datos contiene la estructura inicial de la base de datos, guarda la definición de tablas, dominios de campos, constraints, etc.

La arquitectura de un SGBD consta de tres niveles:

- **Nivel físico:** está constituido del repositorio de información.
- **Nivel conceptual:** es el nivel central, constituido del motor del lenguaje de consulta estructurado (SQL).
- **Nivel de vistas:** una vista es una tabla virtual compuesta principalmente de comandos SQL, no guardan información, solo es una estructura similar al de una tabla.

1.5.2. SQL

Es el lenguaje de consulta estructurado (SQL *Structured Query Language*) utilizado para interactuar con una base de datos del tipo relacional, esta herramienta permite organizar, gestionar y recuperar los datos almacenados en una base de datos.

SQL esta constituido del lenguaje de definición de datos (DDL) y el lenguaje de

modelos de datos (DML).

El DDL permite la definición del esquema de la base de datos, la creación, modificación y eliminación de tablas, creación de disparadores (triggers), especifica la definición de tipos de datos, etc.

El DML permite conocer el estado que se encuentra una base de datos; permitiendo seleccionar, insertar, actualizar y borrar datos.

Algunos comandos DDL son:

Comando	Descripción	Sintaxis
Create Database	Crea una nueva base de datos	Create database *nombreBD*
Create Table	Crea una tabla	Create table *nombreTabla*
Drop Database	Elimina una base de datos	Drop database *nombreBD*
Alter Table	Altera la estructura de una tabla Añade o elimina columnas, llaves primarias, etc.	Alter table *nombreTabla* add || drop [Constraint] || [tipo de dato columna en add] [Primary Key (*nombreCampo*)]
Drop table	Elimina un tabla de la base de datos	Drop table *nombreTabla*

Algunos comandos DML son:

SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

Las cláusulas que más se usan son:

AND	Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Devuelve el valor contrario de la expresión.

Los operadores de comparación son los siguientes:

< , >, <>, < =, > =, =	Menor que, Mayor que, Distinto de, Menor ó Igual que, Mayor ó Igual que, Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

Las funciones de agregación se usan dentro de una cláusula select en grupos de registros para devolver un único valor que se aplica a un grupo de registros son:

AVG	Se utiliza para calcular el promedio de los valores de un campo determinado
COUNT	Se utiliza para devolver el número de registros de la selección
SUM	Se utiliza para devolver la suma de todos los valores de un campo determinado
MAX	Se utiliza para devolver el valor más alto de un campo especificado
MIN	Se utiliza para devolver el valor más bajo de un campo especificado

1.5.3. API JDBC

El API (*Applications Program Interface*) de JDBC (*Java Data Base Connectivity*) es un conjunto de librerías Java estándar que permite la conectividad a diferentes bases de datos. JDBC es un controlador normalmente proporcionado por el fabricante de un SGBD que es capaz de poder establecer una conexión con una base de datos o tener acceso a cualquier fuente de datos tabular, de ejecutar sentencias SQL y procesar los resultados de las consultas.

Los cuatro tipos de drivers JDBC son:

- **Tipo 1: Puente JDBC-ODBC.** Java se conecta con el ODBC vía métodos nativos. Consiste en crear una fuente de datos vía ODBC en el sistema Windows del cliente para poder acceder a la base de datos.
- **Tipo 2: Driver mixto Java – API Nativo.** Las llamadas JDBC son convertidas a llamadas nativas del propio DBMS. Se requiere que algún tipo de código binario deba ser cargado en la máquina del cliente que generalmente es desarrollado por el fabricante del DBMS.
- **Tipo 3: Driver para Middleware de Bases de Datos.** En este driver se traducen las llamadas JDBC a un protocolo de software de tipo middleware y después es traducido al protocolo del DBMS.
- **Tipo 4: Direct-to-Database Pure Java Driver.** En este driver se traducen directamente las llamadas JDBC al protocolo de red utilizado por el DBMS, permite una llamada cliente-servidor por lo que representa una solución básica al acceso por intranet.

1.6 Módulo VI. Reglas de negocio

En este módulo se presentaron las diferentes reglas de negocio que deben

considerarse al diseñar e implementar una aplicación, así como mostrar el funcionamiento del framework struts para su uso en el desarrollo de aplicaciones.

El desarrollo de una aplicación se realiza principalmente para mejorar un sistema existente o para resolver un problema del mundo real, dicho problema debe ser susceptible de modelarlo en computadora y tiene que adaptarse a las restricciones. A las restricciones o reglas que debe seguir el modelo se les conoce como *business rules* o **reglas de negocio**.

1.6.1 Tipos de reglas de negocio

Existen diferentes tipos de reglas de negocio, las que se revisaron son:

- **Reglas del modelo de datos:** controlan que la información básica almacenada para cada atributo o propiedad de una entidad ú objeto sea válida. Normalmente corresponden al dominio que le pertenece a cada atributo de una entidad. Por ejemplo: el atributo fecha de cierta entidad, el dominio que le corresponde para los meses es de 1 a 12, pero debe controlarse el dominio para día, ya que no corresponde un valor 30 para un valor 2 (febrero) de mes.
- **Reglas de relación:** controlan las relaciones entre los datos, por ejemplo, que todo pedido debe ser realizado por un cliente y que el mismo debe estar dado de alta en el sistema.
- **Reglas de derivación:** son el conjunto de reglas que especifican y controlan la obtención de información que se puede calcular a partir de la ya existente. Por ejemplo: el importe de un artículo se deriva del precio de dicho artículo multiplicado por la cantidad de ejemplares más IVA.
- **Reglas de restricción:** son aquellas que restringen el valor de los atributos o propiedades de una entidad mas allá de las restricciones básicas que sobre las mismas existen. Por ejemplo: una regla que indique que el saldo nunca puede ser menor que cierta cantidad tope establecida para cierto tipo de clientes.
- **Reglas de flujo:** son aquellas que determinan y limitan cómo fluye la información a través de un sistema e indican que camino recorre la información y obligan a que se sigan solo los caminos válidos. Por ejemplo: al desear

obtener el inventario físico de un almacén, se tiene que realizar el conteo de cada artículo para posteriormente entregar el resultado, a partir de éste se podrá realizar la captura de la información y de aquí realizar un reporte de los artículos existentes en dicho almacén.

El concepto de capas en un sistema cliente/servidor, es el modelo clásico de dos capas, el cliente y el servidor, éste esta ubicado normalmente en otra máquina como gestor de base de datos relacionales, siendo mejores los que proporcionan soporte para implementar en ellos las reglas de negocio, mediante el uso de llaves primarias, integridad referencial, triggers, stored procedures, etc.

Si en la base de datos se crea una regla de integridad referencial que indica que todo pedido pertenece a un cliente cualquier aplicación que acceda a esta base de datos se beneficiara de esta y otras validaciones automáticamente sin una línea de código. Utilizando una base de datos menos potente, casi todas las reglas de negocio deberán implementarse dentro de los programas que accedan a la base de datos.

Las reglas de flujo son bastante difíciles de implementar dentro de la base de datos, y suelen ser las aplicaciones cliente las que controlan que la información sigue una ruta válida a través del sistema. Pero si la información del negocio se encuentra en distintas bases de datos, gestionadas por distintos gestores se deberá implementar la validación en cada aplicación cliente así que el desarrollador será el encargado de obligar a que se cumplan algunas reglas de negocio; la solución será crear una nueva capa dentro del sistema cliente/servidor llamada capa intermedia o middle-tier creando una arquitectura cliente/servidor en tres capas (three-tier).

1.6.2 Arquitectura de aplicaciones Web con java

La arquitectura de aplicaciones Java que corren sobre el Web, es Web Container, se encarga de dar una infraestructura de ejecución para las aplicaciones, tal infraestructura consiste en los siguientes servicios:

1. Soporte a comunicaciones. Permite a los servlets comunicarse con el servidor Web sin necesidad de programación de bajo nivel por parte del programador.
2. Manejo del ciclo de vida de los servlets.
3. Soporte a ejecución multithreading.
4. Seguridad declarativa. Configurable en tiempo de "publicación" (deployment).
5. Soporte a JSP's.

Con Web Container se maneja la ejecución multithreading, la seguridad y el acceso a servicios de red.

1.6.3 Uso del Patrón de Diseño Modelo Vista Controlador (MVC)

El propósito del MVC es separar la lógica del negocio de la presentación intercalando una capa entre ellos para permitir que la capa de negocio pueda ser independiente de la "vista". Las tareas principales del controlador son:

- Examinar y extraer parámetros del objeto request.
- Toma la entrada del usuario del objeto request y define que servicio se requiere de los que publica el modelo.
- Invocar objetos del negocio, publicados en el modelo, haciendo llegar los parámetros extraídos del objeto request.
- Crear el modelo que las vistas desplegarán basadas en los resultados de invocar objetos del negocio.
- Mantener el estado de la sesión donde sea necesario.

El modelo contiene las reglas del negocio y el estado del sistema. Una vez que el controlador completa su proceso y selecciona una vista para generar contenido el modelo contendrá toda la información a desplegar.

La vista despliega la información contenida en el modelo con la responsabilidad de la generación de contenidos. Es la responsable de la presentación, también es responsable de obtener la entrada del usuario y enviarla al Controlador. Debe ser posible reemplazar cualquier vista en cualquier aplicación web con otra vista que

despliegue los mismos datos sin modificar el código del modelo ni del controlador.

Un patrón de desarrollo implica ordenar la forma de trabajo contando con una serie de estándares definidos y aceptados.

Para el uso del modelo de Vista Controlador en un proyecto de Web se puede utilizar un framework, como status.

1.6.4 Struts

El funcionamiento básico de Struts, parte de la forma que generalmente envía datos a la aplicación a través de alguna acción del negocio, siguiendo el siguiente esquema:

Struts puede definir un enlace de hipertexto a través de un ActionForward. Este objeto tiene una propiedad de nombre y otra de Path, lo que permite definir el enlace hacia el Path que debe dirigirse y posteriormente usarla por su nombre. Los objetos ActionForward son definidos en el archivo de configuración XML de struts. La sintaxis para crear un Actionforward para una página de bienvenida sería:

<forward name=”welcome” path=”/Welcome.do”/>

El framework de struts proporciona una clase tipo ActionForm que se designa para manipular la entrada de una forma HTML validada y de ser necesario, desplegarla nuevamente al usuario.

Una clase tipo ActionForm es un JavaBean con un par de métodos para manejar la validación y ciclo de revisión. Struts hace corresponder automáticamente las propiedades del JavaBean con los atributos de la forma HTML.

Para dar a los objetos Action un URI (Uniform Resource Identifier) el framework de struts proporciona un objeto ActionMapping, que usualmente se define dentro del archivo XML de configuración:

```
<action-mappings>
  <action path="/NewAlumno"
```

```
      type="patito.struts.AlumnoAction"
      name="alumnoForm" scope="request"
      validate="true" input="/addAlumno.jsp">
      <forward name="principal" path="/pages/Welcome.jsp"/>
    </action>
  </action-mappings>
```

La operación del FrameStruts se resume de la siguiente manera:

- 1a. El ActionServlet busca en la configuración un ActionForm que corresponde a una solicitud mediante el Action marcado en una forma HTML.
- 1b. El AcionForm permite calidar la entrada del usuario, en su caso reporta errores y permite instanciar un JavaBean que contiene los datos capturados en la forma HTML y posteriormente procesarlos.
- 2a. El ActionServlet busca en la configuración el objeto Action Asociado al ActionForm elegido.
- 2b. Le pasa al objeto Action, el objeto Request, Response y el ActionForm con los datos capturados y validados.
- 2c. El objeto Action resuelve la petición contenida en el Request, invoca los objetos del negocio que sean requeridos departe del modelo.
- 3a. El ActionServlet localiza en la configuración la vista que debe presentar los resultados de la petición al usuario.
- 3b. se invoca la vista definida en la configuración para presentar los resultados de la petición del usuario.

Por último, se revisó el uso de sesiones utilizando la librería Ibatis para autentificar el acceso de los usuarios a la aplicación. El uso de sesiones permite guardar temporalmente la información que genera un usuario al navegar en un sistema Web, el ejemplo más claro es el modelo del carrito de compras, donde se almacena en forma temporal los artículos a comprar y que posteriormente la información podrá ser procesada y almacenada en el servidor.

Para mostrar la capacitación obtenida en el Diplomado, en los siguientes capítulos se presentará un caso práctico, comprendiendo desde el entorno del problema hasta la implementación del Sistema Web.

CAPÍTULO 2

ANÁLISIS

CAPÍTULO 2. ANÁLISIS

La construcción de un sistema de software por lo general nace por la necesidad de automatizar procesos; estos procesos deben ser susceptibles de ser representados en la computadora para poder analizarlos, diseñarlos y simularlos dando por finalizado en su solución. También surgen por la necesidad de actualizar los sistemas existentes o complementarlos.

Existen diferentes conceptos sobre el análisis de sistemas de computación:

“Es un conjunto o disposición de procedimientos o programas relacionados de manera que juntos forman una sola unidad”.

“Un conjunto de hechos, principios y reglas clasificadas y dispuestas de manera ordenada mostrando un plan lógico en la unión de las partes”.

“Un método, plan o procedimiento de clasificación para hacer algo”.

“Es un conjunto o arreglo de elementos para realizar un objetivo predefinido en el procesamiento de la Información”.

De tal forma, se puede adoptar la definición: el análisis consiste en dividir un todo en sus más mínimas partes para poder resolver las pequeñas fracciones de forma independiente y posteriormente integrar las soluciones para mostrar la solución general.

A continuación se planteará el análisis correspondiente a la empresa Apacsa S.A. de C.V., misma que tiene el interés de automatizar sus procesos; entre los cuales la venta de artículos, es la que representa una de las funcionalidades más relevantes. Se estudiará y proporcionará la mejor solución de acuerdo a sus necesidades y requerimientos de información.

2.1 Análisis empresarial

La empresa a ser analizada pertenece al sector productivo, cuya razón social es Apacsa S.A. de C.V. se ha venido desarrollando en el Estado de México**,** fue fundada en el año 1997 y se clasifica como pequeña de acuerdo a la cantidad de sus empleados.

La actividad o giro primordial es la manufactura y venta de artículos de belleza; en particular cosméticos como: maquillaje, sombra, labial, crema, rubor y rimel.

Dicha empresa tiene registrada la marca *golden lady* para la presentación de sus productos. Tiene presencia en el mercado nacional como internacional, ya sea de manera informal o en tiendas de autoservicio y diversas como Just Price.

Apacsa S.A. de C.V. tiene una planta de producción con área de 1000m^2 donde se producen cientos de productos de belleza diariamente.
La empresa cuenta con las siguientes secciones:

- Recepción de materia prima
- Almacenaje de materia prima
- Sala para fabricación
- Almacenaje de productos acabados
- Oficinas
- Vestuarios

La premisa primordial de Apacsa es tener presencia en la mayoría de la población, introduciendo productos de buena calidad a un precio mucho menor que el de sus competidores, a través de nuevos esquemas de ventas; ofrecer una mayor calidad de servicio.

2.2 Situación Actual

Durante los procesos de producción, la empresa tiene programado un stock de 40,000 unidades por modelo, 150,000 pzas. son enviadas al centro de distribución localizado en la ciudad de México. Es este lugar donde diversos distribuidores viajan desde provincia para poder realizar una compra.

La empresa carece de un programa de publicidad por lo que el cliente al llegar al lugar de distribución compra lo que ve a la vista, desconociendo en algunas ocasiones la variedad de artículos.

El cliente adquiere la cantidad de producto que puede trasladar en su viaje; algunas veces en transporte público o propio, lo cual limita la cantidad de compra a la capacidad del transporte.

El Centro de Distribución (CD) realiza un inventario y reporte de ventas al final del día, el reporte es enviado a las oficinas de la empresa por correo electrónico y confirmado mediante una llamada telefónica, de esta manera permite determinar la cantidad de artículos a ser surtidos. Existen errores humanos al consolidar la información aunque la frecuencia de ocurrencia es mínima.

El siguiente diagrama muestra ilustrativamente la operación de venta descrita anteriormente:

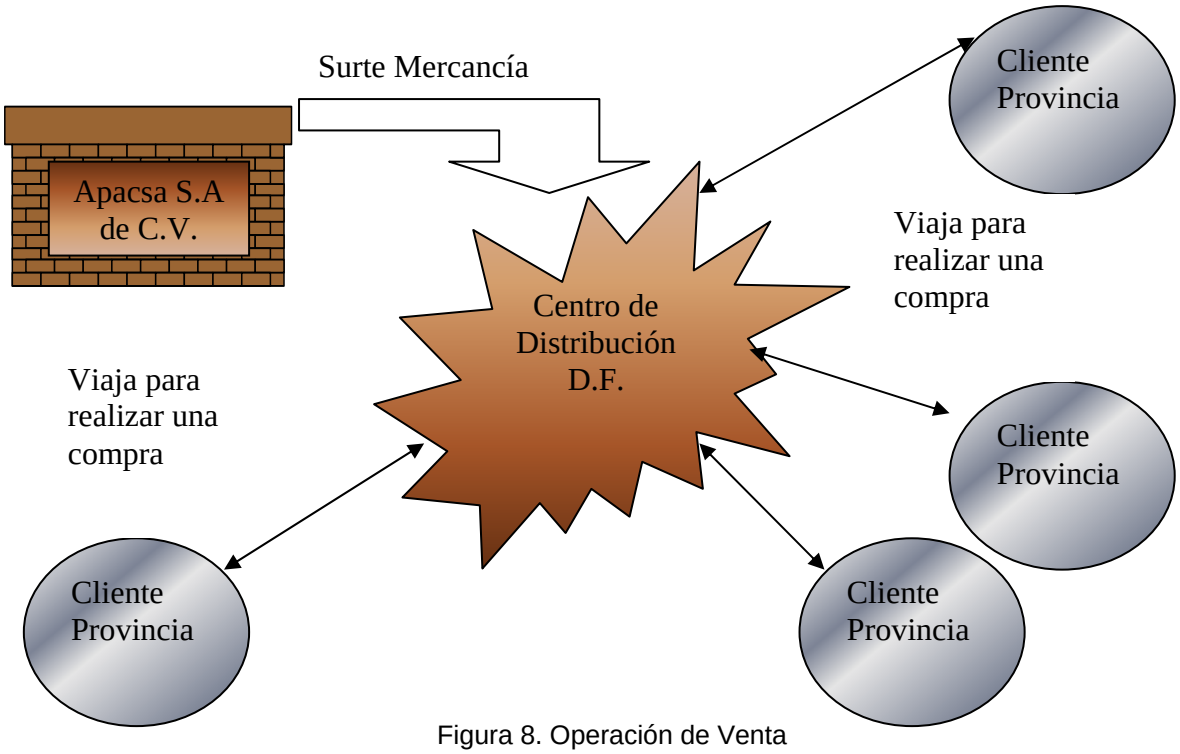


Figura 8. Operación de Venta

En algunos casos las ventas se hacen mediante una llamada telefónica a prospectos de clientes, cuando a éste le interesa hacer una compra; tiene que trasladarse hasta el lugar donde se oferta el producto para realizar un contrato de compra/venta, esto representa una pérdida tiempo/costo de transportación por parte del cliente y pérdida de tiempo/oportunidad por parte de la que oferta para cerrar un contrato.

Actualmente, la empresa cuenta con el paquete comercial ASPEL-PROD para controlar la línea de producción diaria. Y productos como ASPEL-COI que permite procesar y mantener actualizada la información contable y fiscal, ASPEL-NOI es

usado para automatizar el control de percepciones y deducciones de los trabajadores y ASPEL-SAE para controlar el ciclo de todas las operaciones de compra-venta de la empresa.

De tal forma se tienen las siguientes oportunidades de solución:

- Se desconoce la cantidad de producto en stock hasta que se realiza un inventario.
- No se tiene el catálogo de productos ilustrado.
- La variedad de productos se ofrece en un solo lugar físico, disminuyendo las capacidades de venta.
- La información no esta actualizada ya que requiere ser enviada para ser consolidada.
- Existen errores humanos al controlar la información en forma mecánica.
- No se cuenta con un software especializado para el control de las ventas.

2.3 Propuesta de solución

El objetivo de la presente sección es apoyar en la definición de la solución integral para el control de las ventas y administración de Apacsa S.A. de C.V.

2.3.1 Antecedentes

Apacsa S.A. de C.V. tiene la visión de optimizar el proceso de venta a una solución tecnológica robusta, que eficientice sus actividades de comercio aumentando un mejor control en sus tareas administrativas.

2.3.2 Especificación de requerimientos

El proyecto ha ser diseñado debe ser capaz de ofrecer la variedad de artículos con los que dispone la empresa, permitiendo seleccionar cualquier elemento para desplegar información particular, debe ser capaz de registrar un cliente nuevo e iniciar sesión, teniendo la información del cliente en forma persistente y disponible para poderla modificar en cualquier instante, el sistema deberá estar soportado sobre el Web y con la capacidad de generar pedidos de mercancía, notificando la cantidad que el cliente deberá pagar.

Toda información generada por el cliente, deberá ser gestionada por el usuario de la compañía a través del sistema. Por lo que el sistema deberá ser capaz de

registrar altas de usuarios, clientes, registro de artículos, consulta del saldo por cliente, consulta del catálogo de artículos, consulta de existencias y determinar el estado de la mercancía que tiene un pedido. Registrar una orden de compra o pedido.

2.3.3 Especificación de funcionalidades

Observadas las necesidades de dicha empresa, se observan dos funcionalidades primordiales:

a) El lado del cliente que permitirá registrar una orden de pedido.

b) La perspectiva del usuario que permitirá dar seguimiento a una orden de pedido, gestionando toda la información recibida por parte del cliente.

El producto a diseñar deberá cumplir con las siguientes operaciones:

a) Perspectiva del cliente:

- Consultar el catalogo de artículos.
- Consultar las particularidades de cada producto.
- Registrar/modificar los datos de un cliente.
- Acumular los productos en un carrito de compras virtual, obteniendo la cantidad de cada producto a comprar, así como el precio del producto acumulado por categorías y el total de la compra.
- Registrar la orden de pedido.
- Iniciar sesión como cliente, después de haberse registrado.
- Registrar la información correspondiente al pago.

b) Perspectiva del usuario:

- Iniciar sesión como usuario del sistema.
- Realizar mantenimiento (altas, bajas, cambios y eliminación) de categorías.
- Realizar mantenimiento de artículos.
- Realizar mantenimiento de usuarios.
- Realizar mantenimiento de clientes.
- Permiso personalizado de operaciones a usuarios.
- Reporte de Productos.
- Reporte de clientes.

- Reporte de pedidos de acuerdo al estatus (Pendiente, Pagado, En Ruta, Entregado).
- Mantenimiento al Stock.

2.3.4 Características

Las características que se esperan con la solución de software se resumen a continuación:

- Cumplirá con la tarea de publicidad al tener el catálogo en línea.
- Será capaz de registrar una solicitud de pedido.
- La aplicación Web permitirá un control de artículos en stock, notificando existencias.
- La aplicación Web permitirá determinar los pedidos pendientes a surtir.
- El sistema permitirá consultar y conocer el estatus de un pedido.
- La solución permitirá la administración y mantenimiento de la base de datos.

2.3.5 Alcance

La propuesta de solución se consolida en el diseño de una aplicación Web con la funcionalidad primordial de informar las ventas realizadas para la distribución de los productos a sus respectivos clientes.

Los aspectos generales involucrados con el desarrollo del sistema Web son:

- Registrar el pedido de un producto a través de una interfaz Web.
- Registrar clientes.
- Reportar el catálogo de productos.
- Reportar el estatus de los pedidos.
- Controlar el acceso a la aplicación mediante un esquema de permisos.
- Obtener reportes en formato de hipertexto.
- Administración y gestión de información a través de un menú personalizado de acuerdo al nivel de acceso del usuario.

Restricciones y aspectos no pertenecientes al desarrollo del sistema son:

- Sólo se consideran 3 niveles de usuario autentificando a nivel de la base de datos.
- No se instalará algún otro sistema de seguridad como firewalls.

- Los reportes no se podrán exportar a otro formato como *.doc o *.xls ya que la interfaz salida estará constituida de páginas web.
-
- No se hace análisis extensivo de los datos.

2.3.6 Paquete entregable

- Aplicación web: “Sistema Web para Control de Venta”.
- Manual de Usuario.
- Manual de Instalación

2.3.7 Riesgos

La solución se encontrará basada en software de distribución libre, por lo que existe la probabilidad que en un futuro no haya soporte. La solución de software estará respaldada por información técnica para futuro crecimiento.

2.3.8 Beneficios

Los beneficios esperados se listan a continuación:

- Se eficientiza el proceso de compra/venta al reducir los errores humanos.
- La información siempre está actualizada al tener el sistema en línea.
- Se tiene un mayor control en Stock.
- El sistema de venta en línea permite que el negocio pueda ofrecer 24 horas al día los 365 días del año, teniendo costos operativos relativamente bajos.
- El enfoque de venta esta abierta a un mayor mercado.
- El sistema de venta en línea aumenta la comodidad del cliente al poder realizar pedidos desde cualquier lugar del mundo.
- El cliente no se expone a asaltos, logrando su seguridad.
- La aplicación de venta en línea permitirá saber cuándo debe corregir un problema de planificación sobre la venta de productos.
- Mayor control de productos vendidos.
- Información fresca y veraz.
- El coste de los productos vendidos a través de Internet es generalmente bastante menor que los precios que se pueden encontrar en el mercado, ya que una tienda electrónica es libre de costes que una tienda real debe soportar (alquiler, electricidad, agua, etc) y generalmente requiere menos personal.

- El desarrollo de un sistema basado en software libre reduce de forma dramática los costos por uso licencias con respecto al software propietario.

2.3.9 Desventajas

Las desventajas significativas que se consideraron son:

- Habitualmente la seguridad y fiabilidad son difíciles de asegurar en Internet.
- La red se enfrenta a serios problemas de circulación y falta de suficiencia en el transporte de datos debido al rápido crecimiento de usuarios conectados.
- No todos los clientes tienen acceso a Internet.
- Mucha gente se resiste a cambiar y no están acostumbrados a las transacciones impersonales sin la existencia de documentos.

2.4 Casos de Uso

En el capítulo 1 se ha mencionado que los casos de uso modelan los procesos, aplicando ésta metodología al análisis de la empresa se ha obtenido el siguiente diagrama de casos de uso de gestión, y representa la forma en como los Actores interactúan con el sistema en desarrollo.

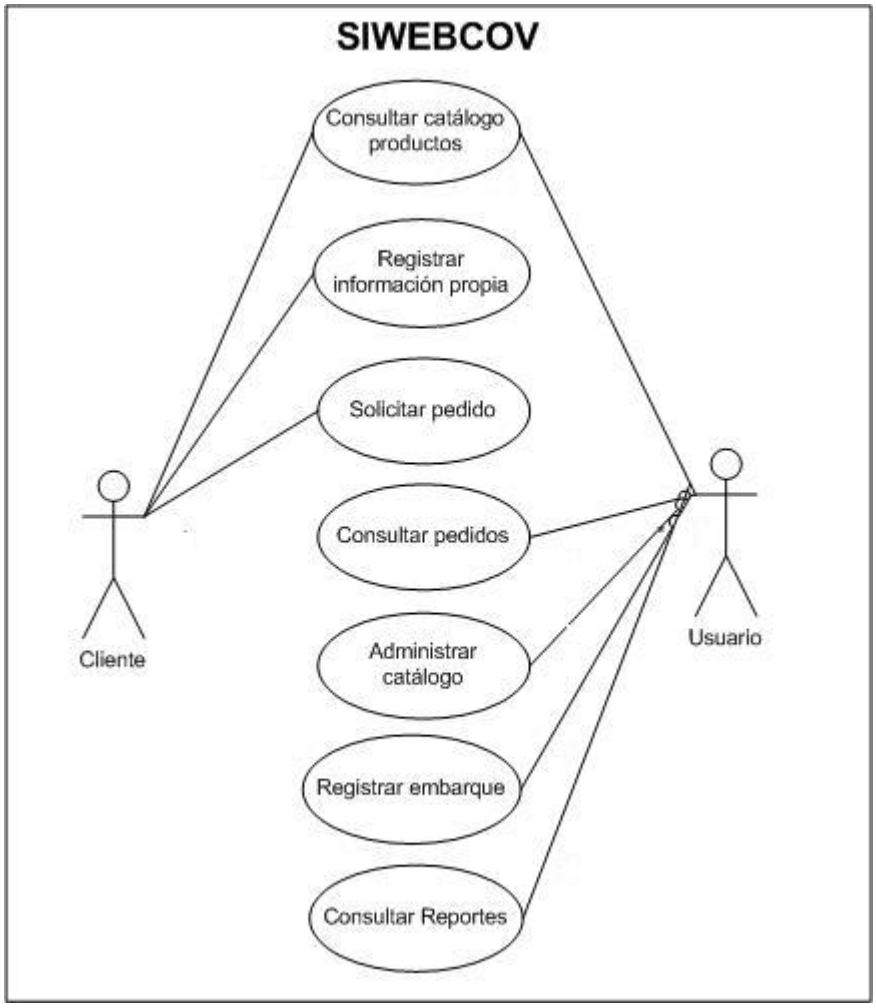


Figura 9. Casos de uso de gestión

A continuación se presentan los casos de uso para mostrar el rol que cada actor juega con respecto al sistema.

Los actores que participan en el sistema se muestran en la siguiente tabla:

Actor	Descripción
Cliente	Persona interesada en adquirir un artículo
SIWEBCOV	Sistema Web para Control de Venta
Usuario	Personal de la empresa que gestiona pedidos

La descripción de los casos de uso se presenta a continuación.

Caso de uso:	Consulta del catálogo de artículos
Actores:	Usuario y cliente
Propósito	Consultar la variedad de artículos
Tipo:	Primario
Descripción:	SIWEBCOV muestra el catalogo de artículos disponibles al cliente o usuario.

El diagrama correspondiente es el siguiente:

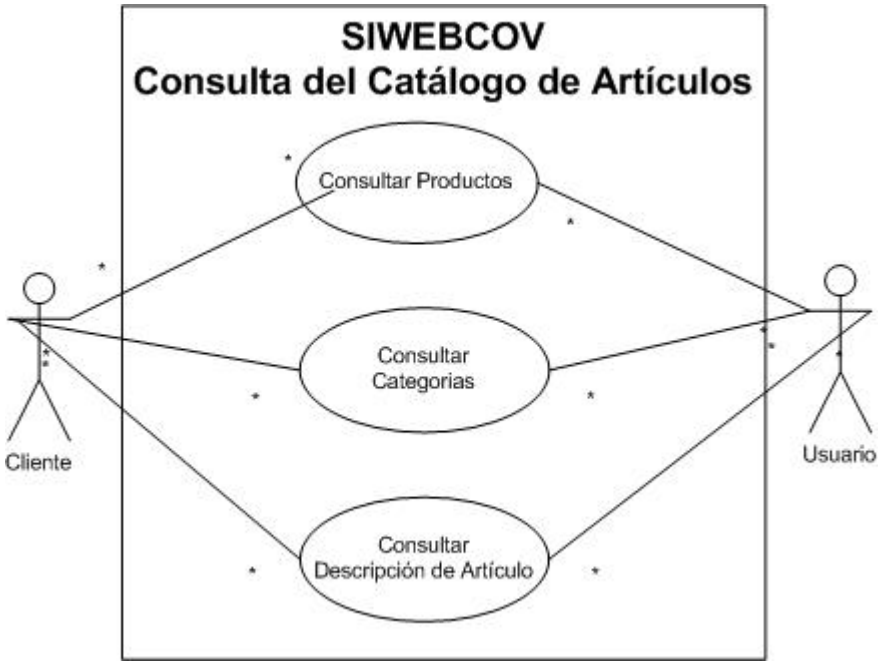


Figura 10. Caso de Uso Consulta del catálogo de artículos

Curso de Eventos:		Consulta del catálogo de artículos	
1.	Usuario o cliente accede a la página de ventas	**2.**	SIWEBCOV muestra las categorías del artículo
3.	Usuario o cliente navega a través del catálogo de artículos.	**4.**	SIWEBCOV muestra la página solicitada.

Caso de uso:	Registro de Cliente
Actores:	SIWEBCOV y Cliente
Propósito	Registrar la información del cliente
Tipo:	Primario
Descripción:	SIWEBCOV muestra los formularios que el Cliente debe llenar, para usar la información en el registro de un pedido.

El diagrama de casos correspondiente al registro del cliente es:

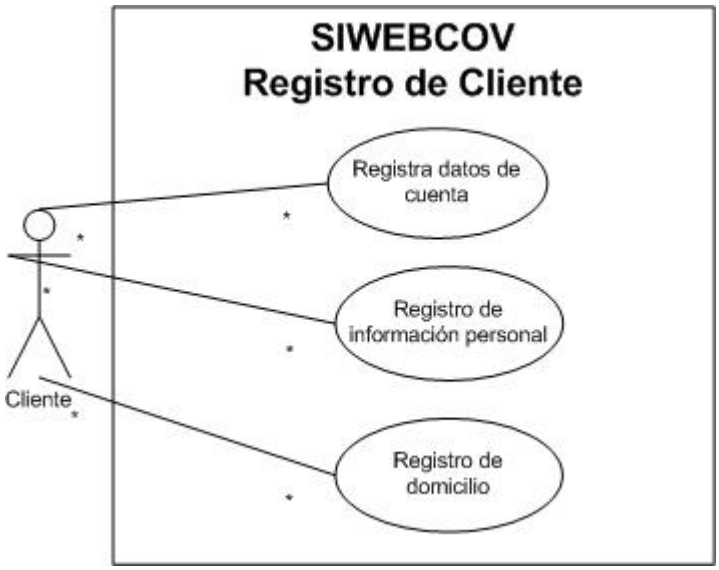


Figura 11. Caso de Uso Registro de Cliente

Caso de uso:	Realizar pedido de productos
Actores:	Cliente, SIWEBCOV
Usa caso:	Consulta del catálogo de artículos
Usa caso:	Registro de Cliente
Propósito	Registrar una venta
Tipo:	Primario
Descripción:	Registrar un pedido hecho por un cliente, captando información necesaria e indispensable para cumplir la operación.

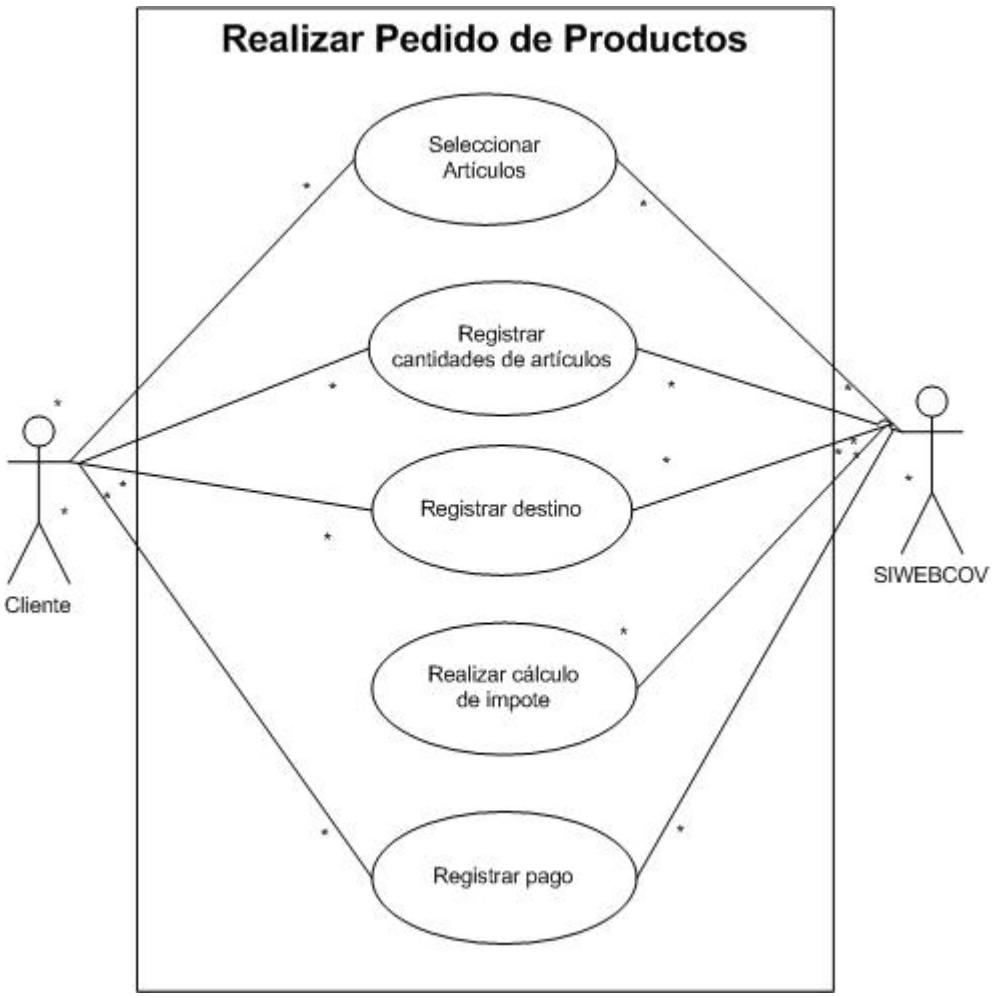


Figura 12. Caso de Uso Realizar pedido de productos

Curso de Eventos:		**Realizar pedido de productos**	
1.	**Usa Curso de Eventos:**	**Consulta del catálogo de artículos**	
2.	Cliente añade un producto al carrito de compras	**3.**	SIWEBCOV calcula el importe de los productos seleccionados
4.	Cliente solicita comprar artículos seleccionados	**5.**	SIWEBCOV proporciona el formulario al Cliente para captar su información
6.	Cliente rellena el formulario y lo envía.	**7.**	SIWEBCOV valida los datos del cliente, guarda los datos y presenta transacción terminada con éxito.

Caso de uso:	**Autenticación de usuario**
Actores:	SIWEBCOV y Usuario
Propósito	Autenticación de usuario al ingresar al sistema.
Tipo:	Primario
Descripción:	Validar al usuario y presentar las operaciones del menú permitidas de acuerdo a su perfil.

El diagrama de caso de uso para la autenticación es el siguiente:

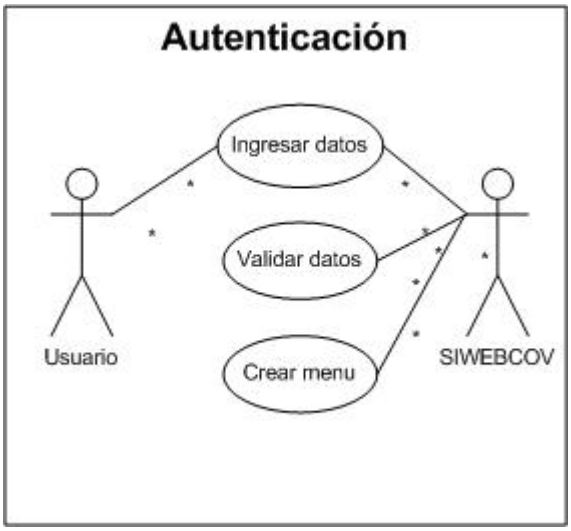


Figura 13. Caso de Uso Autenticar usuario

Curso de Eventos:		**Autenticación de usuario**	
1.	Usuario carga la página de Inicio de sesión.	**2.**	SIWEBCOV pide autenticación
3.	Usuario teclea nombre de la cuenta y palabra clave.	**4.**	SIWEBCOV muestra el menú correspondiente al nivel de usuario

Caso de uso:	**Embarque de artículo**
Actores:	SIWEBCOV y Usuario
Propósito	Generar una orden de Embarque.
Tipo:	Primario
Descripción:	El usuario podrá generar una orden de embarque, para la distribución de mercancía al cliente.

El diagrama de casos uso corresponde al Embarque de artículo es:

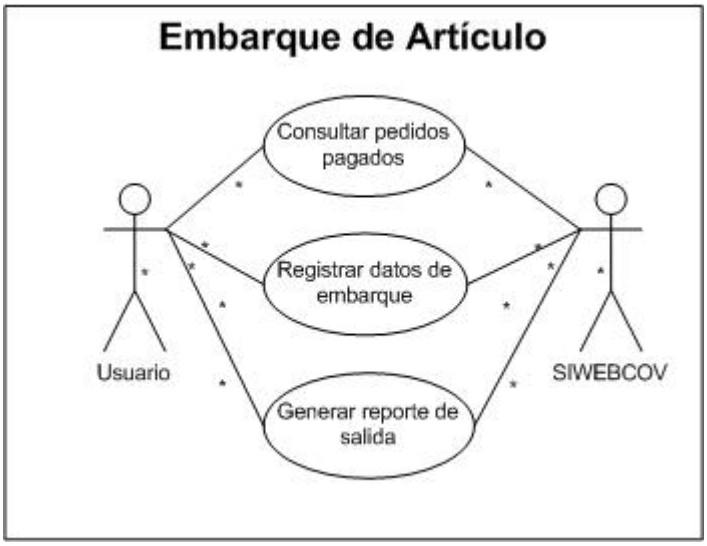


Figura 14. Caso de Uso Embarque de artículo

Curso de Eventos:		**Embarque de artículo**	
1.	Usa curso de Eventos.	Autenticación de usuario.	
2.	Usuario solicita listado de pedidos pendientes por surtir.	**3.**	SIWEBCOV muestra pedidos pendientes solicitados.
4.	Usuario registra los pedidos a Embarcar	**5.**	SIWEBCOV cambia el estatus de “Pendiente” a “En ruta”. Y genera reporte de Embarque.

Caso de uso:	**Seguimiento de entrega**
Actores:	SIWEBCOV y Usuario
Propósito	Generar una orden de Embarque.
Tipo:	Primario
Descripción:	El usuario podrá registrar la correcta entrega, cambiando el estatus a “Entregado”.

Curso de Eventos:		**Seguimiento de entrega**	
1.	Usa curso de Eventos.	Autenticación de usuario.	
2.	Usuario solicita listado de pedidos embarcados.	**3.**	SIWEBCOV muestra pedidos Embarcados.
4.	Usuario selecciona los pedidos “Embarcados” y que fueron entregados.	**5.**	SIWEBCOV cambia el estatus de “Embarcado” a “Entregado”. Y genera reporte de pedidos entregados.

El diagrama del caso de uso Seguimiento de entrega es el siguiente:

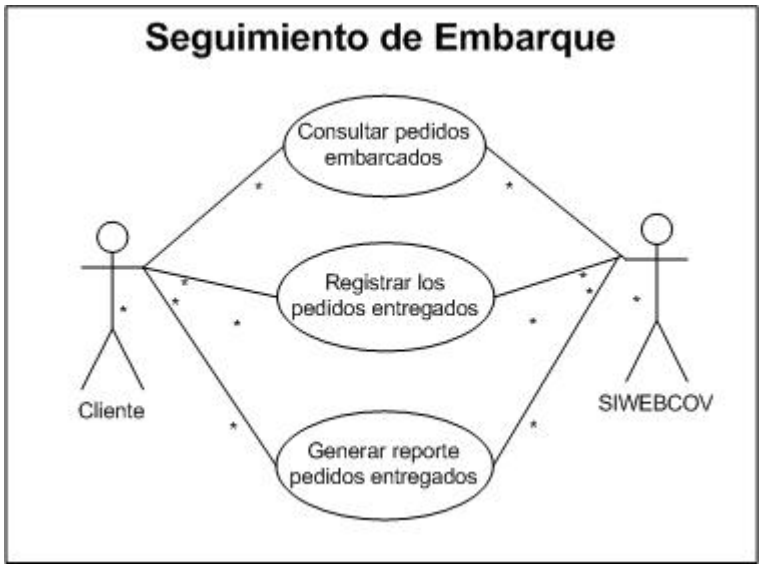


Figura 15. Caso de Uso Seguimiento de entrega

Caso de uso:	Altas, bajas y cambios de catálogos
Actores:	Usuario, SIWEBCOV
Propósito	Ingresar, dar de baja o modificar a un catalogo
Tipo:	Primario
Descripción:	El personal de la empresa, podrá realizar los cambios requeridos en la base de datos siempre y cuando tenga acceso a esas funciones.

El diagrama de casos de uso para realizar Altas, bajas y cambios de catálogos se muestra a continuación:

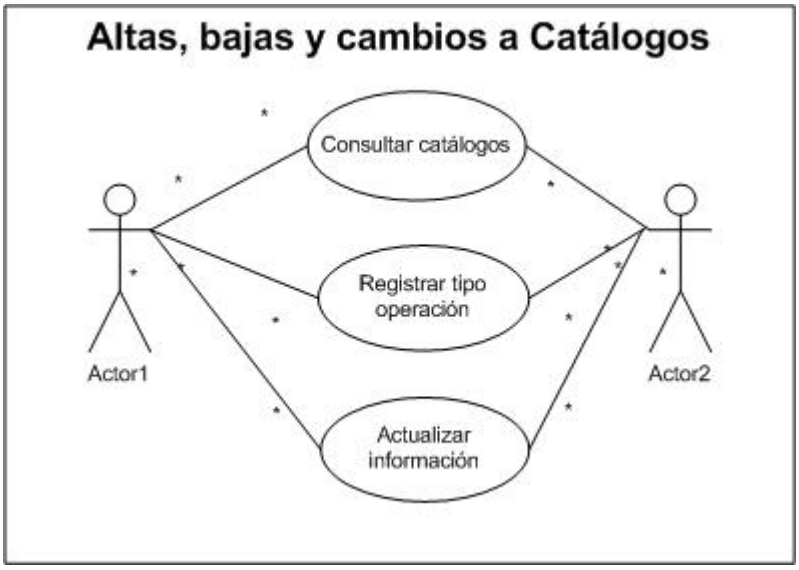


Figura 16. Caso de Uso Altas, bajas y cambios de catálogos

Curso de Eventos:		**Altas, bajas y cambios de catálogos**	
1.	Usa curso de Eventos.	Autenticación de usuario.	
2.	Usuario selecciona el módulo de su interés	**3.**	SIWEBCOV muestra listado correspondiente a la opción elegida del usuario.
4.	Usuario selecciona la operación a realizar (Alta/Baja/Cambio)	**5.**	SIWEBCOV realiza la solicitud del usuario.

2.5 Diagrama de Clases

Al examinar los casos de uso se determinaron los objetos que formarán parte del sistema. Al clasificarlos y estudiar sus relaciones se obtuvo la estructura estática general del sistema, misma que está representada por el siguiente diagrama de clases.

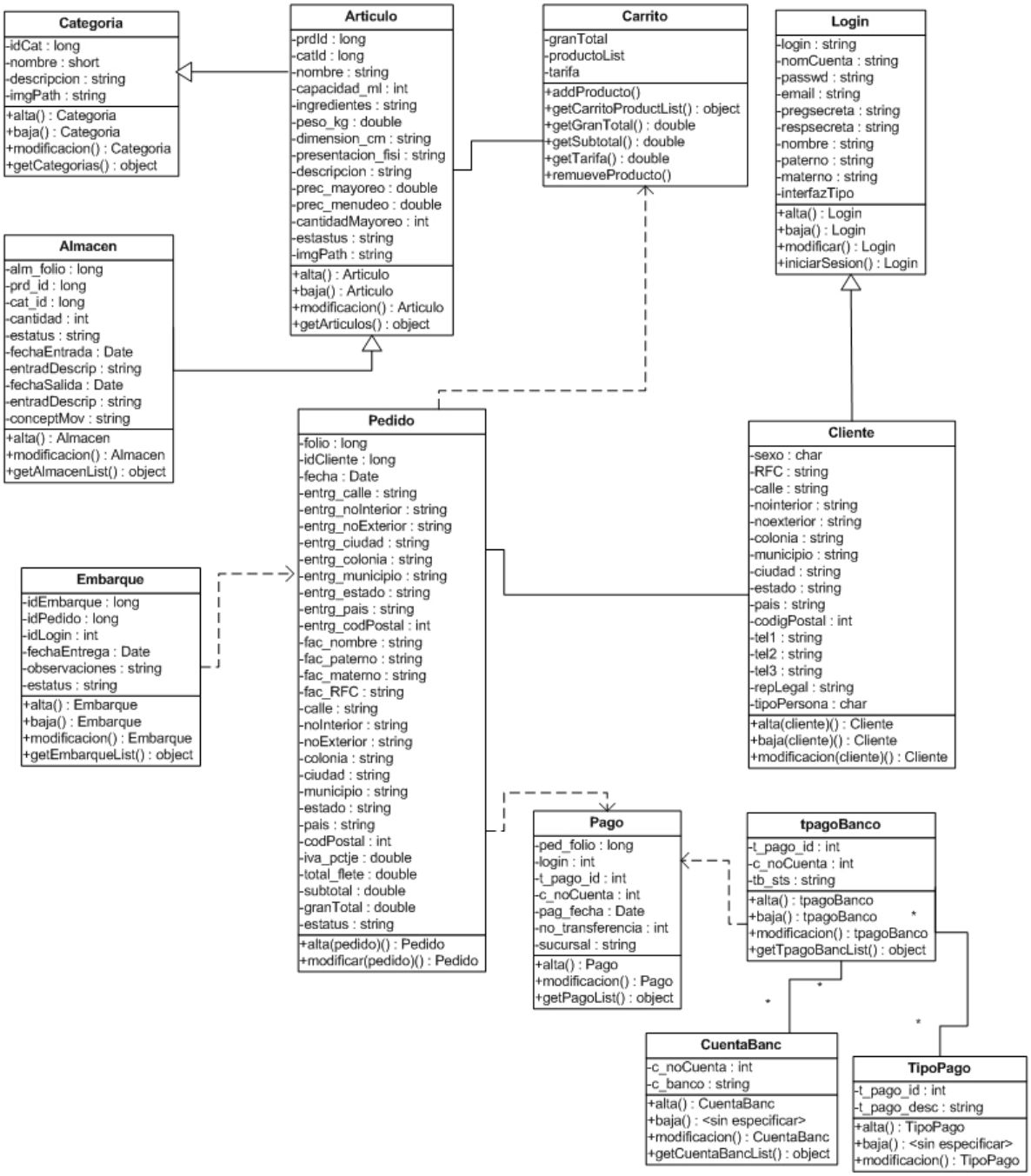


Figura 17. Diagrama de Clases

2.6 Diagrama de secuencia

A continuación se ilustra un diagrama de secuencia que muestra la interacción entre los objetos involucrados para realizar la venta de productos.

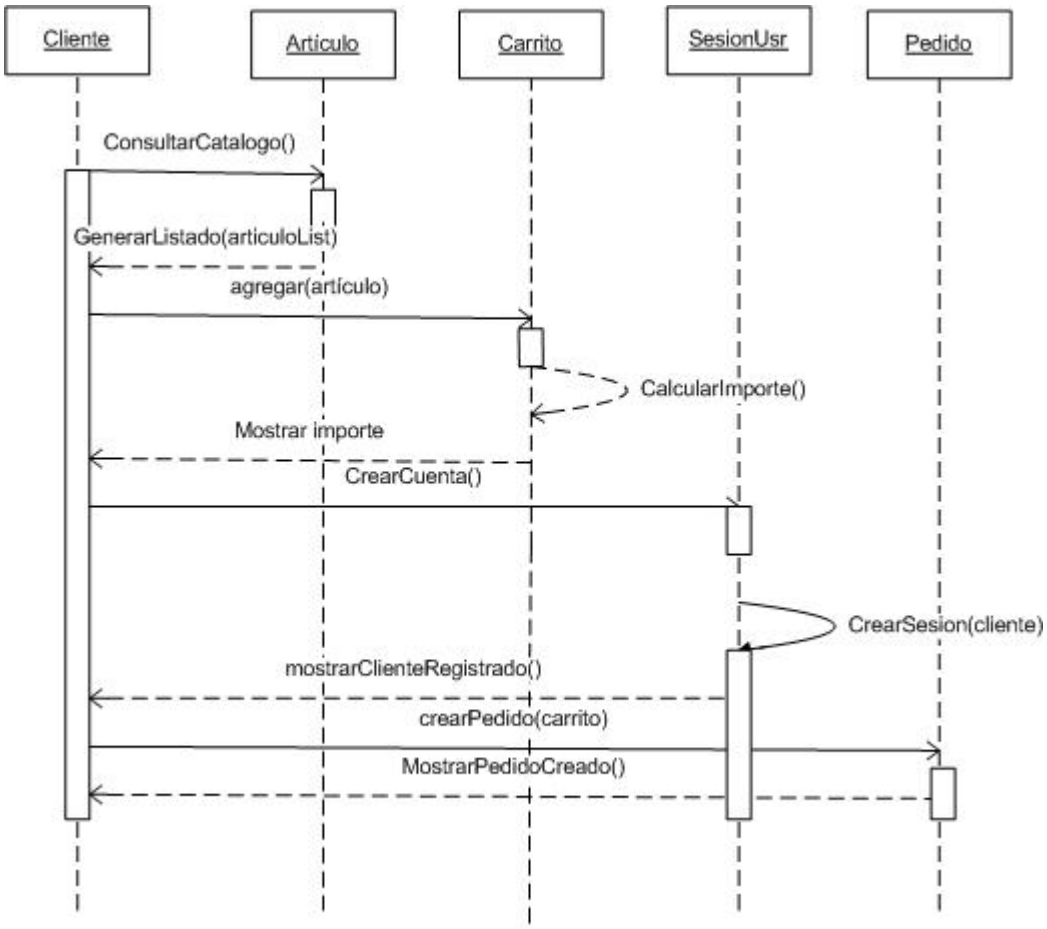


Figura 18. Diagrama de secuencia venta

El siguiente diagrama de secuencia, muestra la interacción del usuario para solicitar el listado de pedidos pendientes.

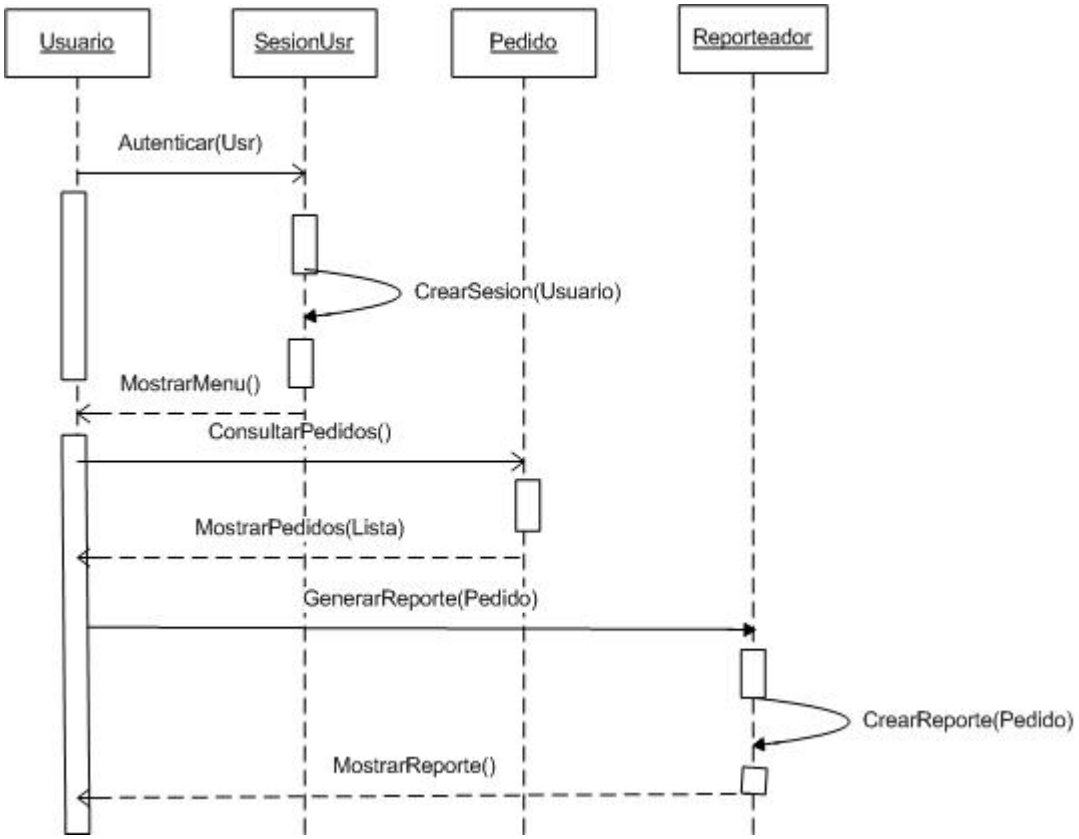


Figura 19. Diagrama de secuencia Listado pedidos pendientes.

El reporte de ventas sigue el siguiente diagrama de secuencia.

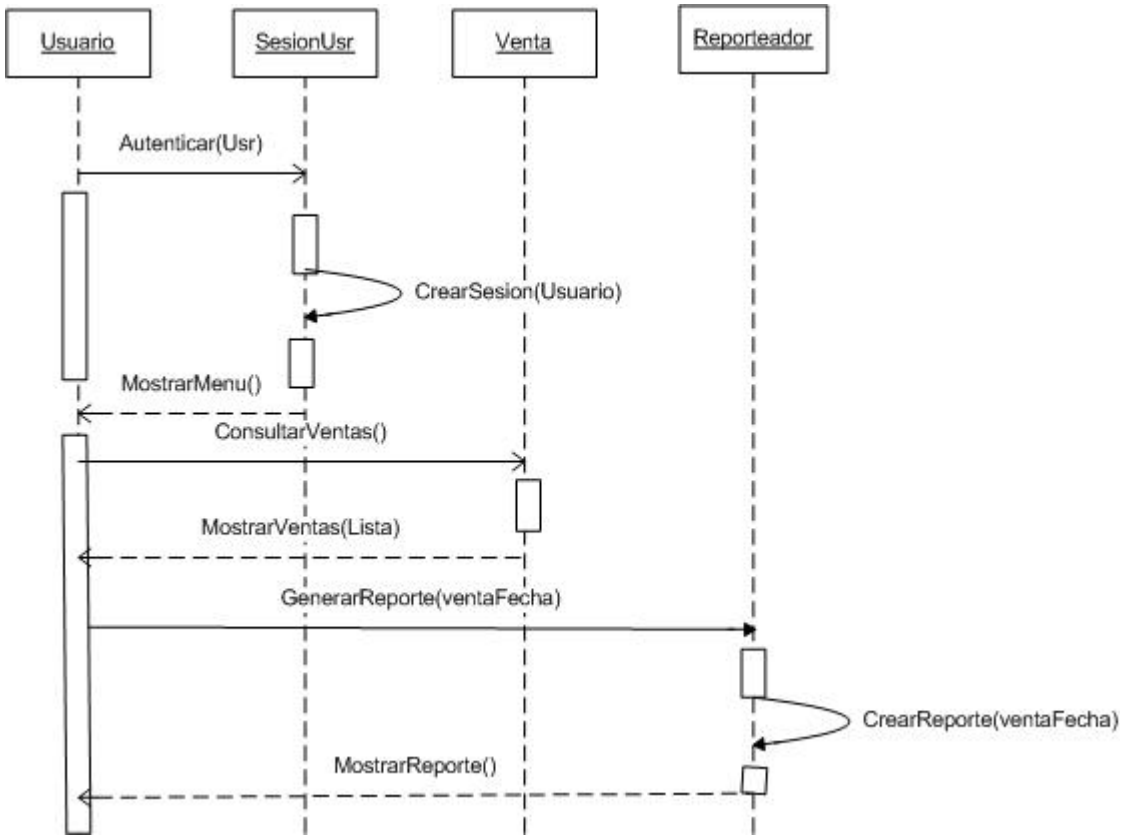


Figura 20. Diagrama de secuencia reporte de ventas

2.7 Soluciones de Software

Existen productos comerciales de software que pueden usarse para la solución del problema; almacenar la información que se deriva de todas las transacciones del negocio, se puede considerar el manejador de base de datos Microsoft SQL Server, el cual corre sobre una plataforma como Windows 2003 Server, la información se requiere que este actualizada y que se puedan realizar pedidos a través de Internet por lo que se requiere de un servidor Web; se propone a Internet Information Server para servir páginas con formato HTML y por su gran integración con el servidor de base de datos. Para acceder a la base de datos usando un front-end Web emplear el lenguaje ASP.

Otra alternativa es usar software libre; el costo de las licencias de la solución propuesta se incrementa considerablemente a comparación del uso del software

libre, además que el software comercial sufre de una gran cantidad de ataques tanto de virus como de hackers; por lo que la convierten más vulnerable.

De tal manera, se recomienda el uso de software libre que cada vez tiene más auge y las funcionalidades son similares que el software propietario. La propuesta es usar al contenedor Apache Tomcat para servir páginas dinámicas, la base de datos MySQL Database Server es una buena alternativa debido a que es muy rápido, confiable y fácil de usar; su conectividad, velocidad y seguridad hacen a MySQL altamente satisfactorio para acceder bases de datos en Internet.

Se presenta un pequeño resumen, en modo de tabla comparativa, con los criterios principales a tener en cuenta:

Criterios	Bases de datos			
	Access	SQL Server	MySQL	PostgreSQL
Plataforma			/	/
Velocidad	↓	↑	↑	↓
Volumen Datos	↓	↑	↑	↑
Integridad	↓	↑	↓	↑
Potencia	↓	↑	↑	↑
Coste/MB	↑	↓	↑	↑
↑ Positivo ↓ Negativo				

Fuente: http://www.arsys.es/soporte/programacion/comparatica.htm

2.8 Requerimientos

Toda aplicación se ejecuta en un ambiente y requiere especificaciones de software y hardware, se especifican los requerimientos a continuación.

2.9.1 Hardware

Los requerimientos mínimos de hardware son los siguientes:

- Procesador Pentium IV a 1.2 Ghz.
- 512 Mb RAM.
- 80 Gb disco duro.
- Monitor a color SVGA.
- Tarjeta de Red.
- Unidad de CD.
- Unidad de respaldo de CD.
- Mouse.
- Teclado.
- No break.

2.9.2 Software

Los requerimientos de software que se necesitan para el uso del sistema son los siguientes:

- Sistema operativo: Linux Red Had 7.
- Servidor de Aplicaciones: Jakarta Tomcat 4.1.
- Gestor de base de datos: MySql.
- Java Virtual Machine 1.4.

2.10 Arquitectura y diseño

La arquitectura representa la infraestructura del software, es decir, las capas sobre las cuales corre una aplicación.

Actualmente el diseño de sistemas se valora más por la flexibilidad que tiene una aplicación de ejecutarse en la mayoría de los ambientes operativos de software. Tal es el caso de las aplicaciones basadas en Java. Los programas basados en Java se caracterizan por la independencia de plataforma.

El software basado en el Web, por lo general corre sobre un servidor de aplicaciones, contenedor o servidor Web. Se recomienda a Jakarta Tomcat 4.1

como mínimo por su facilidad de uso y el rendimiento que tiene al servir aplicaciones basadas en java.

El servidor de base de datos debe ser considerado como una capa de software independiente que puede atender peticiones de servicio de datos a cualquier cliente con permisos especiales.

MySql es una base de datos que se usa principalmente por su velocidad y facilidad de uso, mediante el driver MySQL connector de java puede integrarse a la aplicación y almacenar los datos de forma estructurada y organizada permitiendo la confiabilidad de acceso a los datos.

La interfaz representa la capa de usuario, también se le conoce como front-end. Esta debe ser diseñada de tal forma que su funcionamiento sea independiente a las demás capas, su código estará representado en su mayoría de html y javascript.

La arquitectura multicapa es una de las tecnologías más reciente para el diseño de software, puesto que ofrece grandes ventajas al permitir desarrollar módulos independientes en forma paralela, las capas son independientes entre sí, por lo que un cambio ya sea del front-end o de la misma base de datos no representa una inversión tan costosa de tiempo a comparación de una arquitectura de dos capas. Otra ventaja es que el mantenimiento de código es más rápido.

La arquitectura de tres capas esta representada por:

- La capa de presentación o interfaz.
- La capa de acceso a los datos.
- La capa de reglas de negocio.

El diseño de cada capa requerirá del uso de componentes que deberán desarrollarse primordialmente para la reutilización de código. El diseño de la aplicación se muestra en el siguiente capítulo.

CAPÍTULO 3

DISEÑO

CAPÍTULO 3. DISEÑO

Cuando un arquitecto desea dar a entender mejor la construcción de un edificio lo hace mediante un modelo idéntico pero a una menor escala, el diseño es usado para expresar la arquitectura del objeto a materializar.

1. *El Diseño es una actividad técnica y creativa encaminada a idear un proyecto útil, funcional y estético que pueda llegar a producirse en serie como en el diseño industrial, el diseño gráfico o el diseño de joyas*.[7]

Sin embargo cuando aquello que se desea construir es un software, el modelo debe tomar una forma diferente, debe representar todas las funciones y subfunciones de un sistema. Los modelos se concentran en lo que debe hacer el sistema no en como lo hace, estos modelos pueden incluir notación gráfica, información y comportamiento del sistema.

El diseño de sistemas se puede definir como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física.

Para la etapa del diseño del sistema se consideran los siguientes modelos:

- Modelo de la aplicación.
- Diseño de las reglas de negocio (Diagrama de secuencias).
- Diseño de la base de datos.
- Diseño de la interfaz

3.1 Modelo de la aplicación

Para obtener un modelo adecuado que resuelva el problema con efectividad se usará un patrón de diseño.

[7] Definición tomada de la Enciclopedia Wikipedia http://es.wikipedia.org/wiki/

3.2 Patrón de Diseño

El uso de patrones para poder resolver problemas se ha vuelto tan común que también son usados para resolver proyectos informáticos.

Definición de patrón de diseño:

"*Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software*."[8]

Los patrones de diseño proporcionan una solución probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. De tal manera que para poder elegir el patrón adecuado deben considerarse algunos elementos como: el nombre del patrón, el contexto para poder aplicar el patrón, la solución y el costo beneficio que se obtiene al utilizar el patrón.

Para el área del desarrollo del software existen algunos patrones como:

- Patrones Creacionales: Inicialización y configuración de objetos.
- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

3.2.1 Modelo-Vista-Controlador

Es un patrón del tipo creacional que ha demostrado ser fundamental a la hora de diseñar aplicaciones Web; plantea la separación del problema en tres capas: la capa **modelo**, que representa la realidad; la capa **controlador,** que conoce los métodos y atributos del modelo, recibe y realiza lo que el usuario quiere hacer; y la capa **vista**, que muestra un aspecto del modelo y es utilizada por la capa anterior para interaccionar con el usuario.

La configuración habitual para aplicaciones accesibles desde Internet con acceso a bases de datos, es la que se muestra en la figura siguiente:

[8] Design Patterns (Elements of Reusable Object-Oriented Software)

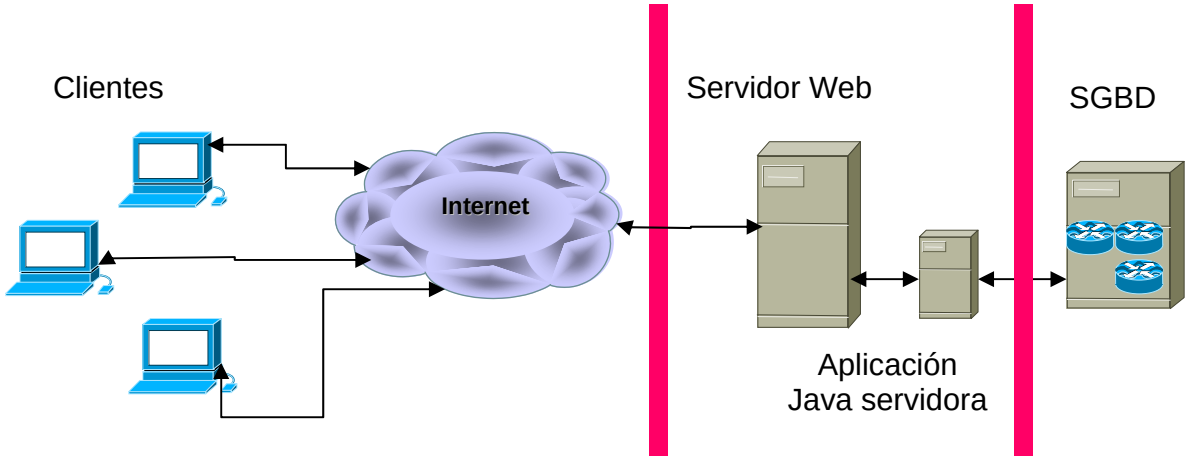


Figura 21. Modelo de tres capas

El diagrama anterior representa un modelo de tres capas, debido a que, entre el cliente constituye el elemento solicitante de información y el sistema de gestión de bases de datos (SGBD), que es el elemento que contiene la información solicitada, se inserta una nueva capa que hace de interfaz entre las dos anteriores.

Esta capa intermedia, es un servidor de aplicaciones y es quien soporta la lógica de negocio para el intercambio de información entre el cliente y el SGBD.

La aplicación se encontrará dividida en tres áreas o niveles:

1. **Nivel de presentación**: será el encargado de generar la interfaz de usuario en función de las acciones llevadas a cabo por el mismo.

2. **Nivel de negocio**: contendrá toda la lógica que modelará los procesos de negocio y es donde se realiza todo el procesamiento necesario para atender a las peticiones del usuario.

3. **Nivel de administración de datos**: será el encargado de hacer persistente toda la información, suministrará y almacenará información para el nivel de negocio.

3.3 Diseño de la Base de Datos

Para el diseño de la base de datos se utilizará el modelo de datos relacional, obteniéndose el modelo lógico y físico de los datos, garantizándose la eliminación de redundancias e inconsistencias.

Se obtuvo el Diagrama Entidad Relación, detallándose en éste las entidades que se relacionan con el sistema, sus atributos y las relaciones entre ellas.

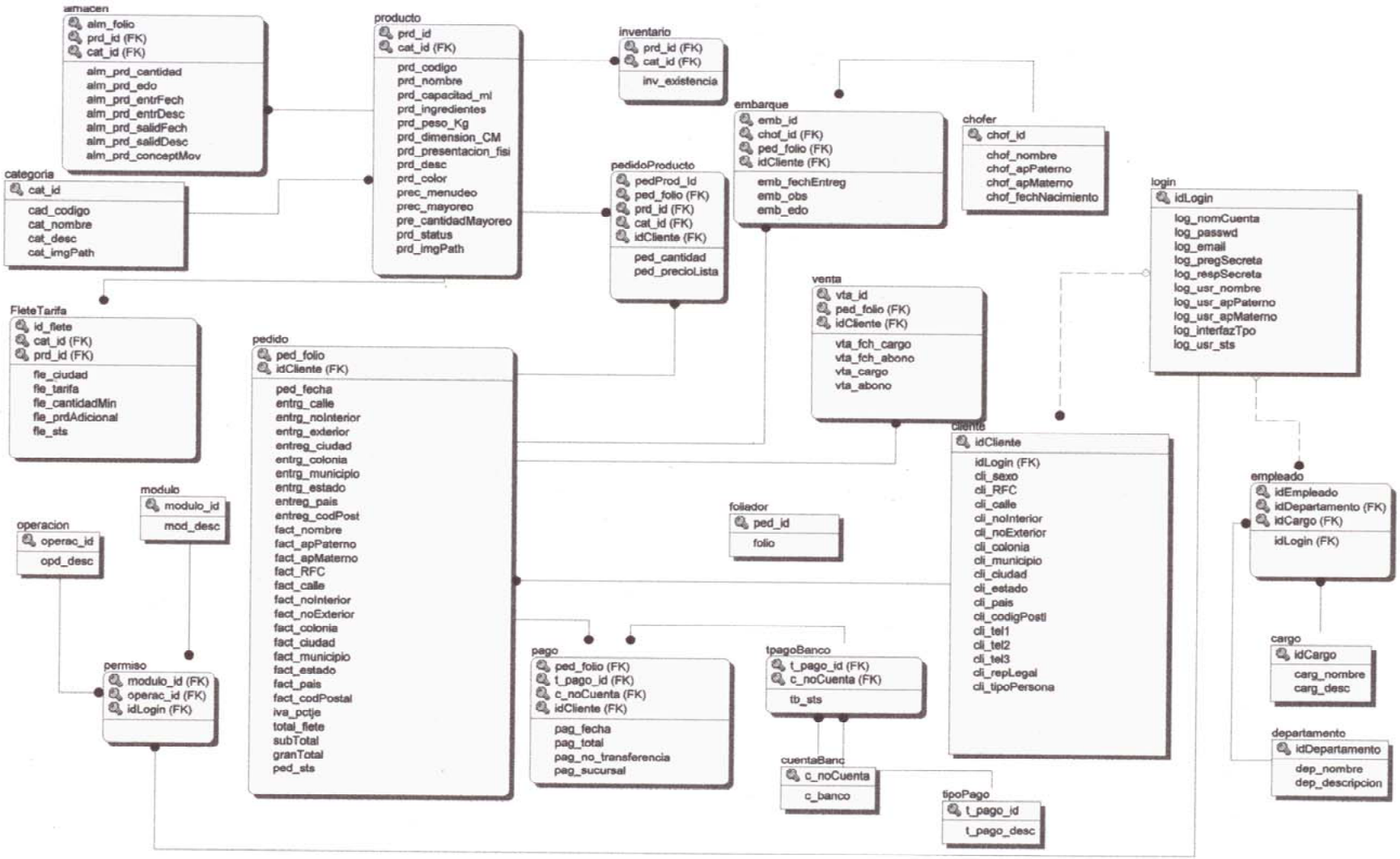


Figura 22. Diagrama ER del modelo propuesto

3.4 Diccionario de Datos

A continuación se muestra el diccionario de datos obtenido

Tabla	almacen	Type Key
alm_folio	INTEGER NOT NULL	PK
prd_id	VARCHAR (10) NOT NULL	FK
cat_id	VARCHAR (10) NOT NULL	FK
alm_prd_cantidad	INTEGER NOT NULL	
alm_prd_edo	VARCHAR (15) NOT NULL	
alm_prd_entrFech	DATE NOT NULL	
alm_prd_entrDesc	DATE NOT NULL	
alm_prd_salidFech	DATE NOT NULL	
alm_prd_salidDesc	DATE NOT NULL	
alm_prd_conceptMov	VARCHAR (20) NULL	

Tabla	Categoría	Type Key
cat_id	VARCHAR (20) NOT NULL	PK
cat_nombre	VARCHAR (40) NULL	
cat_desc	VARCHAR (250) NOT NULL	
cat_imgPath	VARCHAR (255) NULL	

Tabla	chofer	Type Key
chof_id	INTEGER NOT NULL	PK
chof_nombre	VARCHAR (30) NULL	
chof_apPaterno	VARCHAR (30) NULL	
chof_apMaterno	VARCHAR (30) NULL	
chof_fechNacimiento	DATE NULL	

Tabla	cuentaBanc	Type Key
c_noCuenta	INTEGER NOT NULL	PK
banco	VARCHAR (20) NULL	

Tabla	embarque	Type Key
emb_id	INTEGER NOT NULL	PK
chof_id	INTEGER NOT NULL	FK
idCliente	VARCHAR (20) NOT NULL	FK
ped_folio	INTEGER NOT NULL	FK
emb_fechEntreg	DATE NULL	
emb_obs	VARCHAR (255) NULL	
emb_sts	VARCHAR (20) NULL	

Tabla	cliente	Type Key
idCliente	INTEGER NOT NUL	PK
login	VARCHAR (20) NOT NULL	FK
cli_sexo	VARCHAR (10) NULL	
cli_RFC	VARCHAR (13) NULL	
cli_calle	VARCHAR (30) NULL	
cli_noInterior	VARCHAR 5) NULL	
cli_noExterior	VARCHAR (5) NULL	
cli_colonia	VARCHAR (30) NULL	
cli_municipio	VARCHAR (30) NULL	
cli_ciudad	VARCHAR (30) NULL	
cli_estado	VARCHAR (30) NULL	
cli_pais	VARCHAR (30) NULL	
cli_codigostl	INTEGER NULL	
cli_tel1	VARCHAR (22) NULL	
cli_tel2	VARCHAR (22) NULL	
cli_tel3	VARCHAR (22) NULL	
cli_rep_Legal	VARCHAR (60) NULL	
cli_tipoPersona	CHAR (1) NULL	

Tabla	FleteTarifa	Type Key
id_flete	INTEGER NOT NULL	PK
cat_id	VARCHAR (10) NOT NULL	FK
prd_id	VARCHAR (10) NOT NULL	FK
fle_ciudad	VARCHAR (255) NULL	
fle_tarifa	DECIMAL92) NULL	
fle_cantidadMin	INTEGER NULL	
fle_prdAdicional	DECIMAL92) NULL	
fle_sts	VARCHAR (20) NULL	

Tabla	foliador	Type Key
fol_tipo	VARCHAR (10) NOT NULL	PK
folio	INTEGER NOT NULL	

Tabla	inventario	Type Key
prd_id	VARCHAR (10) NOT NULL	FK
cat_id	VARCHAR (10) NOT NULL	FK
inv_existencia	INTEGER NULL	

Tabla	login	Type Key
login	INTEGER NOT NULL	PK
log_nomCuenta	VARCHAR(40) NULL	
log_passwd	VARCHAR (10) NOT NULL	
log_email	VARCHAR (40) NULL	
log_pregSecreta	VARCHAR (255) NOT NULL	
log_respSecreta	VARCHAR (255) NOT NULL	
log_usr_nombre	VARCHAR (255) NOT NULL	
usr_apPaterno	VARCHAR (255) NOT NULL	
usr_apMaterno	VARCHAR (255) NOT NULL	
log_interfazTpo	SMALLINT NOT NULL	
usr_sts	VARCHAR(10) NULL	

Tabla	pago	Type Key
ped_folio	INTEGER NOT NULL	FK
t_pago_id	SMALLINT NOT NULL	FK
noCuenta	INTEGER NOT NULL,	FK
idCliente	INTEGER NOT NULL	FK
pag_fecha	INTEGER NOT NULL	
pag_total	DECIMAL(9,2) NULL	
no_transferencia	INTEGER NULL	
sucursal	VARCHAR (30) NULL	

Tabla	pedidoProducto	Type Key
pedProd_Id	INTEGER NOT NULL	PK
ped_folio	INTEGER NOT NULL	FK
idCliente	INTEGER NOT NULL	FK
prd_id	VARCHAR (10) NOT NULL	FK
cat_id	VARCHAR (10) NOT NULL	FK
ped_cantidad	DECIMAL (9,2) NULL	
ped_precioLista	DECIMAL (7,2) NULL	

Tabla	menuusr	Type Key
modulo_id	INTEGER NOT NULL	FK
operac_id	INTEGER NOT NULL	FK
login	INTEGER NOT NULL	FK

Tabla	modulo	Type Key
modulo_id	INTEGER NOT NULL	PK
mod_desc	VARCHAR (25) NOT NULL	

Tabla	operacion	Type Key
operac_id	INTEGER NOT NULL	PK
opd_desc	VARCHAR (25) NOT NULL	

Tabla	pedido	Type Key
ped_folio	INTEGER NOT NULL	PK
idCliente	INTEGER NOT NULL	FK
ped_fecha	DATE NOT NULL	
entrg_calle	VARCHAR (30) NULL	
entrg_noInterior	VARCHAR (5) NULL	
entrg_exterior	VARCHAR (5) NULL	
entreg_ciudad	VARCHAR (30) NULL	
entrg_colonia	VARCHAR (30) NULL	
entrg_municipio	VARCHAR (30) NULL	
entrg_estado	VARCHAR (30) NULL	
entreg_pais	VARCHAR (30) NULL	
entreg_codPostal	INTEGER NULL	
fact_nombre	VARCHAR (30) NULL	
fact_apPaterno	VARCHAR (30) NULL	
fact_apMaterno	VARCHAR (30) NULL	
fact_RFC	VARCHAR (13) NULL	
fact_calle	VARCHAR (30) NULL	
fact_noInterior	VARCHAR (5) NULL	
fact_noExterior	VARCHAR (5) NULL	
fact_colonia	VARCHAR (30) NULL	
fact_ciudad	VARCHAR (30) NULL	
fact_municipio	VARCHAR (30) NULL	
fact_estado	VARCHAR (30) NULL	
fact_pais	VARCHAR (30) NULL	
fact_codPostal	INTEGER NULL	
total_flete	DECIMAL72) NULL	
subTotal	DECIMAL92) NULL	
granTotal	DECIMAL92) NULL	
ped_sts	VARCHAR (20) NULL	

Tabla	empleado	Type Key
idEmpleado	INTEGER NOT NULL	PK
idDepartamento	INTEGER NOT NULL	FK
idCargo	INTEGER NOT NULL	FK
login	VARCHAR (30) NULL	

Tabla	producto	Type Key
prd_id	INTEGER NOT NULL	PK
cat_id	INTEGER NOT NULL	FK
prd_codigo	VARCHAR (20) NULL	
prd_nombre	VARCHAR (50) NULL	
prd_capacitad_ml	INTEGER NULL	
prd_ingredientes	VARCHAR (255) NULL	
prd_peso_Kg	DECIMAL (5,2) NULL	
prd_dimension_CM	VARCHAR (25) NULL	
prd_presentacion_fisi	VARCHAR (20) NULL	
prd_desc	VARCHAR (255) NULL	
prd_color	VARCHAR (20) NULL	
prec_menudeo	DECIMAL(7,2) NULL	
prec_mayoreo	DECIMAL(7,2) NULL	
prec_cantidadMayoreo	INTEGER NULL	
prd_status	VARCHAR (20) NULL	
prd_imgPath	VARCHAR (255) NULL	

Tabla	tpagoBanco	Type Key
t_pago_id	SMALLINT NOT NULL	FK
noCuenta	INTEGER NOT NULL	FK
tb_sts	VARCHAR (10) NULL	

Tabla	venta	Type Key
vta_id	INTEGER NOT NULL	PK
iva_pctje	INTEGER NOT NULL	FK
idCliente	INTEGER NOT NULL	FK
ped_folio	INTEGER NOT NULL	FK
vta_fch_cargo	DATE NULL	
vta_fch_abono	DATE NULL	
vta_cargo	DECIMAL(9,2) NULL	
vta_abono	DECIMAL(9,2) NULL	

Tabla	tipoPago	Type Key
t_pago_id	SMALLINT NOT NULL	PK
t_pago_desc	VARCHAR (30) NOT NULL	

Tabla	cargo	Type Key
idCargo	INTEGER NOT NULL	PK
carg_nombre	VARCHAR (40) NOT NULL	
carg_desc	VARCHAR (250) NULL	

Tabla	departamento	Type Key
idDepartamento	INTEGER NOT NULL	PK
dep_nombre	VARCHAR (40) NOT NULL	
dep_desc	VARCHAR (250) NULL	

3.5 Mapa de navegación

El mapa de navegación a primer nivel esta representado por la siguiente figura:

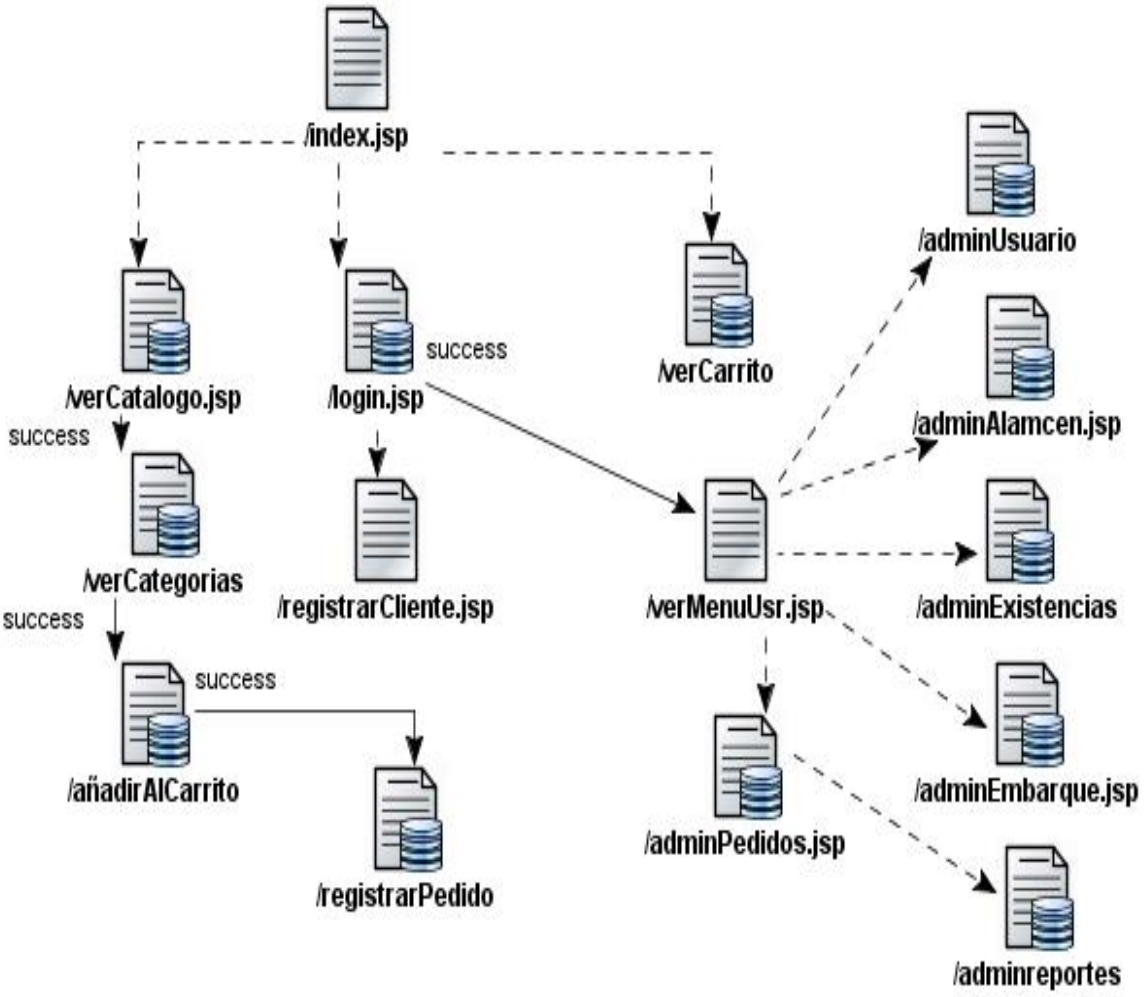


Figura 23. Mapa de navegación.

3.6 Diseño de la interfaz

Otro punto importante sobre el diseño de aplicaciones, es sin duda, el diseño de la interfaz de usuario, puesto que de ello depende la aceptación del sistema por parte del usuario final; éste es la persona que usará de forma concurrida el sistema.

Es importante hacer notar que no solo un sistema amigable y vistoso es suficiente para su aceptación, se requiere que se cumplan los requerimientos del cliente y que cumpla eficazmente las tareas para las que fue diseñado.

El diseño se centrará principalmente en presentar la información en forma congruente, fácil y cómoda. La navegabilidad que el sistema proporcione al usuario será sencilla y objetiva, evitando mal interpretaciones y solicitudes erróneas.

El diseño de la interfaz dependerá de las necesidades del usuario. Para una solución del tipo *E-Busines* o comercio electrónico, se deben atender los siguientes puntos:

- Atraer visitantes.
- Retener su atención.
- Cerrar ventas.

Algunas reglas que se consideran importantes para el diseño del *front-end* y garantizar el objetivo primordial de un sistema enfocado al comercio electrónico de la Empresa a los Clientes o *Bussines to Consumer* (B2C) son:

- La primera regla para diseñar un sitio Web es pensar en las necesidades y deseos de los clientes potenciales.

- Si el sitio está bien diseñado, los clientes obtendrán rápidamente respuestas a sus preguntas. De esta manera el sitio puede ser una herramienta de mercadotecnia exitosa.

- Para eliminar la confusión en los procesos de compra es importante organizar la presentación de la información de forma que se ajuste a la secuencia de preguntas del cliente

- La distribución de la página es muy importante. Hay que reflexionar acerca de su tamaño ideal, dependiendo del giro de la empresa y de la variedad de información y productos que se desean anunciar. Se recomienda utilizar textos cortos y concretos y con una buena redacción

- La página de inicio saluda al cliente y sirve de índice o directorio y es en general más larga que la pantalla de la computadora.

3.6.1 Lineamientos para diseñar un sitio Web

Un sitio Web se debe diseñar de acuerdo con algunos lineamientos fundamentales.

- **Seguir los estándares HTML:** es necesario que se sigan los estándares de HTML debido a que existen diferentes tipos de navegadores de Internet; la información se podría interpretar de manera distinta en diferentes *browsers*. Entre más exacto se especifique el contenido, el navegador podrá mostrarlo mejor.
- **Tipografía:** la tipografía debe ser lo más atractiva posible. El tamaño de la letra que se utilice en un sitio debe ser cómodamente legible. En el caso de los iconos, vínculos y cabezas de texto, es conveniente utilizar letras más grandes y con colores llamativos con respecto al formato normal que se esté utilizando, de ésta forma el visitante podrá identificar fácilmente los enlaces a otras páginas.
- **Redactar en línea:** la redacción debe contribuir para captar rápidamente el interés y la atención del usuario. Se debe proporcionar información útil, práctica y completa, con contenido atractivo y fácil de leer, utilizando las palabras clave en las primeras oraciones. También es importante evitar errores de ortografía y gramática.
- **Gráficos:** se debe aprovechar la naturaleza virtual del Web para incluir imágenes, fotografías o ilustraciones en el sitio y darle un atractivo visual. Se pueden utilizar imágenes simples, como el logotipo o lema de la empresa, fotografías de las instalaciones, fotografías del personal, estadísticas de las metas y logros de las áreas de la empresa, etc. También se pueden diseñar imágenes especiales para la página.
- **Colores:** una buena mezcla de tonos y texturas le darán al sitio una personalidad propia y una vista llamativa.
- **Elementos originales:** es preferible utilizar elementos originales y evitar los muy conocidos por los usuarios, ya que pueden resultar rutinarios y aburridos.
- **Tema:** es necesario mostrar al usuario el contenido y tema del sitio en la primera hoja y dar una idea general de lo que tratará en la página Web para encontrar fácil y rápidamente lo que se busca.

- **Vínculos:** utilizar vínculos ayuda a proporcionar mayor cantidad de información. Usar botones de iconos identificables es un recurso que atrae al usuario. Es recomendable utilizar razonablemente los vínculos. Los expertos recomiendan nunca tener más de cinco clics para llegar a la información, ya que el usuario se puede cansar de buscar la información que requiere y puede provocar que abandone el sitio y no regrese.
- **Imagen profesional:** para tener prestigio y una imagen profesional el mejor camino es decir siempre la verdad en cuanto a lo que se puede ofrecer en el sitio con relación a precios, existencias, calidades, características, tiempos de entrega, formas de pago, etcétera.
- **Seguridad:** el proveedor de Internet debe tener medidas preventivas para proteger la información del sitio, como firewalls, passwords, archivos de contraseñas, encriptación y otros métodos para resguardar datos.

3.7 La interfaz

Se puede notar que en empresas o negocios cuyo giro es la venta de artículos en línea, se requieren por lo menos dos tipos de interfaz: la interfaz del cliente y la del empleado. La interfaz del cliente es la encargada de mostrar el catálogo de artículos al público en general, se publican las promociones, las ofertas y propagandas, etc. Tiene la función de captar clientes, y la capacidad de registrar pedidos, el cliente es quien tiene mayor uso de ésta interfaz.

La interfaz del empleado sirve para que el usuario de la organización pueda realizar la administración de módulos como el de clientes, existencias, pedidos, mantenimiento a catálogos, expedición de reportes, etc.

Después de haber valorado el modelo de la interfaz y obtenido la aprobación, el diseño de interfaz para el cliente es:

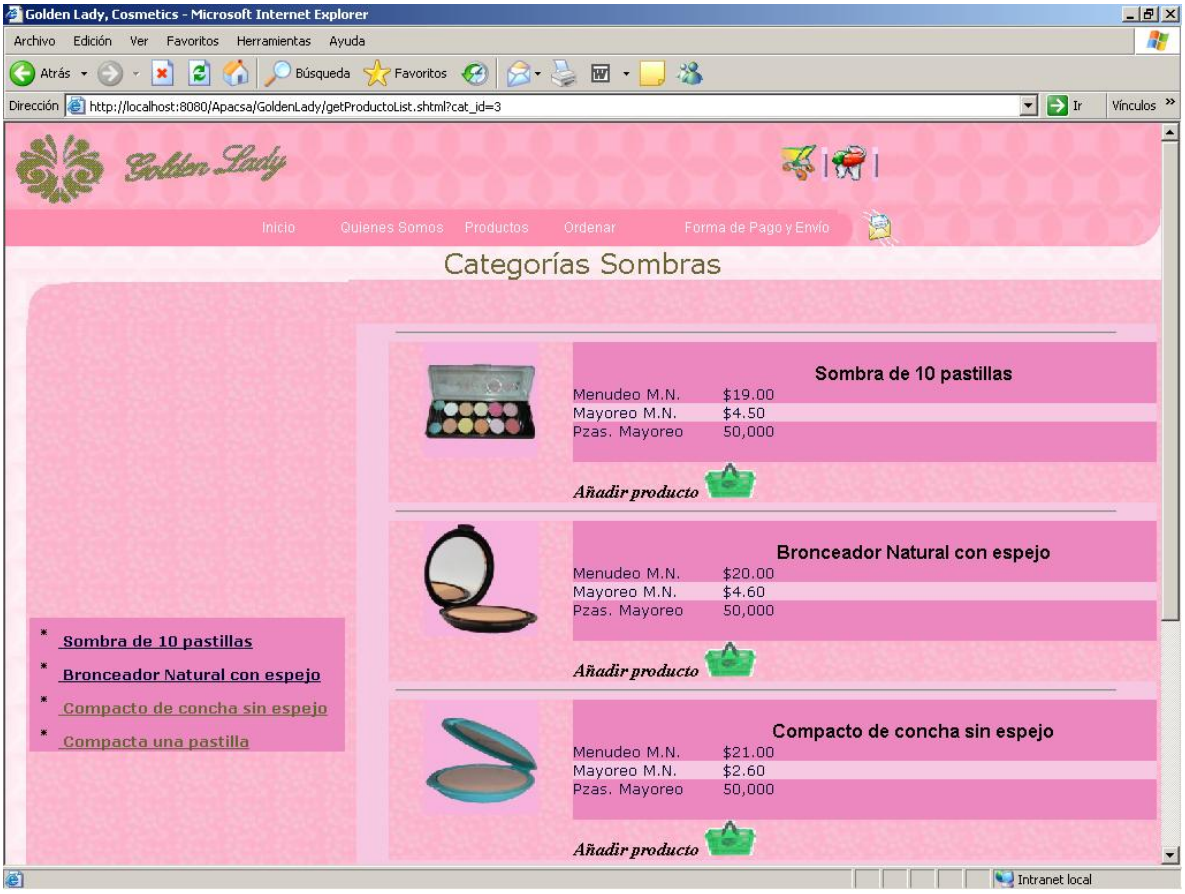


Figura 24. Interfaz del cliente

En la figura anterior se puede observar un esquema o modelo de interfaz para el cliente, el cual estará constituido de tres secciones principales, la parte superior llamada cabecera contendrá en la parte izquierda el logotipo correspondiente a la marca de los productos, este es importante ya que identifica a los productos en el mercado. En el mismo encabezado, en la parte derecha, se encuentran iconos para que el usuario pueda realizar transacciones como; solicitar inicio de sesión, ver el contenido de su carrito de compras, salir de la sesión, ver el perfil de usuario, etc.

Inmediatamente después del encabezado, se encuentra la barra de menú desplegable, éste contendrá la información de la empresa, enlace al catálogo de productos, e información sobre las formas de pago y entrega.

La última sección, llamada de trabajo; se encuentra dividida en dos: en una contendrá el listado de productos en formato de texto con hipervínculo, y en la parte derecha se deberá mostrar en forma de listado el producto y su descripción. Teniendo la funcionalidad de añadir el producto al carrito de compras o lista de pedidos a comprar.

La segunda interfaz corresponde al front-end para administrar los catálogos de la base de datos, Ésta interfaz estará constituida por tres frames, en el encabezado contendrá el logotipo de la empresa y en la parte derecha algunos iconos como el perfil del usuario y salida de sesión. El frame siguiente se encontrará dividido en dos, uno para la generación del menú dinámico de acuerdo a los permisos otorgados al usuario y el otro corresponde al frame de trabajo donde se cargarán todas las peticiones y solicitudes del usuario, el diseño corresponde a la siguiente pantalla.

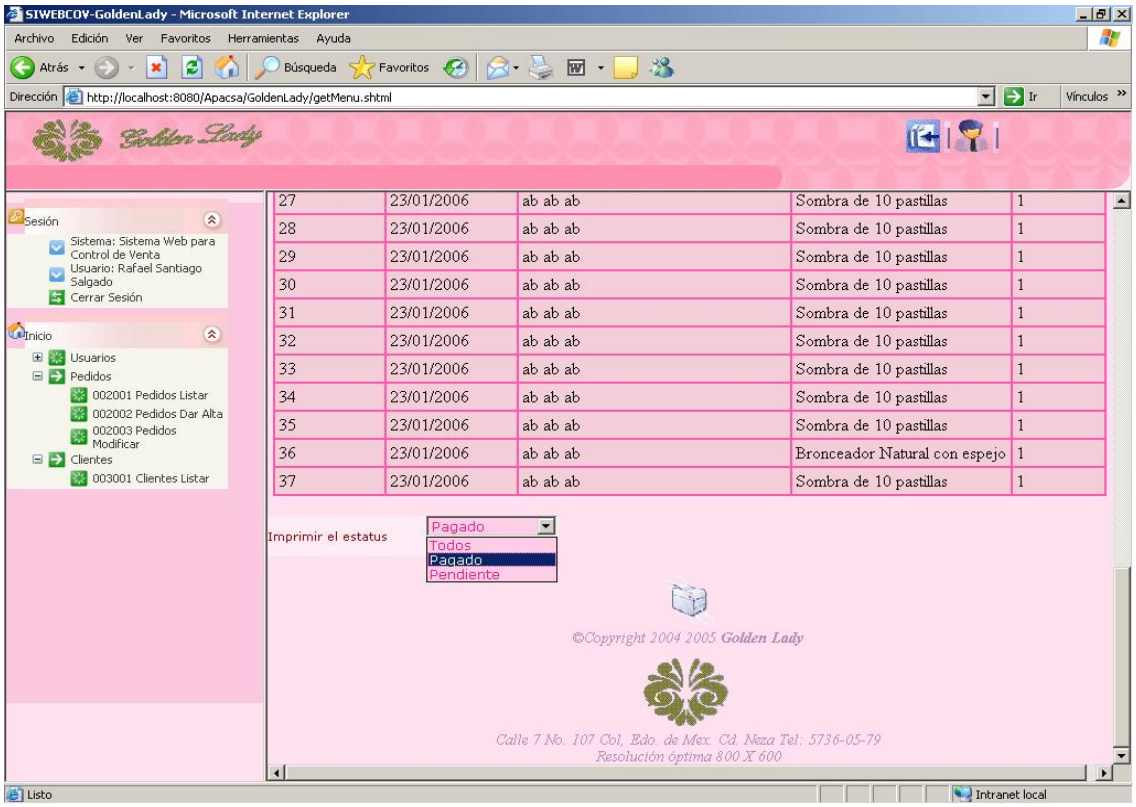


Figura 25. Interfaz del empleado

3.8 Reglas de Negocio

Las reglas del negocio deben ser validadas para cuidar la integridad de los datos, debe aplicarse la validación tanto en el cliente, como en la capa de la aplicación y en la base de datos, para ello se requiere conocer los valores permitidos de cada atributo, la longitud de caracteres que ocupará y las restricciones.

Las reglas de la aplicación se presentan a continuación:

Entidad	Cliente
- El nombre de la cuenta deberá ser único, identificará unívocamente al cliente. - La creación de una cuenta requiere que los datos del cliente no sean nulos.	

Entidad	Carrito de compras
- Se debe añadir un producto registrado en el sistema	
- Al añadir el producto al carrito debe acumularse la cantidad de artículos y calcular el total de la compra - La cantidad de compra de cada producto debe ser mayor que cero - El carrito debe permitir eliminar un producto o vaciarlo completamente - La información del carrito debe estar disponible en cualquier momento	

Entidad	Orden de pedido
- Debe ser expedida por un cliente registrado en el sistema - Debe contener por lo menos un artículo registrado en el sistema - Debe contener datos del cliente - Debe indicar un folio que identifique unívocamente el pedido - Debe indicar la cantidad y descripción de artículos - Debe indicar la cantidad de pago total -Tendrá los estatus “Pendiente”, “En ruta”, “Cancelado” y “Entregado” - El número total de productos a comprar es de 9999999	

Otras reglas de negocio que deben aplicarse correspondientes a la administración y gestión de la información deben ser las siguientes.

Entidad	Usuario
- La creación de una cuenta requiere que los datos del usuario no sean nulos. - El nombre de la cuenta de usuario, debe ser única.	

Entidad	Menú de Usuario
- El menú de administración de usuario debe estar restringido por contraseña - El menú debe generarse dinámicamente de acuerdo al perfil del usuario	

Acción	Mantenimiento de Catálogos
- Las claves y descripción no deben ser nulos - Las operaciones deben ser realizadas por un usuario con autorización	

El diseño corresponde una etapa del desarrollo de sistemas, posterior a esta se procede al desarrollo de la aplicación, misma que se desarrollará en el siguiente capítulo.

CAPÍTULO 4

IMPLEMENTACIÓN

CAPÍTULO 4. IMPLEMENTACIÓN

4.1 Requisitos previos a la implementación

Una vez concluido el análisis y el diseño de un sistema, se puede proceder al desarrollo e implantación del proyecto. Es importante tener planificado una serie de actividades previo a la implantación. Estas actividades corresponden al tiempo de desarrollo, planificación de tareas así como la forma en cómo se realizará dicha implementación, ya que ésta puede ser realizada por facetas, es decir, implantar algunos módulos importantes para la operación de la empresa; esto permitirá poner en funcionamiento la fracción del sistema desarrollado y seguir la operación de la empresa, mientras tanto se sigue desarrollando el resto de la aplicación. Otra forma de implantación es realizarla en forma global que contempla todos los requerimientos del cliente.

La planificación de las tareas, permiten establecer las metas y objetivos conforme a los recursos con los que se cuenta, así como las habilidades que se poseen para llevar a cabo las actividades.

Tomando en cuenta todas las tareas que se deben de realizar hasta la implantación del sistema, se identifican las siguientes:

Tareas	Recursos	Días
Análisis	2	25
Diseño	3	20
Codificación	4	28
Pruebas	2	15
Capacitación	1	5
Implantación	2	7
Documentación	1	14
Total	15	114

Tabla de tareas y tiempos

Cuando el sistema que se desarrolla es una aplicación de venta en línea, se suele tener como tarea primordial el funcionamiento y puesta en marcha del módulo de venta de artículos, y posteriormente el módulo de explotación de información de la base de datos, aunque debe hacerse notar que depende mucho esta cuestión de las necesidades de cada empresa y las decisiones que se tomen, ya que en varios casos similares, es preferible contar con la implantación del sistema completo.

Las subtareas correspondientes a cada etapa se presentan a continuación:

Tarea Análisis

		Días																						
No,	Tarea	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	Planeación de la etapa																									
2	Recopilación de datos																									
3	Realización de entrevistas																									
4	Revisión de los resultados																									
5	Determinación de los requerimientos																									
6	Determinación de procesos																									
7	Descripción de cada proceso identificado																									

Tabla de tiempos para el análisis

Tarea Diseño

		Días																		
	Tarea	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Diseño casos de uso																				
2	Diseño de Reglas de Negocio																				
3	Diseño del modelo de base de datos																				
4	Diseño de la interfaz																				

Tabla de tiempos para el Diseño

Tarea Codificación

		Días					
	Tarea	5	10	15	20	24	28
1	Creación de scripts de base de datos						
2	Codificación de la interfaz						
3	Codificación de clases						
4	Codificación de las reglas de negocio						
5	Codificación módulo de ventas						
6	Codificación módulo administrativo						

Tabla de tiempos para la Codificación

Tarea Capacitación

		Días				
	Tarea	1	2	3	4	5
1	Capacitación al administrador					
2	Capacitación a usuario					

Tabla de tiempos para Capacitación

Tarea Implantación

		Días	
	Tarea	1	2
1	Instalación de Hardware		
2	Instalación de Sistema Operativo		
3	Instalación de JVM		
4	Instalación de SGBD		
5	Instalación del Contenedor Web		
6	Instalación de la aplicación		

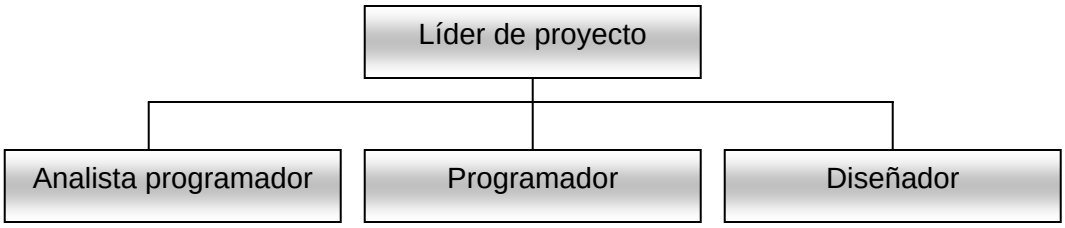
Tabla de Tiempos para la implantación

Tarea Documentación

		Días						
	Tarea	1	2	3	4	5	6	7
1	Manual de programador							
2	Manual de Usuario							

4.2 Equipo de trabajo para el desarrollo e implementación

El equipo de trabajo para el desarrollo del sistema debe estar constituido por el siguiente esquema:



Este esquipo se considera normalmente para el desarrollo de sistemas, pueden variar en cantidad de acuerdo a la magnitud del proyecto. En algunas ocasiones se requiere de un desarrollador de bases de datos, para desarrollar a nivel del SMDB (triggers, stored procedures, vistas, etc).

4.3 Implantación

La implantación se llevará a cabo después que se han cumplido las fases de prueba internas, el sistema estará listo para que sea puesto a prueba por el cliente, permitiendo detectar posibles errores, mismos que deberán corregirse y reevaluarse, hasta que se cumpla con el requerimiento del usuario.

Para poder instalar la aplicación, se requieren características de hardware mínimo el cual se lista a continuación:

- Procesador Pentium IV a 2.4 Ghz.
- 1Gb RAM.
- 80 GB disco duro.
- Monitor a color SVGA.
- Tarjeta de Red.
- Unidad de CD.
- Unidad de respaldo de CD.
- Mouse.
- Teclado.
- No break.

Cabe mencionar que el hardware citado anteriormente corresponde a requerimientos mínimos, y se puede tener otra configuración equivalente de acuerdo al fabricante del procesador.

Después que se cumplan con los requerimientos de hardware, debe instalarse el sistema operativo, y el software restante, el requerimiento es el siguiente:

- Sistema operativo: Linux Red Had 7.3
- Java 2 Standard Edition.
- Gestor de base de datos: MySql.
- Web Container: Apache Tomcat.
- Navegador de Internet: Internet Explorer o Netscape Navigator.

4.4 Instalación de los prerrequisitos

4.4.1 Instalación de Java 2 Standard Edition

1. Establecer permisos de ejecución al archivo: chmod +x jdk-1_5_0-linux-i586.bin.
2. Cambiar de directorio: cd /usr/local/jdk-1_5_0.
3. Ejecutar ./jdk-1_5_0-linux-i586.bin, Se mostrará la licencia y algunos términos de uso.
4. Para configurar las variables de entorno, se debe editar el archivo .bash_profile del usuario, por ejemplo: /home/apacsa/.bash_profile.
5. JAVA_HOME=/home/apacsa/jdk1.5.0
6. PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
7. export PATH JAVA_HOME

4.4.2 Instalación de MySQL

1. Desempaquetar el archivo: gunzip -dfv mysql-4-1-1.tar.gz.

2. Instalar la aplicación mediante el comando: tar -xvf mysql-4-1-1.tar

3. Para crear los permisos de la base de datos por default, se usa: scripts/mysql_install_db

4. La creación de la base de datos se realiza mediante: mysqladmin -u root -pmiclave create apcsa.

5. La forma más rápida de arrancar el servidor es ejecutando: mysql.server Stop y para parar el servicio: mysql.server stop

4.4.3 Instalación de Tomcat

1. Descomprimir el archivo *Tar* de Tomcat en /usr/local/, esto genera un directorio llamado jakarta-tomcat-<numero_de_version>,para dar mayor uniformidad se recomienda cambiar el nombre de este directorio a tomcat.
2. Posteriormente se debe definir una variable de ambiente la cual le indicará al sistema la ubicación de Tomcat , esta variable se llama CATALINA_HOME la cual debe ser agregada a /etc/bashrc, mediante export CATALINA_HOME=/usr/local/tomcat.
3. En el directorio bin de Tomcat (/usr/local/tomcat/bin) ejecutar el archivo startup.sh para iniciar el servicio.
4. El Testeo de la instalación se realiza desde el navegador web, mediante la URL: http://localhost:8080/manager/html, deberá cargarse la página de administrador de aplicaciones como la siguiente:

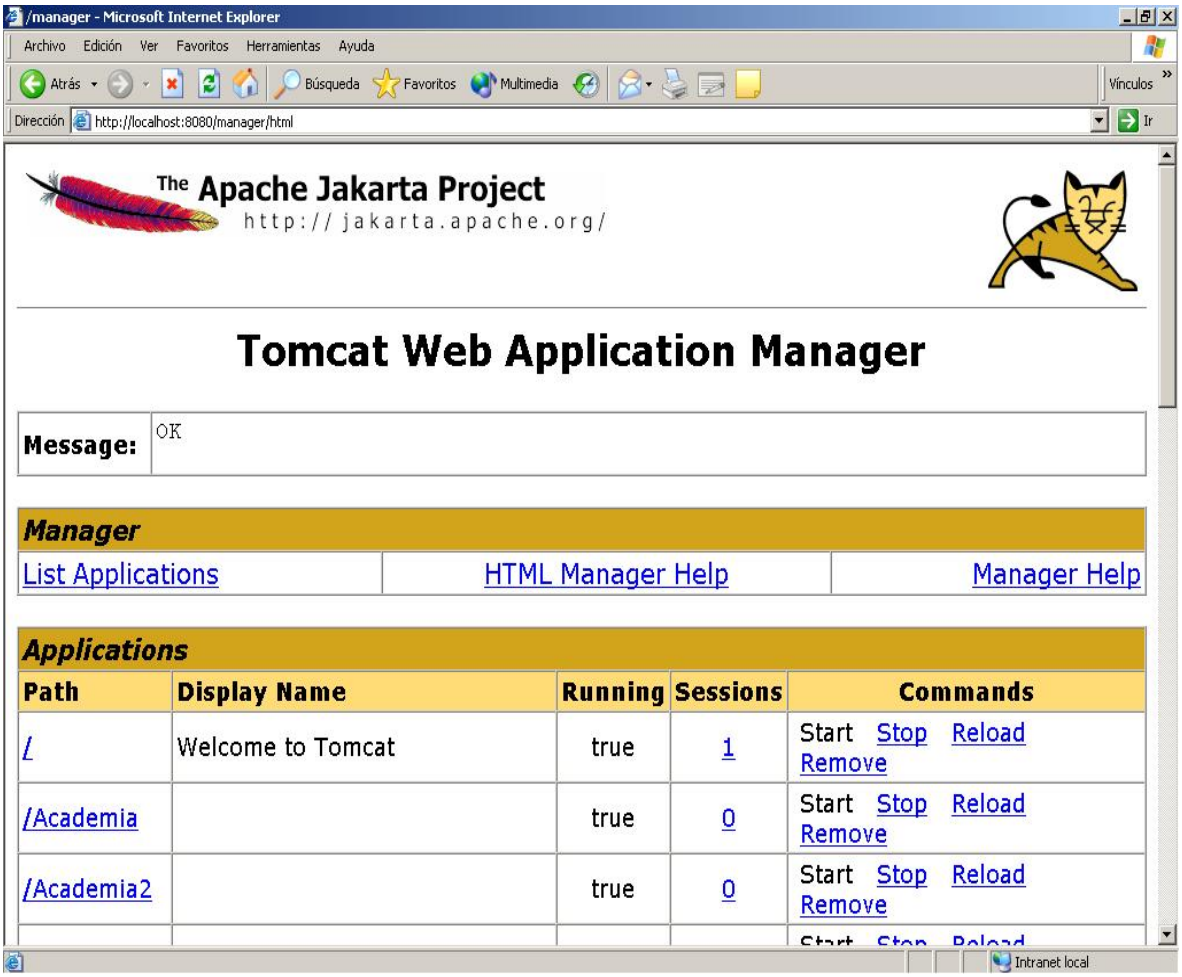


Figura 26. Administrador de Aplicaciones de Tomcat

4.5 Instalación de la Aplicación

Una vez iniciados los servicios de la base de datos, se usa el siguiente comando shell> mysql -h host -u user -p < apacsa.sql, para crear la base de datos

El contenido de apacsa.sql es:

```
CREATE TABLE almacen (
     alm_folio          INTEGER NOT NULL,
     prd_id             INTEGER NOT NULL,
     cat_id             INTEGER NOT NULL,
     alm_prd_cantidad     INTEGER NOT NULL,
     alm_prd_edo         VARCHAR(15) NOT NULL,
     alm_prd_entrFech     DATE NOT NULL,
     alm_prd_entrDesc     DATE NOT NULL,
     alm_prd_salidFech    DATE NOT NULL,
     alm_prd_salidDesc    DATE NOT NULL,
     alm_prd_conceptMov   VARCHAR(20) NULL
);
CREATE UNIQUE INDEX XPKalmacen ON almacen
(
     alm_folio                 ASC,
     prd_id                    ASC,
     cat_id                    ASC
);
ALTER TABLE almacen
     ADD  ( PRIMARY KEY (alm_folio, prd_id, cat_id) ) ;
CREATE TABLE cargo (
     idCargo            INTEGER NOT NULL,
     carg_nombre         VARCHAR2(40) NULL,
     carg_desc          VARCHAR2(250) NULL
);
CREATE UNIQUE INDEX XPKcargo ON cargo
(
     idCargo                   ASC
);
ALTER TABLE cargo
     ADD  ( PRIMARY KEY (idCargo) ) ;
CREATE TABLE categoria (
     cat_id             INTEGER NOT NULL,
     cad_codigo          VARCHAR2(20) NULL,
     cat_nombre          VARCHAR(40) NULL,
     cat_desc           VARCHAR(250) NOT NULL,
     cat_imgPath         VARCHAR(255) NULL
);
CREATE UNIQUE INDEX XPKcategoria ON categoria
(
     cat_id                    ASC
);
ALTER TABLE categoria
     ADD  ( PRIMARY KEY (cat_id) ) ;
CREATE TABLE chofer (
     chof_id            INTEGER NOT NULL,
     chof_nombre         VARCHAR(30) NULL,
     chof_apPaterno       VARCHAR(30) NULL,
     chof_apMaterno       VARCHAR(30) NULL,
     chof_fechNacimiento  DATE NULL
);
```

```
CREATE UNIQUE INDEX XPKchofer ON chofer
(
    chof_id                    ASC
);
ALTER TABLE chofer
    ADD  ( PRIMARY KEY (chof_id) ) ;
CREATE TABLE cliente (
    idCliente          INTEGER NOT NULL,
    login              INTEGER NULL,
    cli_sexo           VARCHAR(10) NULL,
    cli_RFC            VARCHAR(13) NULL,
    cli_calle          VARCHAR(30) NULL,
    cli_noInterior     VARCHAR(5) NULL,
    cli_noExterior     VARCHAR(5) NULL,
    cli_colonia        VARCHAR(30) NULL,
    cli_municipio      VARCHAR(30) NULL,
    cli_ciudad         VARCHAR(30) NULL,
    cli_estado         VARCHAR(30) NULL,
    cli_pais           VARCHAR(30) NULL,
    cli_codigostl      INTEGER NULL,
    cli_tel1           VARCHAR(22) NULL,
    cli_tel2           VARCHAR(22) NULL,
    cli_tel3           VARCHAR(22) NULL,
    cli_rep_Legal      VARCHAR(60) NULL,
    cli_tipoPersona    CHAR(1) NULL
);
CREATE UNIQUE INDEX XPKcliente ON cliente
(
    idCliente                  ASC
);
ALTER TABLE cliente
    ADD  ( PRIMARY KEY (idCliente) ) ;
CREATE TABLE cuentaBanc (
    noCuenta           INTEGER NOT NULL,
    banco              VARCHAR(20) NULL
);
CREATE UNIQUE INDEX XPKcuentaBanc ON cuentaBanc
(
    noCuenta                   ASC
);
ALTER TABLE cuentaBanc
    ADD  ( PRIMARY KEY (noCuenta) ) ;
CREATE TABLE departamento (
    idDepartamento     INTEGER NOT NULL,
    dep_nombre         VARCHAR2(40) NULL,
    dep_descripcion    VARCHAR2(250) NULL
);
CREATE UNIQUE INDEX XPKdepartamento ON departamento
(
    idDepartamento             ASC
);
```

```
ALTER TABLE departamento
    ADD  ( PRIMARY KEY (idDepartamento) ) ;
CREATE TABLE embarque (
    emb_id            INTEGER NOT NULL,
    chof_id           INTEGER NOT NULL,
    ped_folio         INTEGER NOT NULL,
    idCliente         INTEGER NOT NULL,
    emb_fechEntreg    DATE NULL,
    emb_obs           VARCHAR(255) NULL,
    emb_sts           VARCHAR(20) NULL
);
CREATE UNIQUE INDEX XPKembarque ON embarque
(
    emb_id                ASC,
    chof_id               ASC,
    ped_folio             ASC,
    idCliente             ASC
);
ALTER TABLE embarque
    ADD  ( PRIMARY KEY (emb_id, chof_id, ped_folio, idCliente) ) ;
CREATE TABLE empleado (
    idEmpleado        INTEGER NOT NULL,
    idDepartamento    INTEGER NOT NULL,
    idCargo           INTEGER NOT NULL,
    login             INTEGER NULL
);
CREATE UNIQUE INDEX XPKempleado ON empleado
(
    idEmpleado            ASC,
    idDepartamento        ASC,
    idCargo               ASC
);
ALTER TABLE empleado
    ADD  ( PRIMARY KEY (idEmpleado, idDepartamento, idCargo) ) ;
CREATE TABLE FleteTarifa (
    id_flete          INTEGER NOT NULL,
    cat_id            INTEGER NOT NULL,
    prd_id            VARCHAR(10) NOT NULL,
    fle_ciudad        VARCHAR(255) NULL,
    fle_tarifa        DECIMAL(9,2) NULL,
    fle_cantidadMin   INTEGER NULL,
    fle_prdAdicional  DECIMAL(9,2) NULL,
    fle_sts           VARCHAR(20) NULL
);
CREATE UNIQUE INDEX XPKFleteTarifa ON FleteTarifa
(
    id_flete              ASC,
    cat_id                ASC,
    prd_id                ASC
);
ALTER TABLE FleteTarifa
```

```
    ADD  ( PRIMARY KEY (id_flete, cat_id, prd_id) ) ;

CREATE TABLE foliador (
    fol_tipo          VARCHAR(10) NOT NULL,
    folio             INTEGER NOT NULL
);
CREATE UNIQUE INDEX XPKfoliador ON foliador
(
    fol_tipo                  ASC
);
ALTER TABLE foliador
    ADD  ( PRIMARY KEY (fol_tipo) ) ;
CREATE TABLE inventario (
    prd_id            INTEGER NOT NULL,
    cat_id            INTEGER NOT NULL,
    inv_existencia    INTEGER NULL
);
CREATE UNIQUE INDEX XPKinventario ON inventario
(
    prd_id                    ASC,
    cat_id                    ASC
);
ALTER TABLE inventario
    ADD  ( PRIMARY KEY (prd_id, cat_id) ) ;
CREATE TABLE login (
    id_login          INTEGER NOT NULL,
    nomCuenta         VARCHAR2(40) NULL,
    passwd            VARCHAR(10) NOT NULL,
    email             VARCHAR2(40) NULL,
    pregSecreta       VARCHAR(255) NULL,
    respSecreta       VARCHAR(255) NULL,
    usr_nombre        VARCHAR2(20) NOT NULL,
    usr_apPaterno     VARCHAR2(20) NULL,
    usr_apMaterno     VARCHAR2(20) NULL,
    log_interfazTpo   SMALLINT NOT NULL,
    usr_sts           VARCHAR(10) NULL
);
CREATE UNIQUE INDEX XPKlogin ON login
(
    id_login                  ASC
);
ALTER TABLE login
    ADD  ( PRIMARY KEY (id_login) ) ;
CREATE TABLE menuusr (
    modulo_id         INTEGER NOT NULL,
    operac_id         INTEGER NOT NULL,
    login             INTEGER NOT NULL
);
CREATE UNIQUE INDEX XPKmenuusr ON menuusr
(
    modulo_id                 ASC,
    operac_id                 ASC,
    login                     ASC
```

```
);

ALTER TABLE menuusr
    ADD  ( PRIMARY KEY (modulo_id, operac_id, login) ) ;
CREATE TABLE modulo (
    modulo_id          INTEGER NOT NULL,
    mod_desc           VARCHAR(25) NOT NULL
);
CREATE UNIQUE INDEX XPKmodulo ON modulo
(
    modulo_id                ASC
);
ALTER TABLE modulo
    ADD  ( PRIMARY KEY (modulo_id) ) ;
CREATE TABLE operacion (
    operac_id          INTEGER NOT NULL,
    opd_desc           VARCHAR(25) NOT NULL
);
CREATE UNIQUE INDEX XPKoperacion ON operacion
(
    operac_id                ASC
);
ALTER TABLE operacion
    ADD  ( PRIMARY KEY (operac_id) ) ;
CREATE TABLE pago (
    ped_folio          INTEGER NOT NULL,
    t_pago_id          SMALLINT NOT NULL,
    noCuenta           INTEGER NOT NULL,
    idCliente          INTEGER NOT NULL,
    pag_fecha          DATE NULL,
    pag_total          DECIMAL(9,2) NULL,
    no_transferencia   INTEGER NULL,
    sucursal           VARCHAR(30) NULL
);
CREATE UNIQUE INDEX XPKpago ON pago
(
    ped_folio                ASC,
    t_pago_id                ASC,
    noCuenta                 ASC,
    idCliente                ASC
);
ALTER TABLE pago
    ADD  ( PRIMARY KEY (ped_folio, t_pago_id, noCuenta, idCliente) ) ;
CREATE TABLE pedido (
    ped_folio          INTEGER NOT NULL,
    idCliente          INTEGER NOT NULL,
    ped_fecha          DATE NOT NULL,
    entrg_calle        VARCHAR(30) NULL,
    entrg_noInterior   VARCHAR(5) NULL,
    entrg_exterior     VARCHAR(5) NULL,
    entreg_ciudad      VARCHAR(30) NULL,
    entrg_colonia      VARCHAR(30) NULL,
    entrg_municipio    VARCHAR(30) NULL,
```

```
    entrg_estado        VARCHAR(30) NULL,
    entreg_pais         VARCHAR(30) NULL,
    entreg_codPostal    INTEGER NULL,
    fact_nombre         VARCHAR(30) NULL,
    fact_apPaterno      VARCHAR(30) NULL,
    fact_apMaterno      VARCHAR(30) NULL,
    fact_RFC            VARCHAR(13) NULL,
    fact_calle          VARCHAR(30) NULL,
    fact_noInterior     VARCHAR(5) NULL,
    fact_noExterior     VARCHAR(5) NULL,
    fact_colonia        VARCHAR(30) NULL,
    fact_ciudad         VARCHAR(30) NULL,
    fact_municipio      VARCHAR(30) NULL,
    fact_estado         VARCHAR(30) NULL,
    fact_pais           VARCHAR(30) NULL,
    fact_codPostal      INTEGER NULL,
    iva_pctje           DECIMAL(4,2) NULL,
    total_flete         DECIMAL(7,2) NULL,
    subTotal            DECIMAL(9,2) NULL,
    granTotal           DECIMAL(9,2) NULL,
    ped_sts             VARCHAR(20) NULL
);
CREATE UNIQUE INDEX XPKpedido ON pedido
(
    ped_folio                ASC,
    idCliente                ASC
);
ALTER TABLE pedido
    ADD  ( PRIMARY KEY (ped_folio, idCliente) ) ;
CREATE TABLE pedidoProducto (
    pedProd_Id          INTEGER NOT NULL,
    ped_folio           INTEGER NOT NULL,
    prd_id              VARCHAR(10) NOT NULL,
    cat_id              INTEGER NOT NULL,
    idCliente           INTEGER NOT NULL,
    ped_cantidad        DECIMAL(9,2) NULL,
    ped_precio          DECIMAL(7,2) NULL
);
CREATE UNIQUE INDEX XPKpedidoProducto ON pedidoProducto
(
    pedProd_Id               ASC,
    ped_folio                ASC,
    prd_id                   ASC,
    cat_id                   ASC,
    idCliente                ASC
);
ALTER TABLE pedidoProducto
    ADD  ( PRIMARY KEY (pedProd_Id, ped_folio, prd_id, cat_id,
        idCliente) ) ;
CREATE TABLE producto (
    prd_id              INTEGER NOT NULL,
    cat_id              INTEGER NOT NULL,
    prd_codigo          VARCHAR2(20) NULL,
```

```
    prd_nombre          VARCHAR(50) NULL,
    prd_capacitad_ml    INTEGER NULL,
    prd_ingredientes    VARCHAR(255) NULL,
    prd_peso_Kg         DECIMAL(5,2) NULL,
    prd_dimension_CM    VARCHAR(25) NULL,
    prd_presentacion_fisic VARCHAR(20) NULL,
    prd_desc            VARCHAR(255) NULL,
    prd_color           VARCHAR(20) NULL,
    prec_menudeo        DECIMAL(7,2) NULL,
    prec_mayoreo        DECIMAL(7,2) NULL,
    prec_cantidadMayoreo INTEGER NULL,
    prd_status          VARCHAR(20) NULL,
    prd_imgPath         VARCHAR(255) NULL
);
CREATE UNIQUE INDEX XPKproducto ON producto
(
    prd_id                  ASC,
    cat_id                  ASC
);
ALTER TABLE producto
    ADD  ( PRIMARY KEY (prd_id, cat_id) ) ;
CREATE TABLE tipoPago (
    t_pago_id           SMALLINT NOT NULL,
    t_pago_desc         VARCHAR(30) NOT NULL
);
CREATE UNIQUE INDEX XPKtipoPago ON tipoPago
(
    t_pago_id               ASC
);
ALTER TABLE tipoPago
    ADD  ( PRIMARY KEY (t_pago_id) ) ;
CREATE TABLE tpagoBanco (
    t_pago_id           SMALLINT NOT NULL,
    noCuenta            INTEGER NOT NULL,
    tb_sts              VARCHAR(10) NULL
);
CREATE UNIQUE INDEX XPKtpagoBanco ON tpagoBanco
(
    t_pago_id               ASC,
    noCuenta                ASC
);
ALTER TABLE tpagoBanco
    ADD  ( PRIMARY KEY (t_pago_id, noCuenta) ) ;
CREATE TABLE venta (
    vta_id              INTEGER NOT NULL,
    ped_folio           INTEGER NOT NULL,
    idCliente           INTEGER NOT NULL,
    vta_fch_cargo       DATE NULL,
    vta_fch_abono       DATE NULL,
    vta_cargo           DECIMAL(9,2) NULL,
    vta_abono           DECIMAL(9,2) NULL
);
CREATE UNIQUE INDEX XPKventa ON venta
```

```
(
    vta_id                  ASC,
    ped_folio               ASC,
    idCliente               ASC
);
ALTER TABLE venta
    ADD  ( PRIMARY KEY (vta_id, ped_folio, idCliente) ) ;
ALTER TABLE almacen
    ADD  ( FOREIGN KEY (prd_id, cat_id)
                REFERENCES producto ) ;
ALTER TABLE cliente
    ADD  ( FOREIGN KEY (login)
                REFERENCES login ) ;
ALTER TABLE embarque
    ADD  ( FOREIGN KEY (chof_id)
                REFERENCES chofer ) ;
ALTER TABLE embarque
    ADD  ( FOREIGN KEY (ped_folio, idCliente)
                REFERENCES pedido ) ;
ALTER TABLE empleado
    ADD  ( FOREIGN KEY (idCargo)
                REFERENCES cargo ) ;
ALTER TABLE empleado
    ADD  ( FOREIGN KEY (idDepartamento)
                REFERENCES departamento ) ;
ALTER TABLE empleado
    ADD  ( FOREIGN KEY (login)
                REFERENCES login ) ;
ALTER TABLE FleteTarifa
    ADD  ( FOREIGN KEY (prd_id, cat_id)
                REFERENCES producto ) ;
ALTER TABLE inventario
    ADD  ( FOREIGN KEY (prd_id, cat_id)
               REFERENCES producto ) ;
ALTER TABLE menuusr
    ADD  ( FOREIGN KEY (login)
                REFERENCES login ) ;
ALTER TABLE menuusr
    ADD  ( FOREIGN KEY (operac_id)
                REFERENCES operacion ) ;
ALTER TABLE menuusr
    ADD  ( FOREIGN KEY (modulo_id)
                REFERENCES modulo ) ;
ALTER TABLE pago
    ADD  ( FOREIGN KEY (t_pago_id, noCuenta)
                REFERENCES tpagoBanco ) ;
ALTER TABLE pago
    ADD  ( FOREIGN KEY (ped_folio, idCliente)
                REFERENCES pedido ) ;
ALTER TABLE pedido
    ADD  ( FOREIGN KEY (idCliente)
                REFERENCES cliente ) ;
ALTER TABLE pedidoProducto
```

```
    ADD  ( FOREIGN KEY (ped_folio, idCliente)
                REFERENCES pedido ) ;
ALTER TABLE pedidoProducto
    ADD  ( FOREIGN KEY (prd_id, cat_id)
               REFERENCES producto ) ;
ALTER TABLE producto
    ADD  ( FOREIGN KEY (cat_id)
                REFERENCES categoria ) ;
ALTER TABLE tpagoBanco
    ADD  ( FOREIGN KEY (noCuenta)
                REFERENCES cuentaBanc ) ;
ALTER TABLE tpagoBanco
    ADD  ( FOREIGN KEY (t_pago_id)
                REFERENCES tipoPago ) ;
ALTER TABLE venta
    ADD  ( FOREIGN KEY (ped_folio, idCliente)
                REFERENCES pedido ) ;
```

Para la instalación de la aplicación se requiere cargar en el navegador Web, la página de administración de Tomcat.

En la sección de cargar un archivo WAR, se debe seleccionar el archivo web que contiene empaquetada la aplicación.

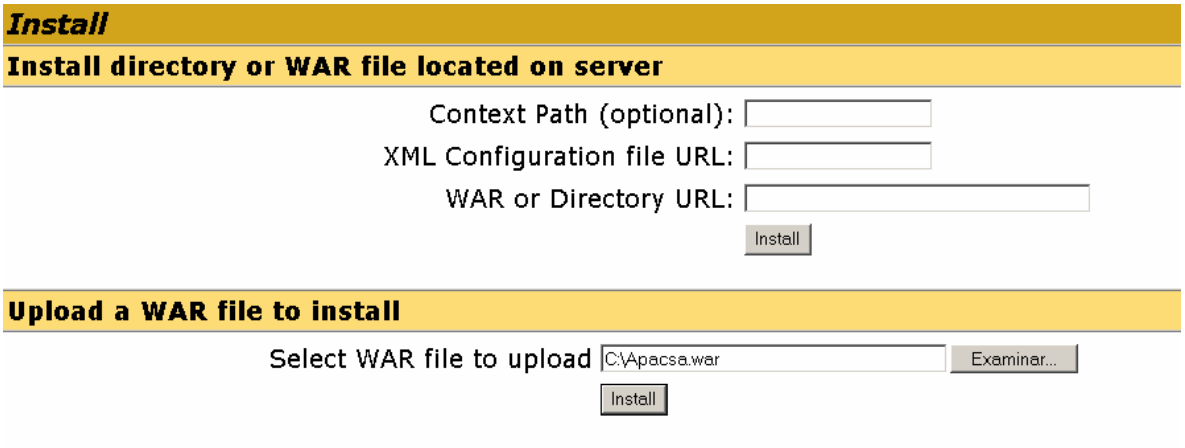


Figura 27. Instalar la aplicación WAR

Después de pulsar install, el nombre de la aplicación se encontrará en la lista de la página, esto indica que la aplicación ha sido instalada.

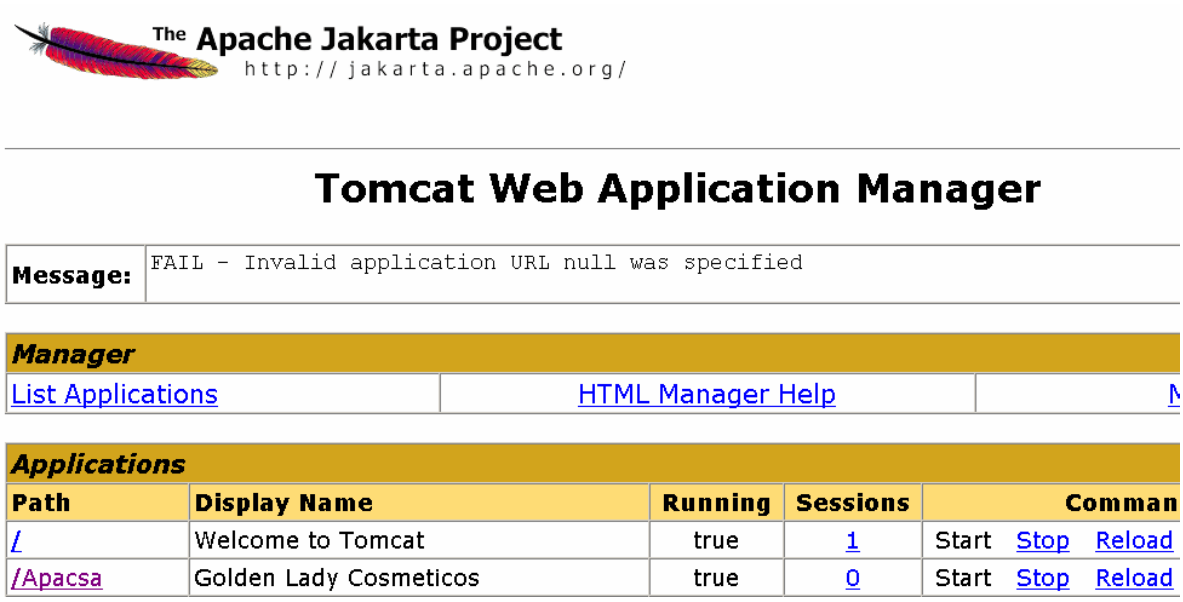


Figura 28. Aplicación instalada

4.6 Ejemplo de uso

Cargar la página de inicio mediante la URL http://localhost:8080/apacsa/index.jsp.

En la página dar clic en el link ENTRAR.

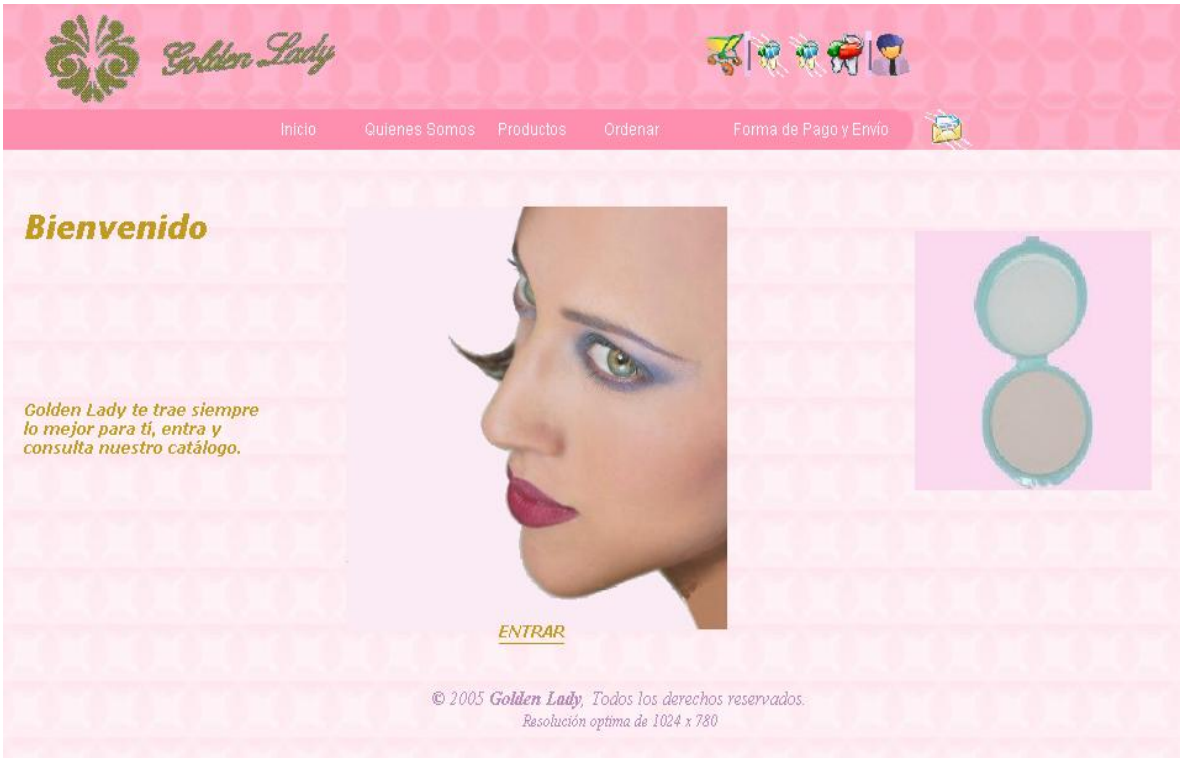


Figura 29. Pantalla de Inicio

Al dar clic en el link entrar, mostrará las categorías existentes como se muestra en la siguiente pantalla:

Figura 30. Pantalla listado de categorías

Al seleccionar un producto, se presenta la descripción general.

Figura 31. Pantalla Subcategorías

Al seleccionar un artículo, se observa una descripción mas detallada.

Figura 32. Pantalla detalle de artículo

Al dar clic en el icono Añadir producto, se guardará en memoria el artículo seleccionado para seguir añadiendo productos a la canasta de compras.

Figura 33. Pantalla carrito de compras

Si el cliente no se ha registrado en el sistema, éste le pedirá sus datos a través del siguiente formulario:

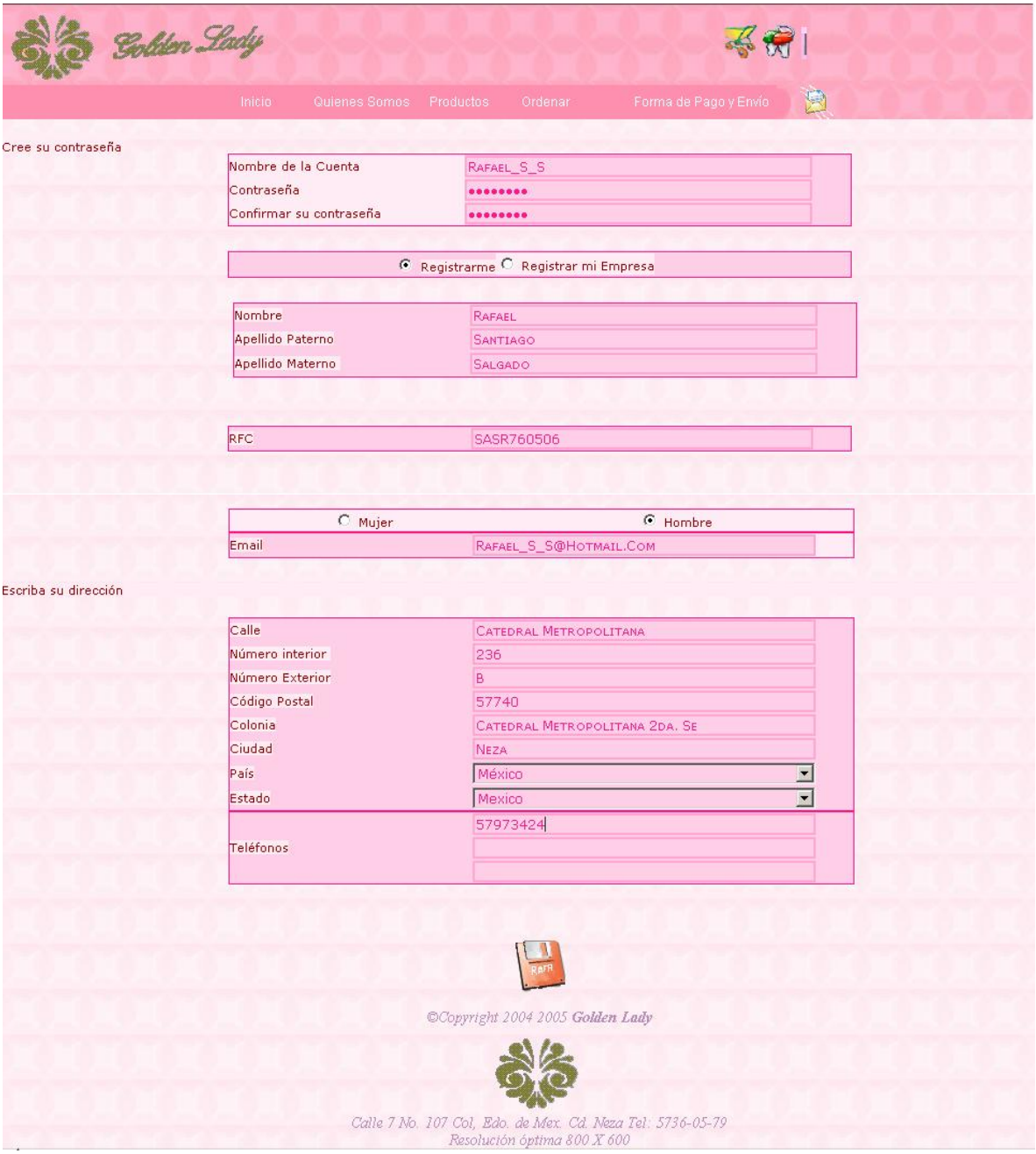


Figura 34. Pantalla Captura datos de cliente

Inmediatamente después se despliega la de la Orden de Pedido como en la siguiente figura:

Figura 35. Pantalla Orden de pedido

Posteriormente se presenta la pantalla que captura la forma de pago.

Figura 36. Pantalla Forma de pago

El sistema cuenta con un módulo de administración por parte del usuario, se encuentra personalizado de acuerdo al perfil autenticado.

Figura 37. Pantalla administración del sistema

SIWEBCOV permite la emisión de reportes, tal como el siguiente:

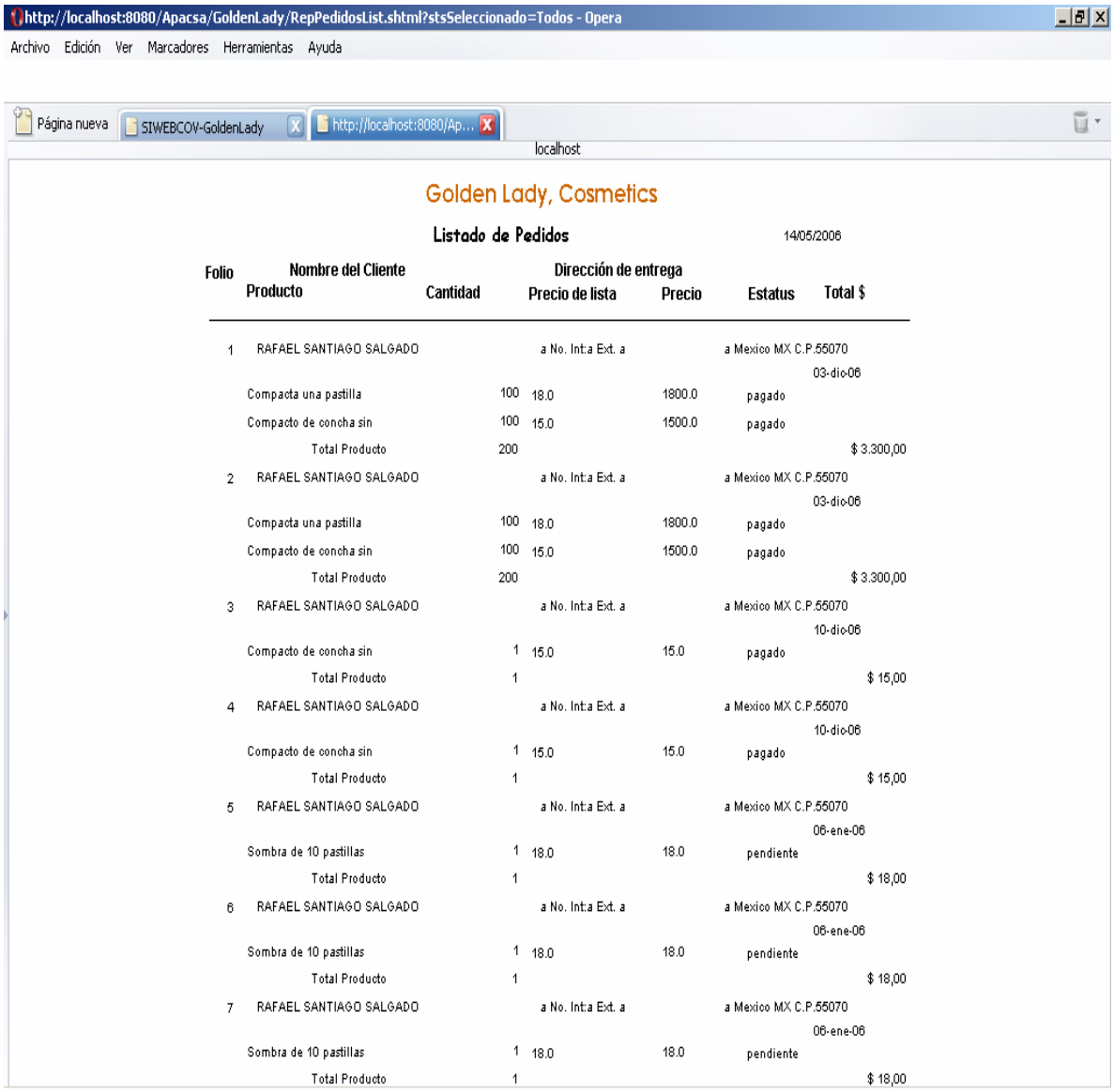


Figura 38. Pantalla Reporte de pedidos.

4.7 Control de calidad

Con el propósito de proporcionar un mejor servicio de calidad al cliente, es recomendable proporcionar una serie cuestionarios que permitan evaluar el servicio brindado y obtener de esta manera puntos de oportunidad de mejora, uno de los cuales se presenta a continuación.

La presente encuesta tiene como objetivo conocer el desarrollo, las condiciones y los resultados de la solicitud recibida, para identificar los factores que incidan en un mejor servicio.

SIWEBCOV

SISTEMA WEB PARA CONTROL DE VENTAS

ENCUESTA DE SERVICIOS

FF / MM / AAAA

Fecha: / /

Nombre: _____

	CON RELACIÓN AL PERSONAL E = Excelente, B = Bueno, R = Regular, M = Malo	Respuestas			
		E	B	R	M
1	¿Cómo califica al tiempo que se le asignó al desarrollo del proyecto?				
2	El nivel de conocimientos de la persona que atendió su solicitud fue				
3	¿Cómo evalúa la actitud de la persona que le atendió?				
4	¿Cómo califica la disponibilidad para atender sus dudas o comentarios?				
5	¿Cómo fue la comunicación con Usted para informarlo del avance del proyecto?				
6	¿Cómo califica de forma general la atención y solución de su solicitud?				

	CON RELACIÓN AL PERSONAL Responda según corresponda a la pregunta S =SI, N = NO ó E = Excelente, B = Bueno, R = Regular, M = Malo	Respuestas S		N	
		E	B	R	M
1	¿Cuál es el nivel de cumplimiento del objetivo de su solicitud, de acuerdo a lo solicitado?				
2	Califique el grado de cumplimiento de la fecha de entrega de la solución de su solicitud				
3	¿Se le entregó un documento de plan de trabajo sobre su requerimiento?				
4	Califique el grado de cobertura del documento de Plan de Trabajo de se requerimiento				
5	¿Cuál fue el nivel de claridad del Documento de Plan de Trabajo que se entregó para la solución de su requerimiento?				
6	Sí hubo modificaciones al documento original y se rechazó, ¿Recibió un nuevo documento actualizado con los cambios?				
7	¿Recibió la capacitación requerida?				
8	¿Cómo califica el manual y/o capacitación requerida?				
9	¿Se entregó la documentación necesaria para entender y operar la nueva funcionalidad solicitada?				
10	¿El requerimiento representó fallas al momento de empezar a utilizarlo en producción?				
11	¿Se atendieron con prontitud las fallas reportadas?				
12	¿Se recibió alguna documentación sobre la solución realizada para la falla reportada?				

Comentarios Adicionales

Gracias por su Colaboración.

CONCLUSIONES

Conclusiones

El diplomado de desarrollo de sistemas en Web representó una fuente de información importante, ya que a través de este curso, fue posible capacitar y actualizar al asistente.

El estudio de las diferentes temáticas como; fundamentos para el desarrollo orientado al Web, permitieron dar a conocer los mecanismos sobre los cuales esta soportada la tecnología de Internet, tal como, los protocolos que se usan para ofrecer servicios, y permitir ejecución de diferentes aplicaciones.

La introducción de UML al diplomado de desarrollo de sistemas en Web, permitió mostrar las técnicas que se usan durante la arquitectura de un software, permitiendo analizarlo y modelarlo, plasmando gráficamente su comportamiento.

Se analizó y practicó de forma clara y concisa el lenguaje java, primeramente su sintaxis básica, seguida de las bases para la conexión con un SMBD mostrando como front páginas Web.

La exposición de la arquitectura conceptual como el uso de un patrón de diseño para el desarrollo de aplicaciones resultó una temática de vital importancia, en la actualidad Struts es uno de los patrones mas usados junto a un mapeador objeto-relacional como Hibernate, Castor o Ibatis, ya que el código se vuelve reutilizable y el desarrollo se separa en tres capas. De esta manera es posible tener a un equipo de desarrollo enfocado a diferentes actividades en forma paralela: desarrollo de la interfaz (html, jsp, javascript, etc), desarrollo de las reglas de negocio (validaciones y reglas que modelan la aplicación), desarrollo de bases de datos (desarrollo de stored procedures, triggers, constraints, etc).

El diseño del front-end resultó interesante, proporcionado al alumno consejos, reglas y utilidades para la creación y diseño armónico de los elementos en una página de Internet.

El curso otorgó al asistente conocimientos para el desarrollo de bases de datos, tanto de Data Definition Languaje como el Data Model Languaje del Structured Query Languaje, y lecciones de triggers en DB2, dándolos a conocer mediante la

práctica se obtuvieron y reforzaron los conocimientos para el diseño y desarrollo de una base de datos.

Los objetivos fueron alcanzados al proporcionar soporte para el análisis, diseño e implantación de soluciones Web, con las herramientas otorgadas por el curso anteriormente mencionado el asistente no solo se convierte en un programador en java, si no un arquitecto de sistemas.

Debe hacerse notar que existen otros frameworks como Oracle Application Development Framework permite crear aplicaciones Web combinando tecnologías de Enterprise Java Beans para el uso de sesiones, jspx como front-end y el mapeador objeto-relacional Toplink, implementa también el patrón Model View Controller ya sea usando el clásico Struts o JavaServer Faces. El entorno de desarrollo integrado Oracle JDeveloper 10g, permite crear aplicaciones rápidamente con la ayuda de sus wizards, y unos cuantos clics obtener la lista de empleados sin introducir código alguno. Permitiendo el ahorro de tiempo y la aceleración del desarrollo.

La tecnología crece aceleradamente y más en el ámbito computacional, por lo que el ingeniero en computación debe actualizarse constantemente.

Se realizó un proyecto práctico utilizando las técnicas y conocimientos adquiridos en el Diplomado de Desarrollo de Sistemas en Web, obteniendo un sistema que resuelve una problemática real. El producto final esta implementado bajo el patrón MVC. El front-end se encuentra conformado de páginas JSP’s y la validación de formularios en el cliente por javascript. El controlador de la aplicación la sigue Struts que determina que página mostrar de acuerdo a los requerimientos del usuario, finalmente las reglas de negocio están soportadas por una base de datos relacional y clases de java que modelan el comportamiento del negocio. Se hace uso de Ibatis para el mapeo de objetos relacionales a objetos de java, permitiendo asegurar la persistencia de información.

Desde la perspectiva de la organización, se observó mayor control en sus procesos de venta, permitiendo obtener rápidamente información fidedigna y fresca. Desde el punto de vista de los clientes de la organización; se les ofreció

una nueva opción de compra, observando detalladamente las características del producto, asegurando una compra más cómoda y rápida al realizar el pago mediante transferencias o depósitos bancarios.

La aportación del presente documento consiste en proporcionar en forma estructurada y organizada la información resumida usada para la arquitectura y desarrollo de aplicaciones basadas en Internet.

La solución a una problemática real consolidada en una aplicación de calidad, bien documentada para el crecimiento futuro de la aplicación de acuerdo a las necesidades de la empresa.

Durante el desarrollo del proyecto, se observaron algunas limitaciones tal como, la realización de la solución integral, de acuerdo a la complejidad del mismo y el tiempo requerido de desarrollo.

BIBLIOGRAFÍA

Bibliografía

Ceballos Fco. Javier; “Java 2 Curso de programación”; Alfaomega, Ra-Ma.

C. Larman. “UML y Patrones” Prentice Hall; 1999.

J. Conallen. Building Web application with UML. Addison Wesley. 1999.

Keogh Jim; “J2EE Manual de referencia”; McGraw Hill

Kendall & Kendall; “Análisis y Diseño de Sistemas”; 3ª Edición; Pearson Educación.

Mercerat Bárbara, Silva Darío Andrés ; “Construyendo Aplicaciones WEB con una Metodología de Diseño Orientada a Objetos”;2002.

Roger S. Pressman; “Ingeniería del Software”; 4ª Edición; McGraw Hill

Rodríguez Rubio Jorge; “Sistema Web para el cálculo de la ruta óptima dentro de una organización de distribución empleado algoritmos de inteligencia artificial”, Tesis 2005.

Referencias en línea

ADF Frame Work. http://www.oracle.com/technology/

UML Resource Center. Rational Software. http://www.rational.com/uml/

Casos de Uso. http://www.usecases.org

Caos de Uso. http://www.pols.co.uk/usecasezone/

IBATIS. http://www.ibatis.com

Introducción a XML http://www.ibm.com/developerWorks

Java Sun, The Java Tutorial: A practical guide for programmers, http://Java.sun.com/docs/books/tutorial, Marzo de 2001.

Toplink:

http://www.oracle.com/technology/obe/obe1013jdev/masterdetailedit_adftoplink/endtoend_toplink_adffaces.htm