9

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

PROCESO UNIFICADO APLICADO AL DESARROLLO DE UN SISTEMA DE COMERCIO ELECTRONICO CON J2EE

T E S I S

QUE PARA OBTENER EL TITULO DE:

LIC. EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

RAYMUNDO ORTEGA LEON

DIRECTOR DE TESIS: DRA. HANNA OKTABA

FACULTAD DE CIENCIAS
UNAM

MEXICO, D.F. 2002

VNIVERSIDAD NACIONAL
AVPNOMA DE
MEXICO

M. EN C. ELENA DE OTEYZA DE OTEYZA
Jefa de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunico a usted que hemos revisado el trabajo escrito:
"PROCESO UNIFICADO APLICADO AL DESARROLLO DE UN SISTEMA DE COMERCIO ELECTRÓNICO CON J2EE"
realizado por RAYMUNDO ORTEGA LEÓN

con número de cuenta 9561488-1 **, quién cubrió los créditos de la carrera de**

CIENCIAS DE LA COMPUTACIÓN

Dicho trabajo cuenta con muestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	DRA. HANNA OKTABA
Propietario	M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ
Propietario	M. EN I. MARÍA DE LUZ GASCA SOTO
Suplente	MAT. MARÍA CONCEPCIÓN ANA LUISA SOLÍS GONZÁLEZ COSÍO
Suplente	ACT. ALEJANDRO TALAVERA ROSALES

Consejo Departamental de MATEMÁTICAS

DRA. AMPARO LÓPEZ GAONA

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL DE
MATEMÁTICAS

Proceso Unificado aplicado al desarrollo de un Sistema de Comercio Electrónico con J2EE

Raymundo Ortega León
2002

Agradecimientos

Esta tesis se logró gracias al esfuerzo conjunto de gente que me apoyó de distintas formas. Agradezco el esfuerzo y confianza que me otorgaron.

A mi directora de tesis Dra. Hanna Oktaba, los sinodales M. en C. María Guadalupe Elena Ibargüengoitia González, M. en I. María de Luz Gasca Soto, Mat. María Concepción Ana Luisa Solís González Cosío, Act. Alejandro Talavera Rosales y al Ing. Germán Santos Jaimes les agradezco su apoyo académico y moral para conseguir la realización del presente trabajo. Su esfuerzo e interés por enseñar es una de las labores más nobles, reciban mi más profundo respeto y admiración. Gracias por darme de su tiempo y conocimientos.

Agradezco a todos mis amigos y en especial a los universitarios con quienes he compartido experiencias inolvidables, mil gracias pues en mi dejaron un ser lleno de vida, alegre y con ganas de superarse.

A mi familia, gracias por su apoyo, consejos, comprensión, dedicación y cariño que me han brindado. Todos ustedes han contribuido en gran parte a mi éxito, les dedico de forma especial éste trabajo, recíbanlo con el amor que les tengo.

En especial agradezco a mis padres y hermana quienes han sido mis principales compañeros en la vida, ustedes me motivan, enseñan y brindan el apoyo para ser mejor. Gracias por darme la oportunidad de vivir.

A las personas con quienes he compartido mi vida y me han enseñado a vivir, mi más sincero agradecimiento.

Dios, gracias.

Raymundo Ortega León

Indice

Introducción

A través del tiempo el comercio ha sido parte de la vida del ser humano. Hoy en día las computadoras se encuentran presentes en una gran variedad de actividades; el comercio no es la excepción. Debido a la creciente aceleración de las actividades humanas impulsadas por las computadoras, el comercio ha tenido que evolucionar y desarrollar nuevas formas para el mismo dando como resultado el comercio electrónico. El comercio electrónico permite realizar las actividades comerciales a través de medios electrónicos los cuales permiten una mayor rapidez en las transacciones comerciales, independencia geográfica de consumidores y proveedores, los compradores tienen una gran variedad de proveedores para elegir, los proveedores tienen amplitud de mercado y las relaciones entre empresas se hacen más eficientes.

El presente trabajo tiene como objetivo el desarrollo de un sistema de comercio electrónico llamado *Sistema de Ventas Café* el cual pone a la venta a través de un sitio Web una variedad de CDs, DVDs y libros. Los clientes pueden revisar en línea las características de los productos disponibles en el sitio además de contar con un carrito de compras en el cual pueden agregar y modificar aquellos que desean adquirir. Una vez que el cliente termina la selección de productos y decide comprarlos se genera un pedido. El sistema también cuenta con un módulo de administración del sitio Web el cual permite añadir, eliminar, modificar y hacer búsquedas sobre los productos además de poder marcarlos como estrenos ó quitarles dicha marca cuando ya no sean considerados como tal.

Para la implementación del sistema de comercio electrónico se escogió la plataforma *Java 2 Enterprise Edition*(J2EE). La plataforma J2EE hace uso de el modelo multicapa distribuido el cual simplifica el desarrollo, instalación y mantenimiento de las aplicaciones empresariales. Permite a los desarrolladores enfocarse sobre la lógica del negocio la cual una vez desarrollada puede ser instalada sobre los servidores apropiados para las necesidades existentes de la organización. Proporciona una infraestructura para el desarrollo de las aplicaciones del lado del cliente que permite la interacción con los usuarios. Además de estas ventajas J2EE permite crear aplicaciones con componentes estandarizados que pueden ser instalados en una gran variedad de plataformas y pueden ser usados para escalar rápidamente aplicaciones que deben adaptarse a las necesidades cambiantes del negocio.

Para producir un software de alta calidad se necesita de un proceso el cual permita eficientar el desarrollo del mismo y entregarlo de manera predecible. El proceso escogido para el desarrollo del sistema de comercio electrónico fue el Proceso Unificado. El Proceso Unificado captura algunas de las mejores prácticas del desarrollo de software en una forma ajustable a una amplia variedad de proyectos y organizaciones. El Proceso Unificado es un método iterativo que apoya un entendimiento incremental del problema a través de refinamientos y crecimiento sucesivos del software. La construcción mediante la estrategia iterativa permite adquirir una gran flexibilidad para acomodar requerimientos y cambios en estos además de poder identificar y resolver de manera temprana los riesgos que pueden hacer fracasar el proyecto. Las actividades del Proceso Unificado hacen un fuerte énfasis en la creación y mantenimiento de modelos los cuales minimizan los costos de generar y mantener documentos y maximizan el contenido relevante de la información. Para visualizar , especificar, construir y documentar los artefactos de los modelos se hace uso del lenguaje gráfico *Unified Modeling Language*(UML). UML es un lenguaje independiente del proceso y puede ser usado con una amplia variedad de estos entre los cuales se destaca el Proceso Unificado por tener un buen acoplamiento.

El trabajo para desarrollar el sistema de comercio electrónico *Sistema de Ventas Café* se dividió en dos tesis complementarias. Las tesis complementarias son "Tecnología J2EE aplicada a un Sistema de Comercio Electrónico" y "Proceso Unificado aplicado al desarrollo de un Sistema de Comercio Electrónico con J2EE" las cuales fueron desarrolladas por Griselda González Rodríguez y Raymundo Ortega León respectivamente.

La tesis "Tecnología J2EE aplicada a un Sistema de Comercio Electrónico" explica la tecnología J2EE y su uso en el desarrollo del sistema de comercio electrónico. J2EE es una plataforma novedosa que maneja un modelo multicapa distribuido y permite el desarrollo de aplicaciones seguras, de alta disponibilidad, confiables y escalables de manera sencilla y a bajo costo.

La presente tesis "Proceso Unificado aplicado al desarrollo de un Sistema de Comercio Electrónico con J2EE" contiene la aplicación del Proceso Unificado para el desarrollo del sistema de comercio electrónico cuyos modelos son representados con UML. El Proceso Unificado es un proceso de reciente aparición el cual captura algunas de las mejores prácticas del desarrollo de software y UML es un lenguaje que día con día gana más adeptos por su gran expresividad y sencilla representación de los artefactos para el modelado de sistemas. Esta tesis expone una serie de valiosas experiencias y aprendizajes surgidos de la aplicación de un nuevo proceso, el Proceso Unificado a una tecnología no menos novedosa, J2EE.

La estructura de la presente tesis "Proceso Unificado aplicado al desarrollo de un Sistema de Comercio Electrónico con J2EE" consta de cuatro capítulos y un apéndice que se describen como sigue:

- **Capítulo 1. Comercio Electrónico**: Desarrolla de manera general el tema de comercio electrónico. También incluye un marco para la identificación de distintos tipos de aplicaciones de comercio electrónico, tecnologías usadas para su desarrollo, tipos de pagos y seguridad.

- **Capítulo 2. Proceso Unificado**: Describe el Proceso Unificado. Explica las fases y flujos de trabajo en los que se divide y cómo las fases a su vez se dividen en iteraciones dentro de los ciclos de desarrollo.

- **Capítulo 3. Los modelos del Proceso Unificado**: Describe a detalle los modelos usados por el Proceso Unificado para el desarrollo del software así como la representación con UML de sus distintos artefactos. Se incluye la descripción de los modelos de casos de uso(y requerimientos suplementerios), análisis, diseño, instalación, implementación y pruebas.

- **Capítulo 4. Aplicación del Proceso Unificado al sistema de comercio electrónico**: Contiene la aplicación del Proceso Unificado al desarrollo del *Sistema de Ventas Café*. Se describe cómo se desarrolla el sistema de comercio electrónico a través de los flujos de trabajo del Proceso Unificado(requerimientos, análisis, diseño, implementación y pruebas). Se desarrollan los modelos de casos de uso(y requerimientos suplementerios), análisis, diseño, instalación, implementación y pruebas para el *Sistema de Ventas Café*.

- **Apéndice A: Diagramas complementarios para los modelos del Proceso Unificado**: Presenta los diagramas complementarios para los modelos desarrollados en el Capítulo 4.

CAPITULO 1

Comercio electrónico

Debido a la creciente aceleración de las actividades humanas impulsadas por las computadoras, el comercio, al igual que una gran variedad de ramas, ha tenido que evolucionar y desarrollar nuevas formas de realizarse dando como resultado el comercio electrónico. El comercio electrónico nos ofrece varias ventajas entre las cuales se destacan la realización con mayor rapidez de las transacciones comerciales, independencia geográfica de consumidores y proveedores, los compradores tienen una gran variedad de proveedores para elegir, los proveedores tienen amplitud de mercado y las relaciones entre empresas se hacen más eficientes.

1.1 El comercio electrónico

El comercio electrónico se define como un arreglo integrado de prácticas de negocio, procesos, configuraciones de aplicaciones técnicas y estructuras organizacionales que usan el Intercambio Electrónico de Datos (*Electronic Data Interchange*, EDI).

El comercio electrónico comienza en 1845 con la primera operación comercial del telégrafo el cual usó la codificación para caracteres. La gran disponibilidad de aplicaciones similares integradas, infraestructura y tecnologías son elementos que aún hacen al comercio electrónico popular en nuestros días. El comercio electrónico describe bien una serie de aplicaciones y tecnologías relacionadas. Los prerrequisitos para el comercio electrónico son los siguientes:

- Debe ser conducido vía comunicaciones electrónicas.
- Debe incluir proveedores.
- Debe incluir consumidores.
- Puede incluir el pago.

El comercio electrónico permite a las empresas ser más eficientes y flexibles en sus operaciones internas, trabajar más estrechamente con sus suministradores y dar mejor respuesta a las necesidades y expectativas de sus clientes. Les permite seleccionar los mejores proveedores sin tener en cuenta su localización geográfica y así vender en un mercado global.

1.2 Historia

En el siglo XIX con la introducción del telégrafo en 1845 se dio la primera oportunidad a proveedores y consumidores de conducir el comercio sin estar localizados físicamente cerca. También permitió que el intercambio de información fuera más rápido que cualquiera en su tiempo. Esta información no estaba restringida a información genérica sino también a aplicaciones específicas como la transferencia de dinero. La independencia en la localización de consumidores y proveedores y la tecnología para procesar más rápido la información fueron y siguen siendo conductores importantes del comercio electrónico . Los beneficios del comercio electrónico para el consumidor y proveedor siguen siendo los mismos:

- Amplitud para la elección.
- Amplitud de mercado.

No se debe olvidar que el comercio electrónico no solamente fue desarrollado para el trato anónimo entre compañías y personas sino también para consumidores y proveedores de diferentes divisiones en la misma compañía. Teniendo esto en mente se tienen cinco reglas que forman la base para los requerimientos y expectativas del usuario del comercio electrónico:

- El comercio electrónico soporta independencia geográfica del proveedor y del consumidor.
- El comercio electrónico permite que el comercio opere de manera más rápida.
- El comercio electrónico maneja aplicaciones inter-empresariales e intra-empresariales.

- El comercio electrónico también debería soportar necesidades interactivas.

  Con la llegada del teléfono en 1875 el comercio electrónico se aceleró aún más ya que permitió realizar transacciones como parte de la misma sesión interactiva, necesidad que permanece hasta ahora. Aún las aplicaciones EDI más sofisticadas necesitan de la voz interactiva.

- El comercio electrónico permite la reingeniería del proceso de negocio para conseguir eficiencia.

  Hasta 1950 el telégrafo y el teléfono fueron las únicas formas de comercio electrónico, es entonces cuando llega el cómputo comercial. Los primeros mecanismos computarizados para comercio electrónico hicieron más eficientes los procesos administrativos de oficina. Entre 1950 y 1960 estos sistemas fueron basados en *mainframe* con usuarios conectados directamente con ligas hacia el *host*. Tales sistemas no iban dirigidos directamente al público comprador pero sí a los intermediarios en el mercado. Los beneficios de estos sistemas de comercio electrónico eran reducir los tiempos y los costos administrativos. El uso de las computadoras fue uno de los primeros ejemplares de reingeniería del proceso de negocio, desplazando los procesos en papel con métodos más eficientes.

A finales de 1970 fueron implementadas otras formas de comercio electrónico: el EDI formal (la transferencia automatizada de datos entre aplicaciones de computadora), *facsímile* (una extensión del teléfono) y el correo electrónico. Durante este tiempo surgieron las redes de valor agregado (*value-added network*, VAN) para soportar aplicaciones negocio-a-negocio. Inicialmente las VAN fueron establecidas como resultado de las necesidades de un sector específico, por ejemplo, la industria automotriz. éstas pronto empezaron a soportar EDI genérico y aplicaciones derivadas para comercio entre distintas compañías. De hecho, las VAN jugaron un papel importante en el comercio electrónico ya que brindaron aplicaciones especializadas y comunicaciones requeridas por la comunidad comercial. A final de la década de los 80 las VAN fueron el mecanismo dominante para soportar el comercio electrónico entre grandes compañías, sus divisiones y socios a través de una amplia gama de tecnologías.

De esta forma, el comercio electrónico dependió en su mayoría de las VAN y redes de mensajería privadas, ambas caracterizadas por altos costos y conectividad limitada. Mientras que las VAN son buenas para funciones como paso de ordenes de compra y facturas, la conectividad de la VAN es muy limitada para publicidad y funciones interactivas como la visualización de productos y proporción de servicios de texto y gráficos. Los puntos de venta de la VAN tienen alta seguridad, confiabilidad y confirmación de recepción. Las características anteriores no se pueden proporcionar del todo en la mayoría de las aplicaciones comerciales pues podría implicar una alta inversión en infraestructura, tiempo y dinero.

Por otra parte, la Internet tiene conectividad mundial, está creciendo en todos los segmentos de la sociedad, puede ser interactiva y usarla es relativamente barato. El problema es que no hay autoridad central para el control de la Internet y la confiabilidad (mientras se realiza) está en duda. La entrega no se garantiza. Los servicios relativamente nuevos de la Internet como el World-Wide Web no poseen mecanismos de seguridad hasta que ésta se implementa explícitamente.

Considerando las fortalezas y debilidades de estos dos caminos para el comercio electrónico, es fácil ver por qué ambos están creciendo y produciendo nuevos desarrollos técnicos.

1.3 Marco de identificación de aplicaciones de comercio electrónico

Para identificar los diferentes tipos de comercio electrónico es útil considerar las dos dimensiones que caracterizan las aplicaciones con tecnología de Internet.

- **Localización del usuario de la aplicación respecto al *firewall*[1] del sistema.**

  Las aplicaciones que utilizan tecnología de Internet pueden hacer disponible a los usuarios información que se encuentra dentro y fuera del *firewall* del sistema. Por ejemplo, los negocios pueden extender su alcance geográfico a nuevos mercados mediante la implementación de un escaparate Web en línea. Mientras que gran parte de la literatura se refiere a este tipo de esfuerzos de mercado mediante el Web, muchos han encontrado un ahorro considerable al implementar *intranet*. Estas *intranet* son usadas para mandar información empleada dentro del *firewall*, como horarios y cambios de proyecto.

- **Tipo de relación afectada.**

  Los sistemas interorganizacionales se pueden categorizar en dos: los que soportan relaciones existentes entre socios comerciales (relaciones ayudadas por la tecnología) y los que establecen nuevas relaciones que hasta la implementación de los sistemas interorganizacionales no eran posibles (relaciones facilitadas por la tecnología). Por ejemplo, el EDI puede ayudar a las relaciones ya existentes promoviendo una integración más cercana entre socios comerciales. Por otro lado, la tecnología de información ha facilitado la creación de relaciones completamente nuevas que antes no eran posibles. Además de encontrar nuevos clientes, la red puede promover grupos virtuales de trabajo descentralizados dentro y fuera de la compañía.

Combinando las dos dimensiones se pueden distinguir varios usos de Internet dentro de la matriz de dominio del comercio electrónico mostrada en la Figura 1.1. Varias de las primeras aplicaciones para Internet fueron dirigidas a un ambiente externo para facilitar las nuevas relaciones de negocio y atraer clientes mediante el sitio Web de la compañía. Estas aplicaciones consumidor-a-negocio se pueden localizar en la Celda 4. La mayoría de las aplicaciones para *intranet*, por definición, están dirigidas de forma interna para ayudar a la relación entre distintas partes dentro de la misma compañía. Estas aplicaciones intraorganizacionales se localizan en la Celda 1. Estos dos tipos de aplicaciones son las que han recibido una mayor atención.

Sin embargo, el clima actual de competencia requiere de distintos socios trabajando como un equipo, facilitando de esta manera el desarrollo de productos complejos con agilidad de manufactura y llamando la atención de clientes a largo plazo. Este equipo de socios ha sido denominado "ecosistema de negocio". Mucha de esta cooperación sólo puede ser lograda mediante el uso de sistemas interorganizacionales como Internet. Como se muestra en la Figura 1.2, hay una amplia región de oportunidades que no está desarrollada en términos de la aplicación de la tecnología de Internet. La tecnología de *extranet* puede llenar ese vacío y ser un puente entre la tecnología de *intranet* e Internet. Una de las definiciones de *extranet* se refiere a ésta como "una red colaborativa que usa tecnología de Internet para relacionar al negocio con sus proveedores, clientes y otros negocios que comparten objetivos comunes". Una *extranet* puede ser vista como parte de la *intranet* de una compañía que es hecha accesible a otras compañías o que es una colaboración con otras compañías. Se pueden clasificar las *extranet* como *intronet* localizadas en la Celda 3 y como *supranet* las cuales son aplicaciones de la Celda 2.

[1] Sistema diseñado para prevenir el acceso no autorizado a una red privada. En general, para prevenir el acceso de intrusos desde Internet.

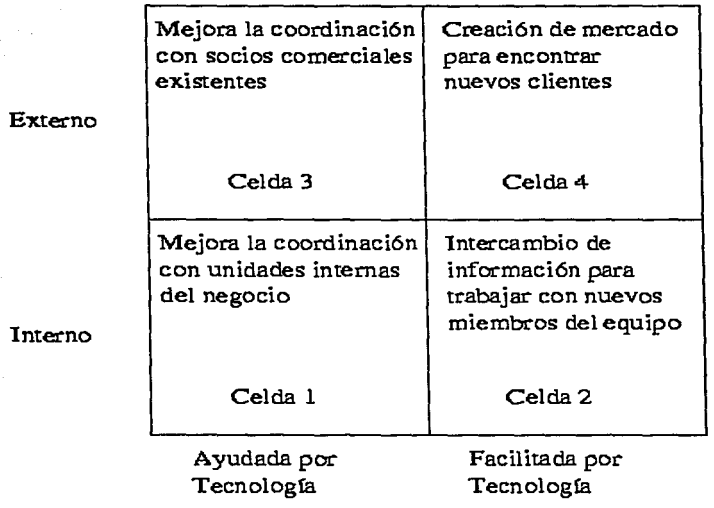


Figura 1.1 Localización del usuario de la aplicación respecto al firewall del sistema

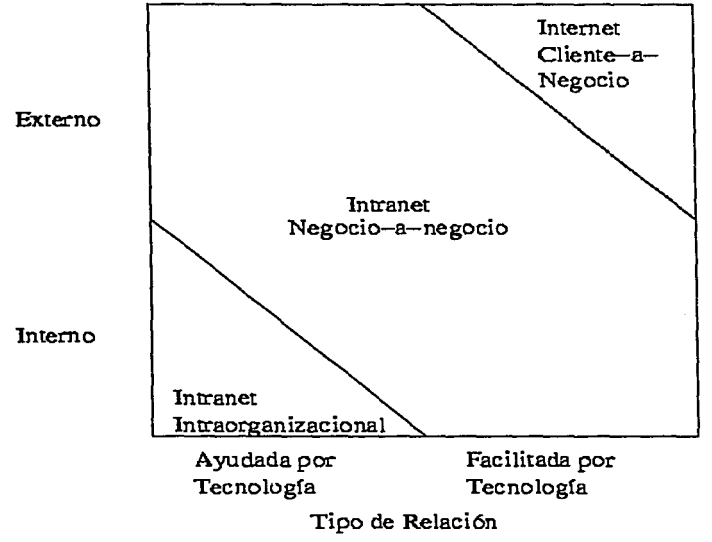


Figura 1.2 Localización del usuario de la aplicación respecto al firewall del sistema.

Las *intronet* son *extranet* donde los socios comerciales externos reciben acceso controlado detrás del *firewall* del iniciador y dentro de la *intranet* del mismo. Estas *intronet* son esencialmente sistemas de soporte para las decisiones interorganizacionales donde un socio comercial externo usa un navegador Web estándar para introducir y sacar información de la aplicación del iniciador. De esta manera, el socio externo controla el uso del sistema, mientras que el iniciador controla la funcionalidad y contenido del mismo. Típicamente, la parte externa obtiene acceso a información única mantenida en una base de datos dentro de la *intranet* del iniciador. Si el iniciador provee de información única y valiosa éste se encontrará en la posición de ganar una ventaja competitiva. En particular, si la *intronet* tiene como resultado cambios en la organización del socio externo, el iniciador puede estar en la posición de atraparlo y crearle una dependencia de la *intronet*.

En contraste, una *supranet* es una red interorganizacional patrocinada y controlada por un consorcio y que provee servicios de comunicación entre los miembros de la organización mediante aplicaciones variadas. Las metas comunes de las *supranet* es la eficiencia de todo el consorcio y la reducción del tiempo para el mercadeo de los entregables negocio-a-negocio de los equipos virtuales, tales como el nuevo diseño de un producto. Estas *extranet* funcionan como sistemas interorganizacionales de grupo donde la información es puesta electrónicamente para la siguiente fase del proceso de valor agregado. El objetivo general de la *supranet* del consorcio es promover la competitividad dentro del mismo contra otros ecosistemas.

1.4 Tecnologías del comercio electrónico

Mientras que muchas tecnologías pueden incluirse dentro de la definición de comercio electrónico , las más importantes son:

1. **Intercambio Electrónico de Datos.** Es el intercambio computadora-a-computadora de información de negocio estructurada en un formato electrónico.

2. **Código de Barras.** Son usados para la identificación automática de productos por una computadora. Los códigos de barras tienen un patrón rectangular de líneas con ancho y espacio variante.

3. **Correo electrónico.** Mensajes escritos por un emisor y mandados en forma digital hacia los receptores.

4. **Internet.** La Internet es una red global descentralizada de millones de computadoras y redes de computadoras. Estas redes pueden comunicarse entre ellas porque han añadido el uso común del protocolo de comunicación TCP/IP. La Internet es una herramienta para la comunicación entre gente, negocios y combinaciones de éstos. La red está creciendo rápidamente y cada vez tiene más usuarios.

5. ***World Wide Web* (WWW).** El WWW es una colección de documentos escritos y codificados con el Lenguaje de Marcado de Hipertexto (*HyperText Markup Language*, HTML). Con la ayuda de piezas pequeñas de software llamadas navegadores, un usuario puede requerir estos documentos y desplegarlos en su computadora local. Los documentos HTML (o "páginas", como se les llama) contienen distintos tipos de información como texto, imágenes, video, audio y ligas que puedan transportar al usuario a otras páginas Web. Ya que las páginas Web están disponibles a través de la Internet, estas ligas pueden llamar páginas de cualquier lugar del mundo. Es esta habilidad de saltar de sitio en sitio la que da lugar al término *World Wide Web*. El WWW es la aplicación más usada de la Internet.

6. **Intercambio de datos del producto.** Datos del producto se refiere a cualquier dato que es necesario para describir un producto. Algunas veces estos datos son gráficos, como en el caso de las imágenes, dibujos y archivos CAD. En otros casos el dato puede basarse en caracteres (números y letras), como en el caso de especificaciones, pagos de material, instrucciones de manufactura. El intercambio de datos del producto difiere de otros tipos de comunicación de negocio en dos formas importantes. Primero, las gráficas son entes complicados que pueden ocupar mucho espacio y pueden crear problemas de compatibilidad entre aplicaciones. Segundo, el control de versiones que cambian rápido es complicado. El diseño de productos en cualquier etapa puede estar sujeto a grandes cambios y como se sabe pequeños cambios de un producto pueden tener consecuencias mayores para poner el producto en producción.

7. **Formas electrónicas.** Las formas electrónicas son una tecnología que combina la familiaridad de las formas en papel con el poder de almacenamiento de información en forma digital. Para el usuario una forma electrónica es el análogo a la forma de papel, es una imagen la cual parece una forma en una pantalla y puede ser llenada mediante *mouse* y teclado. Sin embargo, detrás de la pantalla hay muchas funciones que el papel y lápiz no pueden proporcionar. Estas funciones extra permiten que la información de las formas electrónicas sea almacenada automáticamente e integrada a otras aplicaciones.

1.5 Tipos de pago para el comercio electrónico

Las principales formas en las que un consumidor paga sus productos en el comercio electrónico son:

- **Pago contra entrega.**

  El cobro de los productos se efectúa cuando un representante del proveedor entrega los mismos en el domicilio del consumidor. El pago es directo y sin transacciones electrónicas.

- **Efectivo electrónico.**

  El efectivo electrónico o dinero digital proporciona los medios para transferir dinero entre las partes sobre una red como la Internet. El efectivo electrónico emula las propiedades del dinero convencional y la forma en la que se realizan las transacciones con éste.

- **Cheques electrónicos y transferencias electrónica de fondos.**

  Los cheques electrónicos pertenecen al uso de servicios de red para emitir y procesar pagos que emulan los cheques del mundo real. El consumidor emite un cheque digital al proveedor y éste lo deposita en el banco. Cada transacción es realizada sobre la Internet.

  Los cheques electrónicos difieren de la transferencia electrónica de fondos en varias formas. Para cheques electrónicos, versiones electrónicas de cheques son emitidas, recibidas y procesadas. Así, el consumidor emite un cheque electrónico por cada pago. En el caso de la transferencia electrónica de fondos , retiros automáticos son hechos por recibos mensuales u otros pagos fijos, ningún cheque es emitido. Algunos ejemplos de transferencia electrónica de fondos son el pago de recibos a través de deducciones mensuales a cuentas bancarias y transferencias de grandes sumas de dinero entre bancos a través de todo el mundo.

Una transacción completa de cheque electrónico puede consistir de tres fases. En la primera fase, el consumidor hace una compra. En la segunda fase el proveedor manda el cheque electrónico a su banco para aprobación. En la tercera fase el banco del proveedor accede la oficina de compensación o al banco del consumidor para cobrar el cheque electrónico.

- **Tarjeta de Crédito.**

  Para este tipo de pago las tarjetas de crédito convencionales pueden ser usadas junto con un NIP. El NIP es un código secreto que el consumidor debe introducir mientras usa en línea la tarjeta de crédito. De esta manera se previene el mal uso de la tarjeta de crédito en caso de robo.

  Esta forma de pago tiene cuatro componentes típicos:

  1. Un consumidor con un navegador de Web.
  2. Un servidor del proveedor que proporciona una página. Específicamente, el servidor del proveedor toma las transacciones de la tarjeta de crédito.
  3. El banco del proveedor que toma las transacciones de la tarjeta de crédito.
  4. Una institución que ha emitido la tarjeta de crédito al consumidor.

  Una transacción electrónica de tarjeta de crédito se realiza esencialmente en tres fases. La primera fase termina en la compra de bienes por el consumidor. La segunda fase termina la transferencia de dinero desde la cuenta del consumidor hacia el proveedor. La tercera fase informa al consumidor acerca de las deducciones a su cuenta.

1.6 Seguridad en la Internet comercial

La seguridad en el comercio electrónico y específicamente en las transacciones comerciales es un aspecto de suma importancia. Para ello es necesario disponer de un servidor seguro a través del cual toda la información confidencial es encriptada y viaja de forma segura, esto brinda confianza tanto a proveedores como a compradores que hacen del comercio electrónico su forma habitual de negocios.

Al igual que en el comercio tradicional existe un riesgo en el comercio electrónico al realizar una transacción por Internet, el comprador teme la posibilidad de que sus datos personales (nombre, dirección, número de tarjeta de crédito) sean interceptados por alguien y suplante así su identidad; de igual forma el vendedor necesita asegurarse de que los datos enviados sean de quien dice ser.

Por tales motivos se han desarrollado sistemas de seguridad para transacciones por Internet: encriptación, firma digital y certificado digital, los cuales garantizan la confidencialidad, integridad y autenticidad respectivamente.

Con la encriptación la información transferida solo es accesible por las partes que intervienen (comprador, vendedor y sus bancos). La firma digital, evita que la transacción sea alterada por terceras personas sin saberlo. El certificado digital, que es emitido por un tercero, garantiza la identidad de las partes.

Es muy importante reducir los riesgos que la distancia impone a compradores y a vendedores. Como primer paso es necesario garantizar la confidencialidad, es decir que los datos necesarios para hacer el pago, como son número de tarjeta o cuenta, y su fecha de vencimiento, no sean vulnerables a receptores no autorizados en la red, lo cual se alcanza mediante la encriptación de mensajes.

El segundo paso es garantizar que la integridad de los datos que llevan las instrucciones de pago, no sean modificados a lo largo de su trayecto, esto se logra mediante el uso de firmas digitales.

Por último, la verificación de la autenticidad del comprador y del comerciante, el primero, como usuario legítimo de la tarjeta para el pago del bien adquirido, y el segundo garantizando que mantiene una relación bancaria con una institución financiera que acepta el pago con tarjetas, lo cual se consigue con la emisión de certificados digitales y la generación de firmas digitales.

Dentro del aspecto de seguridad es muy importante considerar el protocolo *Secure Electronic Transaction* (SET) el cual es un estándar técnico abierto para la industria del comercio desarrollada por Visa y Mastercard como forma de facilitar transacciones seguras de pago con tarjeta sobre la Internet. Este permite que emisores múltiples de certificados digitales cooperen en la transacción, integra los servicios de compensación de tarjetas de crédito, débito y otras con el servicio de verificación de autenticidad. Los certificados digitales crean una cadena de confianza a través de la transacción, verificando la validez del titular de la tarjeta y del comerciante. No se revela al comerciante la información de la tarjeta.

Existen instituciones que brindan servicios de *secure server*, para llevar adelante las actividades de comercio electrónico, entre las mas destacadas podemos mencionar las siguientes: Verisign y Terisa Systems.

La corporación *Internet Security Systems* (ISS), es la pionera en proveer sistemas adaptables a la gestión de seguridad y protección de programas, tales como *firewalls*, verificación de la autenticidad y encriptación.

El Sistema *Mondex* emplea la tarjeta de crédito para pagos digitales, opera a nivel mundial a través de bancos e instituciones financieras como una subsidiaria de Master Card Internacional.

La corporación *Cybercash* es reconocida mundialmente por proveer soluciones de pago seguras para el comercio electrónico, así como un sistema de pago y facturación interactivo a través de la WWW, entre empresas y entre empresas y consumidores.

1.7 Conceptos aplicados al desarrollo del sitio de comercio electrónico

Durante la presente tesis se desarrolla un sitio de comercio electrónico el cual es una aplicación sencilla consumidor-a-negocio que aprovecha algunas de las tecnologías revisadas en la Sección 1.4 las cuales son el EDI, la Internet, la WWW y las formas electrónicas.

La seguridad en el sistema y los tipos de pago no se desarrollan y se contempla que sean proporcionados por empresas especializadas en los mismos.

CAPITULO 2

Proceso Unificado

Un proceso es un conjunto de pasos ordenados para alcanzar una meta. En ingeniería de software la meta es hacer eficiente y predecible la entrega un producto de software que cumpla con las necesidades del negocio.

El *Unified Modeling Language* (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema de software. UML es independiente del proceso, esto significa que se puede emplear con distintos procesos de ingeniería de software. El Proceso Unificado es uno de los procesos que se acoplan bien con UML. La meta del Proceso Unificado es permitir la producción de software de alta calidad que satisfaga las necesidades de sus usuarios finales dentro de tiempos y presupuestos predecibles. El Proceso Unificado captura algunas de las mejores prácticas del desarrollo de software en una forma ajustable a una amplia variedad de proyectos y organizaciones. Del lado de la administración, el Proceso Unificado provee una manera disciplinada de cómo asignar tareas y responsabilidades dentro de una organización de desarrollo software.

2.1 Características del Proceso Unificado

El Proceso Unificado es iterativo. Un método iterativo apoya un entendimiento incremental del problema a través de refinamientos sucesivos y crecimiento de una solución efectiva llevada a cabo sobre múltiples ciclos. Construir mediante una estrategia iterativa provee de la flexibilidad para acomodar nuevos requerimientos y cambios tácticos en los objetivos del negocio. También permite identificar y resolver riesgos lo más temprano posible.

Las actividades del Proceso Unificado hacen énfasis en la creación y mantenimiento de modelos los cuales proveen una representación rica del sistema de software bajo desarrollo. Estos minimizan los costos de generar y mantener documentos y maximizan el contenido relevante de información.

El desarrollo bajo el Proceso Unificado está centrado en la arquitectura. El proceso toma especial atención al desarrollo temprano de una arquitectura para el software. Una arquitectura robusta facilita el desarrollo paralelo con varios equipos, minimiza la reconstrucción de lo ya desarrollado, incrementa la probabilidad de reutilizar los componentes y eventualmente facilita la mantenibilidad del sistema. La arquitectura sirve como una base sólida contra la cual planear y administrar un desarrollo de software basado en componentes.

Las actividades de desarrollo bajo el Proceso Unificado son guiadas por los casos de uso. El Proceso Unificado hace un fuerte énfasis en construir sistemas basándose en cómo estos serán usados. Las nociones de casos de uso y escenarios son usadas para alinear el proceso desde la captura de requerimientos hasta las pruebas y para dar un seguimiento del desarrollo del sistema hasta su entrega.

El Proceso Unificado soporta técnicas orientadas a objetos. Cada modelo es orientado a objetos. Los modelos del Proceso Unificado están basados en los conceptos de objetos, clases y las relaciones entre estos, ajustándose bien UML para representarlos.

El Proceso Unificado es configurable. Aunque ningún proceso es aplicable a todas las organizaciones de desarrollo de software, el Proceso Unificado es ajustable y puede ser escalado para alcanzar las necesidades desde proyectos con pequeños equipos de desarrollo hasta grandes organizaciones de desarrollo software.

El Proceso Unificado impulsa el continuo control de calidad y manejo de los riesgos. El control de calidad se logra usando criterios que midan los objetivos para las actividades y sus participantes. Los riesgos se intentan identificar y atacar tempranamente dentro del proceso de tal forma que no se ponga en riesgo el éxito del proyecto y se tenga tiempo de reaccionar.

2.2 Fases e iteraciones

Una fase es el lapso entre dos piedras angulares (*milestones*) mayores del proceso. En éste lapso se alcanza un conjunto bien definido de objetivos, se completan artefactos y se forman decisiones de cuando moverse a la siguiente fase. El Proceso Unificado consiste de las siguientes cuatro fases:

1. Inicio.
2. Elaboración.
3. Construcción.
4. Transición.

El inicio y la elaboración comprenden las actividades de ingeniería del ciclo de desarrollo; la construcción y la transición constituyen su producción.

Dentro de cada fase hay un número de iteraciones. Una iteración representa un paso completo de desarrollo por todos los flujos de trabajo (requerimientos, análisis, diseño, implementación y pruebas) que resulta en la liberación de un proyecto ejecutable, aunque esto puede no ser cierto para las primeras iteraciones del ciclo de desarrollo donde los requerimientos apenas se capturan.

Cada fase e iteración pone atención a la mitigación de algunos riesgos y concluye con una piedra angular bien definida. La revisión de ésta piedra angular provee un punto en el tiempo para saber qué tan bien se han alcanzado las metas, cuándo debe de ser reestructurado el proyecto y cuándo continuar con éste.

2.2.1 Fases

2.2.1.1 Inicio

Durante la fase de inicio se establece a groso modo que debe hacer el sistema para sus principales usuarios, se delimita el alcance del proyecto, se establece una arquitectura tentativa, se valoran los riesgos más importantes y se priorizan, se estiman aproximadamente los recursos a usar y se planea una agenda para piedras angulares mayores.

Al final de la fase de inicio se examinan los objetivos del ciclo de desarrollo del proyecto y se decide cuándo proceder con el desarrollo a gran escala.

2.2.1.2 Elaboración

Las metas de la fase de elaboración son: analizar el dominio del problema, establecer una arquitectura estable, desarrollar el plan del proyecto y eliminar los riesgos más altos de éste. Las decisiones sobre la arquitectura deben hacerse mediante el entendimiento de todo el sistema. Esto implica que se describen la mayoría de los requerimientos del sistema, es decir se capturan de forma detallada la gran mayoría de los casos de uso.

Al final de la fase de elaboración se examinan detalladamente los objetivos del sistema y su alcance, la arquitectura escogida y la resolución de los riesgos más altos para decidir cuándo proceder con la construcción.

2.2.1.3 Construcción

Durante la fase de construcción, de forma iterativa e incremental se desarrolla un producto completo el cual está listo para la transición hacia la comunidad de usuarios. Esto implica describir los requerimientos restantes, completar el diseño, la implementación y pruebas del software.

Al final de la fase de construcción se decide si el software, su sitio de instalación y sus usuarios están listos para operar.

2.2.1.4 Transición

Durante la fase de transición se instala el software para la comunidad de usuarios. Una vez que el sistema ha sido puesto en las manos de los usuarios finales empiezan a surgir requerimientos adicionales para ajustar el sistema, corregir problemas no detectados o terminar algunas características que habían sido pospuestas. Esta fase empieza típicamente con una versión beta del sistema la cual después es reemplazada por el sistema de producción.

Al final de la fase de transición, se decide si los objetivos del ciclo de desarrollo han sido alcanzados y se determina si se debería proceder con el siguiente ciclo. Este también es un punto en el cual se comprenden las lecciones aprendidas durante el proyecto con vistas a mejorar el proceso de desarrollo que será aplicado en el siguiente proyecto.

2.2.2 Iteraciones

Cada fase en el Proceso Unificado puede ser descompuesta en iteraciones. Una iteración es un paso completo de desarrollo que resulta en una versión interna o externa (es decir, para uso del desarrollador o del cliente) de un producto ejecutable el cual constituye un subconjunto del producto final bajo desarrollo que después se aumenta incrementalmente de iteración en iteración para convertirse en el sistema final. Cada iteración pasa por los distintos flujos de trabajo del proceso aunque con distinto énfasis en cada flujo dependiendo de la fase. Durante la fase de inicio la atención se pone a la captura de requerimientos. La fase de elaboración se concentra en el análisis y el diseño. En la construcción, la implementación es la actividad central. En la transición las actividades se centran en la instalación del sistema.

2.2.3 Ciclos de desarrollo

Un ciclo de desarrollo es el paso a través de las fases mayores cuyo resultado es una generación de software. El primer paso a través de las cuatro fases es llamado el ciclo de desarrollo inicial. Un producto existente evolucionará en su siguiente generación mediante la repetición de la misma secuencia de las fases de inicio, elaboración, construcción y transición hasta conseguir el sistema final. Esta es la evolución del sistema, los ciclos de desarrollo después del ciclo inicial son llamados ciclos de evolución.

2.3 Flujos de trabajo

El Proceso Unificado consiste de cinco flujos de trabajo:

1. Requerimientos: Desarrolla el modelo de casos de uso y los requerimientos suplementarios.
2. Análisis: Desarrolla el modelo de análisis.
3. Diseño: Desarrolla el modelo de diseño y el modelo de instalación.
4. Implementación: Desarrolla el modelo de implementación.
5. Pruebas: Desarrolla el modelo de pruebas.

Dentro de cada flujo de trabajo existe un conjunto de artefactos y actividades correlacionados. Un artefacto es cualquier documento, reporte o ejecutable que es producido, manipulado o consumido. Una actividad describe las tareas llevadas a cabo por desarrolladores para crear o modificar artefactos, junto con las técnicas y guías para desarrollarlos.

Hay conexiones importantes entre los artefactos de los distintos flujos de trabajo. El modelo de casos de uso generado durante la captura de requerimientos es *realizado por* el modelo de análisis y el de diseño desde los flujos de análisis y diseño respectivamente, *implementado por* el modelo de implementación desde el flujo de implementación y *verificado por* el modelo de pruebas desde el flujo de pruebas.

2.4 Artefactos

Cada actividad del Proceso Unificado tiene artefactos asociados ya sea requeridos como entrada o generados como salida. Algunos artefactos se usan para dirigir la entrada a actividades subsecuentes, se guardan como recursos de referencia en el proyecto o son generados en un formato de entregables estipulados en un contrato.

2.4.1 Modelos

Los modelos son el tipo de artefactos más importantes en el Proceso Unificado. Un modelo es una simplificación de la realidad creada para entender mejor el sistema a desarrollar. En el Proceso Unificado hay seis modelos que en conjunto cubren las decisiones importantes sobre la visualización, especificación, construcción y documentación de un sistema de software.

1. Modelo de casos de uso: Establece los requerimientos funcionales del sistema.
2. Modelo de análisis: Refina los requerimientos y establece una idea del diseño.
3. Modelo de diseño: Establece la solución del problema.
4. Modelo de instalación: Establece la topología del hardware en la cual el sistema es ejecutado.
5. Modelo de implementación: Establece las partes usadas para configurar y liberar el sistema físico.
6. Modelo de pruebas: Establece los medios por los cuales el sistema es validado y verificado.

En el Capítulo 3 se explica con mayor detalle la creación de los artefactos que contiene cada modelo.

CAPITULO 3

Los modelos del Proceso Unificado

Los modelos son el tipo de artefactos más importantes en el Proceso Unificado. Un modelo es una simplificación de la realidad creada para entender mejor el sistema a desarrollar. Los modelos proveen una representación rica del sistema de software. Estos minimizan los costos de generar y mantener documentos y maximizan el contenido relevante de información. Los modelos del Proceso Unificado están basados en los conceptos de objetos, clases y las relaciones entre estos, ajustándose bien UML para representarlos. El presente Capítulo explica los modelos y sus artefactos.

3.1 Modelo de casos de uso y requerimientos suplementarios

La captura de requerimientos tiene dos objetivos principales: encontrar los verdaderos requerimientos y representarlos en una manera entendible para clientes y desarrolladores. Por verdaderos requerimientos se entienden aquellos requerimientos que al ser implementados agregarán un valor esperado por el usuario. Estos dos objetivos permiten que se llegue a un acuerdo entre clientes y desarrolladores para construir el sistema correcto.

Para capturar los requerimientos del sistema se desarrollan un conjunto de artefactos. Estos artefactos son el modelo de casos de uso y los requerimientos suplementarios. Un modelo de casos de uso captura los requerimientos funcionales y no funcionales que son específicos a cada caso de uso individual. El modelo de casos de uso es descrito por un conjunto de diagramas y una descripción detallada de cada caso de uso. Los requerimientos suplementarios son la especificación de los requerimientos no funcionales que son genéricos y no son particulares de algún caso de uso.

Los requerimientos cambian constantemente durante las iteraciones y el número de cambios decrementa conforme se avanza hacia la fase de construcción. En el flujo de trabajo de captura de requerimientos se realizan los incrementos aproximadamente de la siguiente forma:

- Durante la fase de inicio se identifican casos de uso para delimitar el sistema, saber cual es el alcance del proyecto e identificar los casos de uso más críticos, el 10% aproximadamente.
- Durante la fase de elaboración se captura la mayoría de los requerimientos restantes y sus casos de uso, el 80% aproximadamente.
- El resto de los requerimientos se capturan durante la fase de construcción.
- Casi no hay requerimientos para capturar en la fase de transición de no ser por cambios en estos.

3.1.1 Casos de uso y actores

Los casos de uso son las funciones que un sistema proporciona y agregan valor para sus usuarios. La pregunta clave ¿qué debe hacer el sistema *para cada usuario*? nos mantiene concentrados en entender cómo el sistema necesita soportar a cada uno de sus usuarios. Esta pregunta nos guía en la búsqueda de las funciones que cada usuario necesita y evita sugerir funciones innecesarias.

La mayoría de los sistemas tienen varios tipos de usuarios. Cada tipo de usuario es representado como un actor. Todos los actores y casos de uso hacen el modelo de casos de uso. Un diagrama de casos de uso describe parte del modelo de casos de uso, muestra un conjunto de actores y casos de uso. Los actores y casos de uso están relacionados mediante asociaciones que representan la interacción entre cada par ellos.

No todos los actores representan humanos, pueden ser otros sistemas o *hardware* externo. Cada actor toma un conjunto coherente de roles cuando interactúa con el sistema. Un usuario físico puede actuar como uno o varios actores dependiendo de cómo interactúa con el sistema. Varios usuarios individuales pueden actuar como ocurrencias diferentes del mismo actor.

Los actores se comunican con el sistema mandando y recibiendo mensajes de éste. De la manera como se ha definido lo que los actores y casos de uso hacen, se tiene una clara separación de las responsabilidades de los actores de aquellas del sistema. Esta separación nos ayuda a delimitar el alcance del sistema. Los actores representan el ambiente externo y los casos de uso, la funcionalidad.

El modelo de casos de uso captura todos los requerimientos funcionales del sistema. Se puede definir un caso de uso de manera más precisa como sigue: *Un caso de uso especifica una secuencia de acciones, incluyendo variantes que el sistema puede realizar para obtener un resultado de valor para un actor en particular.*

Hay dos frases clave en la definición anterior "resultado de valor" y "actor en particular". Decir "resultado de valor" se refiere a que cada ejecución de un caso de uso debe proveer al usuario algún valor que le permita alcanzar algún objetivo. La frase "actor en particular" ayuda a identificar casos de uso que den verdadero valor a usuarios individuales.

La secuencia de acciones realizada por un caso de uso es un flujo específico dentro éste. Muchos flujos son posibles y muchos pueden ser bastante similares - son variantes de la ejecución de la secuencia de acciones especificada en el caso de uso. Para hacer el modelo de casos de uso entendible se agrupan descripciones de flujos similares en un mismo caso de uso. Los flujos para el caso de uso pueden ser capturados como una descripción textual de la secuencia de acciones.

Una instancia de caso de uso es la ejecución de un caso de uso. Cuando una instancia de caso de uso es ejecutada ésta interactúa con instancias de actor y realiza una secuencia de acciones como lo especifica su caso de uso. Esta secuencia es un flujo dentro del caso de uso.

Los casos de uso también se ocupan para capturar requerimientos no funcionales que sólo están relacionados con cada Caso de Uso en particular y que necesitan ser manejados en subsecuentes flujos de trabajo tales como análisis, diseño o implementación. Algunos requerimientos pueden estar relacionados con desempeño, disponibilidad, precisión y seguridad, entre otros. Estos requerimientos pueden ser capturados en forma textual.

Hay varias razones por las cuales los casos de uso son buenos y ampliamente utilizados:

- Ofrecen un medio sistemático e intuitivo para capturar requerimientos funcionales centrándose en el valor agregado que éste le puede proporcionar al usuario.
- Conducen el proceso de desarrollo ya que varias actividades como el análisis, diseño y pruebas son realizadas tomando como punto de partida los casos de uso. Son importantes al soportar un seguimiento en el resto de los modelos ya que esto permite mantener fácilmente la integridad y cambios del sistema.
- Los casos de uso son intuitivos. Usuarios y clientes no necesitan aprender notación compleja. En vez de eso se usa el lenguaje natural y del cliente para hacer fácil la lectura y cambios en estos.
- La captura de los casos de uso toma en cuenta a usuarios, clientes y desarrolladores. Los usuarios y clientes son los expertos de requerimientos. El papel de los desarrolladores es facilitar la discusión y ayudar a usuarios y clientes en la comunicación de sus necesidades.
- El modelo de casos de uso es usado para encontrar un acuerdo con los usuarios y clientes acerca de lo que el sistema debería hacer. Se puede pensar el modelo de casos de uso como una especificación completa de todas las maneras posibles de usar un sistema. Esta especificación puede ser usada como parte de un contrato con el cliente.

La Figura 3.1 muestra de manera general como se representa con UML los diagramas de casos de uso con sus actores, casos de uso y las interacciones entre ellos. Los actores "Administrador" y "Cliente" son representados mediante una figura humana abstracta, los casos de uso son representados mediante elipses y la interacción del actor que intenta desencadenar la secuencia de acciones del caso de uso (interactuar con el sistema) es representada mediante la línea dirigida que parte del actor hacia el caso de uso. La Figura también muestra algunas relaciones entre los casos de uso las cuales están representadas por líneas dirigidas y marcadas como extends o include además de la relación de generalización representada por una línea con punta hueca de flecha. Por ejemplo, el caso de uso 10 puede ejecutar su flujo y de manera opcional (relación extends) ejecutar el flujo de 11, 12 o 13. La ejecución del flujo del caso de uso 8 incluye (relación include) siempre la ejecución del caso de uso 10. Los casos de uso 18 y 19 pueden agregar o sobrescribir (relación de generalización) el comportamiento descrito en el caso de uso 18.

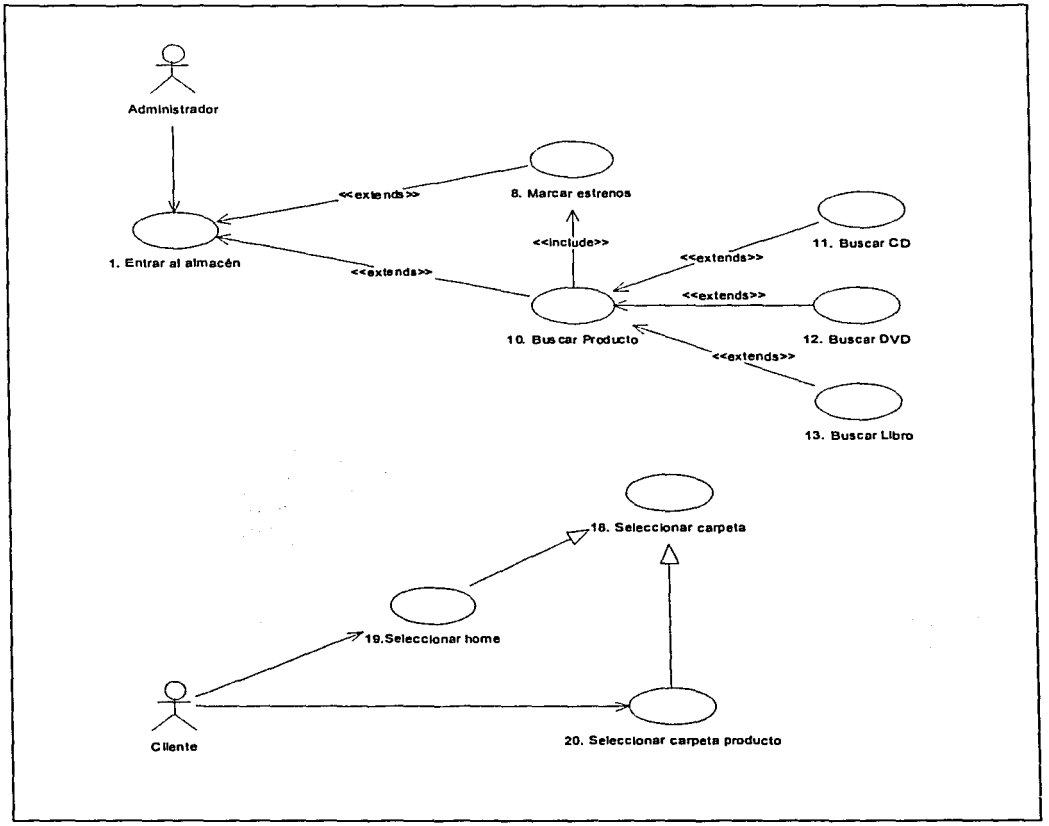


Figura 3.1 Ejemplo de diagrama de caso de uso.

Debido a que el Proceso Unificado no proporciona de una manera con la cual describir cada caso de uso, entonces se propone y aplica una. Esta manera de especificar los casos de uso se describe en la Sección 4.1.2, Subsección "Descripción de los flujos en los casos de uso", donde también se puede encontrar la descripción de los mismos.

3.1.2 Requerimientos suplementarios

Los requerimientos suplementarios son una descripción textual de los requerimientos no funcionales generales que no pueden ser asociados a un caso de uso en especial. Algunos de estos requerimientos son:

- Restricciones de interfaz tales como formatos.
- Restricciones físicas como la configuración física de la red, material y espacio que ocupará el sistema.
- Restricciones de diseño como la mantenibilidad y reutilización de sistemas legados.
- Restricciones de implementación como estándares requeridos, lenguajes de implementación, políticas para la integridad de la base de datos, ambiente de operación.
- Otros requerimientos como los legales y regulatorios.

3.2 Modelo de análisis

Durante el análisis los requerimientos se capturan, estructuran y refinan. El propósito de hacer esto es lograr un entendimiento más preciso de los requerimientos y tener una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema completo.

Al analizar el sistema se utiliza el lenguaje del desarrollador y una notación más formal que la usada durante la captura de requerimientos lo que permite razonar más acerca del interior del sistema. Esto permite refinar los requerimientos dándole una estructura para que se entiendan mejor y se preparen para su cambio, reutilización y en general, su mantenimiento.

Durante el flujo de trabajo de análisis se desarrolla el modelo de análisis. El modelo de análisis realiza los casos de uso mediante clases de análisis y sus objetos. Estas realizaciones son representadas por colaboraciones dentro del modelo de análisis y son denotadas *realizaciones de caso de uso-análisis.*

El uso del análisis difiere de proyecto a proyecto y se pueden notar básicamente tres variantes:

1. El proyecto utiliza el modelo de análisis y mantiene la consistencia del modelo durante todo el ciclo.
2. El proyecto utiliza el modelo de análisis como una herramienta intermedia – tal vez concentrándose en éste durante la fase de elaboración. Posteriormente, cuando la atención se pone al diseño e implementación, el modelo de análisis deja de mantenerse.
3. El proyecto no usa el modelo de análisis. En vez de esto, el análisis se hace como parte de la captura de requerimientos o del diseño.

El análisis es el centro de atención durante la fase de elaboración del sistema. Después, cuando la arquitectura es estable y los requerimientos entendidos, al final de la fase elaboración e inicio de la de construcción el centro es el diseño y la implementación. La estructura de requerimientos hecha en el análisis funciona como entrada fundamental para que el sistema tome forma durante el diseño e implementación.

3.2.1 Clases de análisis

Una clase de análisis representa una abstracción de varias clases y/o subsistemas dentro del diseño del sistema. Esta abstracción tiene las siguientes características:

- Una clase de análisis se centra en el manejo de requerimientos funcionales y pospone el manejo de requerimientos no funcionales hasta las actividades de diseño e implementación. Los requerimientos no funcionales se denotan como "requerimientos especiales" de la clase.
- Rara vez una clase de análisis define operaciones y sus firmas. En vez de esto, su comportamiento se define mediante responsabilidades de alto nivel y con menos formalidad. Una responsabilidad es una descripción textual de un subconjunto cohesivo de comportamiento definido por la clase.
- La clase de análisis puede definir atributos de alto nivel. El tipo de esos atributos es por lo general conceptual y reconocible desde el dominio del problema, mientras que los tipos de los atributos en diseño e implementación están más relacionados con los tipos del lenguaje de programación.
- Una clase de análisis está involucrada en relaciones aunque estas relaciones son más conceptuales que sus contrapartes de diseño e implementación. La navegabilidad de las asociaciones no es tan importante en análisis pero es esencial en diseño.
- Las clases de análisis caben por lo general en tres estereotipos básicos: *Boundary*, *Control* y *Entity*.

Los estereotipos *Boundary*, *Control* y *Entity* son estereotipos estándares del UML. Estos se usan para ayudar a los desarrolladores a distinguir entre distintos intereses sobre las clases. A continuación se describen dichos intereses:

- **Clase *Boundary***: Es usada para modelar la interacción entre el sistema y sus actores. Frecuentemente la interacción involucra recibir y presentar información además de peticiones a(de) usuarios y sistemas externos. Por lo general, representan abstracciones de ventanas, formas, impresoras, sensores, terminales.
- **Clase *Entity***: Es usada para modelar información de larga duración y por lo general persistente. No necesariamente es pasiva y puede tener comportamiento asociado.
- **Clase *Control***: Representan coordinación, secuenciación, transacciones y control de otros objetos. Puede representar una lógica de negocio compleja que involucre varios objetos.

3.2.2 Realización de caso de uso-análisis

Una *realización de caso de uso-análisis* es una colaboración dentro del modelo de análisis la cual describe cómo se realiza y ejecuta un caso de uso en términos de sus clases de análisis y objetos que interactúan. La *realización de caso de uso-análisis* proporciona unseguimiento directo a un caso de uso específico en el modelo de casos de uso.

Una *realización de caso de uso-análisis* tiene diagramas de clase que describen sus clases de análisis y diagramas de interacción que representan la realización de un flujo particular del caso de uso en términos de la interacción de objetos de análisis.

Las clases tienen roles los cuales pueden verse como el conjunto de responsabilidades y atributos de la misma pero que son sólo de interés para una *realización de caso de uso-análisis*. Entonces, al juntar todos los roles de las clases se puede saber cual es el papel de la clase dentro del sistema.

La secuencia de acciones de un caso de uso empieza por lo general cuando el actor invoca el caso de uso enviando algún mensaje al sistema. Si se considera la parte de "adentro" del sistema, un objeto *Boundary* recibirá el mensaje del actor. El objeto *Boundary* manda después el mensaje a algún otro objeto y de esta forma se puede iniciar la interacción de varios objetos los cuales ejecutan el caso de uso. En análisis se prefiere representar esto con diagramas de colaboración ya que el propósito principal es encontrar los requerimientos y responsabilidades de los objetos más que la secuencia cronológica de las interacciones (lo cual se hace con diagramas de secuencia que son más apropiados para el diseño). Los diagramas de colaboración y de secuencia son ambos diagramas de interacción.

Los diagramas de colaboración muestran las interacciones entre objetos mediante ligas (instancias de la asociación entre las clases correspondientes) las cuales incluyen mensajes. El nombre de un mensaje debe denotar el intento del objeto invocador por interactuar con el objeto invocado.

Las Figuras 3.2 y 3.3 ilustran de manera general como expresar mediante UML el diagrama de clases y el diagrama de colaboración necesarios para una *realización de caso de uso-análisis.* La Figura 3.2 es el diagrama de clases donde se encuentran clases típicas de análisis sin atributos ni firmas de operaciones, solo con sus asociaciones y sus estereotipos (*Boundary, Control, Entity*). Las clases se representan mediante las rectángulos y su nombre; las asociaciones mediante las líneas.

Figura 3.2 Ejemplo de diagrama de clases para una realización de caso de uso-análisis.

La Figura 3.3 es el diagrama de colaboración que muestra la interacción entre objetos de las clases de análisis del diagrama anterior. Se muestra el actor "Administrador" que interactúa con el sistema mediante una figura humana abstracta. Los objetos se representan como sus clases, con rectángulo pero con el nombre subrayado. Las ligas se muestran mediante líneas a las cuales se les añade un texto que indica la responsabilidad de la clase y una flecha que indica el orden de las interacciones (aunque en análisis este orden no es tan importante).

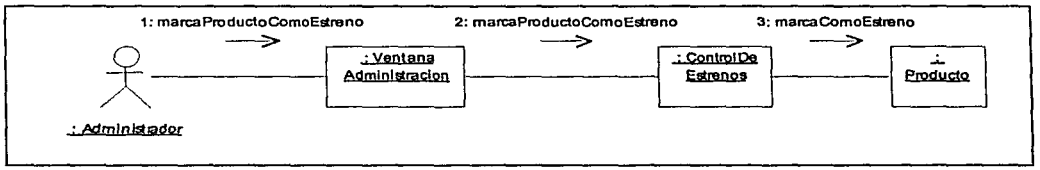


Figura 3.3 Ejemplo de diagrama de colaboración para una realización de caso de uso-análisis.

3.3 Modelo de diseño

El diseño es el centro de atención al final de la fase de elaboración y al principio de la de construcción. Contribuye a elaborar una arquitectura estable y a crear un esbozo del modelo de implementación. Más tarde, durante la fase de construcción, cuando la arquitectura es estable y los requerimientos son bien entendidos, la atención se pone en la implementación. Durante el flujo de diseño el desarrollo de modelos se concentra en la elaboración de los modelos de diseño e instalación.

Ya que el modelo de diseño es muy cercano al de implementación, es natural mantener el modelo de diseño durante todo el ciclo de desarrollo.

El modelo de diseño es un modelo que describe la realización física de los casos de uso concentrándose en los requerimientos funcionales y no funcionales del sistema junto con otras restricciones relacionadas con el ambiente de implementación. Además, el modelo de diseño sirve como una abstracción de la implementación del sistema y se usa como una entrada esencial a la implementación.

Dentro del modelo de diseño los casos de uso son realizados mediante clases de diseño y sus objetos. Esto se representa mediante colaboraciones dentro del modelo de diseño y se denotan *realizaciones de caso de uso-diseño*.

Los subsistemas y clases de diseño representan abstracciones de subsistemas y componentes en la implementación del sistema. Estas abstracciones, representan un mapeo simple y directo entre el diseño y la implementación.

3.3.1 Clases de diseño

Una clase de diseño es una abstracción de una clase o construcción similar en la implementación del sistema. Esta abstracción tiene de manera general las siguientes características:

- El lenguaje usado para especificar una clase de diseño es el mismo que el del lenguaje de programación. Por consiguiente, operaciones, parámetros, atributos y tipos son especificados usando la sintaxis del lenguaje de programación.
- Frecuentemente se especifican la visibilidad de los atributos y operaciones.
- Las relaciones en las que una clase de diseño se involucra con otras, con frecuencia tienen un significado directo con la implementación de la clase. Por ejemplo, la generalización tiene una semántica correspondiente a la herencia en los lenguajes de programación como Java. Para proveer referencias entre objetos las asociaciones y agregaciones de diseño a menudo corresponden a variables o atributos de clase en la implementación.
- Los métodos de una clase de diseño tienen mapeo directo a los métodos correspondientes en la clase de implementación (es decir, el código).
- Una clase de diseño puede posponer el manejo de algunos requerimientos a actividades de implementación subsecuentes denotándolos como "requerimientos de implementación de la clase". Esto hace posible posponer decisiones que son inapropiadas para su manejo durante el diseño.
- Si se usan estereotipos en las clases de diseño estos tienen un mapeo parecido a la construcción dada por el lenguaje de programación. Por ejemplo, una clase de diseño para una aplicación de Java puede estereotiparse como "JavaBean".

La Figura 3.4 muestra como diagramar con UML una clase típica de diseño. La clase se representa mediante un rectángulo dividido en tres partes, la de arriba para el nombre de la clase, la de en medio para los atributos y la de abajo para los métodos. En la clase se muestran los atributos con los tipos correspondientes al lenguaje de programación y los métodos se despliegan con su firma completa. Además se incluye la visibilidad de atributos y métodos.

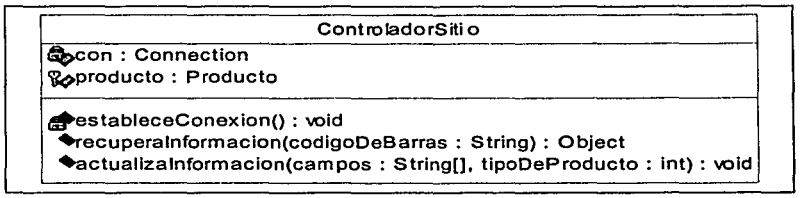


Figura 3.4 Ejemplo de una clase de diseño.

Respecto a la visibilidad de los atributos y métodos de las clases, en el presente trabajo no se usa la notación estándar de UML y se sustituye por los íconos de la herramienta Rational Rose 98 que se utilizó para crear los diagramas. La clase de la Figura 3.5 muestra dichos íconos. Para los atributos, las visibilidades en íconos se muestran en el siguiente orden *public*, *protected*, *default* y *private* que corresponden a las visibilidades manejadas por el lenguaje de programación Java. De manera análoga se disponen en la clase las visibilidades de los métodos.

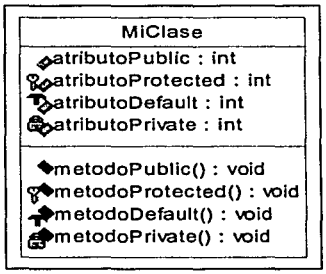


Figura 3.5 Visibilidades de una clase de diseño.

3.3.2 Realización de caso de uso-diseño

Una *realización de caso de uso-diseño* es una colaboración dentro del modelo de diseño que describe cómo se realiza un caso de uso específico en términos de sus clases de diseño y objetos. Una *realización de caso de uso-diseño* tiene seguimiento hacia una *realización de caso de uso-análisis* dentro del modelo de análisis. Nótese que una *realización de caso de uso-diseño* puede entonces tener seguimiento hacia un caso de uso en el modelo de casos de uso a través de la *realización de caso de uso-análisis*.

Cuando no se mantiene el modelo de análisis a través del ciclo de desarrollo del software pero se crea para poder obtener un buen diseño entonces es más natural dirigir el seguimiento de la *realización de caso de uso-diseño* hacia su caso de uso correspondiente en el modelo de casos de uso.

Una *realización de caso de uso-diseño* tiene diagramas de clase que ilustran sus clases de diseño participantes y diagramas de interacción donde se muestra la ejecución de un flujo particular del caso de uso en términos de interacciones entre objetos de diseño.

Una *realización de caso de uso-diseño* maneja de la mayoría de los requerimientos no funcionales del sistema. Sin embargo, una *realización de caso de uso-diseño*, al igual que las clases de diseño, puede posponer el manejo de algunos requerimientos hasta la implementación denotándolos como "requerimientos de implementación de la realización".

La Figura 3.6 muestra de manera general cómo expresar mediante UML el diagrama de clases para diseño. Las clases del diagrama aquí mostradas se denotan mediante rectángulos que contienen el estereotipo (entre los símbolo << >>), el nombre de la clase y su paquete (escrito entre paréntesis, éste no es parte de la notación de UML y es usada por la herramienta Rational Rose 98 para una mayor expresividad en los diagramas). En la Figura se muestran los tipos de relaciones que son utilizadas en el desarrollo del presente trabajo. La relación de generalización se denota mediante una línea sólida con una punta hueca de flecha, al inicio de la línea se encuentra la clase hija que hereda atributos y operaciones de la clase padre la cual está del lado de la punta hueca de flecha. La relación de agregación se denota con una línea sólida con punta hueca de diamante, ésta representa una relación *todo/parte* donde un objeto de una clase (el *todo*) se compone de objetos de otras clases (las *partes*); el diamante se pone del lado de la clase que representa el *todo*. La relación de dependencia se denota mediante una línea punteada con flecha en un extremo; expresa que el cambio en la especificación de la clase independiente (clase en el extremo de la flecha) puede afectar la semántica de la clase dependiente (clase que se encuentra al inicio de la línea).

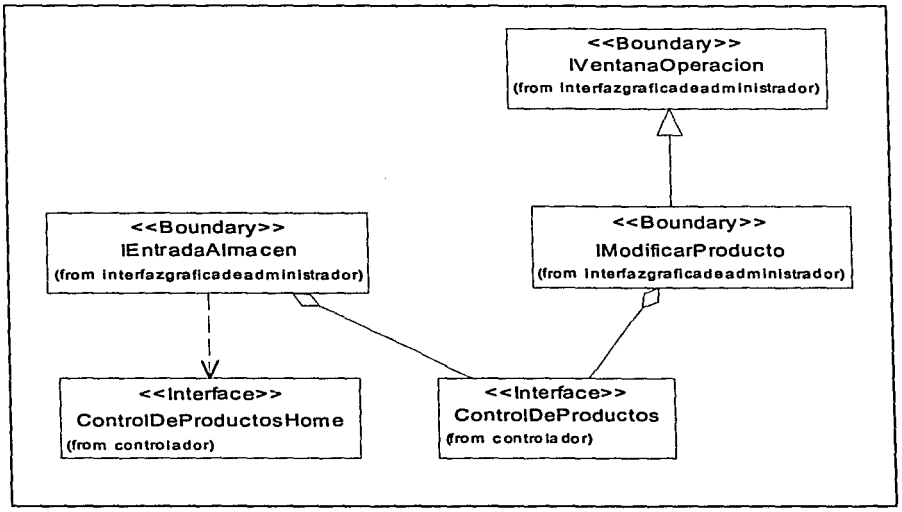


Figura 3.6 Ejemplo de diagrama de clases para diseño.

3.3.2.1 Diagramas de interacción

La secuencia de acciones en un caso de uso empieza cuando un actor invoca un caso de uso mandando algún mensaje al sistema. Si se considera la parte de "adentro" del sistema habrá algunos objetos de diseño que reciben los mensajes del actor. Después, el objeto de diseño llama a algún otro objeto y así los objetos interactúan para llevar a cabo el caso de uso. En diseño se prefiere denotar esto con diagramas de secuencia (un tipo de diagrama de interacción como los diagramas de colaboración que son mejor usados para el análisis) que proporcionen una secuencia cronológica detallada de las interacciones.

En los diagramas de secuencia se ilustra la interacción entre objetos mediante la transmisión de mensajes entre las líneas de vida de los objetos. El nombre de un mensaje debe indicar una operación del objeto invocado.

La Figura 3.7 muestra de manera general cómo expresar mediante UML un diagrama de secuencia para el diseño. En el diagrama se muestran en la parte superior el actor y los objetos de las clases participantes en la interacción. El actor se representa con la figura humana abstracta y los objetos con rectángulos que contienen el nombre subrayado de la clase a la cual pertenecen. La línea de vida del objeto es la línea punteada vertical que proviene de cada objeto y representa su existencia durante un periodo de tiempo. Se ilustra la interacción entre los objetos mediante la transmisión de mensajes (líneas con flecha al extremo) entre las líneas de vida de los objetos. El nombre de un mensaje debe indicar una operación del objeto invocado (objeto en la punta de la flecha). En el diagrama se muestra el control de las interacciones mediante rectángulos delgados sobre las líneas de vida de los objetos, estos indican el tiempo durante el cual el objeto está realizando una acción. La parte superior del rectángulo se alinea con el inicio de la acción y la parte inferior con su terminación. La terminación de la acción puede ser marcada con un mensaje de regreso como "daPagina" el cual se muestra en el diagrama.

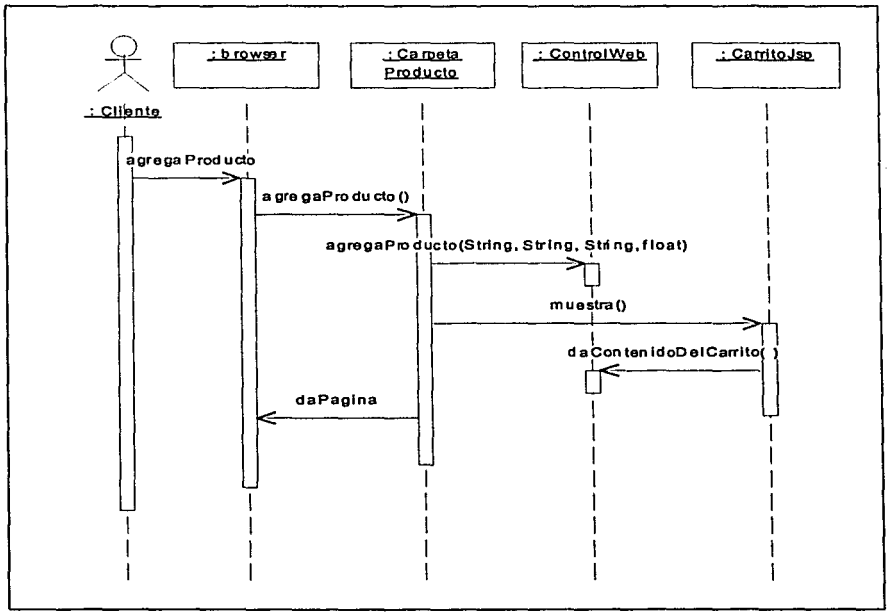


Figura 3.7 Ejemplo de diagrama de secuencia para diseño.

3.3.3 Subsistemas

Los subsistemas de diseño proporcionan un medio para organizar los artefactos del modelo de diseño en piezas más manejables. Un subsistema puede consistir de clases de diseño, *realizaciones de caso de uso-diseño* y otros subsistemas.

Un subsistema debe ser cohesivo; esto es, su contenido debe estar fuertemente relacionado. También los subsistemas deben estar débilmente acoplados; esto es, las dependencias entre ellos deben ser mínimas.

Además de las anteriores, los subsistemas deberían tener las siguientes características:

- Los subsistemas pueden representar una separación de intereses de diseño. Por ejemplo, en un sistema grande algunos subsistemas pueden ser diseñados en forma separada y posiblemente de manera concurrente por diferentes equipos de desarrollo con distintas habilidades de diseño.
- Los subsistemas pueden representar componentes de alta granularidad en la implementación del sistema; es decir, componentes que son creados mediante componentes de granularidad más fina tales como aquellos que especifican implementaciones individuales de clases que en conjunto se manifiestan como ejecutables, binarios o entidades similares que pueden ser instaladas en diferentes nodos.
- Los subsistemas pueden representar productos de software que se reutilizan.
- Los subsistemas pueden representar sistemas legados.

En UML los subsistemas se representan mediante el símbolo de paquete que consiste de un fólder el cual contiene el nombre del paquete (y algunas veces su contenido). En este caso se muestra el contenido de los paquetes que consiste del nombre de las clases que contienen y sus visibilidades. La visibilidad *public* de la clase se denota mediante el símbolo "+" y para denotar la visibilidad por *default* no se pone algún símbolo en especial. La Figura 3.8 muestra dos paquetes y las dependencias entre ellos. Las dependencias (línea punteada dirigida) entre los paquetes se definen cuando los contenidos de los paquetes tienen alguna relación unos con otros. La dirección de la dependencia debe ser la misma que la dirección definida por la relación entre los contenidos de los paquetes.

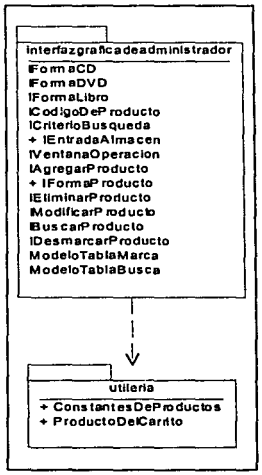


Figura 3.8 Ejemplo de paquetes.

3.3.4 Modelo de aplicación multicapa distribuido para J2EE

La plataforma J2EE usa un modelo de aplicación multicapa distribuido. Esto significa que la lógica de la aplicación está divida en componentes de acuerdo a la función y a los diversos componentes de aplicación que constituyen a la aplicación J2EE para que sean instalados sobre diferentes máquinas dependiendo sobre que capa del ambiente J2EE multicapa pertenece el componente de la aplicación.

Mientras que una aplicación J2EE puede consistir de tres o cuatro capas, las aplicaciones multicapa J2EE son generalmente consideradas como aplicaciones de tres capas porque son distribuidas sobre diferentes lugares: máquina cliente, máquina servidor y máquinas de base de datos o sistemas legados en el *back end*. Las aplicaciones de tres capas que se ejecutan de esta forma extienden el modelo cliente servidor estándar de dos capas poniendo un servidor de aplicación multicapa entre la aplicación cliente y el medio de almacenaje.

Un componente es una unidad de software a nivel de aplicación. La plataforma J2EE soporta los siguientes tipos de componentes: componentes *Java Bean*, *applet*, clientes de aplicación, componentes *Enterprise Java Bean* (EJB) y componentes de Web. Los clientes de aplicación y los *applets* se ejecutan sobre la plataforma cliente y los componentes de Web y EJB se ejecutan sobre la plataforma del servidor.

Debido al uso de la plataforma J2EE en el sistema de comercio electrónico del presente trabajo, los modelos de diseño e instalación son influenciados por la misma. Para una información más detallada del modelo de aplicación multicapa distribuido y sus configuraciones usadas en el presente trabajo se sugiere consultar la tesis complementaria [1] en sus secciones "2.2 J2EE", "2.3.1 Escenario de Aplicación Multicapa" y "2.3.2 Escenario Cliente *Stand-Alone*". Las últimas dos secciones citadas servirán como marco de referencia para el desarrollo de los modelos de diseño e instalación del siguiente Capítulo, en el cual se explicará cuál fue el uso particular de esos dos escenarios.

3.4 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño tales como las clases de diseño se implementan en términos de componentes como archivos fuente, ejecutables y estructuras similares. El modelo de implementación también describe cómo se organizan y relacionan los componentes de acuerdo a los mecanismos de modularización del ambiente de implementación y de los lenguajes de programación que se utilizan. De manera más específica, las metas de la implementación son:

- Planear la integración del sistema en cada iteración. La estrategia es incremental; es decir, el sistema se implementa poco a poco a través de una sucesión de pasos pequeños y planeados.
- Distribuir el sistema mediante el mapeo de componentes ejecutables en los nodos del modelo de instalación.
- Implementar las clases y subsistemas de diseño. En particular, las clases de diseño se implementan como archivos que contienen el código fuente.
- Hacer pruebas unitarias en los componentes para después compilarlos y ligarlos en uno o más ejecutables. Después se integran al sistema y se les aplican pruebas para el sistema.

Los artefactos que se desarrollan en el modelo de implementación son los componentes, subsistemas de implementación y el plan de integración.

La implementación es la principal actividad durante la fase de construcción. La implementación también se lleva a cabo durante la fase de elaboración para crear una arquitectura ejecutable estable y durante la transición para corregir los defectos de las versiones beta del sistema.

3.4.1 Componentes

Un componente es el empacado físico de los elementos del modelo tales como las clases de diseño en el modelo de diseño. Los componentes tienen las siguientes características.

- Tienen seguimiento hacia los elementos del modelo que están implementando.
- Frecuentemente los componentes implementan varios elementos, como las clases de diseño; sin embargo, la manera en que se hace el seguimiento depende de cómo los archivos fuente se estructuran dependiendo del lenguaje de programación que se esté utilizando.
- Pueden existir dependencias de compilación entre componentes, éstas denotan qué componentes se requieren para compilar un componente específico.

Hay algunos estereotipos estándares para los componentes tales como:

- *executable*: Programa que podría correr en un nodo.
- *file*: Archivo que contiene código fuente o datos.
- *library*: Biblioteca estática o dinámica.

Dependiendo del ambiente de implementación los estereotipos pueden ser modificados para que reflejen de manera adecuada lo que los componentes hacen.

La Figura 3.9 muestra de manera general cómo expresar mediante UML un diagrama de componentes para la implementación. En el diagrama se muestran los componentes y sus relaciones. El componente se representa mediante un rectángulo con un par de tablas el cual incluye el nombre del componente y en ocasiones su estereotipo. En este diagrama se puede definir las dependencias entre los componentes como dependencias de compilación para crear el ejecutable IEntradaAlmacen.class a partir de los componentes de estereotipo *class*. Es un diagrama que en particular, puede expresar componentes en el lenguaje de programación Java.

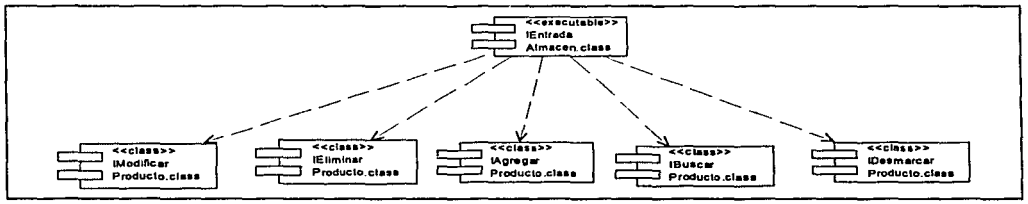


Figura 3.9 Ejemplo de diagrama de componentes para implementación.

3.4.2 Subsistemas de implementación

Los subsistemas de implementación proporcionan un medio para organizar los artefactos del modelo de implementación en piezas más manejables. Un subsistema puede consistir de componentes y otros subsistemas (recursivamente).

Un subsistema de implementación se manifiesta como un "mecanismo de empaquetado" del ambiente de implementación, por ejemplo; un *paquete* de Java o un *proyecto* en Visual Basic. De esta forma, el subsistema de implementación adquiere una semántica refinada correspondiente al ambiente de implementación.

Los subsistemas de implementación se relacionan fuertemente con los subsistemas de diseño del modelo de diseño. De hecho, un subsistema de implementación debería de tener un seguimiento uno-a-uno con un subsistema de diseño. Los subsistemas de diseño definen dependencias sobre otros subsistemas y las clases de diseño que le sirven como interfaz. Un subsistema de implementación debería definir:

- Dependencias análogas sobre otros (los correspondientes) subsistemas de implementación.
- Los componentes que le sirven de interfaz. Estos componentes deberían tener seguimiento hacia las clases de diseño que implementan.

En UML los subsistemas de implementación se representan de manera similar a los subsistemas de diseño mediante el símbolo de paquete que consiste de un fólder el cual contiene el nombre del paquete.

3.4.3 Plan de integración

En el Proceso Unificado el software se construye incrementalmente en varios pasos de forma que cada paso conduzca a una pequeña integración de funcionalidad y pocos problemas en las pruebas. El resultado de cada paso se llama un *incremento* el cual es una versión ejecutable del sistema. A cada *incremento* se le aplican pruebas de integración antes de construir el siguiente *incremento*.

Si se pone lo anterior en un contexto iterativo, cada iteración resulta en al menos un *incremento* (varios *incrementos* si la funcionalidad es muy compleja). Estos *incrementos* añaden una pequeña parte de funcionalidad al sistema.

Un plan de integración controla la construcción de la aplicación y describe la secuencia de *incrementos* requerida para cada iteración.

3.5 Modelo de instalación

El modelo de instalación es un modelo que describe la distribución física del sistema en términos de cómo la funcionalidad es distribuida entre nodos computacionales. El modelo de instalación es usado como una entrada esencial a el diseño e implementación ya que la distribución del sistema tiene gran impacto sobre su diseño.

Algunas consideraciones del modelo de instalación son las siguientes:

- Cada nodo representa un recurso computacional, con frecuencia un procesador o un dispositivo de *hardware* similar .
- Las relaciones entre los nodos representan medios de comunicación entre éstos tales como Internet, intranet y *bus*.
- El modelo de instalación puede describir varias configuraciones de red.
- La funcionalidad (o procesos) de un nodo está definida por los componentes instalados en el nodo.
- El modelo de instalación manifiesta el mapeo entre la arquitectura del software y la arquitectura del sistema (el *hardware*).

La Figura 3.10 muestra de manera general cómo expresar mediante UML un diagrama de instalación. En el diagrama se muestran los nodos representados por cubos con sus nombres. Las asociaciones entre los nodos muestran conexiones entre estos. Las asociaciones pueden estereotiparse para darles un mayor semántica (en éste caso se muestran dos protocolos, HTTP y JDBC).

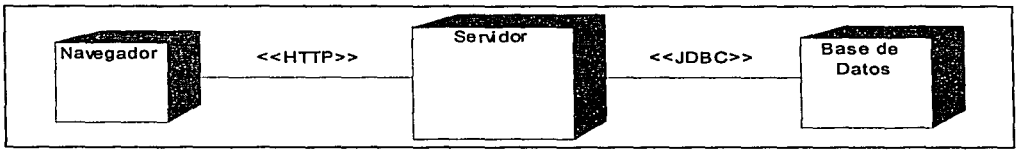


Figura 3.10 Ejemplo de diagrama de instalación.

3.6 Modelo de pruebas

El propósito principal del modelo de pruebas es describir cómo son probados los componentes ejecutables del modelo de implementación mediante pruebas de integración y del sistema. El modelo de pruebas es una colección de casos de prueba, procedimientos y componentes de prueba.

Durante el flujo de pruebas se realizan y evalúan las pruebas descritas en el modelo de pruebas. Esto se inicia con la planeación del esfuerzo requerido (por ejemplo tiempo, personal) para las pruebas dentro de cada iteración. Después se describen los casos de prueba y los procedimientos para su aplicación. Si es posible, se crean componentes de prueba que automaticen algunos de los procedimientos de prueba. Lo anterior se realiza para cada incremento que se libera en el flujo de implementación. Ya que son creados los casos, procedimientos y componentes de prueba, estos se usan para probar el incremento y probablemente detectar algunos defectos. Los defectos detectados se mandan a otros flujos de trabajo como el diseño e implementación para su corrección.

La planeación de las pruebas para las iteraciones incluyen pruebas de integración y de sistema. Las pruebas de integración de realizan para cada incremento dentro de la iteración mientras que las pruebas de sistema se requieren solo al final de cada iteración.

Las aplicación de pruebas se realiza principalmente durante la fase de elaboración, cuando se prueba la base de la arquitectura ejecutable y durante la fase de construcción donde se implementa la mayor parte del sistema. La fase de transición se concentra en la corrección de los defectos detectados.

3.6.1 Casos de prueba

De manera general, un caso de prueba especifica una manera de probar el sistema, incluyendo que hay que probar, con que entradas y resultados y bajo que condiciones se realiza la prueba. En la práctica, lo que hay que probar es cualquier requerimiento o colección de requerimientos del sistema cuya implementación justifica la elaboración de una prueba que sea posible realizar y no sea demasiado costosa (por ejemplo, una prueba que ocupe demasiado tiempo de desarrollo). Los siguientes son casos de prueba comunes:

- Un caso de prueba que especifique cómo probar un caso de uso o escenario particular del caso de uso. Puede notarse que un caso de prueba basado en un caso de uso es una típica "prueba de caja negra" del sistema; es decir, una prueba del comportamiento observable externo del sistema.
- Un caso de prueba que especifica cómo probar una *realización de caso de uso-diseño* el cual puede incluir el verificar la interacción entre los componentes que implementan el caso de uso. Puede notarse que un caso de prueba basado en una *realización de caso de uso* es una típica "prueba de caja blanca" del sistema; es decir, una prueba de la interacción interna entre los componentes del sistema.

Existen otros casos de prueba para probar el sistema como un todo y son conocidos como casos de prueba para el sistema. Algunos de estos son:

- Pruebas de instalación: Verifican que el sistema pueda ser instalado en la plataforma y que el sistema opere correctamente al ser instalado.
- Pruebas de configuración: Verifica que el sistema trabaje correctamente en diferentes configuraciones tales como distintas configuraciones de red.
- Pruebas negativas: Intentan que el sistema falle para que revele sus debilidades. Un ejemplo puede ser someter al sistema a una gran carga de trabajo.
- Pruebas de *stress*: Intentan identificar problemas cuando el sistema sufre de insuficiencia de recursos o existe competencia por los mismos.

3.6.2 Procedimientos de prueba

Un procedimiento de prueba especifica la manera en que se realizan uno o varios casos de prueba o parte de ellos. Por ejemplo, un procedimiento de prueba puede ser un conjunto de instrucciones que digan cómo llevar a cabo un caso de prueba manualmente o puede ser una especificación de cómo interactuar con una herramienta que automatiza las pruebas.

La manera de realizar un caso de prueba se puede especificar en un procedimiento de prueba aunque lo usual es reutilizar un procedimiento de prueba para varios casos de prueba o también reutilizar varios procedimientos de prueba para un caso de prueba.

3.6.3 Componentes de prueba

Un componente de prueba automatiza uno o varios procedimientos de prueba o varios de ellos. Los componentes de prueba se usan para probar los componentes del modelo de implementación dando una serie de entradas, controlando y monitoreando la ejecución del componente en prueba y posiblemente dando el reporte de los resultados. Los componentes de prueba pueden ser desarrollados usando algún lenguaje de programación o de *script*.

CAPITULO 4

Aplicación del Proceso Unificado al sistema de comercio electrónico

El desarrollo del sistema de comercio electrónico se presenta en este Capítulo a través de los flujos de trabajo del Proceso Unificado. Para cada flujo de trabajo existen algunos modelos y artefactos centrales a desarrollar. Los modelos solo muestran los artefactos que fueron necesarios para la especificación y construcción del software puesto que el Proceso Unificado es configurable para distintos tipos de proyectos y organizaciones. En éste caso, el proyecto es un sistema de comercio electrónico y el equipo de desarrollo estuvo conformado por dos personas. Las experiencias y aprendizajes que tuvieron lugar durante el desarrollo de este trabajo son de gran valor pues demuestran que el desarrollo de software de calidad no requiere solamente de un conjunto de buenos programadores sino también de un proceso que lo guíe y de un equipo cuya experiencia en el proceso y tecnología será fundamental en el éxito del proyecto.

4.1 Requerimientos

La captura de requerimientos es esencial en el desarrollo de cualquier sistema. De una buena captura de requerimientos depende su uso futuro. Muchos sistemas son desechados o poco utilizados debido a que no cumplen con la funcionalidad que proporciona valor agregado al usuario. Un sistema que no se usa o se usa poco es un sistema que representa pérdidas para las organizaciones. Se pierde tiempo, recursos y dinero que no serán remunerados por su funcionalidad.

4.1.1 Planteamiento del sistema a desarrollar

Antes de empezar a resolver cualquier problema se debe tener en cuenta de qué se trata éste. Por esta razón, lo primero que se realizó para empezar a desarrollar el sistema de comercio electrónico fue un planteamiento textual del problema permitiendo descubrir de manera general cual era el alcance del proyecto y partiendo de ese punto empezar a precisar cuales serían los recursos necesarios para el desarrollo del mismo. A continuación se describe:

Planteamiento

El propósito del proyecto es realizar un sistema de comercio electrónico llamado *Sistema de Ventas Café* el cual pone a la venta a través de un sitio Web una variedad de CDs, DVDs y libros.

Los clientes visitan el sitio Web donde realizan la selección de productos que desean adquirir. Los clientes pueden revisar en línea las características de los productos disponibles en el sitio además de contar con un carrito de compras en el cual pueden agregar y modificar aquellos que desean adquirir. Una vez que el cliente decide hacer la compra se le solicita la forma de pago las cuales pueden ser, mediante tarjeta de crédito o pagar sus productos en efectivo cuando éstos sean entregados en su domicilio. Después de haber decidido el tipo de pago se genera un pedido.

El sistema de comercio electrónico cuenta con un módulo de administración de los productos del sitio Web. Dicho módulo sirve para añadir, eliminar, modificar y hacer búsquedas sobre los productos además de poder marcarlos como estrenos o quitarles dicha marca cuando ya no sean considerados como tal. El módulo de administración es independiente de la información correspondiente a la disponibilidad física para la venta que puedan tener los productos.

La seguridad en el sistema y las transacciones de pago con tarjeta de crédito no se desarrollan en el proyecto y se contempla que sean proporcionados por empresas especializadas en los mismos.

A partir de este planteamiento ya se puede empezar a refinar la funcionalidad del sistema, esta necesidad es resuelta por los casos de uso.

Otro punto después del planteamiento es proponer una arquitectura tentativa para el software. Al principio se pensó en un sistema cliente/servidor donde la administración del sistema estuviera programada en Java con el *Java 2 SDK, Standard Edition, v 1.3* (J2SDK1.3) y el sitio Web estuviera programado con *Java Server Pages* (JSP). De hecho, la administración del sistema fue terminada con este esquema dando como resultado un cliente pesado, es decir, que tenía programada dentro de él la lógica de negocio.

Después de que se desarrolló la administración se pensó usar una tecnología innovadora que permitiera probar una arquitectura más rica donde por principio se prometía poder desarrollar de manera natural un cliente ligero donde la lógica del negocio quedara del lado del servidor. Dicha tecnología fue *Java 2 Enterprise Edition* (J2EE). Esta decisión impactó fuertemente el tiempo de entrega del proyecto. La experiencia antes descrita se explicará con más detalle posteriormente.

El Proceso Unificado hace fuerte énfasis en la administración del proyecto mediante planes sobre las iteraciones, fases y ciclos creando criterios que permitan medir tiempos, calidad y posibilidades de seguir con las siguientes etapas. Además, la parte administrativa también se encarga de manejar y asignar las tareas dentro de la organización. En el caso del presente trabajo, no se pudieron llevar a cabo varias de estas tareas administrativas debido a la falta de experiencia en la aplicación del proceso y no sólo del proceso sino en la tecnología *Java 2 Enterprise Edition* (J2EE) que se utitlizó en el sistema. Esto hacia difícil el poder distribuir las actividades entre los dos miembros del equipo pues se tenía que descubrir conjuntamente a cada paso lo que el proceso y la tecnología requerían. En vez de hacer un gran énfasis en las tareas administrativas, el presente trabajo centra la atención en la creación de modelos y otros artefactos que colaboran en la construcción de un software de calidad.

La planeación de cómo sería construido el sistema quedó influenciada por la identificación temprana de dos subsistemas: la administración y el sitio Web. Se planeó usar dos ciclos, uno inicial que tuviera como resultado el subsistema de administración y el segundo donde se creaba el sitio Web. No se planearon iteraciones controladas durante estos dos ciclos debido a la dificultad ya antes mencionada de aprendizaje del proceso y tecnología. Sin embargo, sí se usó el método iterativo e incremental que permite construir poco a poco el sistema mediante varios pasos por los flujos del trabajo.

4.1.2 Modelo de casos de uso

Para la captura de requerimientos se careció de un usuario real que pusiera los límites, usos y restricciones para el sistema. La consecuencia de no tener al usuario real fue que se intentaban realizar funcionalidades que adquirían una mayor complejidad de la que podría requerirse o que tarde o temprano hacían surgir preguntas acerca de la necesidad de incluirlas. Durante el desarrollo del sistema fue fácil ver que este problema es bien resuelto por los casos de uso. Para su captura era necesario siempre tener en mente al usuario. La pregunta ¿qué debe hacer el sistema *para cada usuario*? obligó a pensar en el valor agregado que el sistema puede proporcionar al usuario real. Sin esta pregunta se corre el riesgo de intentar desarrollar funcionalidades sin utilidad o más complejas de lo necesario que harán el sistema más grande y costoso.

Se percibió que los casos de uso ahorran el tiempo en la captura de requerimientos pues ésta se realiza mediante el lenguaje natural y del usuario sin que sea necesario aprender una notación difícil. Con esto no se quiere decir que no es importante usar una notación menos ambigua que el lenguaje natural pero éste problema queda resuelto en etapas posteriores las cuales ayudan a refinar los requerimientos. Debido al uso del lenguaje natural los casos de uso también proporcionan una excelente manera de llegar a un acuerdo entre desarrolladores y usuarios de forma que el sistema haga lo que estos esperan.

La buena captura de los requerimientos ahorra tiempo, recursos y dinero, además permite construir un sistema que realmente proporcione los valores requeridos por el usuario.

En las siguientes secciones se muestran actores, flujos y diagramas de casos de uso para el sistema. Estos se muestran en dos partes para su mejor lectura, una para el subsistema de administración y la otra para el subsistema del sitio Web

4.1.2.1 Diagramas de casos de uso

Los diagramas de casos de uso de las Figuras 4.1 y 4.2 muestran las interacciones de los actores con el sistema mediante las asociaciones entre estos. A cada caso de uso se le asocia un identificador dado por un número natural el cual servirá para darle seguimiento en el resto de los modelos.

Administración

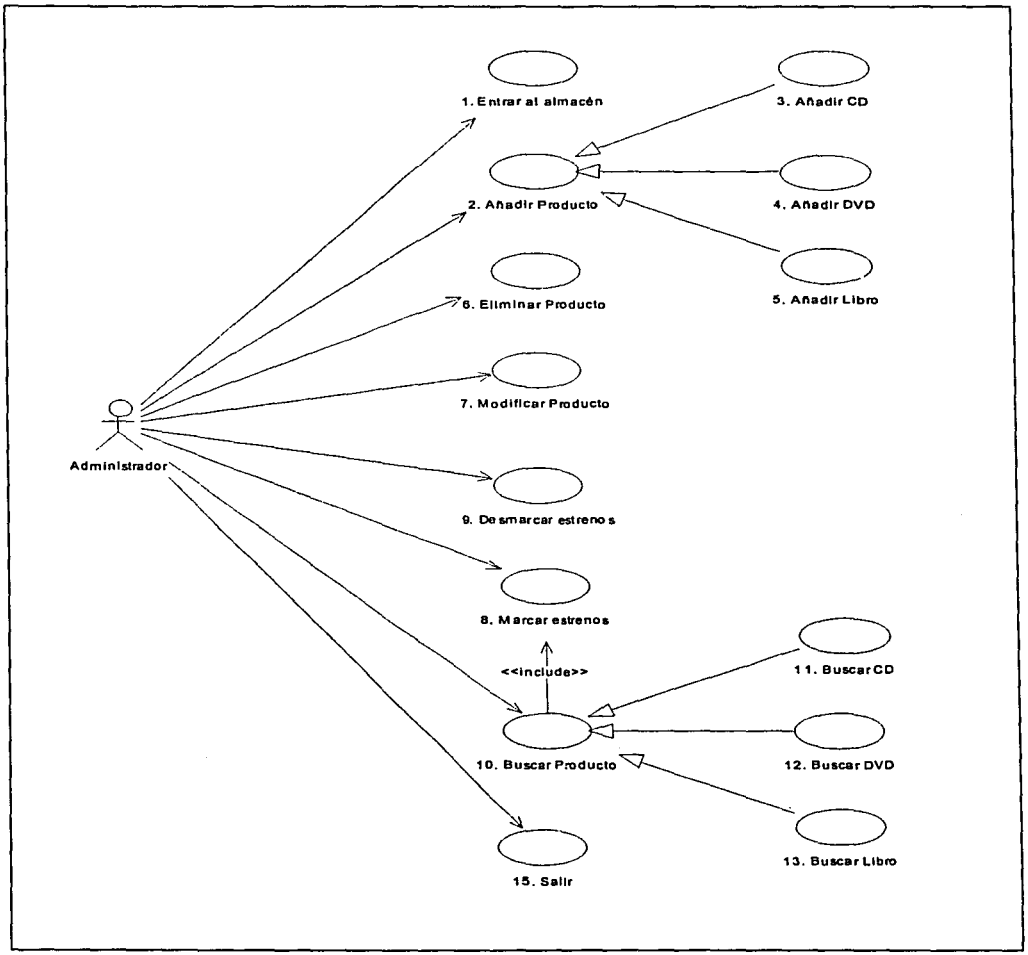


Figura 4.1 Diagrama de casos de uso para la administración.

El resto de los diagramas de casos de uso para la administración se muestran en el Apéndice A, Sección A.1.

Sitio Web

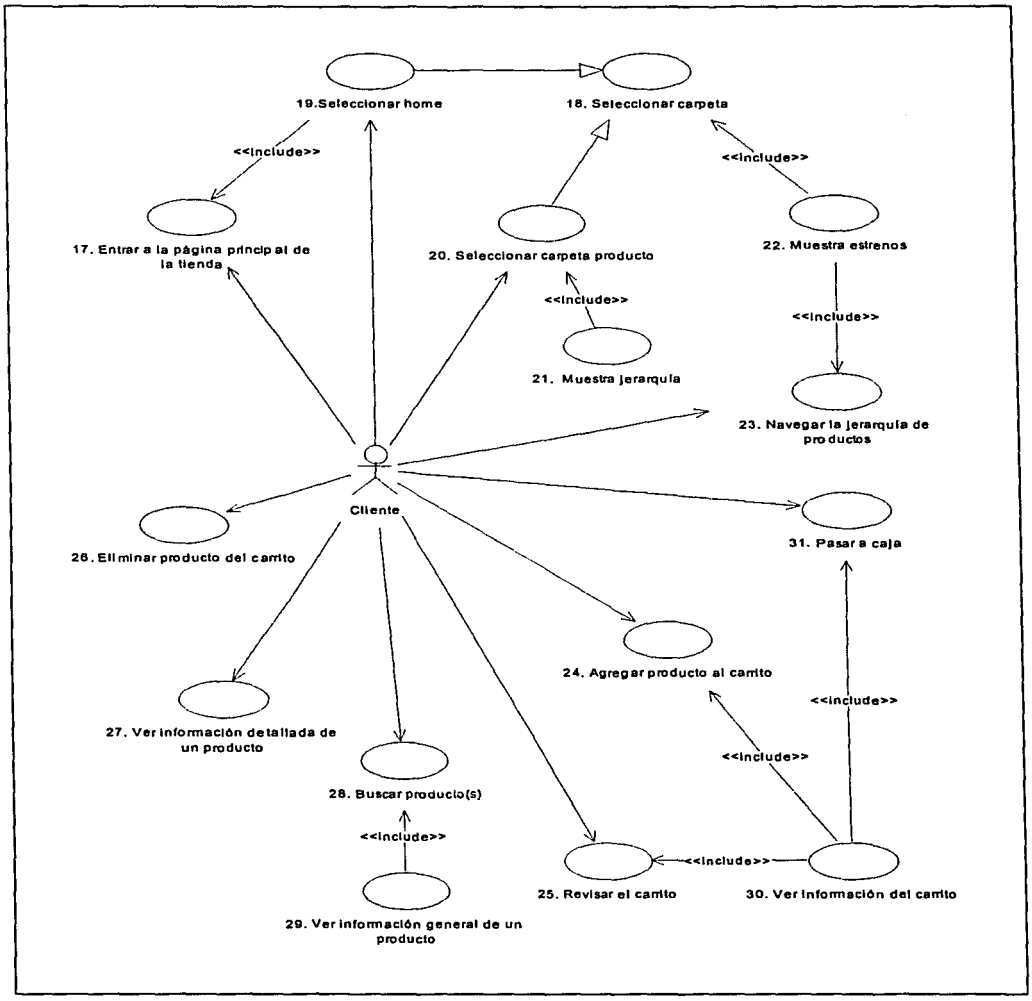


Figura 4.2 Diagrama de casos de uso para el sitio Web.

4.1.2.2 Descripción de los actores

Los actores representan el entorno del sistema. Para el sitio de comercio electrónico se tienen dos actores:

Administrador : Realiza las altas, bajas y cambios de la información de los productos.
Cliente : Funge como comprador en la tienda la cual se implementa como sitio Web.

4.1.2.3 Descripción de los flujos en los casos de uso

Los casos de usos capturan los requerimientos funcionales del sistema además de los no funcionales para cada uno de ellos. En este trabajo no hubo algún requerimiento no funcional específico para algún caso de uso. Los casos de uso son descritos mediante sus flujos. Debido a que el Proceso Unificado no especifica una forma de describir esos flujos entonces en esta Sección se propone y se aplica una. Para cada caso de uso se muestran los flujos alternativos y el principal además de su precondición (estado del sistema antes de iniciar el caso de uso) y poscondición (estado del sistema después de ejecutar el caso de uso). El flujo principal es aquel que representa mejor la función que ofrece el caso de uso. Los flujos alternativos son pequeñas variantes del flujo principal. La precondición o poscondición que no se hace explícita se asume como *"cualquier estado"*.

En los flujos de los casos de uso se denota mediante el enunciado:

- *extends(<algún caso de uso>)* la relación extends del diagrama de casos de uso.
- *include(<algún caso de uso>)* la relación include del diagrama de casos de uso.
- *inherit(<algún caso de uso>)* la relación de generalización del diagrama de casos de uso.

A cada flujo se le asocia un identificador formado de la siguiente manera:

- Al flujo principal se le asocia el mismo identificador que al caso de uso del cual proviene.
- Cada flujo alternativo es identificado por un número de punto flotante cuya parte entera es el identificador del caso de uso del cual proviene y la parte decimal se usa para distinguirlo de otros flujos.

Administración

1. Caso de uso: "Entrar al Almacén"

Flujo Principal: El Administrador inicia el Sistema. El Administrador puede hacer cualquiera de las siguientes acciones: añadir, eliminar, modificar y buscar productos, además de poder marcarlos como estrenos o quitarles dicha marca. El caso de uso reinicia.
Poscondición: El Sistema permite al Administrador escoger cualesquiera de las acciones descritas en el flujo principal.

Flujos Alternativos:

1.1 El Administrador indica al Sistema que desea salir del mismo. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, la ejecución del Sistema y el caso de uso terminan. Si no confirma la salida el caso de uso continua.
1.2 extends(Problema en la Base de Datos). El caso de uso termina.

2. Caso de uso: "Añadir producto"

Precondición: El Administrador seleccionó el caso de uso "Añadir producto".
Flujo Principal: El Administrador proporciona al Sistema el tipo de producto que desea añadir (CD, DVD, Libro). El Administrador indica al Sistema que guarde el producto. El Sistema guarda la información del producto. El caso de uso reinicia.
Poscondición: El producto es agregado al Almacén. El sistema esta en el estado en el que permite volver a hacer "Añadir producto".

Flujos Alternativos:

2.1 El Administrador proporciona al Sistema el tipo de producto que desea añadir (CD, DVD, Libro). El Administrador indica al Sistema que guarde el producto. El Sistema no guarda el producto y avisa que el producto ya existe. El caso de uso reinicia.
2.2 extends(Limpiar). El caso de uso reinicia.
2.3 El Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, el caso de uso termina. En caso contrario el caso de uso continua.
2.4 extends(Problema en la Base de Datos). El caso de uso reinicia.

3. Caso de uso: "Añadir CD" inherit(Añadir producto)

Precondición: El Administrador seleccionó el caso de uso "Añadir CD".
Flujo Principal: El Administrador proporciona los datos del CD que serán posteriormente guardados por el Sistema. Los datos son: código de barras, título, género, portada, precio, intérprete y disquera. De estos datos es obligatorio proporcionar el código de barras, el título y el precio, el resto de los datos pueden ser no proporcionados.
Poscondición: Los datos del CD están capturados.

4. Caso de uso: "Añadir DVD" inherit(Añadir producto)

Precondición: El Administrador seleccionó el caso de uso "Añadir DVD".
Flujo Principal: El Administrador proporciona los datos del DVD que serán posteriormente guardados por el Sistema. Los datos son: código de barras, título, género, portada, precio, elenco, región y clasificación. De estos datos es obligatorio proporcionar el código de barras, el título y el precio, el resto de los datos pueden ser no proporcionados.
Poscondición: Los datos del DVD están capturados.

5. Caso de uso: "Añadir Libro" inherit(Añadir producto)

Precondición: El Administrador seleccionó el caso de uso "Añadir Libro".
Flujo Principal: El Administrador proporciona los datos del Libro que serán posteriormente guardados por el Sistema. Los datos son: código de barras, título, género, portada, precio, autor y editorial. De estos datos es obligatorio proporcionar el código de barras, el título y el precio, el resto de los datos pueden ser no proporcionados.
Poscondición: Los datos del Libro están capturados.

6. Caso de uso: "Eliminar producto"

Precondición: El Administrador seleccionó el caso de uso "Eliminar producto".
Flujo Principal: El Administrador proporciona el código de barras del producto que desea eliminar. El Sistema elimina el producto. El caso de uso reinicia.
Poscondición: El producto es eliminado del Almacén. El sistema esta en el estado en el que permite volver a hacer "Eliminar producto".

Flujos Alternativos:

6.1 El Administrador proporciona el código de barras del producto que desea eliminar. El Sistema le avisa que el producto no existe. El caso de uso reinicia.
6.2 extends(Limpiar). El caso de uso reinicia.
6.3 El Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, el caso de uso termina. En caso contrario el caso de uso continua.
6.4 extends(Problema en la Base de Datos). El caso de uso reinicia.

7. Caso de uso: "Modificar producto"

Precondición: El Administrador seleccionó el caso de uso "Modificar producto".
Flujo Principal: El Administrador proporciona el código de barras del producto que desea modificar. El Sistema recupera la información del producto y admite su modificación. El Administrador hace los cambios de información. El Administrador solicita al Sistema que guarde la información. El Sistema guarda los cambios. El caso de uso reinicia.
Poscondición: La información del producto es modificada. El sistema esta en el estado en el que permite volver a hacer "Modificar producto".

Flujos Alternativos:

7.1 El Administrador proporciona el código de barras del producto que desea modificar. El Sistema avisa que el producto no existe. El caso de uso reinicia.
7.2 extends(Limpiar). El caso de uso reinicia.
7.3 El Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, el caso de uso termina. En caso contrario el caso de uso continua.
7.4 extends(Problema en la Base de Datos). El caso de uso reinicia.

8. Caso de uso: "Marcar estrenos"

Precondición: El Administrador seleccionó el caso de uso "Marcar estrenos". Los productos que se quieren marcar como estreno existen.
Flujo Principal: include(Buscar Producto). El Administrador selecciona del resultado de la búsqueda los productos que desea marcar como estrenos. El Sistema marca los productos seleccionados como estrenos. El caso de uso reinicia.
Poscondición: Los productos que fueron seleccionados por el Administrador para ser marcados como estrenos están marcados como estrenos. El sistema esta en el estado en el que permite volver a hacer "Marcar estrenos".

Flujos Alternativos:

8.1 extends(Limpiar). El caso de uso reinicia.
8.2 El Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, el caso de uso termina. En caso contrario el caso de uso continua.
8.3 extends(Problema en la Base de Datos). El caso de uso reinicia.

9. Caso de uso: "Desmarcar estrenos"

Precondición: El Administrador seleccionó el caso de uso "Desmarcar estrenos". Los productos a los que se les quiere quitar la marca de estreno están marcados como estrenos.
Flujo Principal: El Sistema muestra todos los productos marcados como estreno al Administrador. El Administrador selecciona los productos a desmarcar. El Sistema quita la marca de estreno a los productos seleccionados. El caso de uso reinicia.
Poscondición: Los productos que fueron seleccionados por el Administrador ya no tienen la marca de estreno. El sistema esta en el estado en el que permite volver a hacer "Desmarcar estrenos".

Flujos Alternativos:

9.1 El Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, el caso de uso termina. En caso contrario el caso de uso continua.
9.2 extends(Problema en la Base de Datos). El caso de uso termina.

10. Caso de uso: "Buscar producto"

Precondición: El Administrador seleccionó el caso de uso "Buscar producto" o "Desmarcar estrenos".
Flujo Principal: El Administrador proporciona al Sistema el tipo de producto a buscar. El Administrador solicita al Sistema efectuar la búsqueda. El Sistema le muestra los productos solicitados resultantes de la búsqueda. El caso de uso reinicia.
Poscondición: Se muestran los productos solicitados. El sistema esta en el estado en el que permite volver a hacer "Buscar producto".

Flujos Alternativos:

10.1 El Administrador proporciona al Sistema el tipo de producto a buscar (CD, DVD, Libro). El Administrador solicita al Sistema efectuar la búsqueda. El Sistema le avisa que no hay productos resultantes de la búsqueda. El caso de uso reinicia.
10.2 El Administrador proporciona al Sistema el tipo de producto a buscar (CD, DVD, Libro). El Administrador no proporciona un criterio de búsqueda. El Sistema le muestra todos los productos relacionados al tipo de producto seleccionado. El caso de uso reinicia.
10.3 extends(Limpiar). El caso de uso reinicia.
10.4 El Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, el caso de uso termina. En caso contrario el caso de uso continua.
10.5 extends(Problema en la Base de Datos). El caso de uso reinicia.

11. Caso de uso: "Buscar CD" inherit(Buscar producto)

Precondición: El Administrador seleccionó el caso de uso "Buscar CD".
Flujo Principal: El Administrador proporciona un criterio para la búsqueda de los CDs. El criterio de búsqueda se forma mediante la combinación de la siguiente información: título, intérprete, género.
Poscondición: El criterio de búsqueda está capturado.

12. Caso de uso: "Buscar DVD" inherit(Buscar producto)

Precondición: El Administrador seleccionó el caso de uso "Buscar DVD".
Flujo Principal: El Administrador proporciona un criterio para la búsqueda de los DVD. El criterio de búsqueda se forma mediante la combinación de la siguiente información: título, elenco, género.
Poscondición: El criterio de búsqueda está capturado.

13. Caso de uso: "Buscar Libro" inherit(Buscar producto)

Precondición: El Administrador seleccionó el caso de uso "Buscar Libro".
Flujo Principal: El Administrador proporciona un criterio para la búsqueda de los libros. El criterio de búsqueda se forma mediante la combinación de la siguiente información: título, autor, editorial.
Poscondición: El criterio de búsqueda está capturado.

14. Caso de uso: "Limpiar"

Precondición: El Administrador seleccionó el caso de uso "Limpiar".
Flujo Principal: El caso de uso se inicia cuando el Administrador solicita al Sistema limpiar los datos capturados. Los datos capturados son los datos que proporciona el Administrador correspondientes al caso de uso desde el que se llama "Limpiar". El Sistema limpia los datos. El caso de uso termina.
Poscondición: Los campos son limpiados.

15. Caso de uso: "Salir"

Precondición: El Administrador está realizando alguna actividad.
Flujo Principal: El caso de uso se inicia cuando el Administrador indica al Sistema que desea salir de la actividad que está realizando. El Sistema pide al Administrador la confirmación de la salida. Si el Administrador confirma la salida, la ejecución del Sistema termina. Si no confirma la salida, el Administrador puede continuar con la actividad mencionada en la precondición. El caso de uso termina.
Poscondición: Si el Administrador confirma la salida, la ejecución del Sistema termina. Si no confirma la salida, el Administrador continua con la actividad mencionada en la precondición.

16. Caso de uso: "Problema en la Base de Datos"

Precondición: El Administrador realiza alguna transacción en la Base de Datos.
Flujo Principal: El caso de uso se inicia cuando surge un problema en la Base de Datos. El Sistema avisa al Administrador sobre dicho problema. El caso de uso termina.
Poscondición: El Sistema avisa al Administrador de un problema en la Base de Datos.

Sitio Web

17. Caso de uso: "Entrar a la página principal de la tienda"

Precondición: El Sitio existe.
Flujo Principal: El caso de uso inicia cuando el Cliente solicita entrar al Sitio de la tienda mediante la Internet. Se observan dentro de la página referencias hacia la carpeta *home* y carpetas relacionadas con los tipos de producto. include(Seleccionar home). El caso de uso termina.
Poscondición: Se muestra la página principal de la tienda al Cliente.

18. Caso de uso: "Seleccionar carpeta"

Precondición: El Cliente está en el Sitio.
Flujo Principal: El caso de uso inicia cuando el Cliente selecciona una carpeta. El Sistema muestra los estrenos include(Muestra estrenos). El caso de uso termina.
Poscondición: Una carpeta está seleccionada.

19. Caso de uso: "Seleccionar home" Inherit(Seleccionar carpeta)

Precondición: El Cliente selecciona la carpeta *home*.
Flujo Principal: El Sistema muestra los estrenos de todos los productos. El caso de uso termina.
Poscondición: La carpeta *home* está seleccionada.

20. Caso de uso: "Seleccionar carpeta producto" Inherit(Seleccionar carpeta)

Precondición: El Cliente selecciona la carpeta de un tipo de producto.
Flujo Principal: El Sistema muestra los estrenos correspondientes al tipo de producto de la carpeta seleccionada por el Cliente en la precondición, include(Muestra jerarquía). El caso de uso termina.
Poscondición: El Cliente está en la carpeta que seleccionó en la precondición.

21. Caso de uso: "Muestra jerarquía"

Precondición: El Cliente está en la carpeta de un tipo de producto.
Flujo Principal: Dependiendo del tipo de producto en la carpeta se muestra una jerarquía:

- La jerarquía de CDs tiene las siguientes categorías: Pop, Clásica, Alternativo.
- La jerarquía de DVDs tiene las siguientes categorías: Acción, Suspenso.
- La jerarquía de libro tiene las siguientes categorías: Computación, Novela.

El caso de uso termina.
Poscondición: El Cliente está en la carpeta de la precondición.

22. Caso de uso: "Muestra estrenos"

Precondición: El Cliente está en la carpeta de un tipo de producto.
Flujo Principal: Dependiendo del tipo de producto en la carpeta se muestra la información:

- CD: Título, portada, precio, intérprete.
- DVD: Título, elenco, portada, precio.
- Libro: Título, portada, precio, autor, editorial.

El caso de uso termina.
Poscondición: El Cliente está en la carpeta de la precondición.

23. Caso de uso: "Navegar la jerarquía de productos"

Precondición: El Cliente está en la carpeta de un tipo de producto. El Cliente escoge una de las categorías de la jerarquía mostrada en la carpeta antes referida.
Flujo Principal: El Sistema despliega los estrenos que pertenecen a la categoría elegida por el Cliente, include(Muestra estrenos). El caso de uso termina.
Poscondición: El Cliente está en la carpeta de la precondición.

24. Caso de uso: "Agregar producto al carrito"

Precondición: El Cliente está en una carpeta. El Cliente escoge comprar un producto.
Flujo Principal:. El Sistema añade al carrito del Cliente el producto que éste escogió en la precondición. include(Ver información del carrito). El caso de uso termina.
Poscondición: Un producto es agregado al carrito. El Cliente está en la carpeta de la precondición.

25. Caso de uso: "Revisar el carrito"

Precondición: El Cliente está en una carpeta. El Cliente indica al Sistema que desea ver el carrito.
Flujo Principal: include(Ver información del carrito). El caso de uso termina.
Poscondición: El Cliente está en la carpeta de la precondición.

Flujos Alternativos:
25.1 El Cliente modifica el número de unidades de algún producto en el carrito. El caso de uso termina.

26. Caso de uso: "Eliminar producto del carrito"

Precondición: El Cliente está en una carpeta y puede ver el carrito.
Flujo Principal: El caso de uso inicia cuando el Cliente indica al Sistema que deséa eliminar un producto del carrito. El Sistema elimina del carrito el producto elegido por el Cliente. El caso de uso termina.
Poscondición: El Cliente está en la carpeta de la precondición y los productos elegidos por el Cliente fueron eliminados del carrito.

27. Caso de uso: "Ver información detallada de un producto"
Precondición: El Cliente está en una carpeta. El Cliente indica al Sistema que desea ver la información detallada de un producto.
Flujo Principal: El Sistema muestra la información del producto elegido por el Cliente en la precondición. Dependiendo del tipo de producto se muestra la información:

- CD: Título, género, portada, precio, intérprete, disquera.
- DVD: Título, género, portada, precio, elenco, región, clasificación.
- Libro: Título, género, portada, precio, autor, editorial.

El caso de uso termina.
Poscondición: La información detallada del producto seleccionado por el Cliente en la precondición es desplegada.

28. Caso de uso: "Buscar productos"

Precondición: El Cliente está en la carpeta de un tipo de producto.
Flujo Principal: El Cliente proporciona un criterio simple para la búsqueda de los productos. El criterio de búsqueda depende del tipo de producto en la carpeta:

- CD: Título, intérprete, género.
- DVD: Título, elenco, género.
- Libro: Título, autor, editorial.

El Cliente solicita al Sitio efectuar la búsqueda. El Sitio le muestra los productos solicitados resultantes de la búsqueda. Para cada producto resultante de la búsqueda se realiza include(Ver información general de un producto). El caso de uso termina.
Poscondición: El resultado de la búsqueda es desplegado en la carpeta de la precondición.

29. Caso de uso: "Ver información general de un producto"

Precondición: El Cliente está la carpeta de un tipo de producto.
Flujo Principal: El caso de uso inicia cuando el Sistema muestra la información general de un producto. Dependiendo del tipo de producto se muestra la información:

- CD: Título, precio, intérprete, disquera.
- DVD: Título, precio, región.
- Libro: Título, precio, autor, editorial.

El caso de uso termina.
Poscondición: La información general del producto es desplegada en la carpeta de la precondición.

30. Caso de uso: "Ver información del carrito"

Precondición: El Cliente está en una carpeta.
Flujo Principal: El caso de uso inicia cuando el Sistema muestra la información del carrito. La información del carrito es:

- Título del producto.
- Costo unitario del producto.
- Número de unidades de cada producto.
- Subtotal por producto.
- Total de la compra.

El caso de uso termina.
Poscondición: La información del carrito es desplegada. El Cliente está en la carpeta de la precondición.

31. <u>Caso de uso:</u> "Pasar a caja"

Precondición: El Cliente tiene productos en el carrito y éste esta siendo visto por el cliente. El Cliente indica al Sistema que desea comprar los productos del carrito.
Flujo Principal: El Sistema pide al Cliente los datos para la compra. Todos los datos son obligatorios. Estos son:

Datos personales del Cliente:
- Nombre.
- Apellido.
- Teléfono.

Datos para la entrega de los productos de la compra:
- Dirección.
- Colonia.
- Código Postal.
- Ciudad.
- Estado.

Forma de pago:
- Puede ser Tarjeta de Crédito o Pago contra Entrega pero no ambos.

Si la forma de pago es la Tarjeta de Crédito entonces se agregan los datos de la misma:
- Nombre del tarjetahabiente.
- Número de la tarjeta.
- Tipo (Visa o MasterCard).
- Fecha de expiración (mes y año).

El Cliente proporciona al Sistema los datos anteriores. El Sistema muestra al Cliente los datos que proporcionó y los del carrito, include(Ver información del carrito). El Cliente indica al Sistema que realice la compra. El Sistema guarda los datos de la compra y devuelve al Cliente su número de pedido. El caso de uso termina.
Poscondición: El Sistema tiene los datos de la compra del Cliente.

Flujos Alternativos:
31.1 El Cliente cancela la compra. El Sistema elimina los productos del carrito. La compra no es realizada. El caso de uso termina.

4.1.3 Requerimientos suplementarios para el sitio

Es una buena práctica tener en cuenta desde una etapa temprana los requerimientos no funcionales del sistema. Estos influyen fuertemente en los flujos de trabajo posteriores al de captura de requerimientos. Teniendo en cuenta estos requerimientos se evita cometer errores y por consiguiente pérdida de recursos.

La importancia de tener en cuenta desde una etapa temprana los requerimientos no funcionales del sistema se hizo evidente durante el desarrollo del sitio de comercio electrónico cuando se eligió usar la plataforma J2EE y no solamente el J2SDK1.3 y JSPs los cuales se habían contemplado para el desarrollo en un principio. Se tuvo que invertir una gran cantidad de tiempo (aproximadamente 6 meses) para entender, aprender, instalar y manejar de buena manera la tecnología J2EE así como para encontrar alguna base de datos compatible con dicha tecnología (en este caso ORACLE8i ya que era la opción más viable por su disponibilidad y compatibilidad de su driver con J2EE). Este hecho representó cambios en el diseño y la fecha prevista para la entrega del sistema.

4.1.3.1 Restricción para la interfaz de usuario

Las interfaces serán gráficas.

4.1.3.2 Restricciones físicas

Hardware

Servidores :

- El Servidor donde reside JBoss 2.2.2 y Tomcat 3.2.2 requiere 64 Mb RAM y 30 Mb en disco duro.
- El Servidor donde reside la Base de Datos ORACLE8i requiere como mínimo 128 Mb RAM y 985 Mb en disco duro.

Nota: JBoss 2.2.2, Tomcat 3.2.2 y la Base de Datos ORACLE8i pueden residir en el mismo Servidor.

Administrador :

- Computadora que tenga como mínimo un procesador Pentium 166 Mhz, 32 Mb RAM y 40 Mb en disco duro.

Cliente :

- Computadora que soporte un navegador de Internet .

4.1.3.3 Restricciones de diseño / implementación

Lenguajes

- Java.
- HTML.
- Scripts para Base de Datos ORACLE.

Tecnologías

- Java 2 SDK, Standard Edition, v 1.3.
- J2EE (JSP, EJB 1.1, JDBC 2.0, JNDI, RMI, JTA)

Software

Servidores :

- Java 2 Runtime Environment, Standard Edition, v 1.3.
- Servidor de Aplicaciones JBoss 2.2.2.
- Servidor para Servlets Tomcat 3.2.2.
- Servidor de Base de Datos ORACLE8i.

Administrador :

- Java 2 Runtime Environment, Standard Edition, v 1.3.

Cliente :

- Navegador de Internet.

Instalación

Instalar en el Administrador :

- Java 2 Runtime Environment, Standard Edition, v 1.3.

Instalar en el Cliente :

- Navegador de Internet.

Instalar en el Servidor :

- Java 2 Runtime Environment, Standard Edition, v 1.3.
- Servidor de Aplicaciones JBoss 2.2.2.
- Servidor para Servlets Tomcat 3.2.2.
- Servidor de Base de Datos ORACLE8i.

Colocar en el Administrador :

- .jar para la Administración del Almacén.

Colocar en el Servidor de Aplicaciones JBoss 2.2.2 :

- .jar para los EJB y componentes Web.

Correr el script para la creación de la Base de Datos ORACLE.

4.1.3.4 Estándar de nombrado en la codificación

- *Se usarán los estándares de programación para el lenguaje Java.*
- *Para EJB se usarán adicionalmente las siguientes convenciones para el nombrado de clases:*
  - La interfaz Home se nombra: (Nombre de la Interfaz Remota)Home.
  - La clase de definición del EJB se nombra: (Nombre de la Interfaz Remota)Bean.

4.1.3.5 Otros requerimientos

Documentación

Se escribirá el Manual de Usuario para el Administrador del Sistema.

4.2 Análisis

4.2.1 Modelo de análisis

La manera en que se utilizó el análisis fue como una herramienta intermedia que permitiera refinar los requerimientos y prepararse para el diseño. El modelo de análisis no fue mantenido durante todo el ciclo de desarrollo y el seguimiento con el modelo de diseño puede no ser tan directo, sin embargo conserva un buen seguimiento con el modelo de casos de uso.

El análisis permitió poder empezar a pensar en la funcionalidad interna del sistema de manera que se pudiera esbozar cuales eran las clases que iban a participar en la resolución del problema. En este rubro fue de gran utilidad separar las clases participantes mediante los estereotipos *Boundary*, *Control* y *Entity* pues obliga a pensar en términos de actividades más especializadas que puedan ser repartidas entre varias clases. Ayuda a empezar a pensar en una arquitectura del sistema que permita relacionar clases que posteriormente serán parte de paquetes en el diseño.

Conforme se avanzaba en el análisis surgían nuevas preguntas acerca de los requerimientos. Las clases de análisis al refinar la funcionalidad descrita por los casos de uso permitieron descubrir inconsistencias y mejores maneras de realizar funcionalidades. También se pudo empezar a organizar y agrupar el comportamiento en clases. Se empezó a intuir como se podía reutilizar el comportamiento de una clase en varias *realizaciones de caso de uso*, esto fue más evidente en las clases de estereotipo *Control*. Durante el análisis no se detectó algún requerimiento no funcional específico para alguna clase por lo que no se especifican "requerimientos especiales" para las clases.

La mayor parte del modelo de análisis fue desarrollado de manera temprana durante los ciclos pues era pieza fundamental para empezar a realizar un buen diseño.

Desarrollar el modelo de análisis fue una buena opción para refinar los requerimientos además de sentar una base más sólida para el diseño.

En las siguientes secciones se muestran el diagrama general de clases de análisis y las *realizaciones de caso de uso-análisis* para el sistema. Estos diagramas se muestran en dos partes para su mejor lectura, una para el subsistema de administración y la otra para el subsistema del sitio Web.

4.2.1.1 Diagramas de clases

Las Figuras 4.3 y 4.4 presentan dos diagramas de clases. La Figura 4.3 muestra las clases de análisis de las cuales provienen los objetos que participan en las *realizaciones de caso de uso-análisis* para el subsistema de administración. De forma análoga la Figura 4.4 muestra las clases de análisis para el subsistema del sitio Web.

Administración

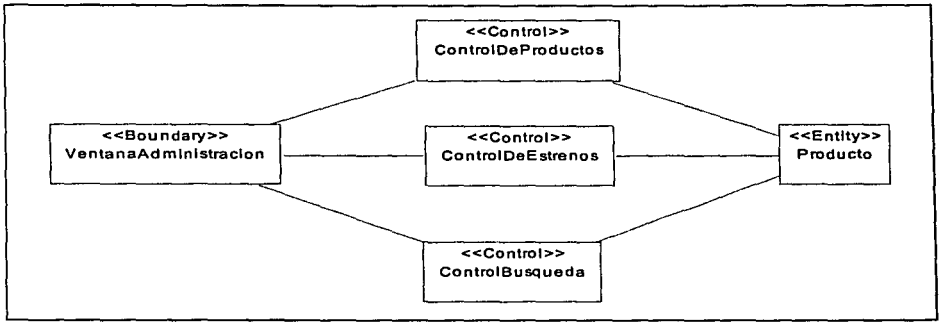


Figura 4.3 Diagrama de clases para la administración.

Sitio Web

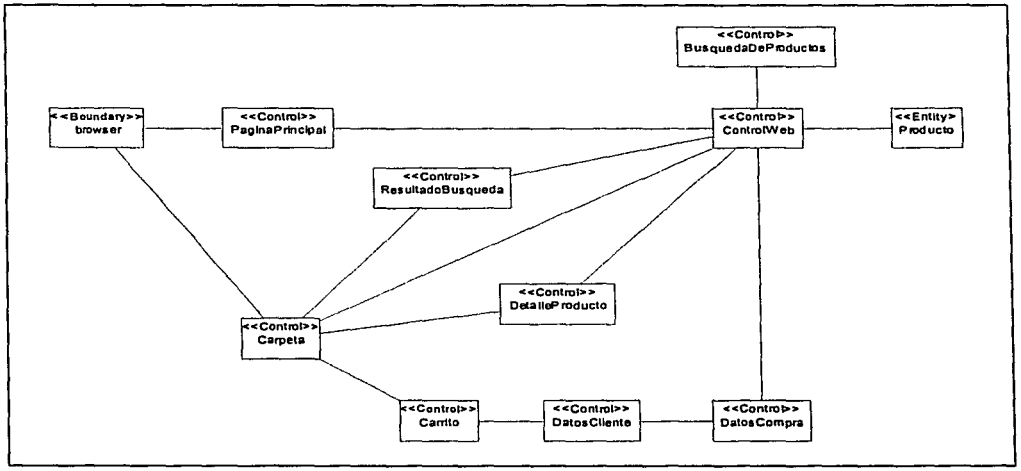


Figura 4.4 Diagrama de clases para el sitio Web.

4.2.1.2 Realizaciones de caso de uso-análisis

En las Figuras 4.5 y 4.6 se muestran las *realizaciones de caso de uso-análisis.* En los diagramas se observa el actor que interactúa con el sistema y las responsabilidades de cada objeto. Cada diagrama tiene a su lado derecho una nota que especifica el seguimiento de las *realizaciones de caso de uso-análisis* con los casos de uso.

El análisis del sitio fue hecho para gran parte de los casos de uso. Las necesidades de los casos de uso que no requirieron análisis fueron mejor capturadas durante el flujo de diseño. Estas dependieron en gran parte de clases de diseño posteriores que se realizaron para los casos de uso que sí tuvieron análisis. Por esta razón, no aparecen las *realizaciones de caso de uso-análisis* para todos los casos de uso.

Administración

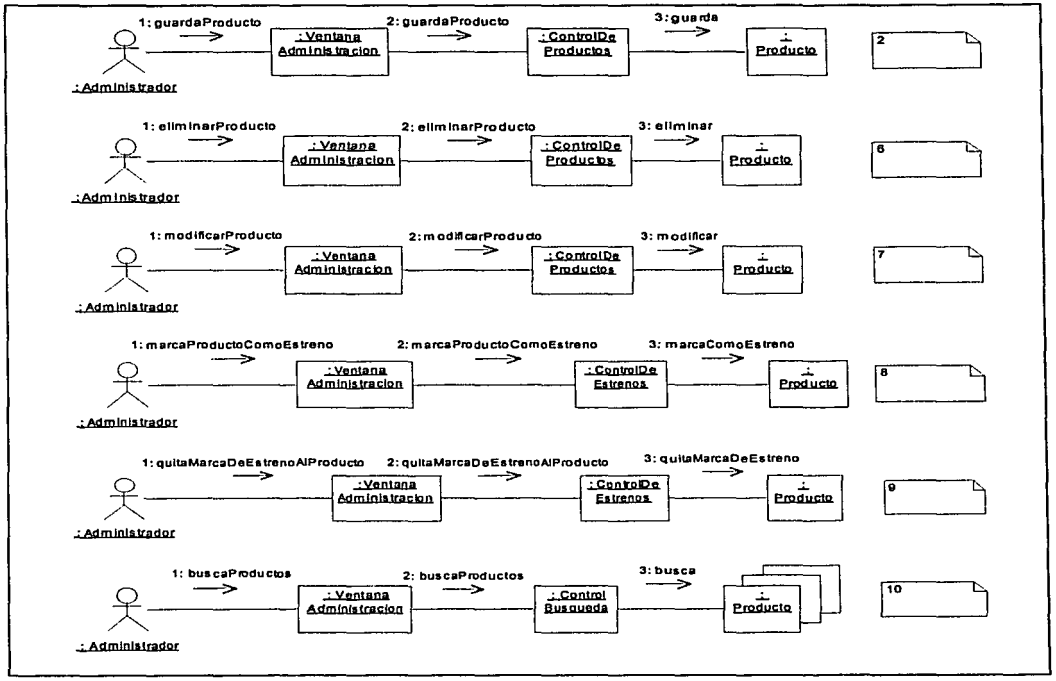
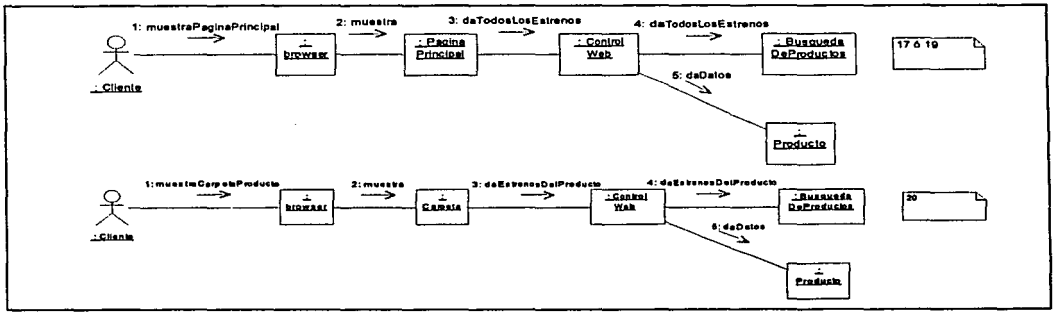


Figura 4.5 Realizaciones de caso de uso-análisis para la administración.

Sitio Web



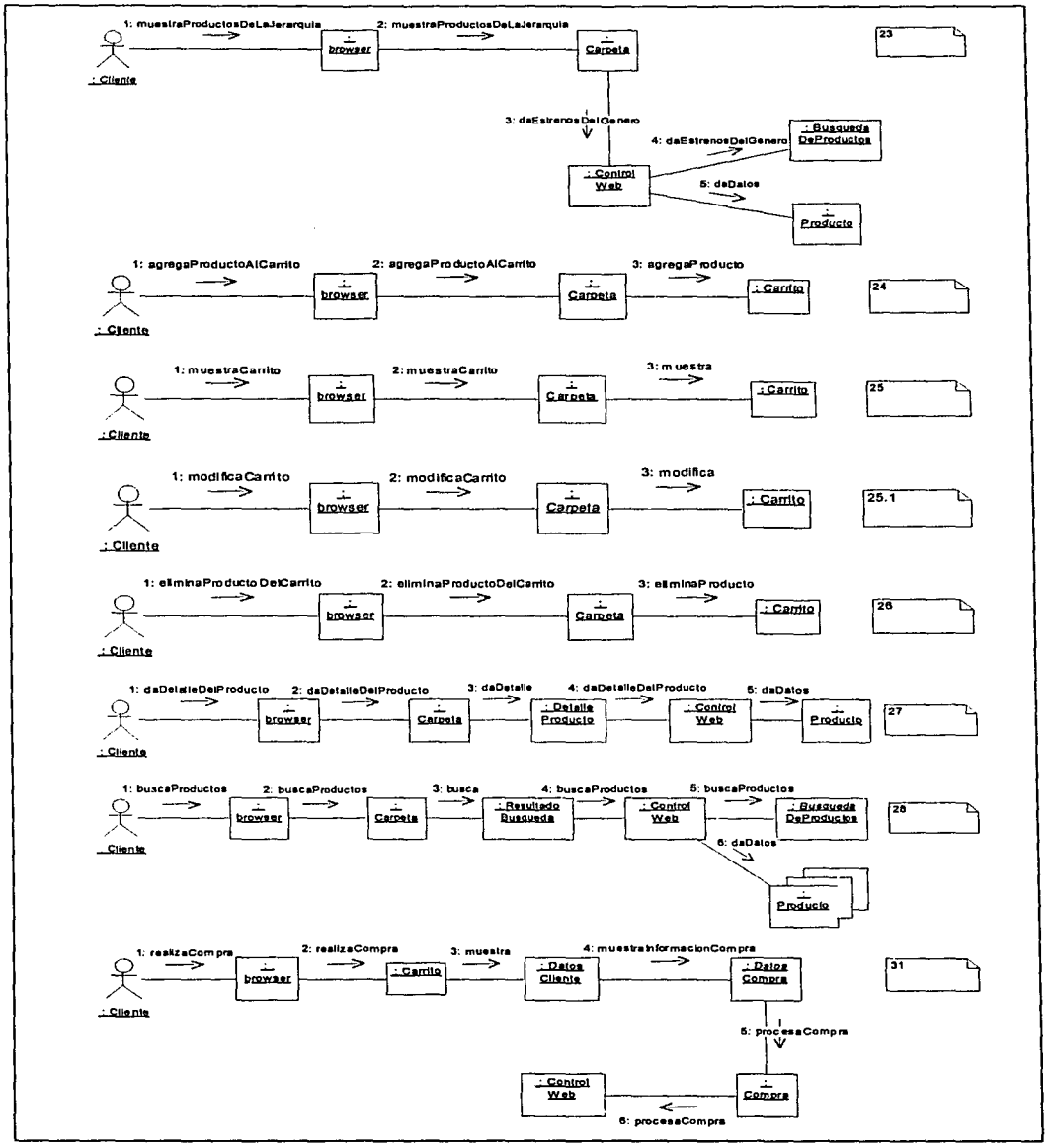


Figura 4.6 Realizaciones de caso de uso-análisis para el sitio Web.

4.3 Diseño

4.3.1 Modelo de diseño

El diseño del sitio, como la teoría lo indica, se vio altamente influenciado por la plataforma de desarrollo utilizada. Es este caso, la plataforma que se usó fue J2EE de la que se puede encontrar una descripción detallada dentro de la tesis complementaria [1] la cual es conveniente consultar antes de empezar con la lectura de los modelos de diseño, instalación e implementación. Dentro de esta Sección de diseño se irán haciendo algunas recomendaciones de qué partes de la tesis complementaria [1] consultar para entender lo que en su momento se esté desarrollando.

El seguimiento del modelo de diseño se dirige hacia el modelo de casos de uso pues el modelo de análisis solo fue usado como una herramienta intermedia que permitiera refinar los requerimientos y prepararse para el diseño, razón por lo cual no se le dio mantenimiento durante todo el ciclo de desarrollo.

El diseño fue realizado paralelamente con la implementación pues la tecnología J2EE era por primera vez experimentada por los desarrolladores y no se podía saber con anticipación cuales eran las decisiones convenientes para distribuir la funcionalidad del sistema entre las clases de diseño. Dado este marco, se tuvieron que realizar pruebas de implementación antes de empezar a diseñar y cambiar algunas decisiones de diseño después de haber programado. Todo esto permitió experimentar la conveniencia de poder contar dentro del equipo con gente que conozca la plataforma de desarrollo y así evitar los riesgos inherentes tales como la entrega retrasada del producto y en caso extremo el fracaso del mismo.

En las subsecuentes secciones dentro del diseño se tendrá con frecuencia la separación de la parte de la administración y del sitio Web que fueron identificadas desde la captura de requerimientos para una distribución más comprensible de los diagramas.

4.3.1.1 Subsistemas

Empecemos por analizar desde lo general a lo particular la manera en que es diseñado y distribuido el sitio de comercio electrónico.

La Figura 4.7 muestra los subsistemas en los que fue separado el sistema. Estos subsistemas fueron los subsistemas propios de la aplicación. Dentro de cada subsistema se muestran sus clases. También se muestra la relación que existe entre los subsistemas.

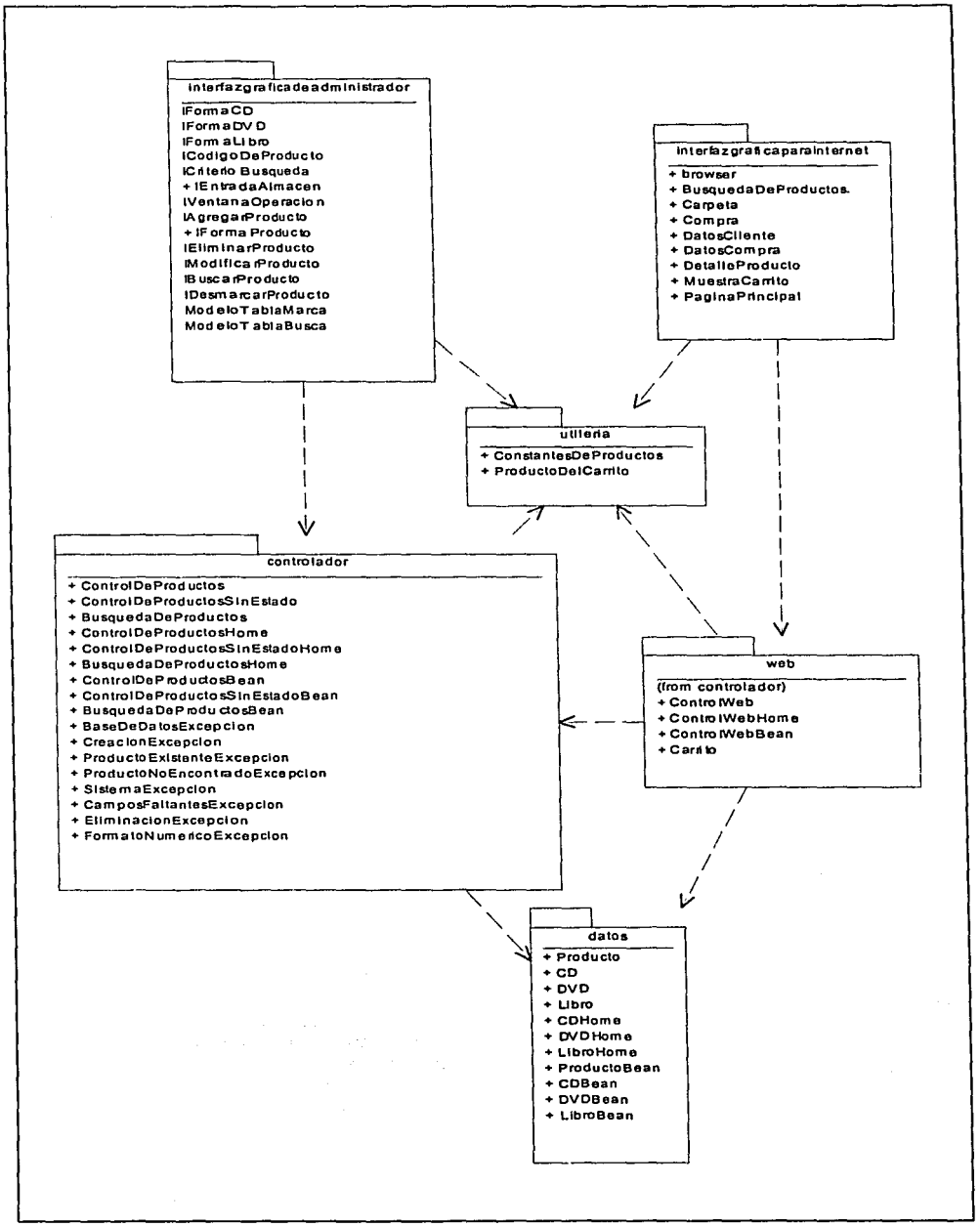


Figura 4.7 Subsistemas para el sitio de comercio electrónico.

Las Figuras 4.8 y 4.9 presentan dos diagramas que muestran como fueron utilizados los subsistemas para las partes de administración y sitio Web del sistema. La distribución de los subsistemas refleja el modelo multicapa distribuido que emplea J2EE; para un mejor entendimiento de dicho modelo consultar la tesis complementaria [1] en su Sección "2.2 J2EE."

Administración

La parte de administración refleja el modelo multicapa distribuido en su escenario cliente *stand-alone* descrito en "2.3.2 Escenario Cliente *Stand-Alone*" de la tesis complementaria [1]. Para la administración se utiliza el primero de los tipos descritos en "2.3.2 Escenario Cliente *Stand-Alone*". El tipo se describe de manera general como sigue:

El cliente EJB interactúa directamente con un servidor, esto es enterprise beans intstalados sobre un contenedor EJB. Se asume que RMI-IIOP será usado en este escenario y que el servidor EJB accederá los recursos EIS usando JDBC.

La manera en que el tipo descrito es usado es la siguiente:

- Las clases para el cliente EJB se encuentran en el subsistema *interfazgraficadeadministrador.*
- Los *enterprise beans* para el contenedor EJB se encuentran dentro de los subsistemas *controlador* y *datos* los cuales contienen *session beans* y *entity beans* respectivamente.
- El recurso EIS es una base de datos ORACLE8i.

RMI-IIOP y JDBC son protocolos de comunicación que adquieren interés hasta el modelo de instalación.

Sitio Web

La parte del sitio Web refleja el modelo multicapa distribuido en su escenario de aplicación multicapa descrito en "2.3.1 Escenario de Aplicación Multicapa" de la tesis complementaria [1].

La manera en que se usa el escenario es la siguiente:

- Se usan JSPs en el contenedor Web los cuales se encuentran en el subsistema *interfazgraficaparainternet.*
- Los *enterprise beans* para el contenedor EJB se encuentran dentro de los subsistemas *controlador*, *web* y *datos* de los cuales los dos primeros contienen *session beans* y el último *entity beans.*
- El recurso EIS es una base de datos ORACLE8i.

Para la comunicación entre el navegador de Internet y el contenedor Web se usa HTML y para la comunicación entre el contenedor EJB y la base de datos se usa JDBC. Estos dos últimos protocolos son tomados en cuenta hasta el modelo de instalación.

Cabe señalar que los subsistemas de *control* y *datos* son compartidos por la administración y el sitio Web del sistema. Esta reutilización de funcionalidad se manifestó durante el segundo ciclo mientras se implementaban los *enterprise beans* del subsistema *web* el cual pudo hacer uso de la funcionalidad definida en los subsistemas *controlador* y *datos.*

La Figura 4.8 muestra la distribución y relación entre los subsistemas para la parte de administración. Los subsistemas propios de la aplicación mostrados son: *datos, controlador, interfazgraficadeadministrador* y *utileria*. El resto de los subsistemas no son propios de la aplicación y son necesarios para desarrollar el sistema basado en la plataforma J2EE.

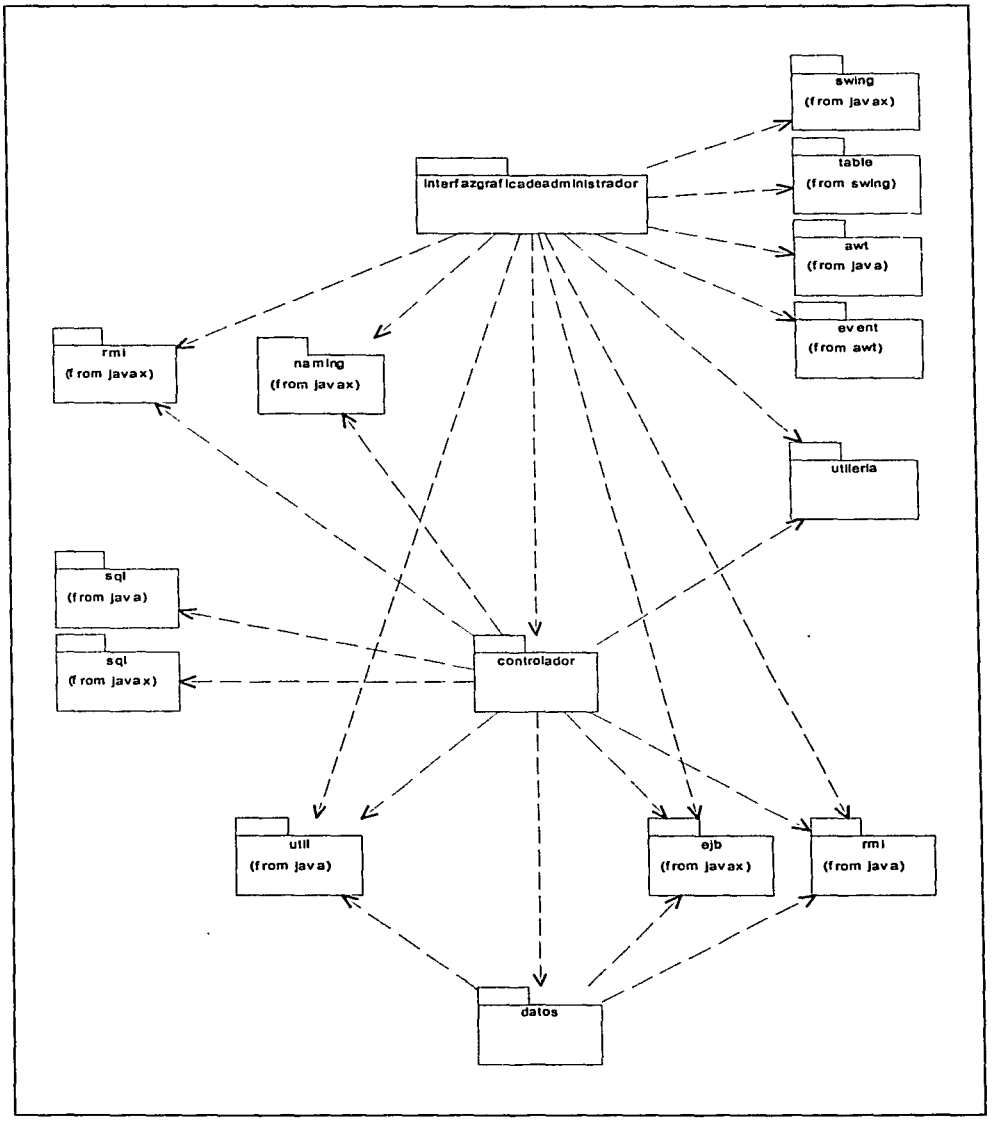


Figura 4.8 Subsistemas para la administración.

La Figura 4.9 muestra la distribución y relación entre los subsistemas para la parte del sitio Web. Los subsistemas propios de la aplicación mostrados son: *datos*, *web*, *controlador*, *interfazgraficaparainternet* y *utileria*. El resto de los subsistemas no son propios de la aplicación y son necesarios para desarrollar el sistema basado en la plataforma J2EE.

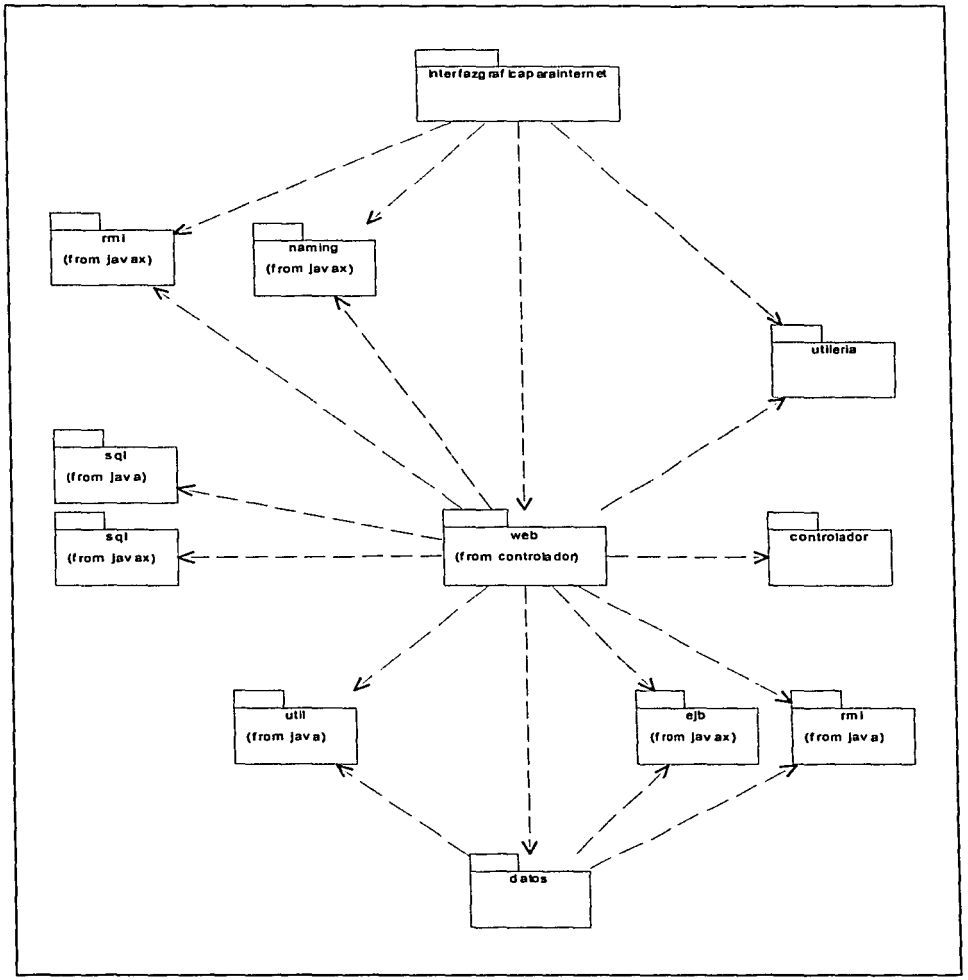


Figura 4.9 Subsistemas para el sitio Web.

4.3.1.2 Diagramas de clases para la relación entre subsistemas

Los siguientes diagramas de clases enfatizan la relación existente entre las clases que pertenecen a distintos subsistemas. Más tarde, en la siguiente Sección se muestra la relación de las clases que pertenecen al mismo subsistema.

Se puede notar dentro de algunas clases el uso de estereotipos que ya son propios de la plataforma de desarrollo J2EE a diferencia de los estereotipos usados en análisis donde solo se usaban *Boundary*, *Control* y *Entity*. Los estereotipos que se usaron dan una mayor expresividad a las clases en términos de la plataforma y se usan los siguientes:

- *Boundary*: Se conservó del análisis para denotar las clases de la interfaz gráfica para la administración.
- *Interface*: Para denotar una interfaz.
- *exception*: Denota las excepciones.
- *EJB*: Denota un *enterprise bean*.
- *JavaBean*: Denota un *Java bean*.
- *JSP*: Denota un JSP.

Administración

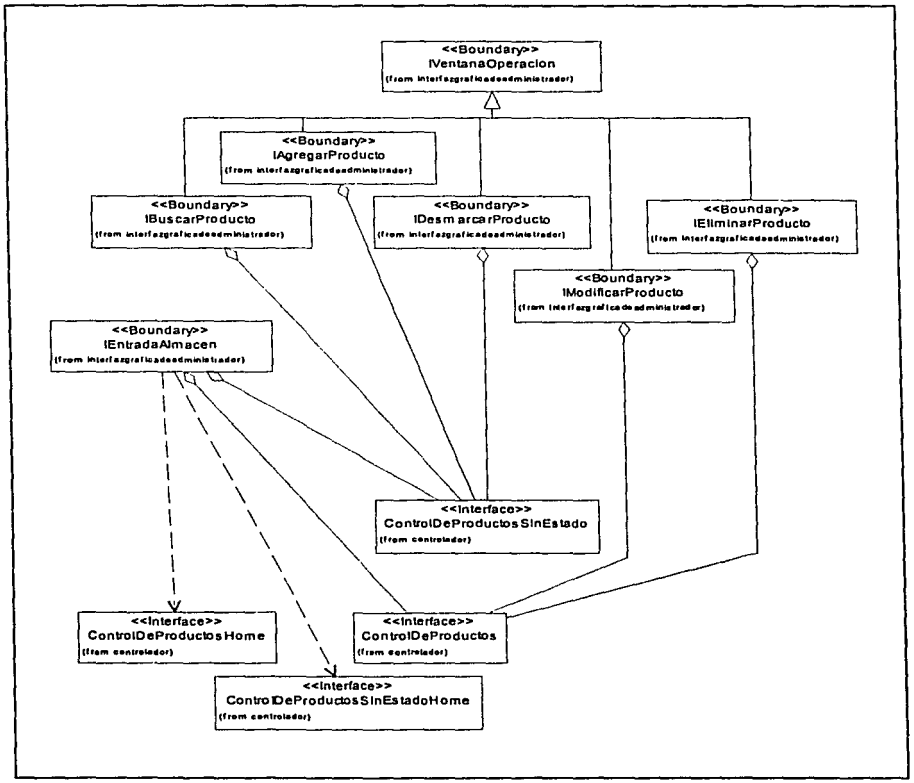


Figura 4.10 Diagrama de clases enfatizar la relación entre los subsistemas controlador e interfazgraficadeadministrador.

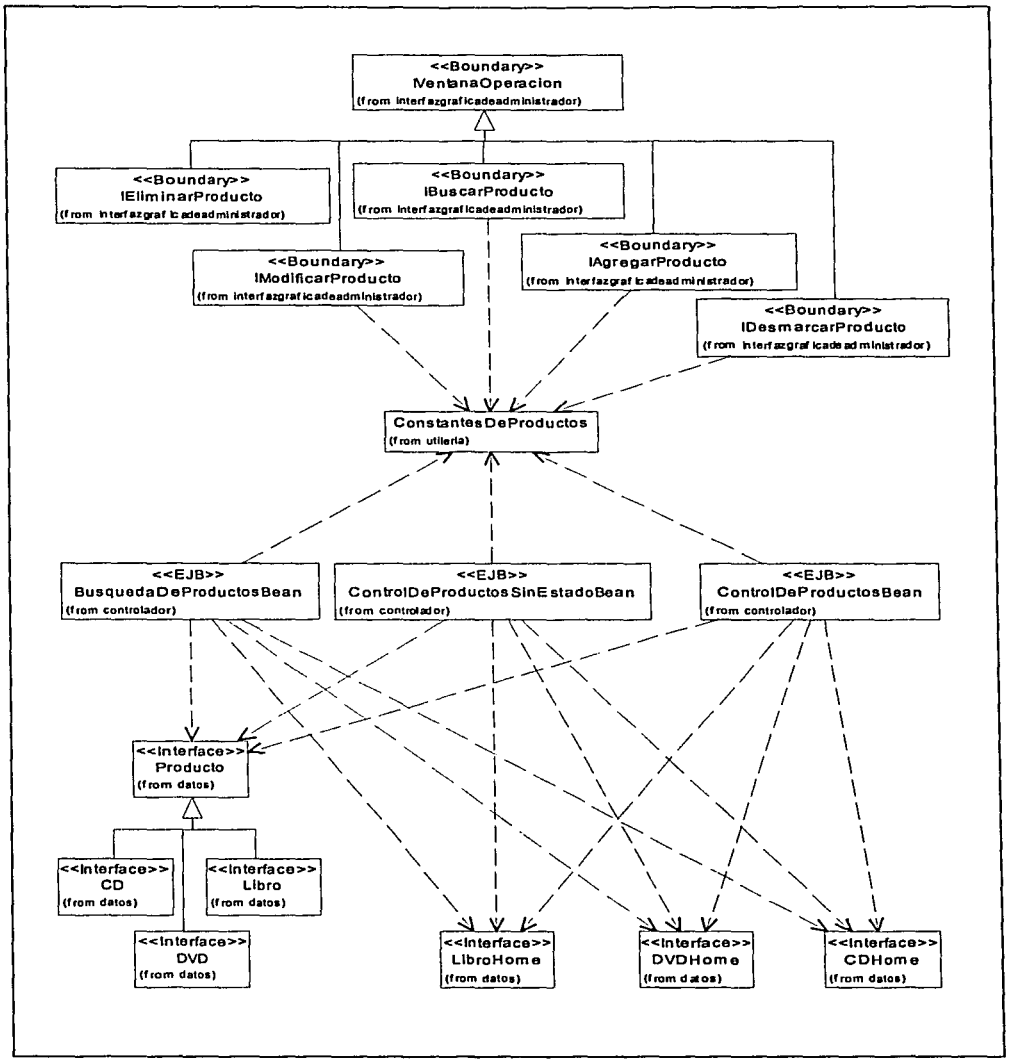


Figura 4.11 Diagrama de clases para enfatizar la relación entre los subsistemas controlador e interfazgraficadeadministrador con utileria; y la relación de controlador con datos.

El resto de los diagramas de clases para la administración de esta Sección se muestran en el Apéndice A, Sección A.2.

Sitio Web

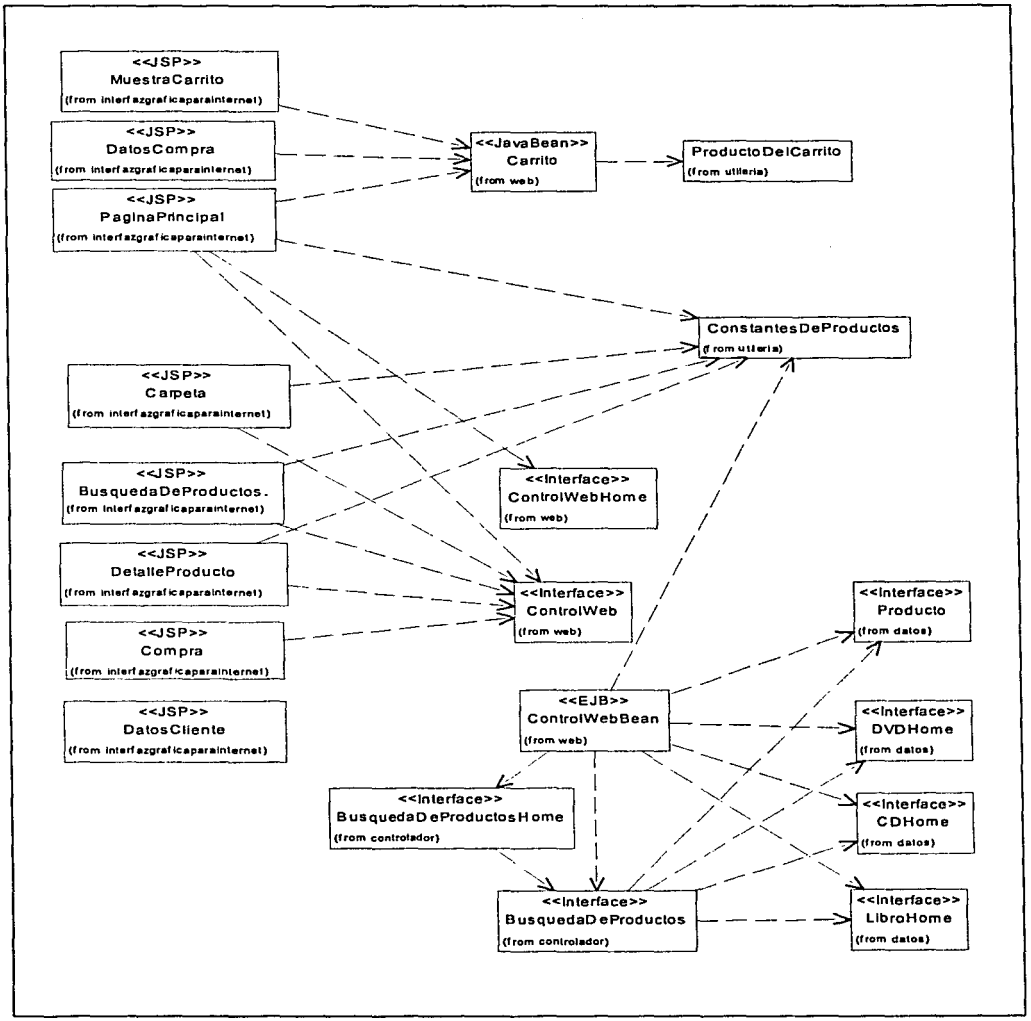


Figura 4.12 Diagrama de clases para enfatizar la relación entre los subsistemas del sitio Web.

4.3.1.3 Diagramas de clases de los subsistemas

En esta Sección se presentan los diagramas de clases para mostrar la relación existente entre las clases del mismo subsistema.

Subsistema interfazgraficadeadministrador

La Figura 4.13 muestra un diagrama de clases que enfatiza los atributos y métodos de las mismas. Las clases mostradas son ventanas correspondientes a la administración del sistema y son la interfaz con el administrador.

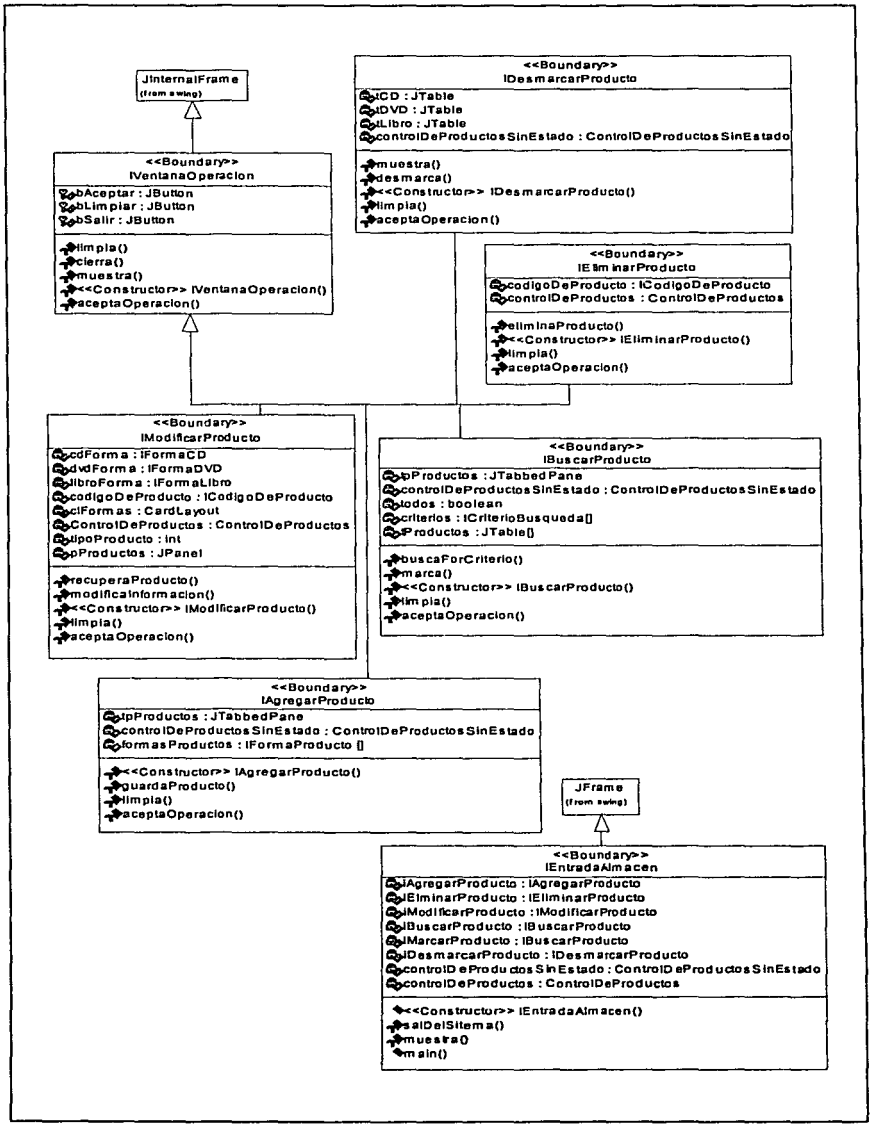


Figura 4.13 Diagrama de clases para el subsistema interfazgraficadeadministrador.

La Figura 4.14 muestra un diagrama de clases que enfatiza la relación entre las ventanas que sirven para la interacción del administrador con el sistema. La ventana principal de la administración es IEntradaAlmacen la cual contiene las ventanas que ofrecen las distintas funcionalidades de la administración; éstas se muestran con estereotipo *Boundary*. El resto de las clases mostradas solo son auxiliares de subsistemas que no son propios de la aplicación.

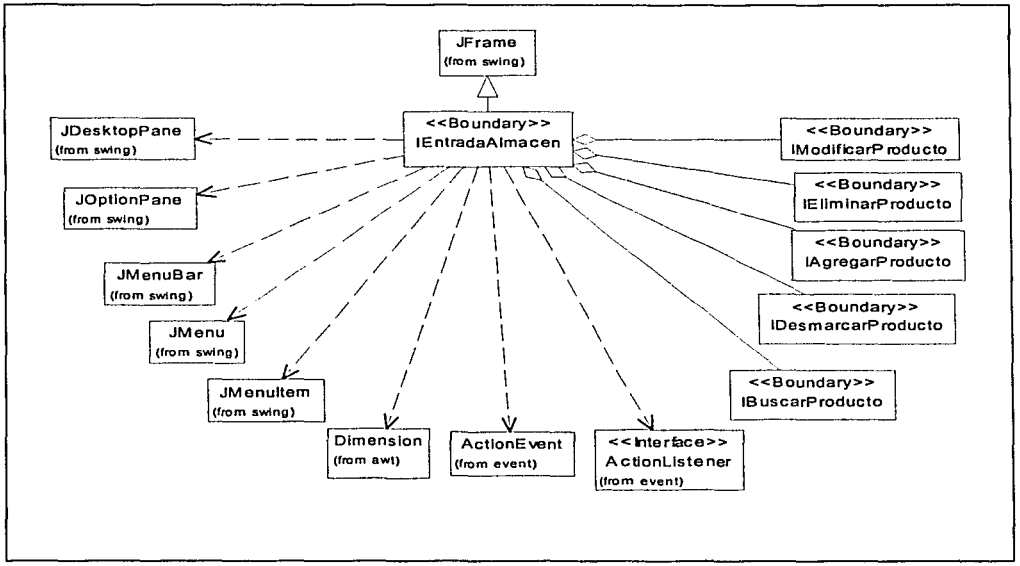


Figura 4.14 Diagrama de clases para el subsistema interfazgraficadeadministrador.

El resto de los diagramas de clases para la administración de esta Sección se muestran en el Apéndice A, Sección A.2.

Subsistema interfazgraficaparainternet

La Figura 4.15 muestra un diagrama de clases que enfatiza la relación entre las clases que conforman la interfaz gráfica de Internet para que el cliente interactúe con el sistema. Las clases que se muestran son en gran parte JSPs los cuales no poseen métodos ni atributos pues representan la lógica de presentación; es decir, solo darán formato a la información la cual es presentada como código HTML solicitado por un navegador de Internet. El navegador es representado por una clase del mismo nombre.

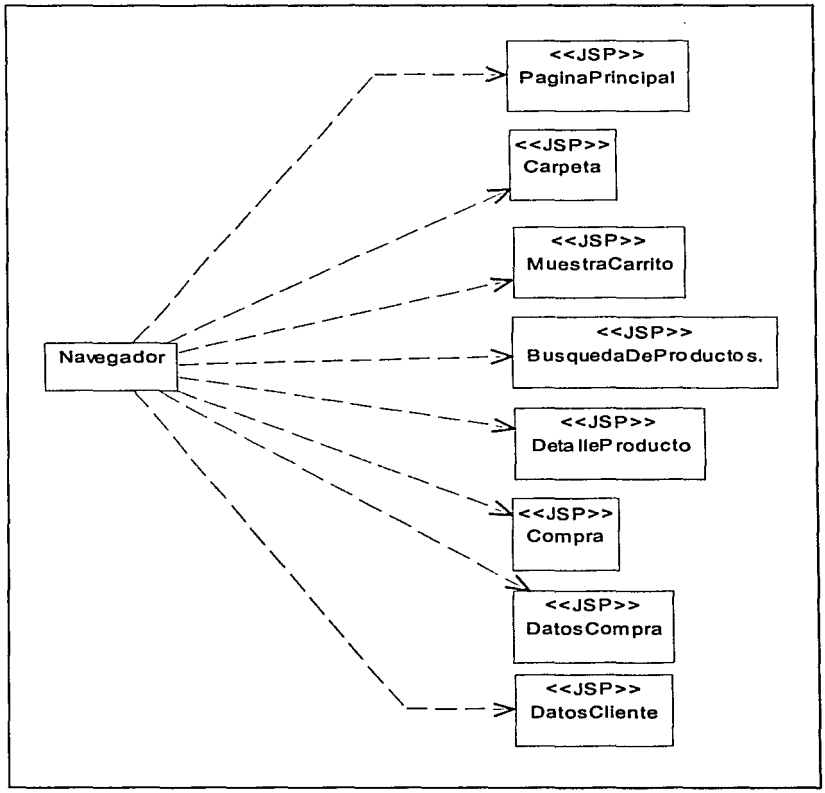


Figura 4.15 Diagrama de clases para el subsistema interfazgraficaparainternet.

Subsistema controlador

La Figura 4.16 muestra un diagrama de clases que enfatiza los atributos y métodos de las mismas. Se presentan las interfaces *remote* y *home* de los *session beans* además de sus relaciones.

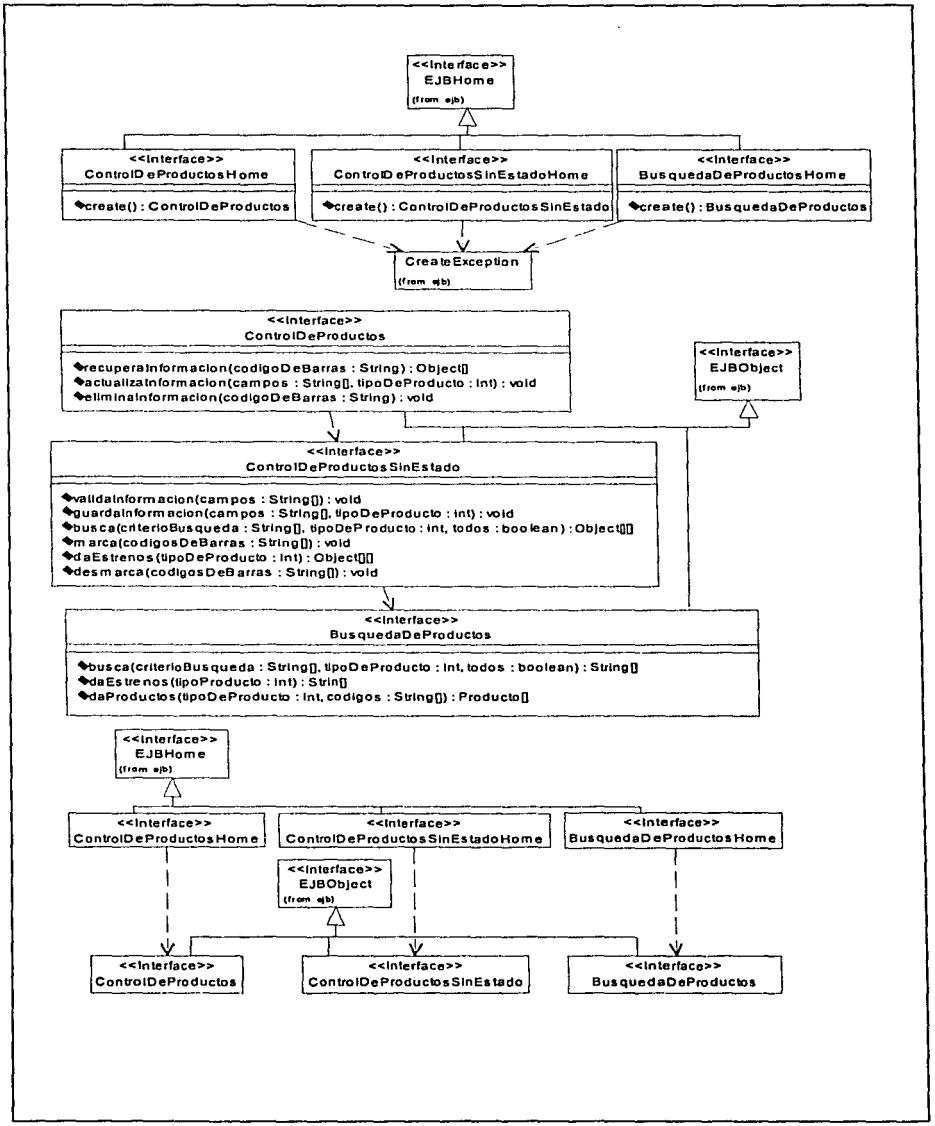


Figura 4.16 Diagrama de clases para el subsistema controlador.

La Figura 4.17 muestra un diagrama de clases que enfatiza los atributos y métodos de las mismas. Se presentan las clases *bean* de los *session beans* además de sus relaciones con excepciones y clases de subsistemas no propios de la aplicación que sirven para desarrollar con la plataforma J2EE.

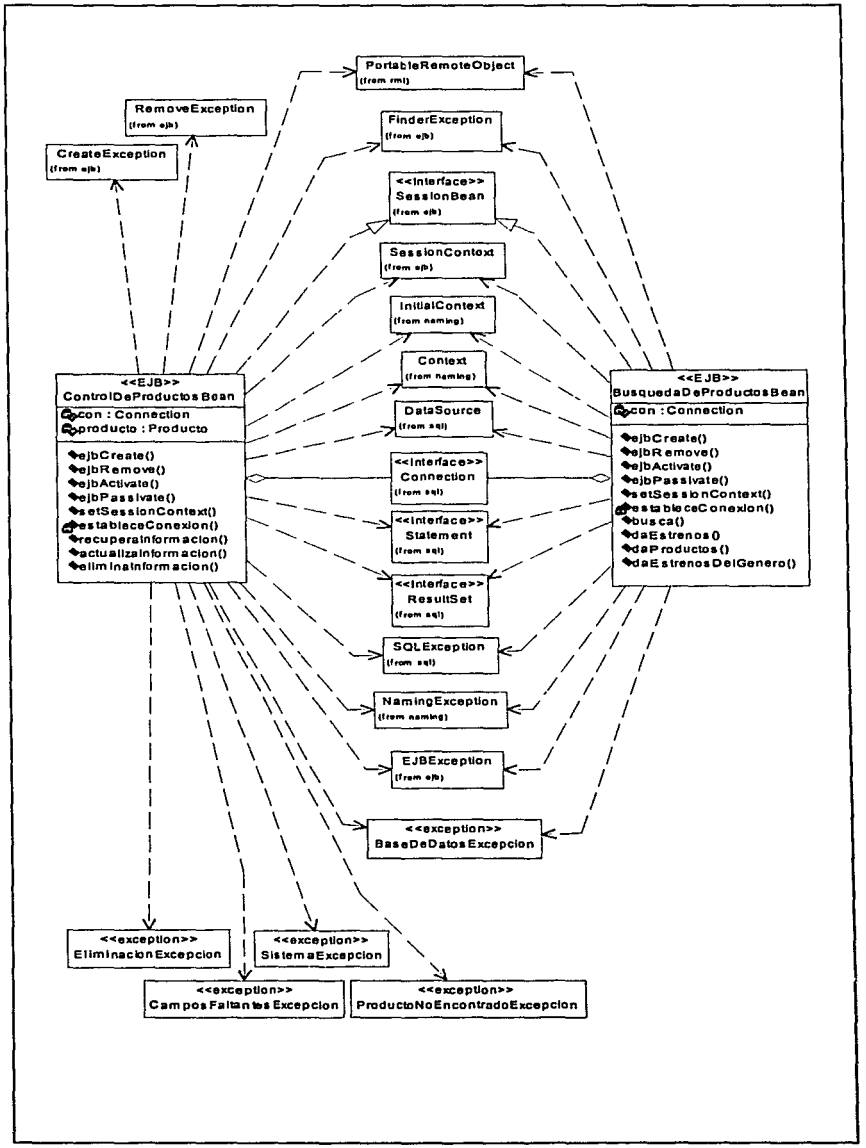


Figura 4.17 Diagrama de clases para el subsistema controlador.

La Figura 4.18 muestra un diagrama de clases que enfatiza los atributos y métodos de la clase *bean* de un *session bean* además de sus relaciones con excepciones y clases de subsistemas no propios de la aplicación que sirven para desarrollar con la plataforma J2EE.

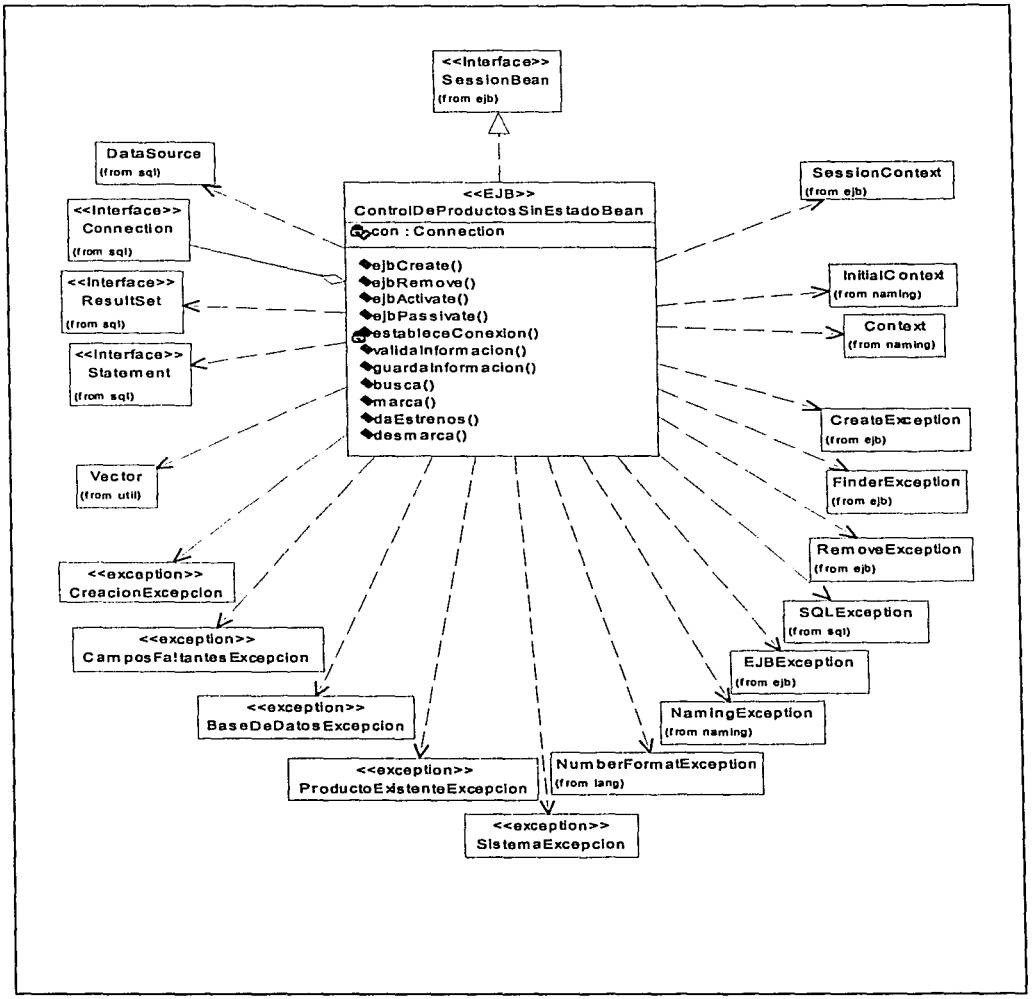


Figura 4.18 Diagrama de clases para el subsistema controlador.

El resto de los diagramas de clases para el subsistema controlador de esta Sección se muestran en el Apéndice A, Sección A.2.

Subsistema web

La Figura 4.19 muestra un diagrama de clases que enfatiza los atributos y métodos de las clases que sirven para el control del sitio Web. Se muestran *session beans* además de sus relaciones con excepciones y clases de subsistemas no propios de la aplicación que sirven para desarrollar con la plataforma J2EE. También se muestra un *Java bean* "Carrito" el cual es usado para simular el carrito de compras del cliente en el sitio de comercio electrónico.

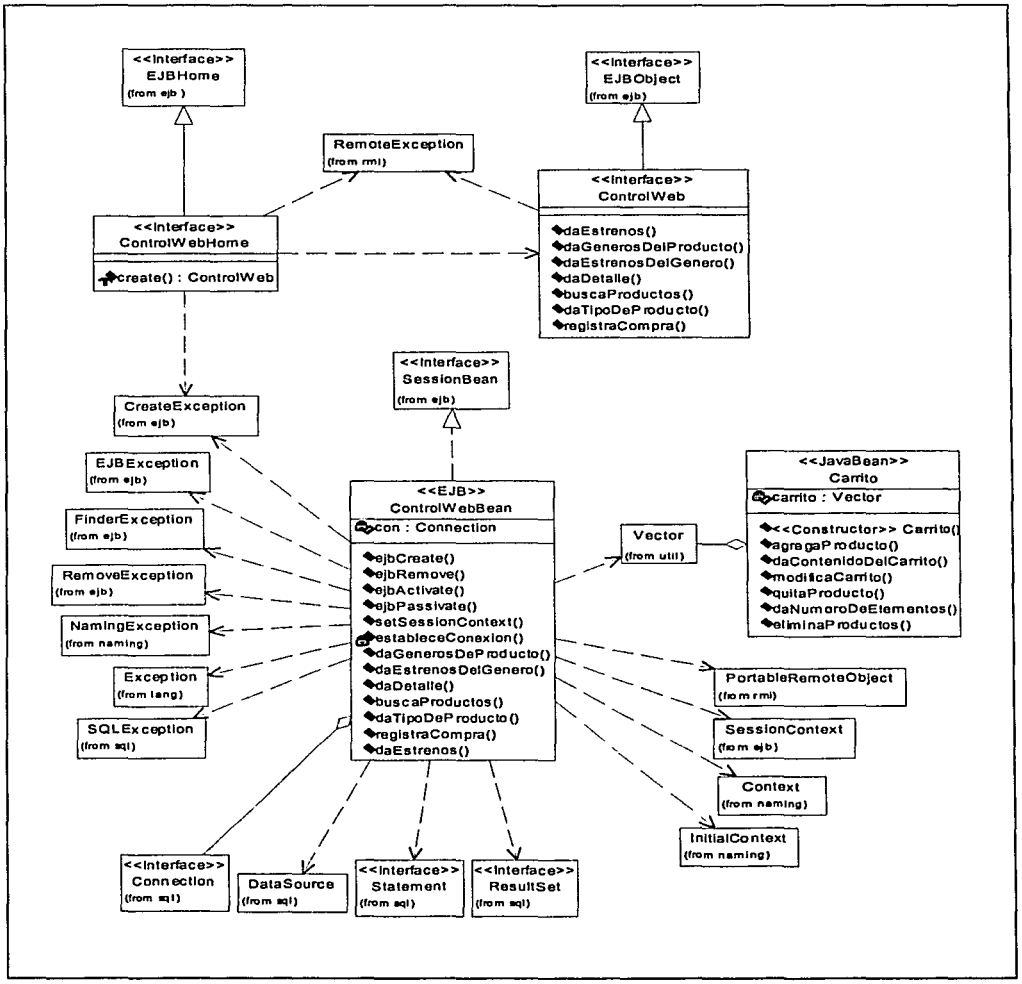


Figura 4.19 Diagrama de clases para el subsistema web.

Subsistema datos

La Figura 4.20 muestra un diagrama de clases que enfatiza los atributos y métodos de las mismas. Se muestran la clase *bean*, las interfaces *remote* y *home* de los *entity beans* usados para representar la información persistente de los productos que se venden en el sitio de comercio electrónico.

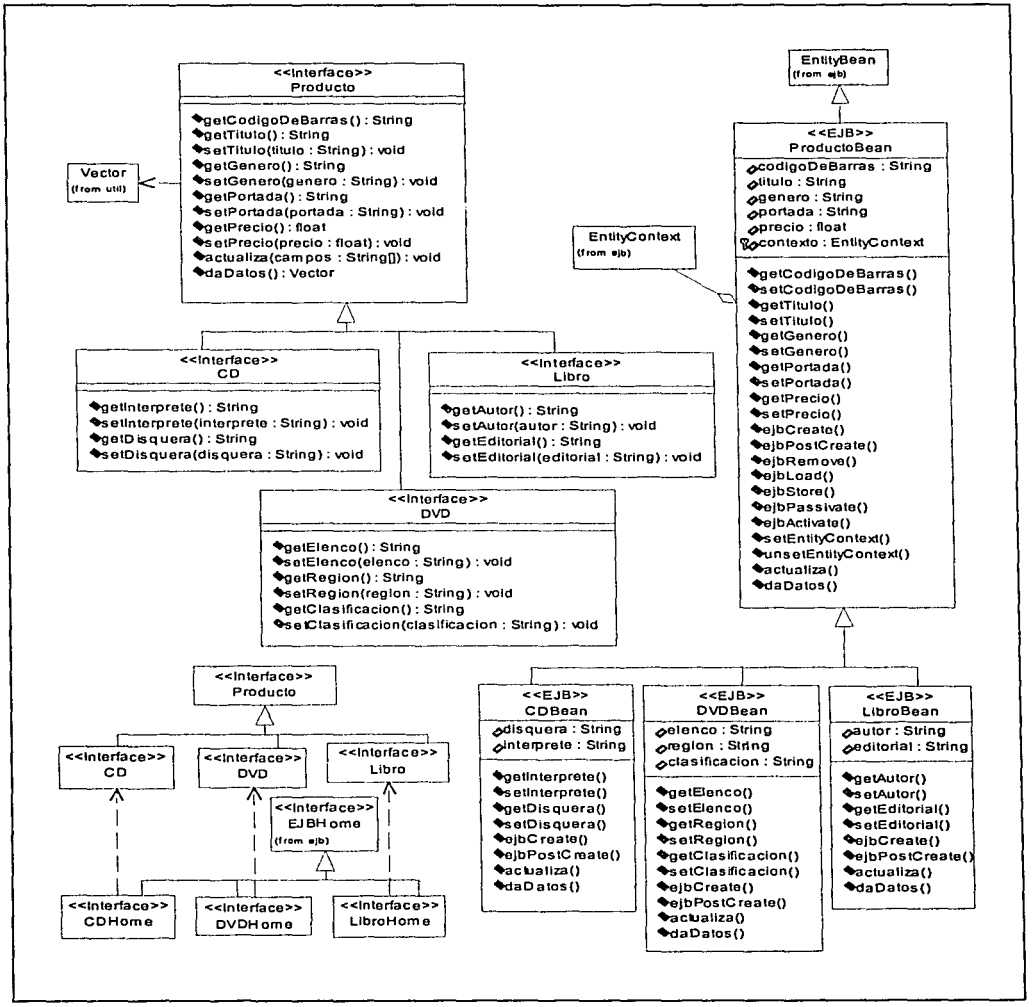


Figura 4.20 Diagrama de clases para el subsistema datos.

El resto de los diagramas de clases para el subsistema *datos* de esta Sección se muestran en el Apéndice A, Sección A.2.

Subsistema utileria

La Figura 4.21 muestra un diagrama de clases que muestra dos clases auxiliares para la aplicación. La primera para el manejo de constantes asociadas a los productos que se venden en el sitio de comercio electrónico y la segunda para representar los artículos del carrito de compras del cliente.

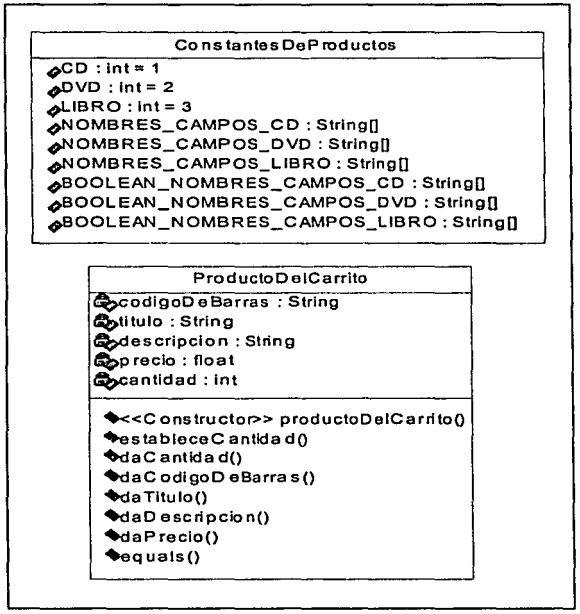


Figura 4.21 Diagrama de clases para el subsistema utileria.

4.3.1.4 Realizaciones de caso de uso-diseño

A continuación se presentan las *realizaciones de caso de uso-diseño*. Cada diagrama tiene en la esquina superior derecha una nota que especifica el seguimiento de las *realizaciones de caso de uso-diseño* con los casos de uso. En esta Sección se presentan los diagramas de secuencia de las *realizaciones de caso de uso-diseño* y no se presentan sus diagramas de clases debido a que la información que estos últimos contienen puede ser fácilmente inferida a partir de los tipos de los objetos presentados en los diagramas de secuencia. Los diagramas de secuencia solo fueron hechos para los flujos que aportaban la mayor funcionalidad de los casos de uso y se evitó diagramar flujos triviales. En este Capítulo solo se presentan diagramas de secuencia significativos para la administración y el sitio Web, el resto de los diagramas se encuentran en el Apéndice A, Sección A.2.

Administración

La Figura 4.22 muestra un diagrama de secuencia para el caso de uso *2.Añadir Producto*. El diagrama incluye la ejecución del caso de uso *3.Añadir CD* el cual podría sustituirse por *4.Añadir DVD* o *5.Añadir Libro*, ejecutándose estos de manera análoga.

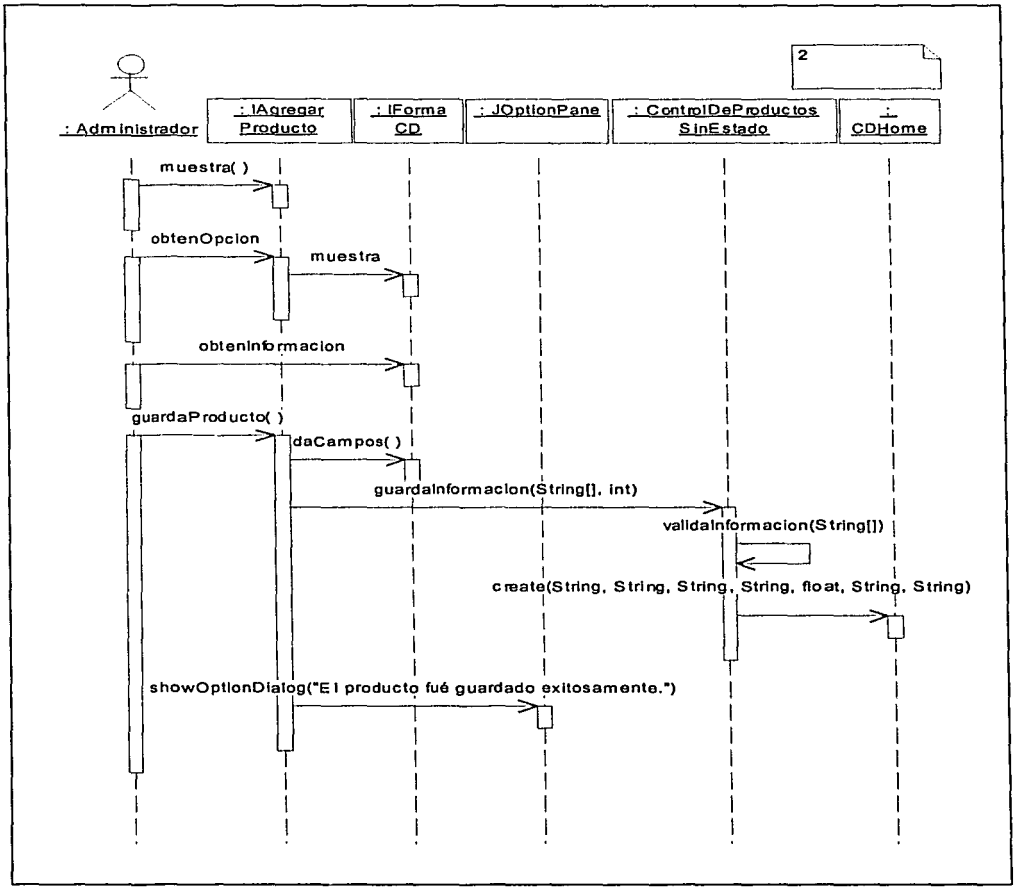


Figura 4.22 Diagrama de secuencia para el caso de uso 2.Añadir Producto.

La Figura 4.23 muestra un diagrama de secuencia para el caso de uso *10.Buscar Producto.* El diagrama incluye la ejecución del caso de uso *11.Buscar CD* el cual podría sustituirse por *12.Buscar DVD* o *13.Buscar Libro*, ejecutándose estos de manera análoga.

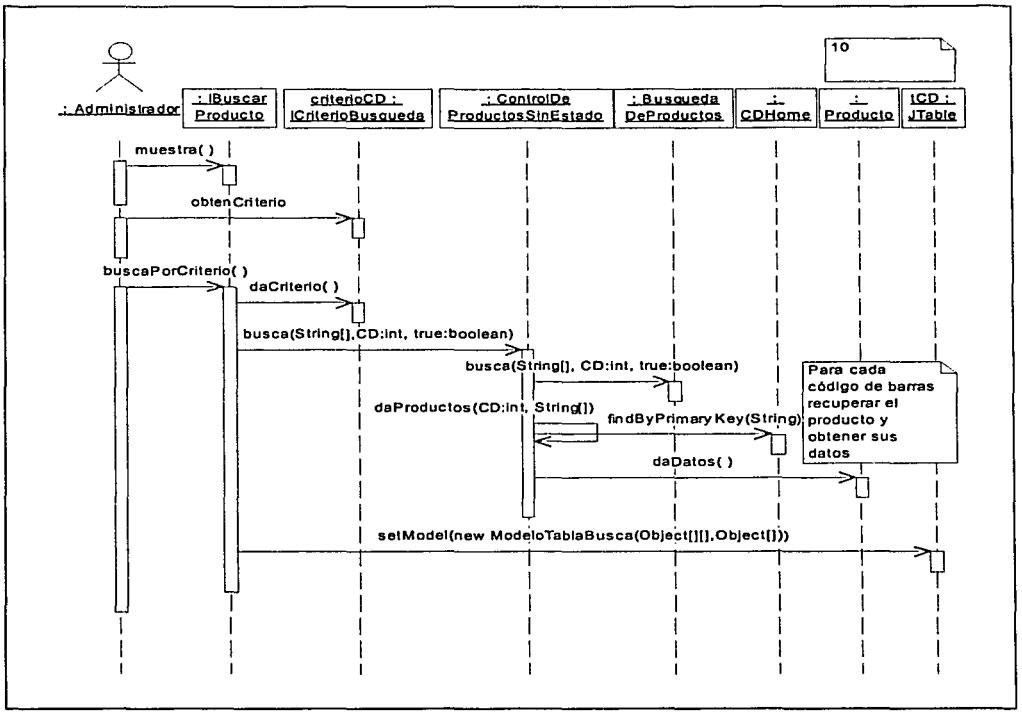


Figura 4.23 Diagrama de secuencia para el caso de uso 10.Buscar Producto.

Sitio Web

La Figura 4.24 muestra un diagrama de secuencia para el caso de uso *20.Seleccionar carpeta producto.* El diagrama incluye la ejecución del caso de uso *21.Muestra jerarquía.*

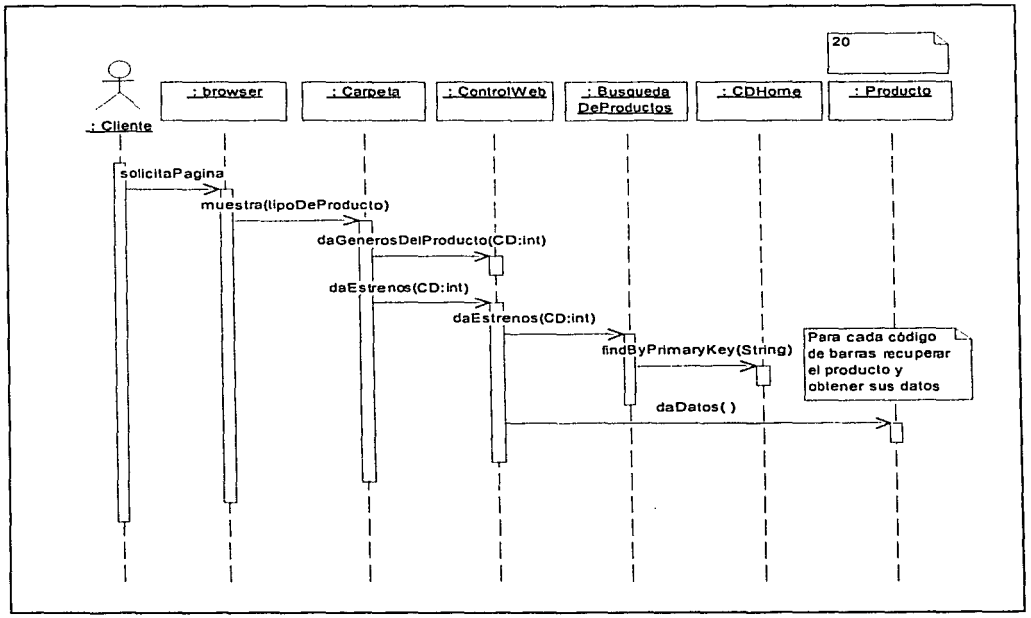


Figura 4.24 Diagrama de secuencia para el caso de uso 20.Seleccionar carpeta producto.

La Figura 4.25 muestra un diagrama de secuencia para el caso de uso *25.Revisar el carrito.* El diagrama incluye la ejecución del caso de uso *30.Ver información del carrito.*

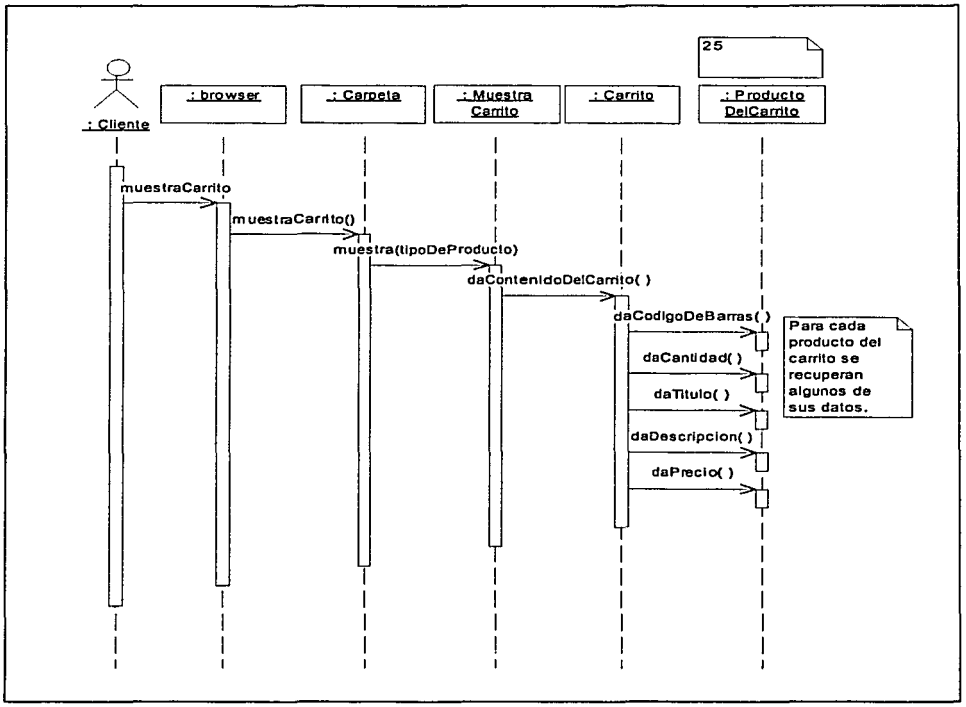


Figura 4.25 Diagrama de secuencia para el caso de uso 25.Revisar el carrito.

4.4 Implementación

4.4.1 Modelo de implementación

El modelo de implementación fue usado como una herramienta para documentar la manera en que fue construido (en términos de la programación de componentes) el sistema de comercio electrónico más que para planear su construcción, que es el verdadero objetivo. La manera en que el modelo se usó fue producto de la falta de experiencia en el uso de la plataforma J2EE y de nuevo se pudo experimentar la conveniencia de poder contar dentro del equipo con gente que conozca la plataforma de desarrollo para evitar los riesgos inherentes tales como la entrega retrasada del producto y en caso extremo el fracaso del mismo.

Otra consecuencia de la falta de experiencia en el uso de la plataforma J2EE, como se mencionó en el modelo de diseño, fue que la implementación fue realizado paralelamente con el diseño pues no se podía saber con anticipación cuales eran las decisiones convenientes para distribuir la funcionalidad del sistema entre las clases de diseño hasta no haber empezado a experimentar con la implementación de algunos componentes.

Dentro del modelo de implementación no se hizo un plan de integración ya que se careció de una planeación de iteraciones como ya antes se ha mencionado.

Los detalles de la plataforma J2EE se pueden encontrar dentro de la tesis complementaria [1] la cual es conveniente consultar antes de empezar con la lectura del modelo de implementación.

En las subsecuentes secciones dentro de la implementación se tendrá con frecuencia la separación de la parte de la administración y del sitio Web que fueron identificadas desde la captura de requerimientos para una distribución más comprensible de los diagramas.

4.4.1.1 Subsistemas de implementación

Como la teoría lo indica, los subsistemas de diseño tienen un mapeo uno-a-uno con los subsistemas de implementación. Como un subsistema de implementación se manifiesta como un "mecanismo de empaquetado" del ambiente de implementación (en este caso J2EE), entonces podemos identificar los siguientes paquetes para el modelo de implementación:

- Paquete *interfazgraficadeadministrador:* Tiene un seguimiento con el subsistema *interfazgraficadeadministrador.*
- Paquete *interfazgraficaparainternet:* Tiene un seguimiento con el subsistema *interfazgraficaparainternet.*
- Paquete *controlador:* Tiene un seguimiento con el subsistema *controlador.*
- Paquete *web:* Tiene un seguimiento con el subsistema *web.*
- Paquete *utileria:* Tiene un seguimiento con el subsistema *utileria.*

El paquete *web* no es implementado como tal pues su contenido tendrían que ser archivos JSP los cuales no pueden ser puestos en paquetes. Solo se muestra de esta forma para no perder el seguimiento de los paquetes con los subsistemas de diseño.

Los diagramas de los paquetes son los mismos que los usados en "4.3.1.1 Subsistemas" en las Subsecciones de administración y sitio Web. En el resto de esta Sección de implementación solo se muestran los componentes de los paquetes que son propios de la aplicación (*controlador ,web, interfazgraficadeadministrador, interfazgraficaparainternet* y *utileria*), el resto de los paquetes son propios de la plataforma J2EE y ayudan para la implementación en la misma.

4.4.1.2 Diagramas de componentes para el código fuente y *byte code* de Java.

El lenguaje de programación Java utiliza archivos con extensión *.java* para el código fuente. El archivo *.java* es compilado para producir uno o varios archivos de extensión *.class* los cuales contienen el *byte code* de Java que posteriormente será interpretado por una *Java Virtual Machine* (JVM). El número de *.class* producido a partir de un *.java* es directamente proporcional al número de clases o interfaces definidas en el *.java*, es decir, un archivo *.class* por clase o interfaz definida. Los archivos *.class* tienen el mismo nombre que el de la clase o interfaz que definen.

En el sistema desarrollado se utiliza un archivo *.java* por clase o interfaz definida de forma que hay un solo *.class* por cada *.java*. Cada *.java* tiene seguimiento hacia una clase de diseño en el modelo de diseño cuyo nombre coincide con el nombre del archivo *.java*.

Se utilizan los estereotipos siguientes en los componentes:

- *file*: Archivo con código fuente.
- *executable*: Archivo .class que se puede ejecutar en un nodo.
- *class*: *byte code* de Java.
- *class anonymous*: Archivo .class para una clase anónima de Java.

Las clases anónimas en Java se generan debido a la creación de objetos que no son asociados a una variable.

Los diagramas de componentes de esta Sección muestran el código fuente y el *byte code* de Java para el sistema de comercio electrónico. Se muestran los diagramas de componentes por paquete para una lectura más sencilla.

Paquete interfazgraficadeadministrador

La Figura 4.26 presenta un diagrama de componentes que muestra el archivo para el código fuente y el *byte code* de Java en el paquete *interfazgraficadeadministrador.*

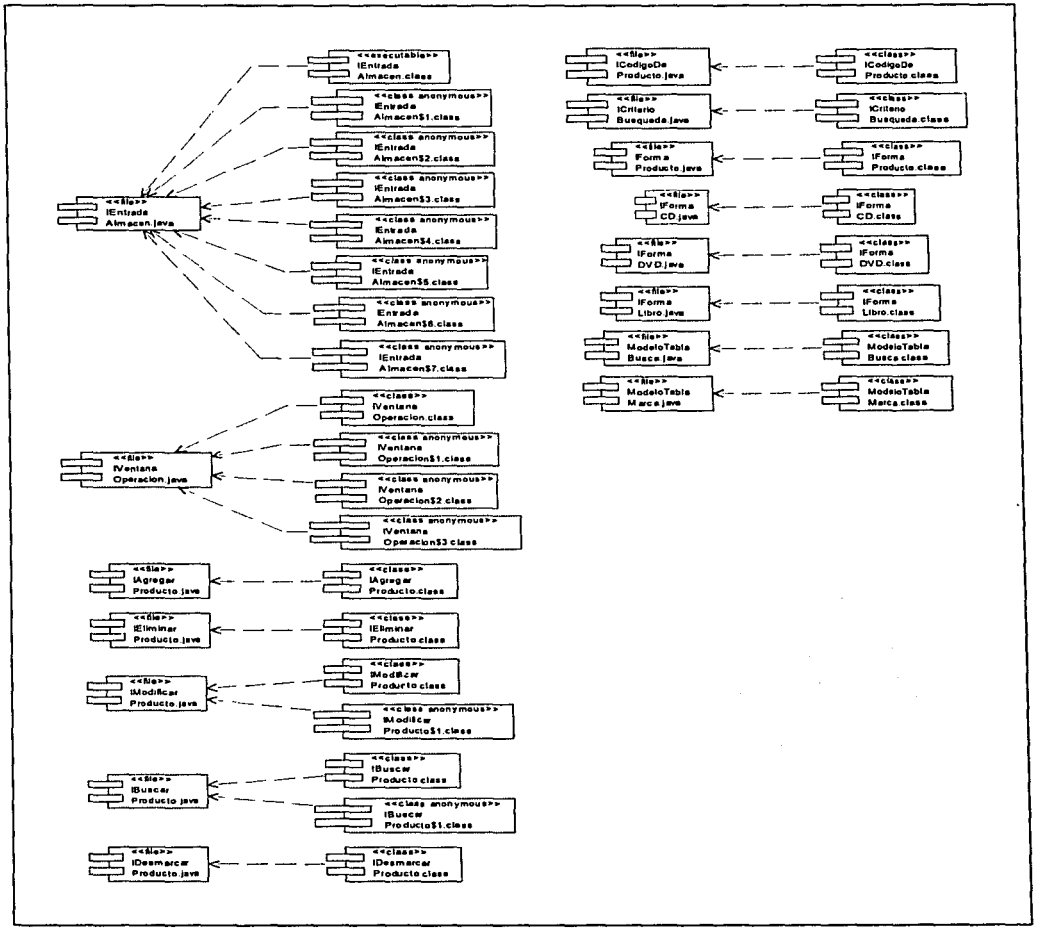


Figura 4.26 Diagrama de componentes para el paquete interfazgraficadeadministrador.

Paquete interfazgraficaparainternet

Los archivos JSP tienen una extensión .jsp asociada, estos no producen un .class en especial pues son manejados directamente con la extensión .jsp por el contenedor Web donde son colocados. Cada *.jsp* tiene seguimiento hacia una clase de diseño en el modelo de diseño cuyo nombre coincide con el nombre del archivo *.jsp.*

La Figura 4.27 presenta un diagrama de componentes que muestra el archivo para el código fuente de los JSPs en el paquete *interfazgraficaparainternet.*

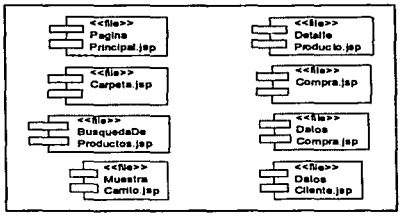


Figura 4.27 Diagrama de componentes para el paquete interfazgraficaparainternet.

Paquete controlador

La Figura 4.28 presenta un diagrama de componentes que muestra el archivo para el código fuente y el *byte code* de Java en el paquete *controlador*. En la primera columna de pares de componentes file-class se muestran los *beans* definidos con sus interfaces *home*, *remote* y su clase *bean*. La segunda columna muestra las excepciones definidas en el paquete.

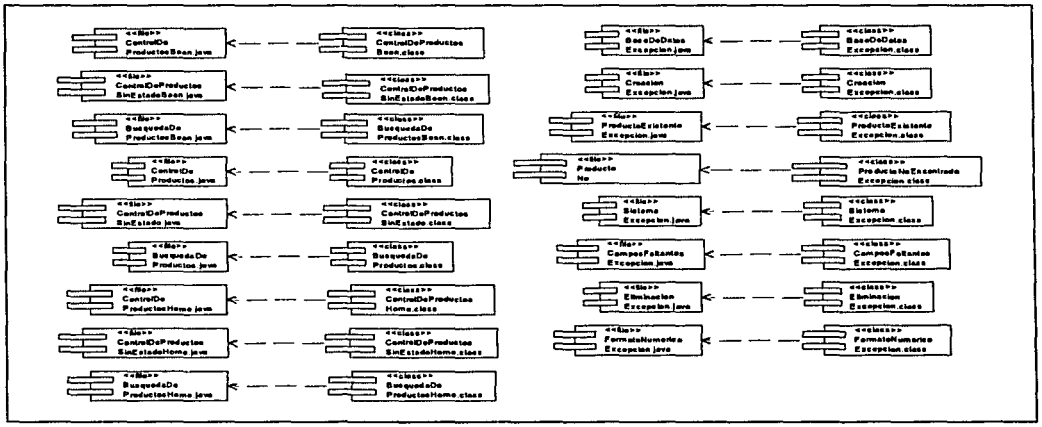


Figura 4.28 Diagrama de componentes para el paquete controlador.

Paquete web

La Figura 4.29 presenta un diagrama de componentes que muestra el archivo para el código fuente y el *byte code* de Java en el paquete *web*. En la primera columna de pares de componentes file-class se muestra el *bean* definido con sus interfaces *home*, *remote* y su clase *bean*. La segunda columna muestra el *Java bean* usado para representar el carrito de compras del cliente.

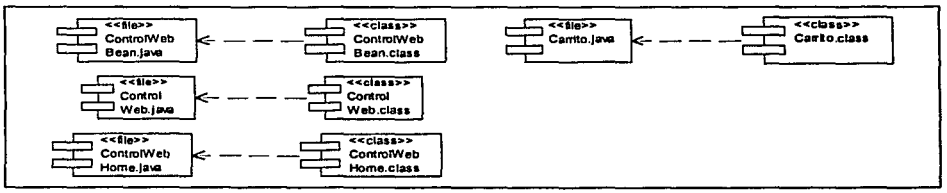


Figura 4.29 Diagrama de componentes para el paquete web.

Paquete datos

La Figura 4.30 presenta un diagrama de componentes que muestra el archivo para el código fuente y el *byte code* de Java en el paquete *datos*. Se muestran los *beans* definidos con sus interfaces *home*, *remote* y su clase *bean*.

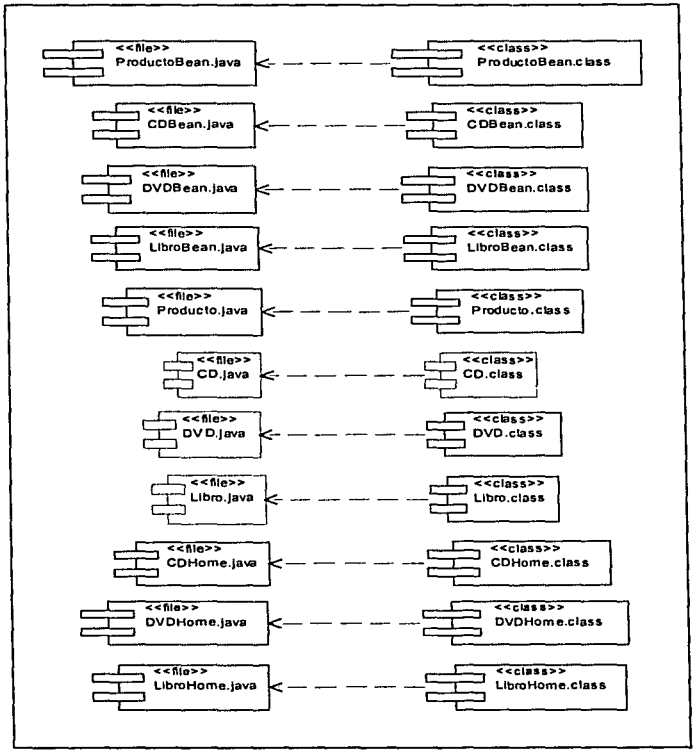


Figura 4.30 Diagrama de componentes para el paquete datos.

Paquete utileria

La Figura 4.31 presenta un diagrama de componentes que muestra el archivo para el código fuente y el *byte code* de Java en el paquete *utileria*.

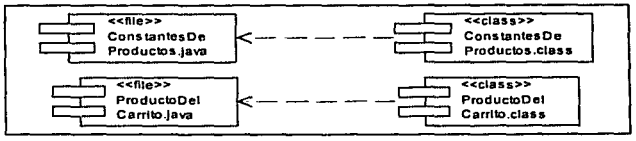


Figura 4.31 Diagrama de componentes para el paquete utileria.

4.4.1.3 Diagramas de componentes para la dependencia de compilación entre componentes de distintos paquetes

Los diagramas de componentes de esta Sección enfatizan las dependencias de compilación existentes entre componentes que pertenecen a distintos paquetes. Más tarde, en la siguiente Sección se muestran las dependencias de compilación entre componentes que pertenecen al mismo paquete.

Administración

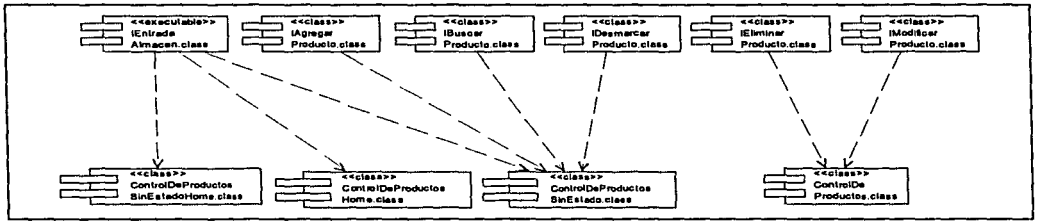


Figura 4.32 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes de los paquetes interfazgraficadeadministrador y controlador.

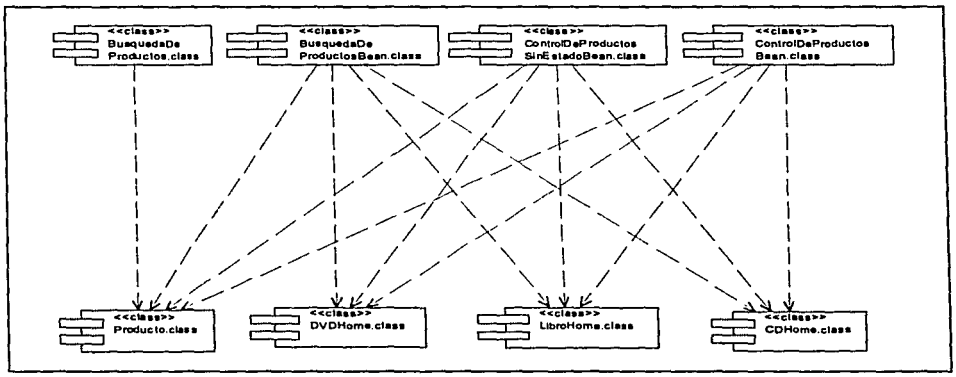


Figura 4.33 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes de los paquetes controlador y datos.

El resto de los diagramas de componentes para la administración de esta Sección se muestran en el Apéndice A, Sección A.3.

Sitio Web

Como se mencionó antes los archivos JSP no producen un .class pues no son compilados explícitamente y son manejados directamente con la extensión .jsp por el contenedor Web donde son colocados. Las dependencias de compilación mostradas para los componentes JSP son meramente ilustrativas y muestran los componentes que necesitan existir para que estos pueda correr en un momento dado.

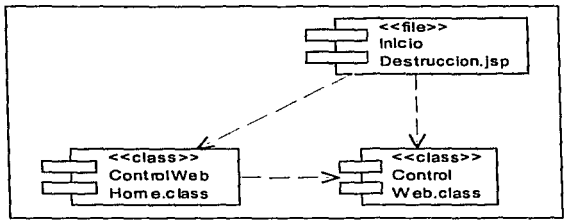


Figura 4.34 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes de los paquetes interfazgraficaparainternet y web.

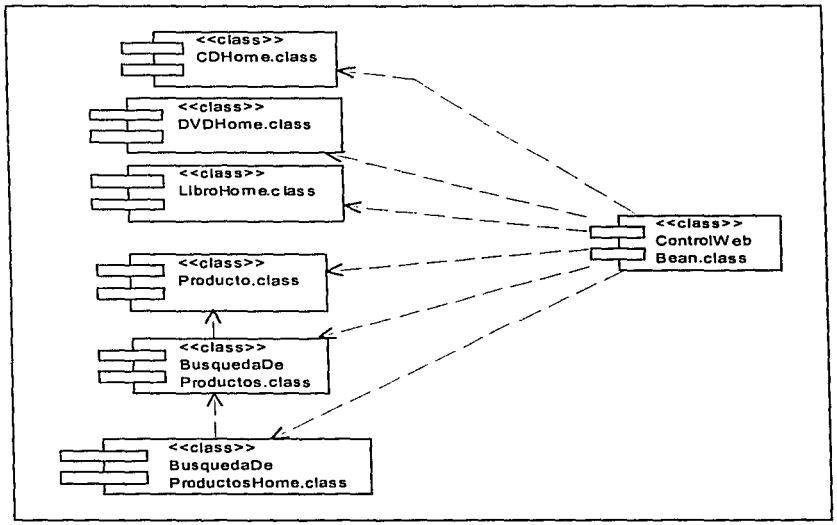


Figura 4.35 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete web con los paquetes datos y controlador.

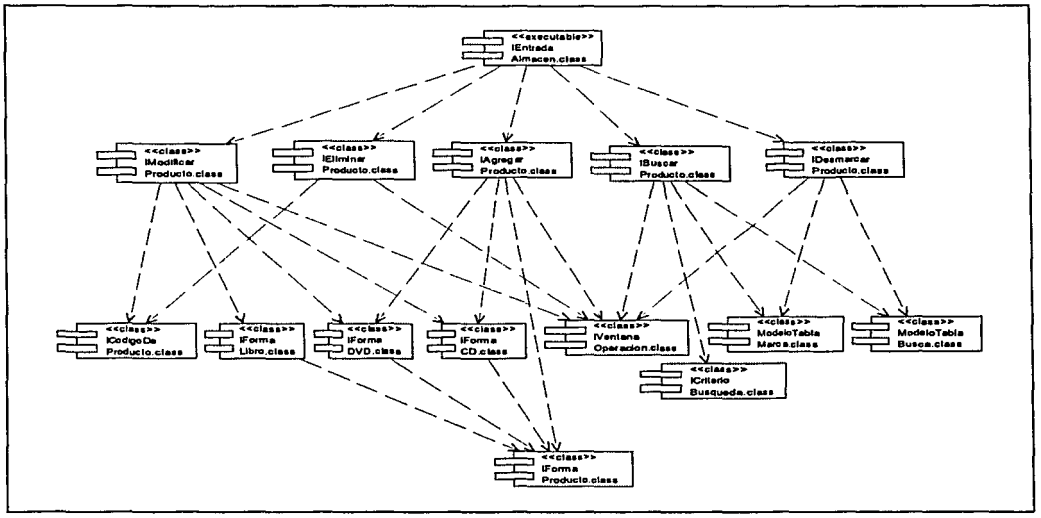
El resto de los diagramas de componentes para el sitio Web de esta Sección se muestran en el Apéndice A, Sección A.3.

4.4.1.4 Diagramas de componentes para la dependencia de compilación entre componentes del mismo paquete

Los diagramas de componentes de esta Sección enfatizan las dependencias de compilación existentes entre componentes que pertenecen al mismo paquete.

Paquete interfazgraficadeadministrador

La Figura 4.36 muestra un diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete *interfazgraficadeadministrador.* El diagrama muestra un ejecutable, IEntradaAlmacen.class.

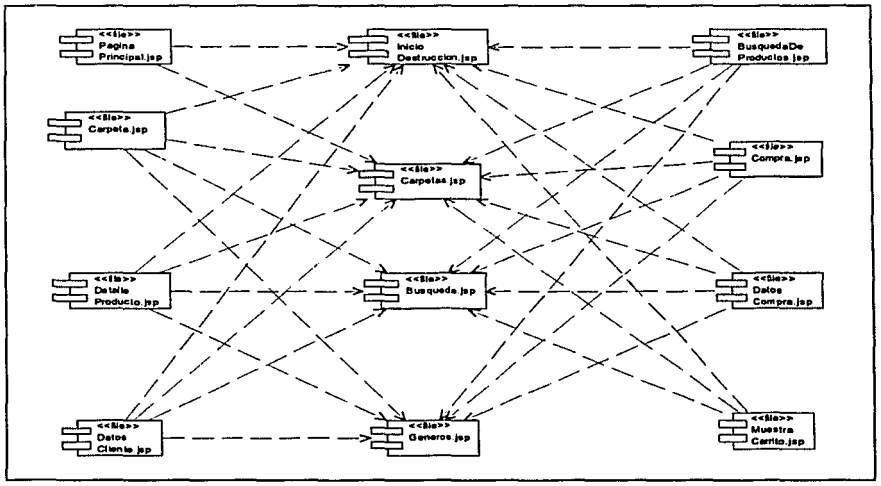


4.36 Diagrama de componentes para el paquete interfazgraficadeadministrador.

Paquete Interfazgraficaparainternet

Las dependencias de compilación mostradas para los componentes JSP son meramente ilustrativas y muestran los componentes que necesitan existir para que estos pueda correr en un momento dado.

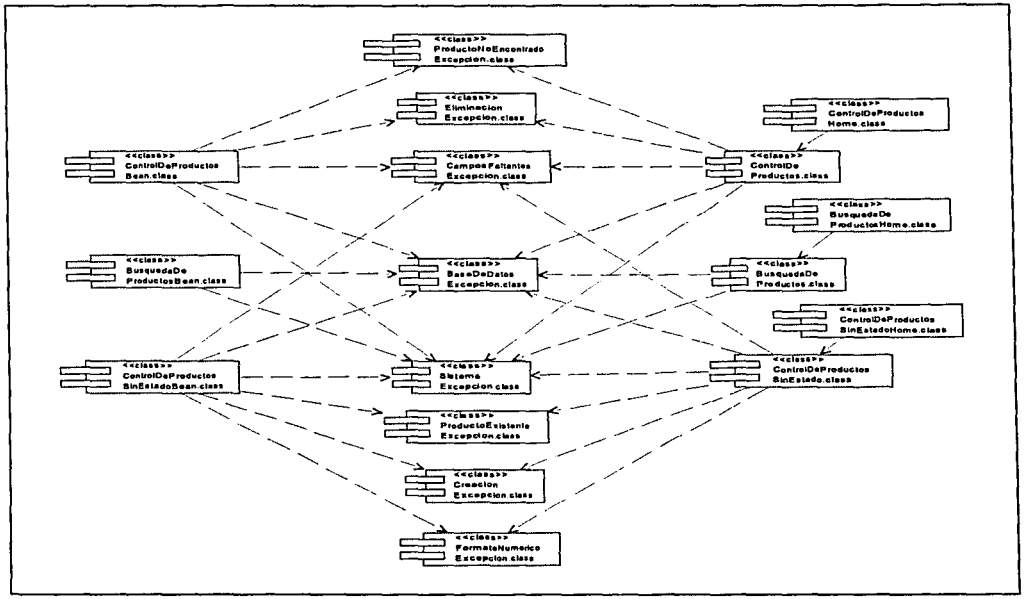
La Figura 4.37 muestra un diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete *interfazgraficaparainternet.*



4.37 Diagrama de componentes para el paquete interfazgraficaparainternet.

Paquete controlador

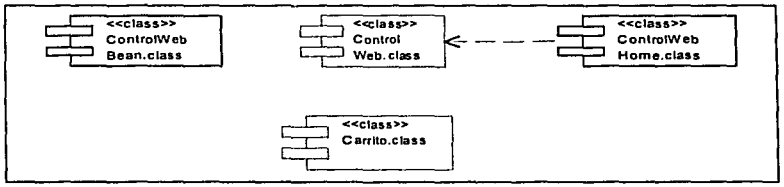
La Figura 4.38 muestra un diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete *controlador*.



4.38 Diagrama de componentes para el paquete controlador.

Paquete web

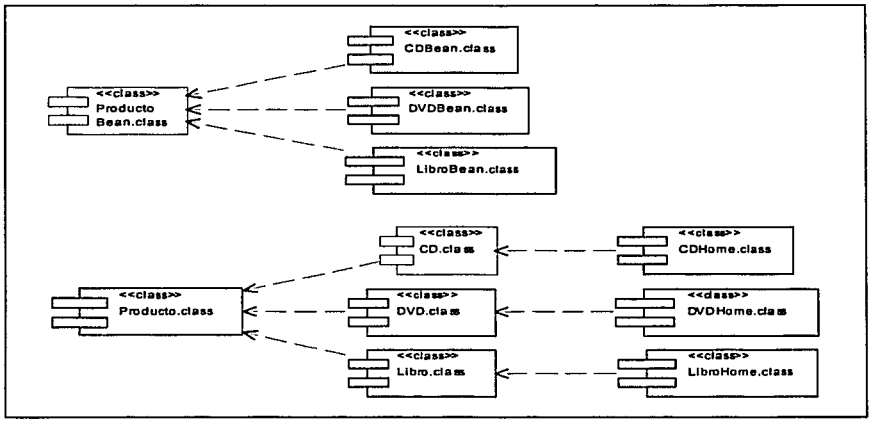
La Figura 4.39 muestra un diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete *web*. Los componentes que muestran el *bean* (interfaces *remote*, *home* y clase *bean*) definido no tienen dependencias de compilación con la clase *bean* pues éstas son manejadas directamente por el contenedor EJB. El componente Carrito.class no tiene dependencias dentro del paquete.



4.39 Diagrama de componentes para el paquete web.

Paquete datos

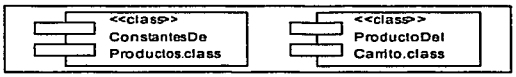
La Figura 4.40 muestra un diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete *datos.* Los componentes que muestran los *beans* (interfaces *remote*, *home* y clases *bean*) definidos no tienen dependencias de compilación con las clases *bean* pues éstas son manejadas directamente por el contenedor EJB.



4.40 Diagrama de componentes para el paquete datos.

Paquete utileria

Los componentes del paquete *utileria* no tienen dependencias de compilación dentro del paquete.



4.41 Diagrama de componentes para el paquete utileria.

4.4.1.5 Pruebas unitarias para los componentes

Caso de prueba

Las pruebas unitarias solamente fueron realizadas para los componentes del paquete *interfazgraficadeadministrador* pues estos componentes pueden ser probados por separado sin la existencia de otros componentes a diferencia de los componentes del resto de los paquetes los cuales están altamente relacionados y no pueden ser probados de manera aislada.

Los componentes del paquete *interfazgraficadeadministrador* a los cuales se les aplican pruebas son:

- IEntradaAlmacen.class
- IAgregarProducto.class
- IEliminarProducto.class
- IModificarProducto.class
- IBuscarProducto.class
- IDesmarcarProducto.class

El resultado de aplicar la prueba al componente es la visualización del despliegue gráfico de la ventana representada por éste.

Procedimiento de prueba

Estas pruebas unitarias para los componentes del paquete *interfazgraficadeadministrador* son automáticas y consisten de un método *main* dentro de la clase correspondiente a cada componente.

El main del componente corre de manera tradicional en que se ejecutan las clases de Java. Se aplica el intérprete de Java al componente .class mediante el comando *java*. Por ejemplo, para IEntradaAlmacen.class se escribe:

> *java IEntradaAlmacen*

4.4.2 Modelo de instalación

El modelo de instalación debe desarrollarse durante el flujo de trabajo de diseño, sin embargo, se decide incluirlo hasta esta fase de implementación pues es donde se refina y quedan bien definidos los componentes que se instalan en cada nodo.

El modelo de instalación fue influenciado por la plataforma de desarrollo. La arquitectura del software pudo ser mapeada de manera sencilla y directa a la arquitectura del hardware debido al modelo multicapa distribuido que emplea J2EE.

La Figura 4.42 presenta el diagrama de instalación para el sitio de comercio electrónico. El diagrama muestra el software instalado en cada nodo y los componentes propios de la aplicación para el funcionamiento del sistema.

Los archivos .jar es el mecanismo típico de Java para el empaquetado de componentes. En la Figura 4.42 se pueden notar los archivos administración.jar y svc.war. Ambos archivos son archivos creados con la herramienta de empaquetado *jar* de Java de tal forma que los dos son archivos tipo *jar*, solamente que svc.war tiene esa extensión pues así lo requiere la especificación de J2EE para componentes que se instalan en el contenedor Web.

El archivo administracion.jar contiene los componentes de todos los paquetes a excepción de los componentes del paquete *web*. Este archivo se instala en el nodo Servidor (contenedor EJB) y en el Cliente *stand-alone*. En el nodo Cliente *stand-alone* se corre el componente IEntradaAlmacen.class del paquete *Interfazgraficadeadministrador* que está dentro de administración.jar para iniciar la ejecución de la administración del sitio de comercio electrónico. El contenedor EJB sabe como manejar internamente el archivo administracion.jar.

El archivo svc.war contiene todos los componentes JSP correspondientes al paquete interfazgraficadeadministrador y el componente ConstantesDeProductos.class del paquete *utileria*. Este archivo se instala en el contenedor Web del nodo Servidor el cual sabe cómo manejarlo internamente. Los JSPs de svc.war se encargan de hacer las páginas html dinámicas para la interacción de los clientes con sitio de comercio electrónico.

El nodo Navegador es el nodo desde el cual el cliente interactúa con el sistema mediante un navegador de Internet.

El nodo Base de Datos tiene instalada la base de datos ORACLE8i.

Las ligas del diagrama de instalación muestran los protocolos usados para la comunicación entre los nodos del sistema.

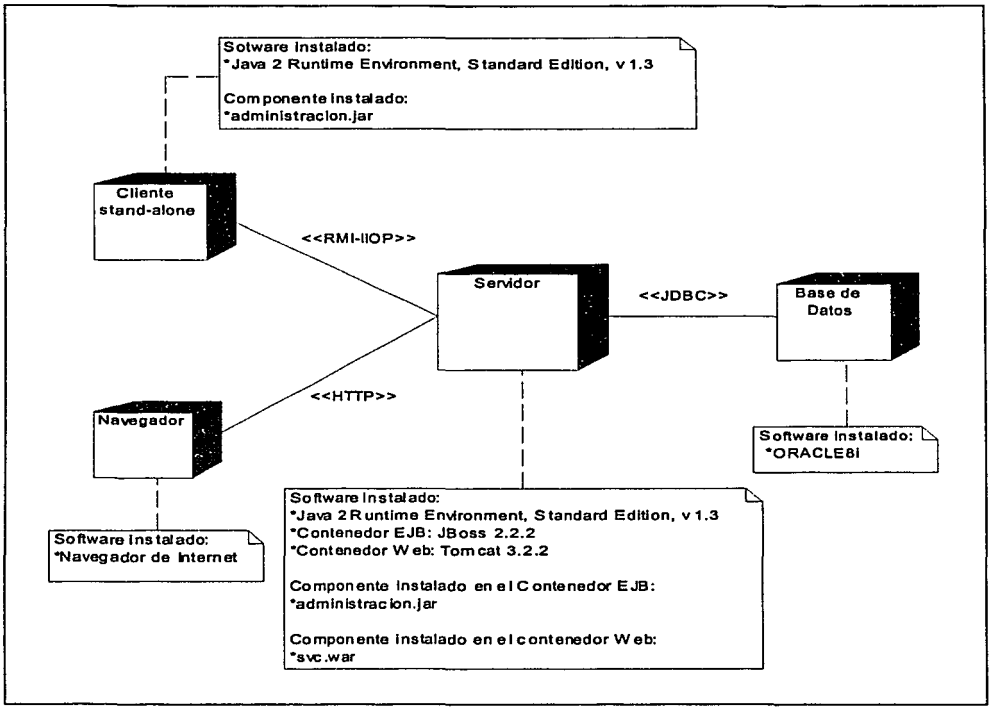


Figura 4.42 Diagrama de instalación para el sitio de comercio electrónico.

4.5 Pruebas

4.5.1 Modelo de Pruebas

Como ya se ha dicho, durante el desarrollo del sitio de comercio electrónico se careció de una planeación de las iteraciones y solamente se usaron dos ciclos, uno para el subsistema de administración y el segundo donde se creaba el sitio Web. Este hecho influenció el contenido del modelo de pruebas. Las pruebas de integración fueron omitidas debido a que no existieron iteraciones planeadas. Las pruebas se hicieron más generales y tuvieron que ver directamente con los casos de uso; es decir, se aplicaron el tipo de pruebas de "caja negra" para donde se observa el comportamiento externo del sistema.

Las pruebas planeadas en fases tempranas fueron cambiando poco a poco conforme se hacían cambios a los casos de uso. Cuando llegó el momento de aplicar las pruebas se detectaron defectos mínimos que tenían que ver con pequeños errores de implementación. Esto permitió experimentar el hecho de que un software bien planeado y documentado con bases sólidas dadas desde las etapas tempranas del desarrollo puede alcanzar el éxito. La corrección de errores no ocupa demasiados recursos y su cantidad se minimiza.

A continuación se presentan los casos de prueba y sus procedimientos para aplicarlos. Estos se muestran en dos partes para su mejor lectura, una para el subsistema de administración y la otra para el subsistema del sitio Web.

4.5.1.1 Casos de prueba y procedimiento

Los casos de prueba presentados en esta Sección son casos de prueba para cada uno de los casos de uso requeridos para el sistema. Cada caso de prueba recibe un identificador numérico idéntico al del caso de uso correspondiente en el modelo de casos de uso. Se presentan casos de prueba solamente para los casos de uso o flujos en los cuales se consideró que así lo requerían.

El procedimiento para la aplicación de las pruebas es totalmente manual. Para la parte de administración es necesario correr el cliente *stand-alone* del sistema y para la parte del sitio Web es necesario entrar al mismo mediante un navegador de Internet.

Administración

1. Caso de prueba: "Entrar al Almacén"

(1) "Nancy entra al Almacén"

Nancy inicia el Sistema. El Sistema le muestra las siguientes acciones a realizar: Añadir producto, Eliminar producto, Modificar producto, Buscar producto, Marcar estreno, Desmarcar estreno, Salir del Almacén. Nancy escoge Eliminar producto.

(1.1) Similar al Escenario 2.3 solo que la confirmación de la salida termina la ejecución del Sistema.

2. Caso de prueba: "Añadir producto"

(2) "Nancy selecciona el tipo de producto que desea añadir"

Nancy solicita al Sistema añadir un producto. El Sistema muestra las opciones y Nancy elige añadir CD. El Sistema muestra una forma. Nancy llena la forma con los datos del CD. Nancy indica al Sistema que guarde la información. El Sistema guarda la información.

(2.1) "Nancy solicita añadir un producto y el Sistema avisa que el producto existe en el Almacén".

Nancy solicita al Sistema añadir un producto. El Sistema muestra las opciones y Nancy elige añadir CD. El Sistema muestra una forma. Nancy llena la forma con los datos del CD. Nancy indica al Sistema que guarde la información. El Sistema no guarda el CD y avisa que el CD ya existe.

(2.2) "Nancy limpia la forma para añadir un CD".

Nancy solicita al Sistema añadir un producto. El Sistema muestra las opciones y Nancy elige añadir CD. El Sistema muestra una forma. Nancy llena algunos campos de la forma con datos del CD. Nancy indica al Sistema que limpie la forma. El Sistema limpia los campos de la forma.

(2.3) "Nancy solicita salir mientras añade un CD ".

(2.3.1) "Nancy solicita salir mientras añade un CD y confirma la salida".

Nancy solicita al Sistema añadir un producto. El Sistema muestra las opciones y Nancy elige añadir CD. El Sistema muestra una forma. Nancy indica al Sistema que salga de la operación de añadir el CD. El Sistema pide a Nancy confirmar la salida. Nancy confirma la salida. El Sistema cierra la forma. Ninguna información se agrega al Almacén.

(2.3.2) "Nancy solicita salir mientras añade un CD pero no confirma la salida".

Nancy solicita al Sistema añadir un producto. El Sistema muestra las opciones y Nancy elige añadir CD. El Sistema muestra una forma. Nancy indica al Sistema que salga de la operación de añadir el CD. El Sistema pide a Nancy confirmar la salida. Nancy no confirma la salida. La forma permanece abierta permitiendo a Nancy continuar operando sobre la misma.

(2.4) "La Base de Datos manda un error cuando Nancy intenta añadir un CD"

Nancy solicita al Sistema añadir un producto. El Sistema muestra las opciones y Nancy elige añadir CD. El Sistema muestra una forma. Nancy llena la forma con los datos del CD. Nancy indica al Sistema que guarde la información. El Sistema no guarda el CD y avisa a Nancy que hubo algún problema en la Base de Datos. La forma permanece abierta permitiendo a Nancy continuar operando sobre la misma.

3. Caso de prueba: "Añadir CD"

(3) "Nancy añade un CD exitosamente".

El Sistema muestra una forma donde se pueden capturar los datos del CD. Nancy proporciona al Sistema la siguiente información:

Código de barras: 1259374.
Título: Chocolate Starfish And The Hot Dog Flavoured Water
Género: Alternativo.
Portada: limpbizkit.jpg.
Precio: 170.00.
Intérprete: Limp Bizkit.
Disquera: Universal.

4. Caso de prueba: "Añadir DVD"

(4) Similar al Escenario 3 solo que con los datos del DVD.

5. Caso de prueba: "Añadir Libro"

(5) Similar al Escenario 3 solo que con los datos del libro.

6. Caso de prueba: "Eliminar producto"

(6) "Nancy elimina un producto exitosamente".

Nancy solicita al Sistema eliminar un producto del Almacén. El Sistema le pide el código de barras del producto. Nancy le proporciona el código de barras 1259374. El Sistema elimina el producto del Almacén.

(6.1) "Nancy elimina un producto y el Sistema avisa que el producto no existe en el Almacén".

Nancy solicita al Sistema eliminar un producto del Almacén. El Sistema le pide el código de barras del producto. Nancy le proporciona el código 1259374. El Sistema informa que no eliminó el producto pues no existe en el Almacén.

(6.2) Similar al Escenario 2.2.

(6.3) Similar al Escenario 2.3.

(6.4) Similar al Escenario 2.4.

7. Caso de prueba: "Modificar producto"

(7) "Nancy modifica la información de un producto exitosamente".

Nancy solicita al Sistema modificar la información de un producto. El Sistema le pide el código de barras del producto. Nancy proporciona el código de barras 1259374. El Sistema reconoce que la información que se va a modificar es de un Libro y muestra una forma con su información:

Código de barras: 995478.
Título: Mil y una ... la herida de Paulina, Las.
Género: Novela.
Portada: elenap.jpg.
Precio: 100.00.
Autor: Poniatowska Elena.
Editorial: Plaza & Janés.

Nancy decide modificar la información de precio con un costo de $120.00. Nancy le indica al Sistema que guarde los cambios. El Sistema guarda los cambios.

(7.1) "Nancy solicita modificar un producto y el Sistema avisa que el producto no existe en el Almacén".

Nancy solicita al Sistema modificar la información de un producto. El Sistema le pide el código de barras del producto. Nancy proporciona el código 995477. El Sistema avisa que no existe el producto en el Almacén.

(7.2) Similar al Escenario 2.2.

(7.3) Similar al Escenario 2.3.

(7.4) Similar al Escenario 2.4.

8. Caso de prueba: "Marcar estrenos"

(8) "Nancy marca los productos como estrenos".

Nancy solicita al Sistema buscar productos para hacer una selección de nuevos estrenos. El Sistema le muestra los tipos de productos. Nancy elige buscar CDs. Nancy emite la consulta según el siguiente criterio:

Título: (vacío).
Intérprete: (vacío).
Género: Rock.

El Sistema le muestra la siguiente información resultante de la búsqueda de CDs:

Código de barras	Título	Intérprete	Precio	Género	Portada	Disquera
1654389	In the Flesh Live	Waters, Roger	250	Rock Progresivo	Live.jpg	Sony Music
7498320	Is Anybody Out There?	Pink Floyd	300	Rock Progresivo	Pinkf.jpg	EMI
1189600	Genesis Archive	Genesis	650	Rock Progresivo	Genesis.jpg	EMI

Nancy marca como estrenos los productos con códigos de barras 1654389 y 1189600. El Sistema marca como estrenos los productos con códigos de barra 1654389 y 1189600.

(8.1) Similar al Escenario 2.2.

(8.2) Similar al Escenario 2.3.

(8.3) Similar al Escenario 2.4.

9. Caso de prueba: "Desmarcar estrenos"

(9) "Nancy quita la marca de estreno a los productos".

Nancy solicita al Sistema desmarcar productos de estreno. El Sistema le muestra los siguiente estrenos:

Código de barras	Título	Intérprete	Precio	Género	Portada	Disquera
1654389	In the Flesh Live	Waters, Roger	250	Rock Progresivo	Live.jpg	Sony Music
7498320	Is Anybody Out There?	Pink Floyd	300	Rock Progresivo	Pinkf.jpg	EMI
1189600	Genesis Archive	Genesis	650	Rock Progresivo	Genesis.jpg	EMI

Nancy selecciona desmarcar los estrenos con códigos de barras 1654389 y 1189600. El Sistema quita la marca de estreno a los productos con códigos de barras 1654389 y 1189600.

(9.1) Similar al Escenario 2.3.

(9.2) Similar al Escenario 2.4.

10. Caso de prueba: "Buscar producto"

(10) "Nancy busca productos exitosamente".

Nancy solicita al Sistema buscar productos. El Sistema le muestra los tipos de productos. Nancy elige buscar CDs. Nancy emite la consulta según el siguiente criterio:

Título: (vacío).
Intérprete: (vacío).
Género: Rock.

El Sistema le muestra la siguiente información resultante de la búsqueda de CDs:

Código de barras	Título	Intérprete	Precio	Género	Portada	Disquera
1654389	In the Flesh Live	Waters, Roger	250	Rock Progresivo	Live.jpg	Sony Music
7498320	Is Anybody Out There?	Pink Floyd	300	Rock Progresivo	Pinkf.jpg	EMI
1189600	Genesis Archive	Genesis	650	Rock Progresivo	Genesis.jpg	EMI

(10.1) "Nancy emite una búsqueda pero no hay productos asociados".

Nancy solicita al Sistema buscar productos. El Sistema le muestra los tipos de productos. Nancy elige buscar CDs. Nancy emite la consulta según el siguiente criterio:

Título: Marron.
Intérprete: (vacío).
Género: Rock.

El Sistema le avisa que no hay productos resultantes de la búsqueda.

(10.2) "Nancy emite una búsqueda sin proporcionar un criterio".

Nancy solicita al Sistema buscar productos. El Sistema le muestra los tipos de productos. Nancy elige buscar CDs. Nancy emite la consulta sin proporcionar un criterio. El Sistema le muestra todos los CDs.

(10.3) Similar al Escenario 2.2.

(10.4) Similar al Escenario 2.3.

(10.5) Similar al Escenario 2.4.

11. Caso de prueba: "Buscar CD"

(11) "Nancy llena el criterio de búsqueda de un CD".

El Sistema le muestra las opciones que tiene para buscar:

- Título.
- Intérprete.
- Género.

Nancy elige buscar por intérprete e introduce "Mark".

12. Caso de prueba: "Buscar DVD"

(12) Similar al caso de prueba 11 solo que con los criterios de búsqueda del DVD.

13. Caso de prueba: "Buscar Libro"

(13) Similar al caso de prueba 11 solo que con los criterios de búsqueda de los libros.

14. Caso de prueba: " Limpiar"

El caso de uso correspondiente a este caso de prueba se realiza como opción de otros casos de uso.

15. Caso de prueba: "Salir"

Nancy indica al Sistema que salga de la operación de buscar CDs. El Sistema pide a Nancy confirmar la salida. Nancy confirma la salida. Termina la ejecución del Sistema.

16. Caso de prueba: "Problema en la Base de Datos"

El caso de uso correspondiente a este caso de prueba se realiza como opción de otros casos de uso.

Sitio Web

17. Caso de prueba: "Entrar a la página principal de la tienda"

(17) "Ana entra a la página principal".

Ana entra a la página principal a través de un navegador proporcionando la dirección electrónica del Sitio. El Sistema muestra la página principal que contiene las carpetas pertenecientes a las categorías de *home* y una para cada tipo de producto. La página principal corresponde a la carpeta *home* donde se despliega la información de todos los estrenos.

18. Caso de prueba: "Seleccionar carpeta"

El caso de uso correspondiente a este caso de prueba se realiza como parte de otros casos de uso.

19. Caso de prueba: "Seleccionar home" (heredar "Seleccionar carpeta")

(19) "Ana selecciona la carpeta home".

Ana selecciona la carpeta home. El Sistema muestra los estrenos:

- Un CD con título "Reveal", portada "Reveal.gif", precio "150" e intérprete "REM".
- Un DVD con título "El Patriota", elenco "Mel Gibson", portada "Patriota.gif", precio "223".
- Un Libro con título "Así son las cosas", portada "Cosas.gif", precio "210", autor "John Brockman" y editorial "P & J".

20. Caso de prueba: "Seleccionar carpeta producto" (heredar "Seleccionar carpeta")

(20) "Ana selecciona la carpeta de CD".

Ana selecciona la carpeta de CD. El Sistema muestra los productos de estreno para CD con la siguiente información:

- Título "Reveal", portada "Reveal.gif", precio "150", intérprete "REM".
- Título "J. LO", portada "JLO.gif", precio "180" e intérprete "Jennifer López".
- Título "Maquillaje", portada " Maquillaje.gif", precio "140" e intérprete "Zurdoc".

El Sistema también muestra la jerarquía de CD dada por las categorías de Pop, Clásica y Alternativo.

21. Caso de prueba: "Muestra jerarquía"

El caso de uso correspondiente a este caso de prueba se realiza como parte de otros casos de uso.

22. Caso de prueba: "Muestra estrenos"

El caso de uso correspondiente a este caso de prueba se realiza como parte de otros casos de uso.

23. Caso de prueba: "Navegar la jerarquía de productos"

(23) "Ana selecciona la categoría de Pop de la carpeta de CD".

Ana selecciona la categoría de Pop de la carpeta de CD. El Sistema muestra los estrenos de CD para la categoría de Pop:

- Título "Reveal", portada "Reveal.gif", precio "150", intérprete "REM".
- Título "J. LO", portada "JLO.gif", precio "180", intérprete "Jennifer López".

24. Caso de prueba: "Agregar producto al carrito"

(24) "Ana agrega un CD al carrito".

Ana agrega al carrito el CD de título "Reveal", precio "150", intérprete "REM" y disquera "Warner Music".
El Sistema muestra el carrito con el producto agregado. La información del carrito es:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
1	Reveal	150	150
1	J. LO	180	180
			Total 330

25. Caso de prueba: "Revisar el carrito"

(25) "Ana revisa el carrito".

Ana solicita al Sistema revisar el carrito. El Sistema muestra los productos que contiene el carrito mostrando la siguiente información:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
2	Reveal	150	300
1	J. LO	180	180
			Total 480

(25.1) "Ana modifica el número de unidades de un producto en el carrito".

La información del carrito antes de que Ana modifique la el número de unidades del segundo producto es:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
2	Reveal	150	300
1	J. LO	180	180
			Total 480

Ana modifica el número de unidades del segundo producto a 3 unidades. El Sistema modifica el carrito y éste tiene la nueva información:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
2	Reveal	150	300
3	J. LO	180	540
			Total 840

26. Caso de prueba: "Eliminar producto del carrito"

(26) "Ana elimina un producto del carrito".

La información del carrito antes de que Ana elimine el segundo producto es:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
2	Reveal	150	300
3	J. LO	180	540
			Total 840

Ana elimina el segundo producto. El Sistema modifica el carrito y éste tiene la nueva información:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
2	Reveal	150	300
			Total 300

27. Caso de prueba: "Ver información detallada de un producto"

(27) "Ana ve la información detallada de un CD".

Ana selecciona un CD para ver su información detallada. El Sistema le muestra la siguiente información del CD:

- Título "Reveal", género "Pop", portada "Reveal.gif", precio "150", intérprete "REM" y disquera "Warner Music".

28. Caso de prueba: "Buscar productos"

(28) "Ana busca CDs".

Ana está en la carpeta de CD. Ana emite una búsqueda con el siguiente criterio: intérprete "REM". El Sistema le muestra la información general de los CD que cumplen el criterio de búsqueda con el siguiente resultado:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
Reveal	REM	150	Warner Music
Automatic For The People	REM	180	Warner Music

29. Caso de prueba: "Ver información general de un producto"

El caso de uso correspondiente a este caso de prueba se realiza como parte de otros casos de uso.

30. Caso de prueba: "Ver información del carrito"

El caso de uso correspondiente a este caso de prueba se realiza como parte de otros casos de uso.

31. Caso de prueba: "Pasar a caja"

(31) "Ana pasa a la caja".

Ana solicita al Sistema que desea comprar los productos del carrito. El Sistema pide a Ana los datos para la compra. Ana proporciona los siguientes datos:

- Nombre: Ana.
- Apellido: Lorite.
- Teléfono: 34 54 65 76.
- Dirección: Fuentes #204
- Colonia: San Pedro.
- Código Postal: 32100.
- Ciudad: Cuernavaca.
- Estado: Morelos.
- Forma de pago: Tarjeta de Crédito.
- Nombre del tarjetahabiente: Ana Lorite.
- Número: 2311992323.
- Tipo: Visa.
- Fecha de expiración: Julio 2002.

El Sistema muestra al Cliente los datos proporcionados con anterioridad y la siguiente información del carrito:

Cantidad	Productos seleccionados	Costo unitario	Subtotal
2	Reveal	150	300
1	J. LO	180	180
		Total	480

Ana indica al Sistema que realice la compra. Ana recibe el número de pedido 324.

(31.1) "Ana cancela la compra".

Ana solicita al Sistema que desea comprar los productos del carrito. El Sistema pide a Ana los datos para la compra. Ana proporciona los siguientes datos:

- Nombre: Ana.
- Apellido: Lorite.
- Teléfono: 34 54 65 76.
- Dirección: Fuentes #204
- Colonia: San Pedro.
- Código Postal: 32100.
- Ciudad: Cuernavaca.
- Estado: Morelos.
- Forma de pago: Tarjeta de Crédito.
- Nombre del tarjetahabiente: Ana Lorite.
- Número: 2311992323.
- Tipo: Visa.
- Fecha de expiración: Julio 2002.

Ana cancela la compra. El Sistema elimina los productos del carrito. La compra no es realizada.

Conclusiones

En el desarrollo del sistema de comercio electrónico *Sistema de Ventas Café* se tuvieron una serie de experiencias y aprendizajes valiosos pues se pudo experimentar que el desarrollo de software de calidad no requiere solamente de un conjunto de buenos programadores sino también de un proceso que lo guíe y de un equipo cuya experiencia en el proceso y tecnología es fundamental en el éxito del proyecto.

Durante el desarrollo del sistema se decidió cambiar la plataforma de desarrollo. Se decidió reemplazar el uso de J2SDK1.3 y JSPs por *Java 2 Enterprise Edition* (J2EE). Este hecho tuvo asociadas una serie de experiencias:

- Se hizo evidente la importancia de tener en cuenta los requerimientos no funcionales del sistema desde una etapa temprana. Se tuvo que invertir una gran cantidad de tiempo (aproximadamente 6 meses) para entender, aprender, instalar y manejar de buena manera la tecnología J2EE así como para encontrar alguna base de datos compatible con dicha tecnología (en este caso ORACLE8i ya que era la opción más viable por su disponibilidad y compatibilidad de su *driver* con J2EE). Este hecho representó cambios en el diseño y la fecha prevista para la entrega del sistema.

- No se pudieron llevar a cabo varias tareas administrativas del Proceso Unificado debido a la falta de experiencia tanto en la aplicación del proceso como en la tecnología J2EE. Esto hacia difícil el poder distribuir las actividades entre los dos miembros del equipo pues se tenía que descubrir conjuntamente a cada paso lo que el proceso y la tecnología requerían. Por ejemplo, dentro del modelo de implementación no se hizo un plan de integración ya que se careció de una planeación de iteraciones. La falta de planeación puede conducir al desarrollo de un software de baja calidad pues es posible intentar implementarlo de manera rápida y poco detallada para alcanzar las fechas de entrega previstas.

- La implementación se realizó paralelamente con el diseño pues no se podía saber con anticipación cuales eran las decisiones convenientes para distribuir la funcionalidad del sistema entre las clases de diseño hasta no haber empezado a experimentar con la implementación de algunos componentes. El modelo de implementación fue usado como una herramienta para documentar la manera en que fue construido (en términos de la programación de componentes) el sistema de comercio electrónico más que para planear su construcción, que es el verdadero objetivo. Esto permitió experimentar la conveniencia de poder contar dentro del equipo con gente que conozca la plataforma de desarrollo y así evitar los riesgos inherentes tales como la entrega retrasada del producto y en caso extremo el fracaso del mismo.

El hecho de no contar con un usuario real que pusiera los límites, usos y restricciones para el sistema hizo que se intentaran realizar funcionalidades que adquirían una mayor complejidad de la que se podría requerir o que tarde o temprano hacían surgir preguntas acerca de la necesidad de incluirlas. Los casos de uso resuelven bien este problema pues para su captura

es siempre necesario tener en mente al usuario mediante la pregunta ¿qué debe hacer el sistema *para cada usuario*? la cual obliga a pensar en el valor agregado que el sistema puede proporcionar al usuario real.

Se percibió que los casos de uso ahorran el tiempo en la captura de requerimientos pues ésta se realiza mediante el lenguaje natural y del usuario sin que sea necesario aprender una notación difícil. Debido al uso del lenguaje natural los casos de uso también proporcionan una excelente manera de llegar a un acuerdo entre desarrolladores y usuarios de forma que el sistema haga lo que estos esperan.

La buena captura de los requerimientos ahorra tiempo, recursos y dinero además de permitir construir un sistema que realmente proporcione los valores requeridos por el usuario.

Las clases de análisis al refinar la funcionalidad descrita por los casos de uso permitieron descubrir inconsistencias y mejores maneras de realizar funcionalidades. También se pudo empezar a organizar y agrupar el comportamiento en clases. Se empezó a intuir como se podía reutilizar el comportamiento de una clase en varias *realizaciones de caso de uso*. Desarrollar el modelo de análisis fue una buena opción para refinar los requerimientos además de sentar una base más sólida para el diseño.

La arquitectura del software pudo ser proyectada de manera sencilla y directa a la arquitectura del hardware debido al modelo multicapa distribuido que emplea J2EE.

Cuando llegó el momento de aplicar las pruebas se detectaron defectos mínimos que tenían que ver con pequeños errores de implementación. Un software bien planeado y documentado con bases sólidas dadas desde las etapas tempranas del desarrollo permite una minimizar el número de errores los cuales pueden ser corregidos sin ocupar demasiados recursos.

UML es un lenguaje que permite una buena representación de los modelos del Proceso Unificado. Minimiza el costo de generar y mantener la documentación y maximiza el contenido relevante de información.

Dentro de los trabajos futuros para el *Sistema de Ventas Café* se contemplan:

- La implantación de un esquema de seguridad el cual es indispensable en los sistemas de comercio electrónico.
- La implementación del pago con tarjeta de crédito.
- Creación de cuentas para los clientes del sitio lo cual evita pedir los datos del mismo cada vez que realiza una compra.

Con la experiencia adquirida durante el desarrollo de esta tesis pienso que el uso de la tecnología J2EE, el Proceso Unificado y UML son buenas opciones para el desarrollo de sistemas de comercio electrónico y los aplicaría en el desarrollo de un siguiente sistema. Conseguiría un grupo de desarrolladores que conozcan el Proceso Unificado y la plataforma J2EE pues su carencia impactó fuertemente la fecha de entrega del sistema de comercio electrónico. Pondría especial atención a la aplicación de las actividades de planeación propuestas en el Proceso Unificado pues conducen a la obtención de un software de calidad cuya fecha de entrega e inversión de recursos pueden ser bien calculados.

Apéndice A

Diagramas complementarios para los modelos del Proceso Unificado

Dentro de éste Apéndice se muestran los diagramas complementarios correspondientes al Capítulo 4. Para los diagramas se muestran referencias a su lugar de procedencia dentro del Capítulo antes mencionado.

A.1 Diagramas complementarios para el modelo de casos de uso

Administración

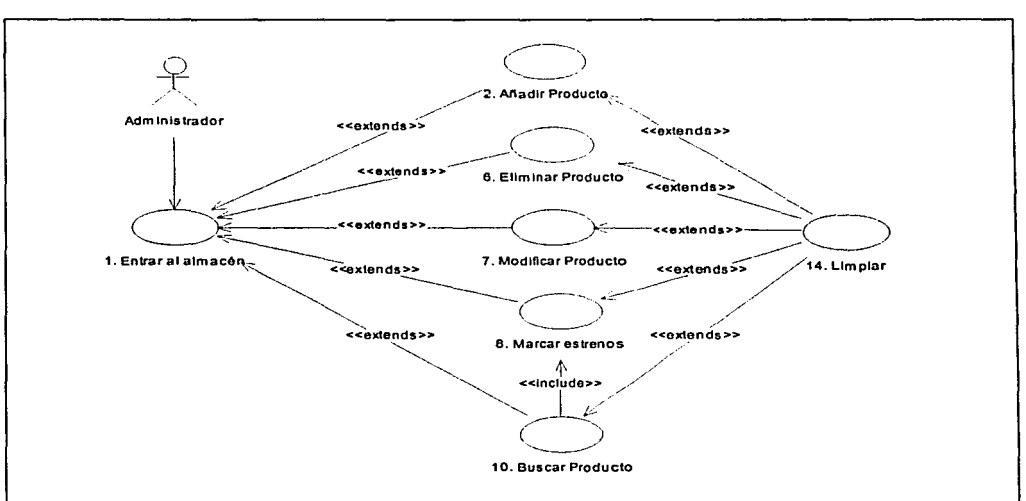


Figura A.1 Diagrama de casos de uso para enfatizar la relación del caso de uso 14.Limpiar con el resto de los casos de uso.

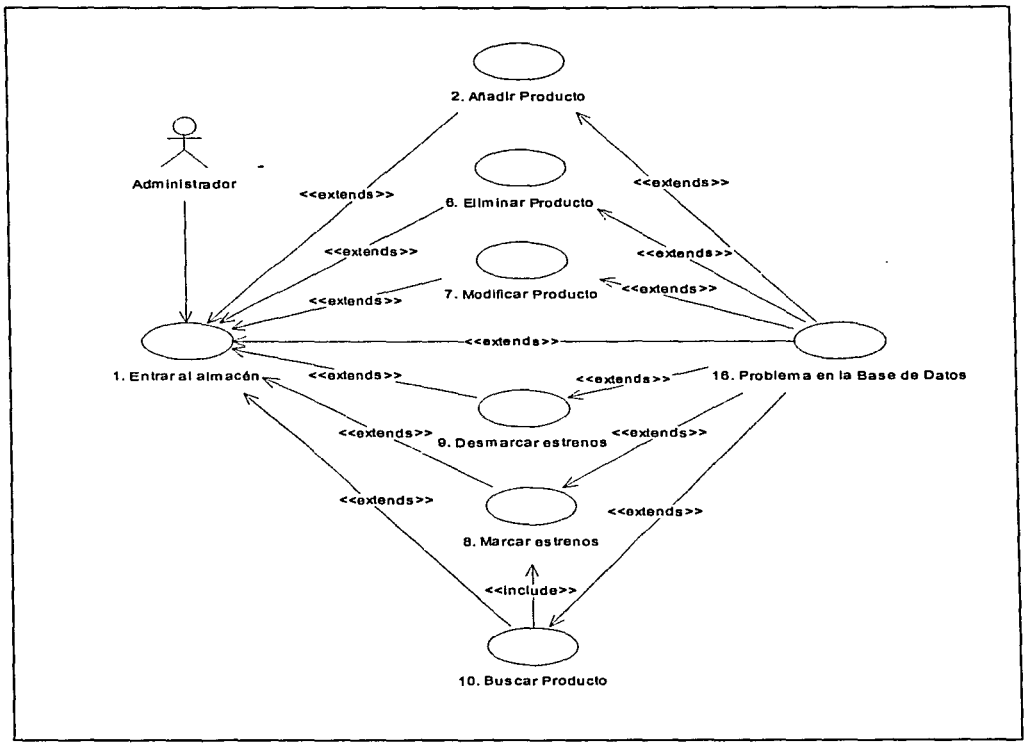


Figura A.2 Diagrama de casos de uso para enfatizar la relación del caso de uso 16.Problema en la Base de Datos con el resto de los casos de uso.

A.2 Diagramas complementarios para el modelo de diseño

4.3.1.2 Diagramas de clases para la relación entre paquetes

Administración

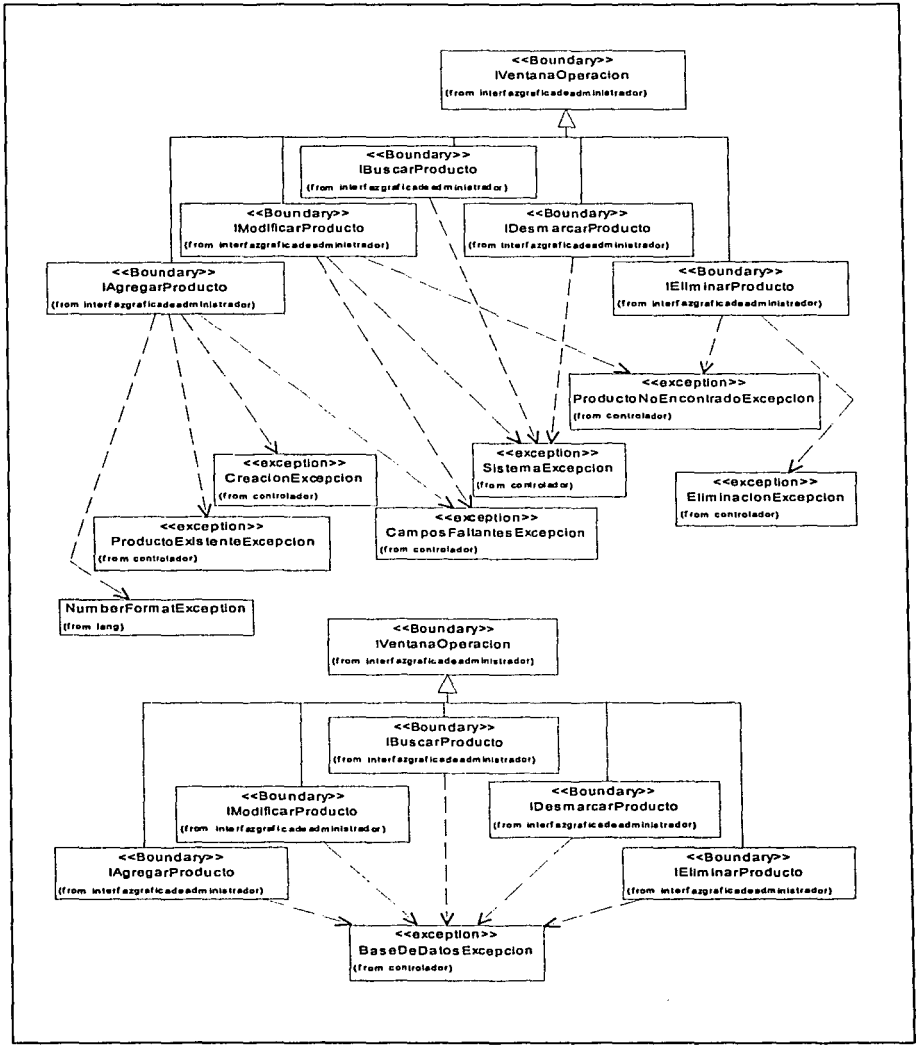


Figura A.3 Diagrama de clases para enfatizar la relación entre el subsistema interfazgraficadeadministrador con las excepciones definidas en el subsistema controlador.

4.3.1.3 Diagramas de clases de los subsistemas

Subsistema interfazgraficadeadministrador

La Figura A.4 muestra un diagrama de clases que enfatiza los atributos y métodos de las mismas. Las clases mostradas son auxiliares para el despliegue de las ventanas correspondientes a la administración del sistema.

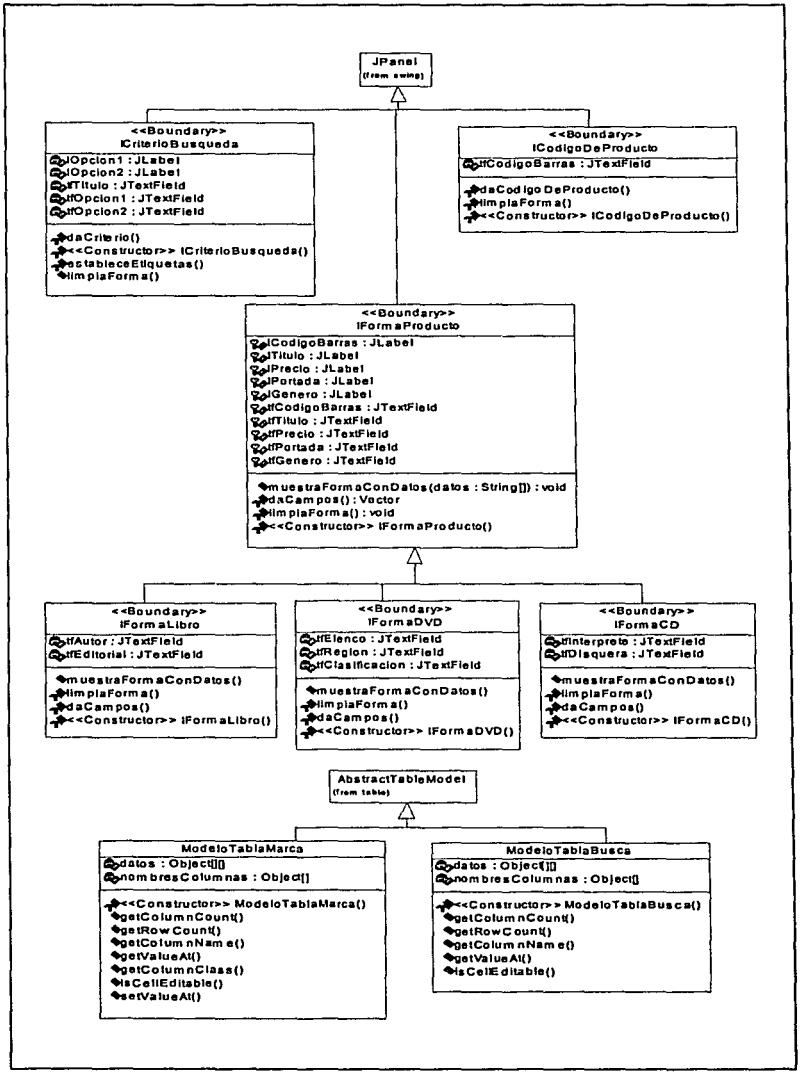


Figura A.4 Diagrama de clases para el subsistema interfazgraficadeadministrador.

Las Figura A.5, A.6 y A.7 muestran resto de los diagramas de clases complementarios para el subsistema *interfazgraficadeadministrador* los cuales enfatizan la relación de las ventanas principales mostradas en el Capítulo 4 con sus clases auxiliares (Figura A.4) y las clases de subsistemas no propios de la aplicación que ayudan al desarrollo con J2EE.

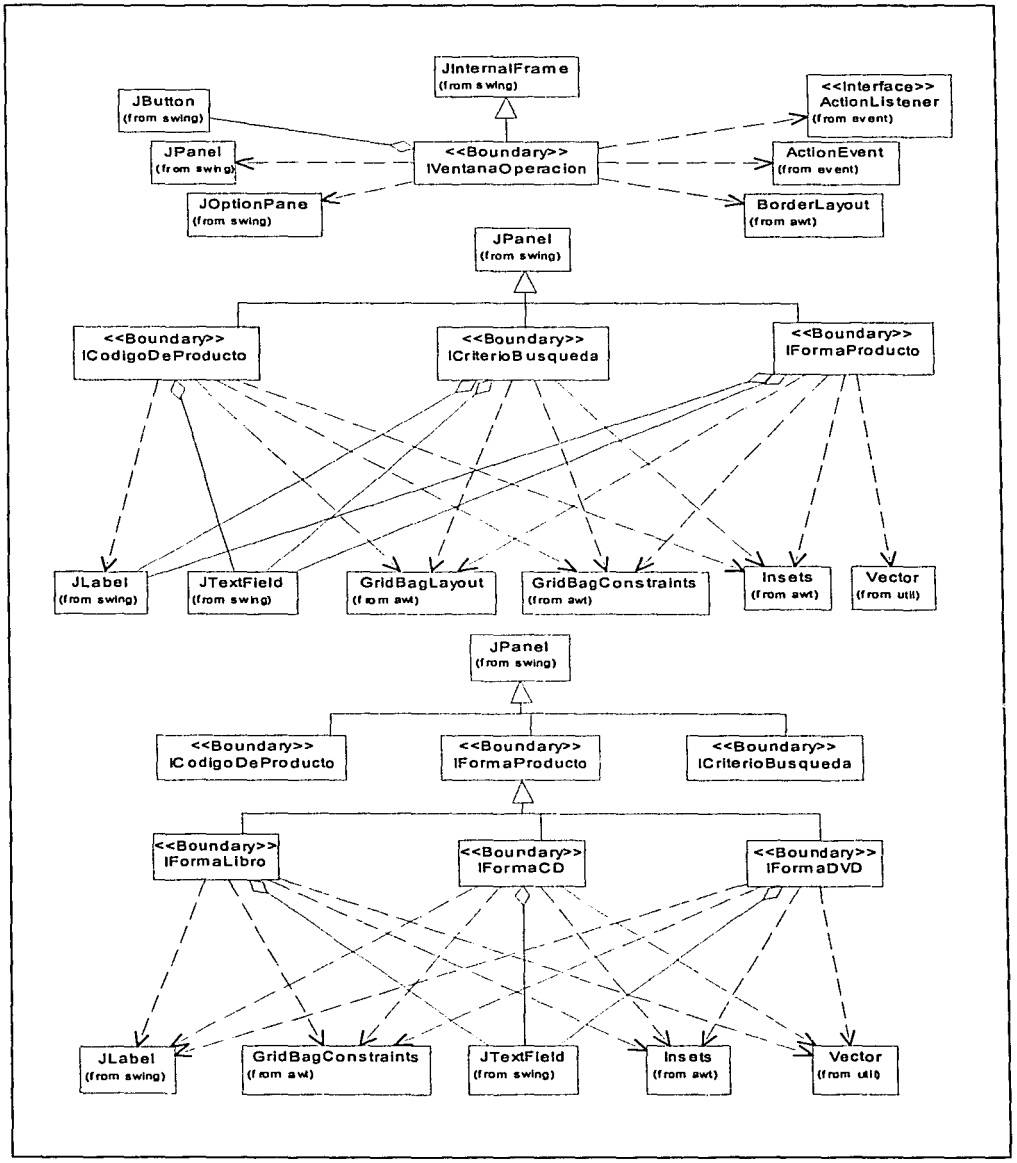


Figura A.5 Diagrama de clases para el subsistema interfazgraficadeadministrador.

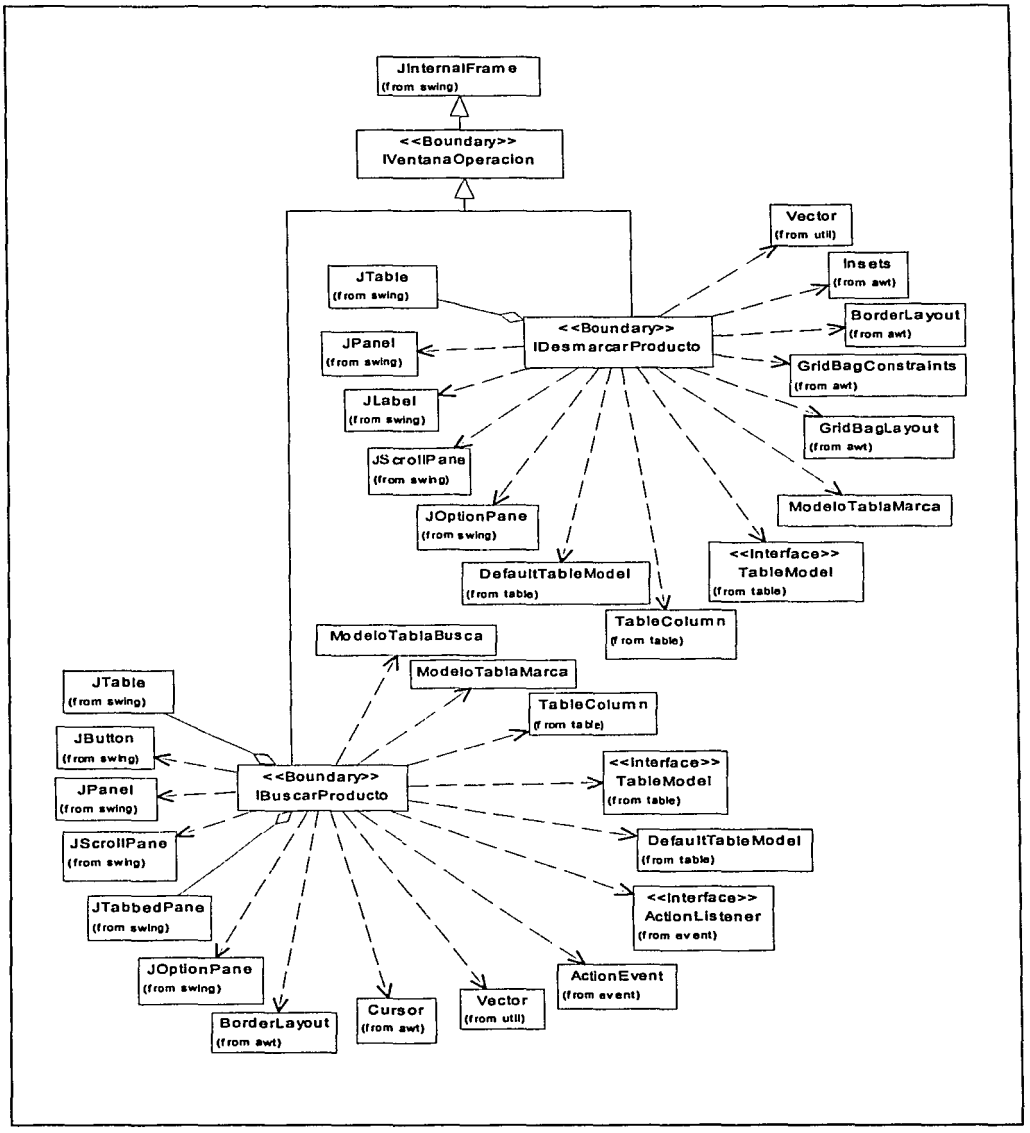


Figura A.6 Diagrama de clases para el subsistema interfazgraficadeadministrador.

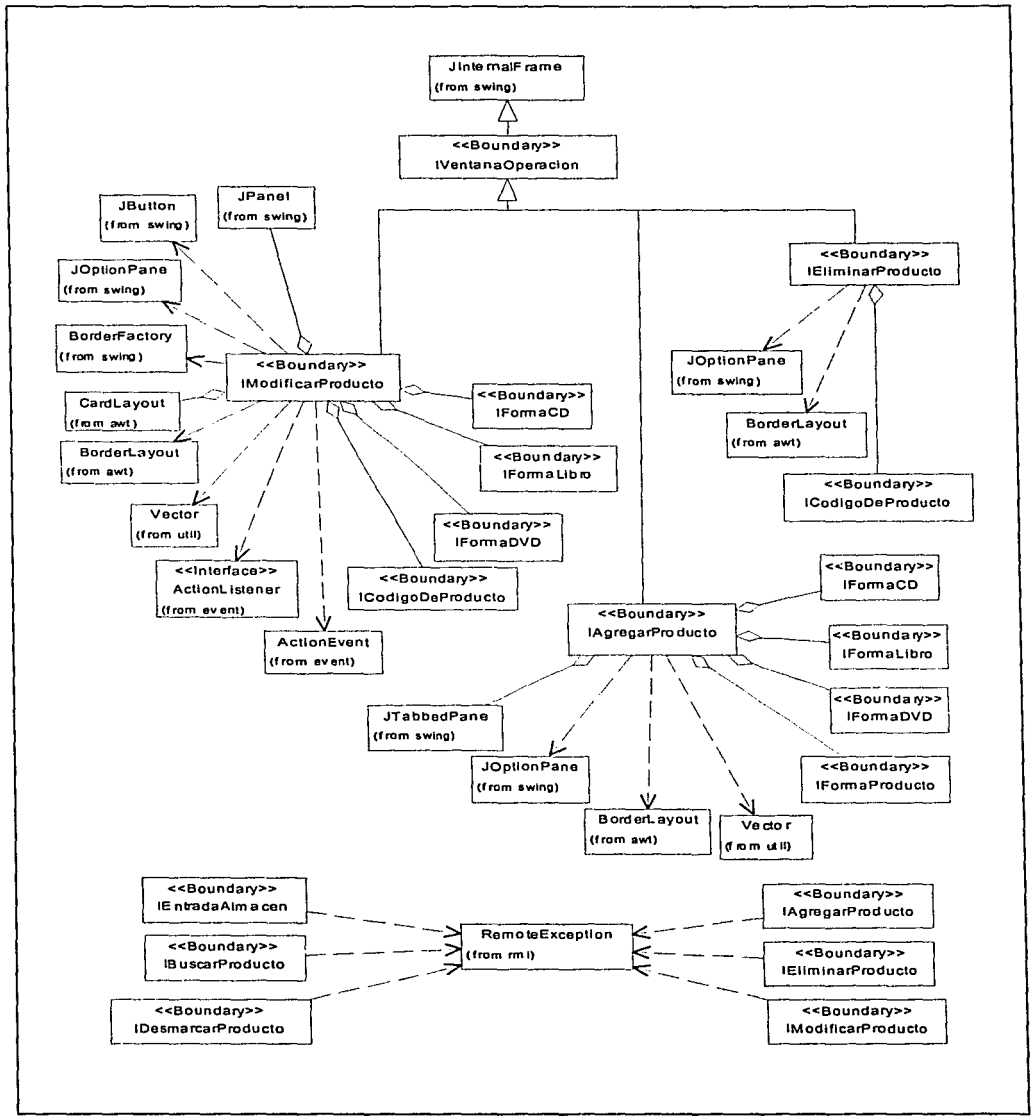


Figura A.7 Diagrama de clases para el subsistema interfazgraficadeadministrador.

Subsistema controlador

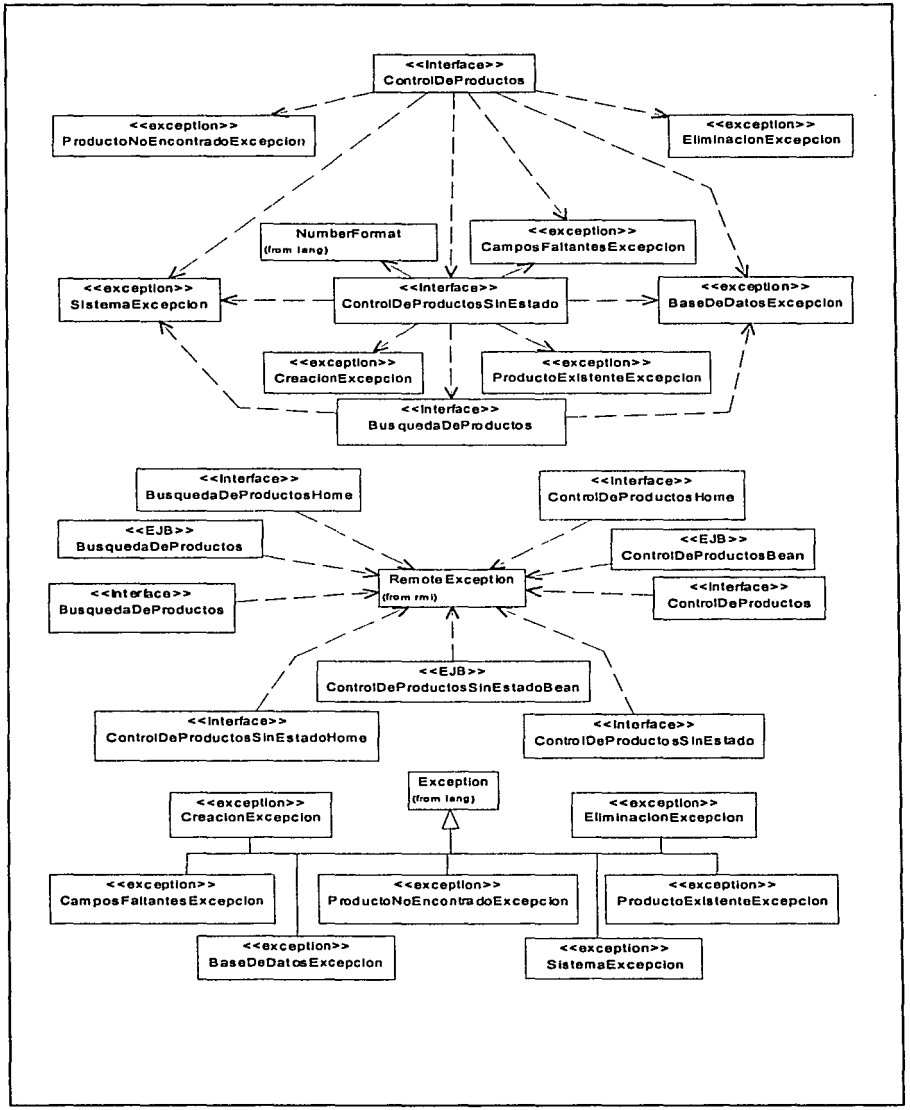


Figura A.8 Diagrama de clases que muestra las relaciones de las interfaces de los session beans con las excepciones.

Subsistema datos

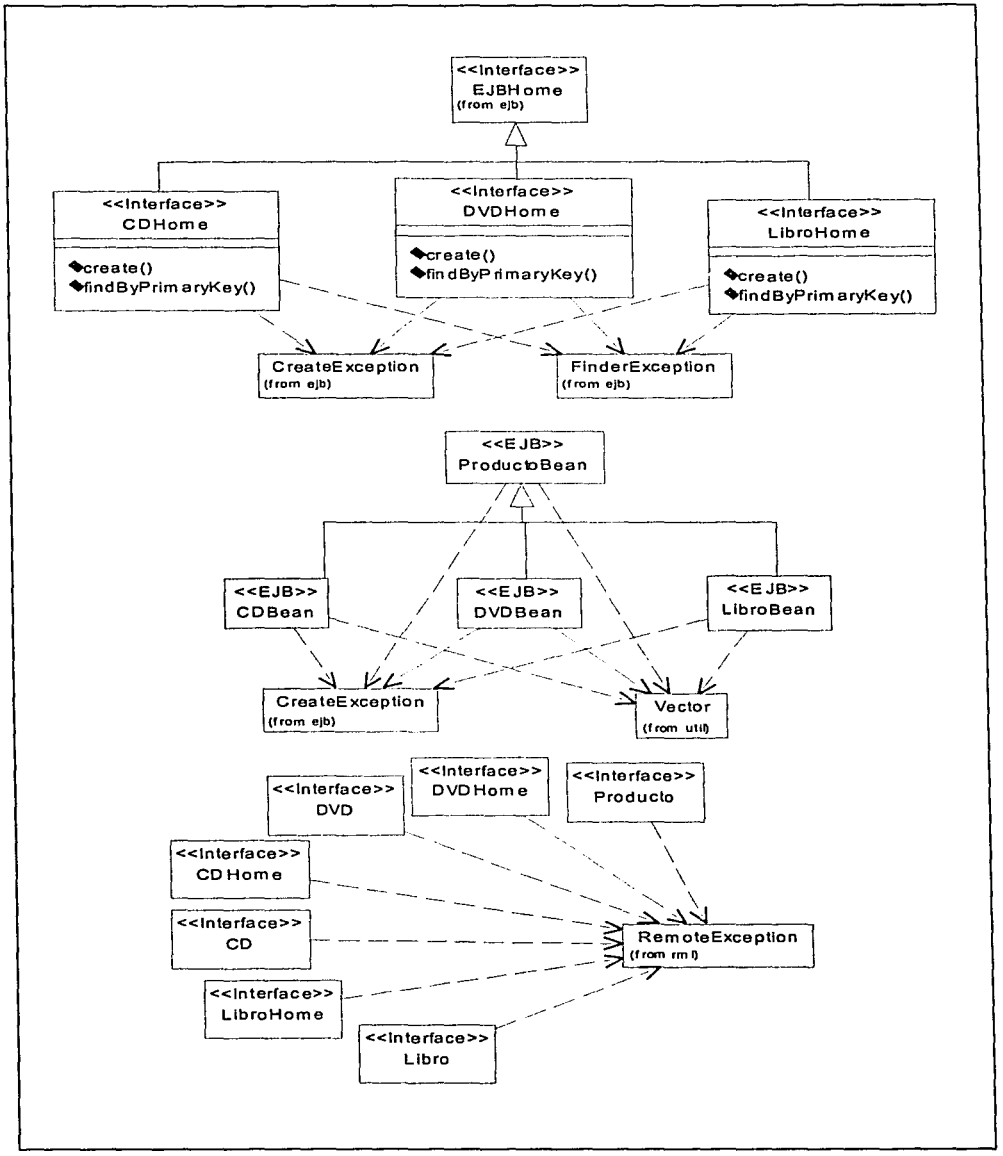


Figura A.9 Diagrama de clases que muestra las relaciones de las interfaces de los session beans con las excepciones.

4.3.1.4 Realizaciones de caso de uso-diseño

Administración

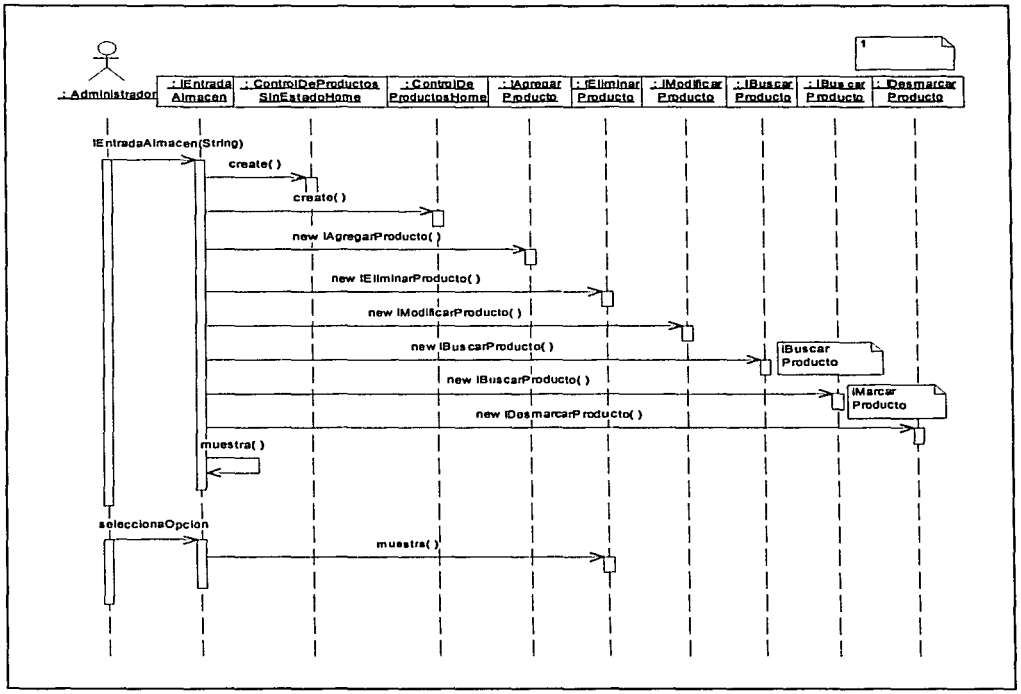


Figura A.10 Diagrama de secuencia para el caso de uso 1.Entrar al almacén.

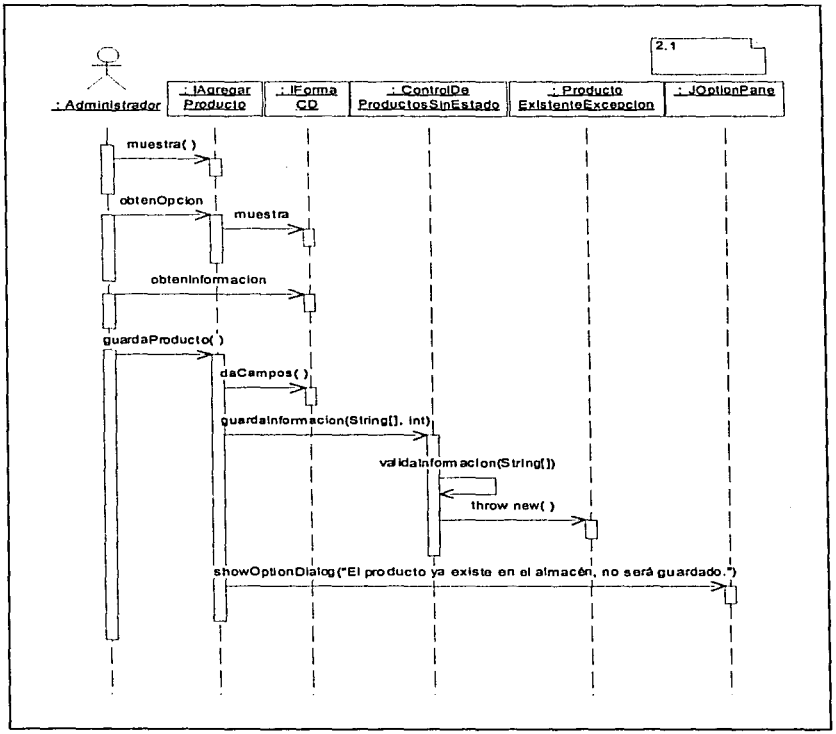


Figura A.11 Diagrama de secuencia para el caso de uso 2.Añadir Producto, flujo alternativo 2.1.

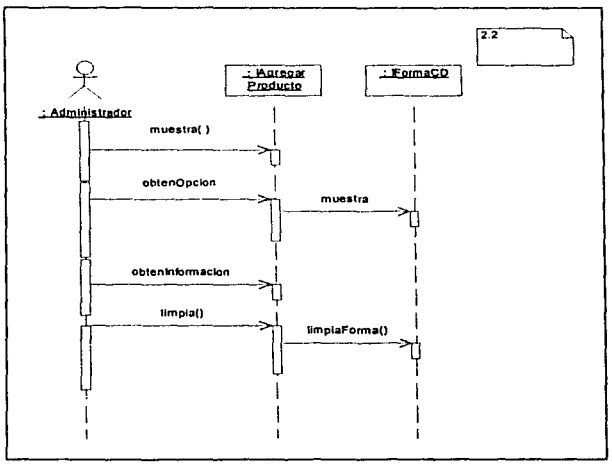


Figura A.12 Diagrama de secuencia para el caso de uso 2.Añadir Producto, flujo alternativo 2.2.

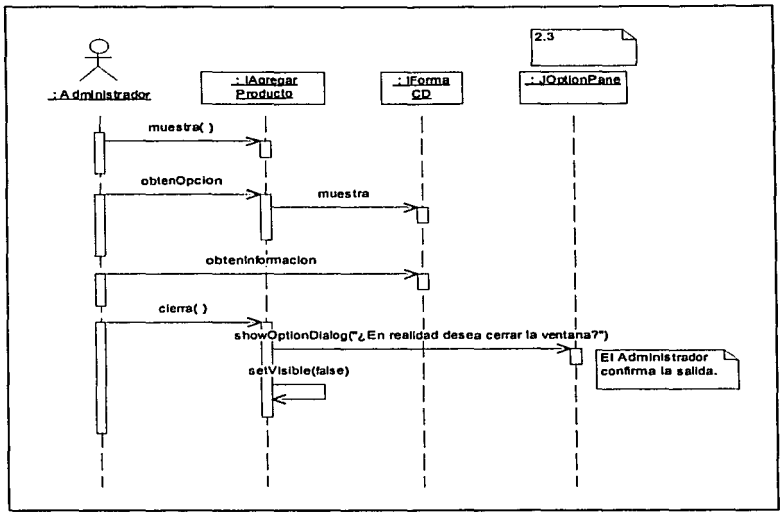


Figura A.13 Diagrama de secuencia para el caso de uso 2.Añadir Producto, flujo alternativo 2.3.

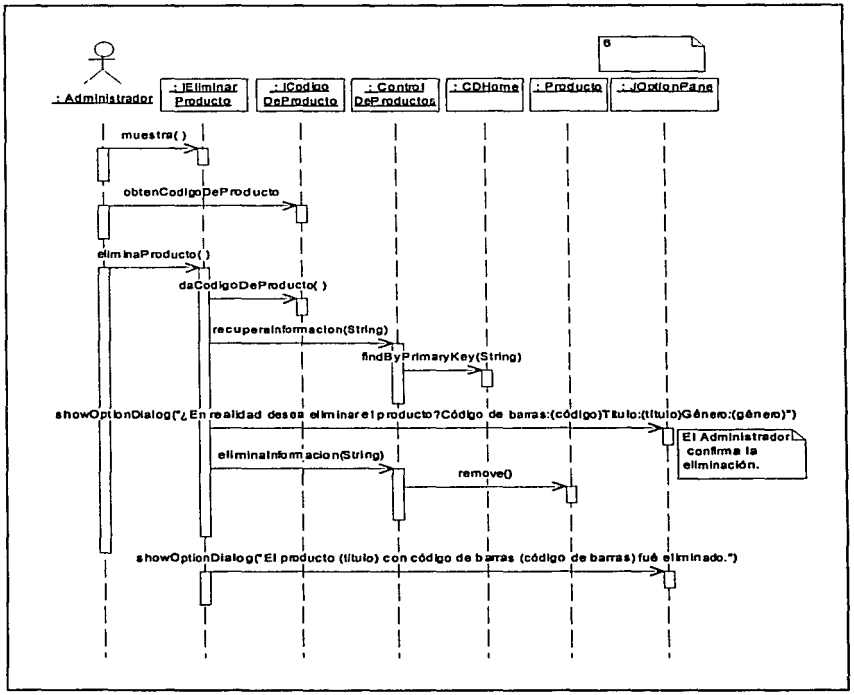


Figura A.14 Diagrama de secuencia para el caso de uso 6.Eliminar Producto.

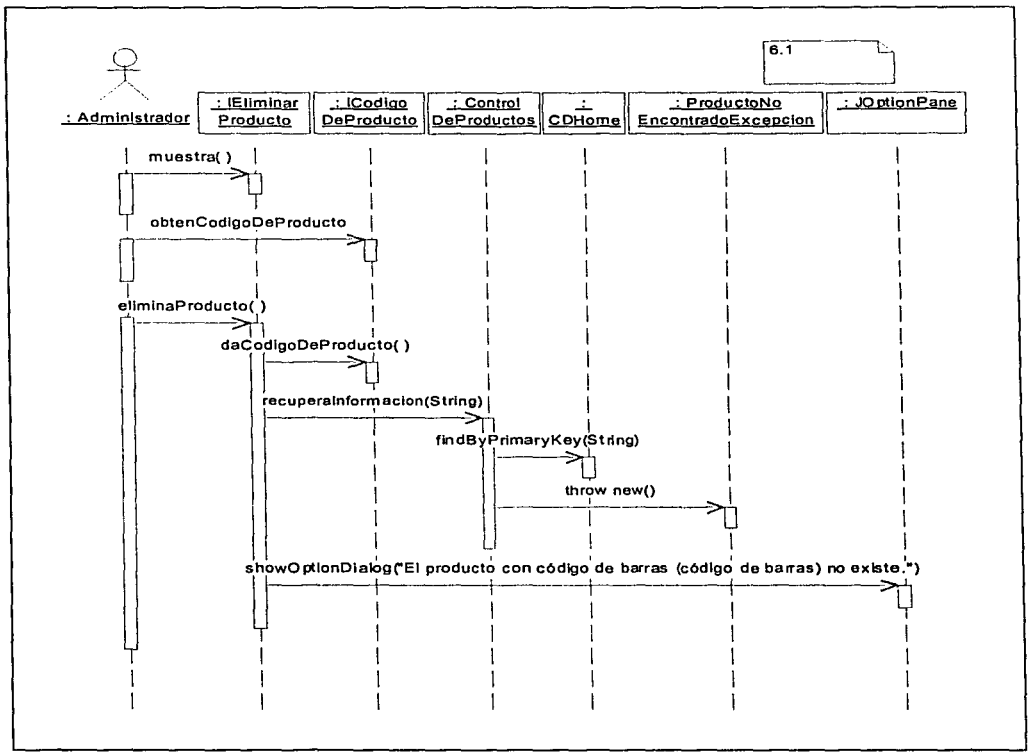


Figura A.15 Diagrama de secuencia para el caso de uso 6.Eliminar Producto, flujo alternativo 6.1.

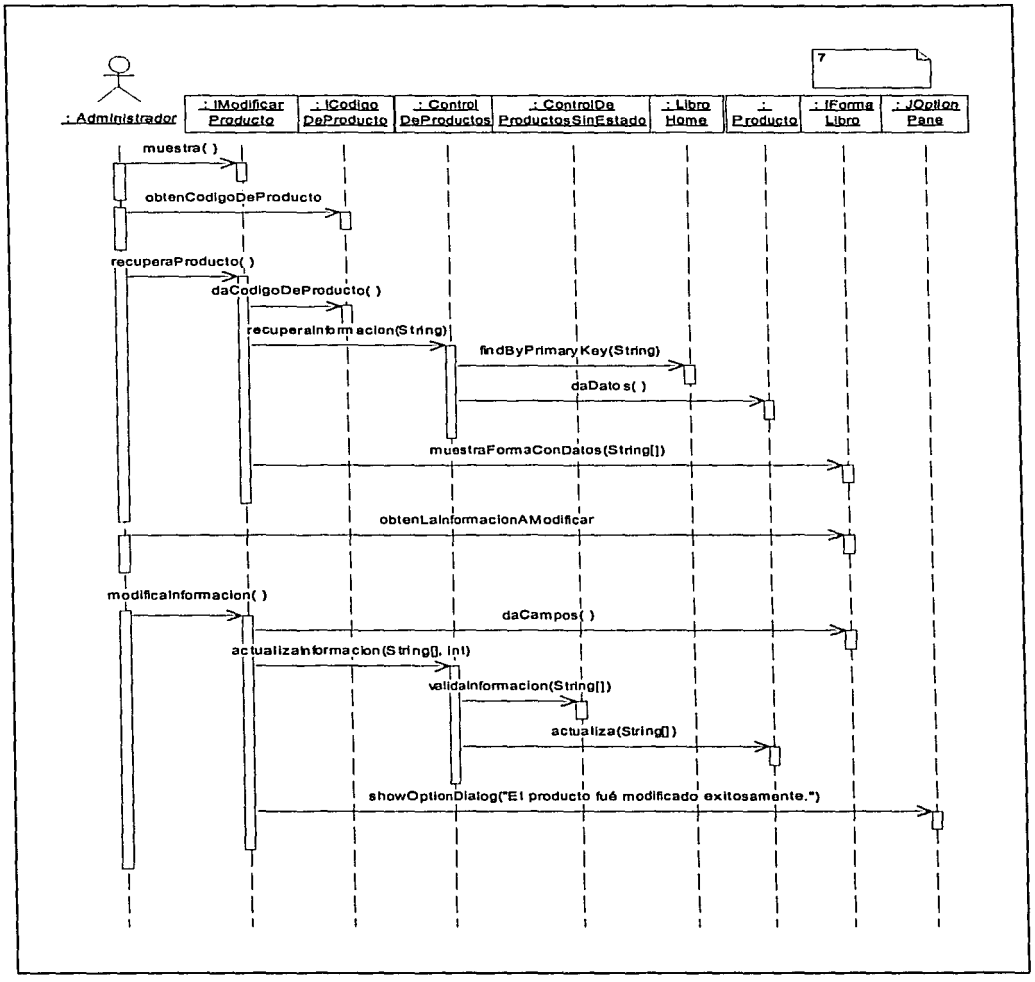


Figura A.16 Diagrama de secuencia para el caso de uso 7.Modificar Producto.

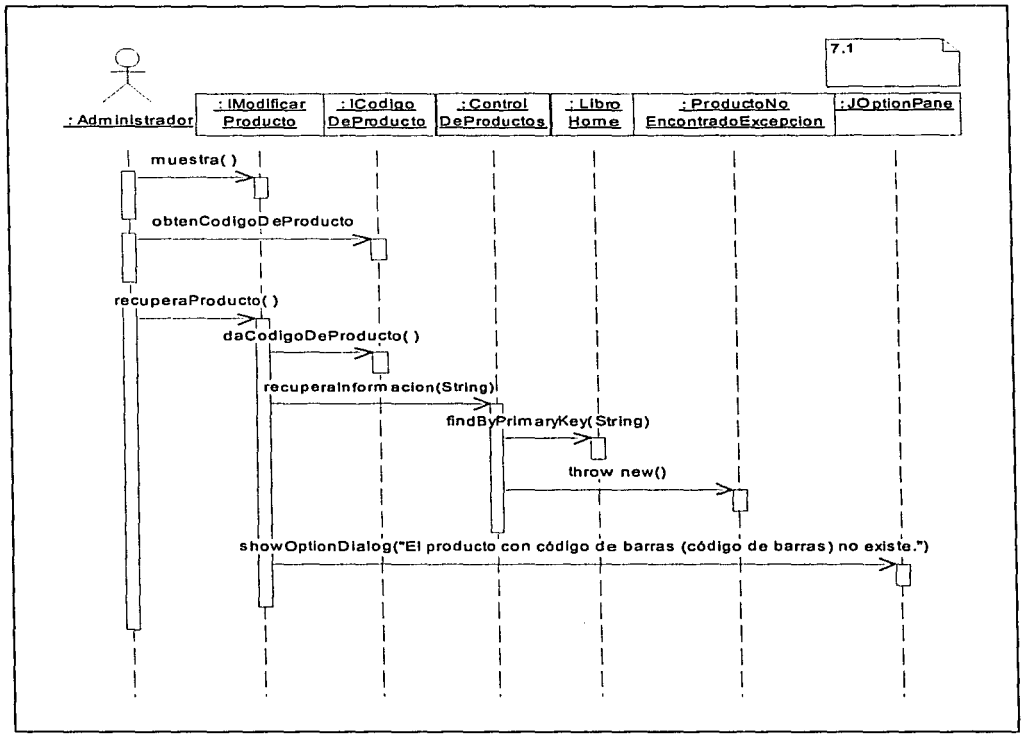


Figura A.17 Diagrama de secuencia para el caso de uso 7.Modificar Producto, flujo alternativo 7.1.

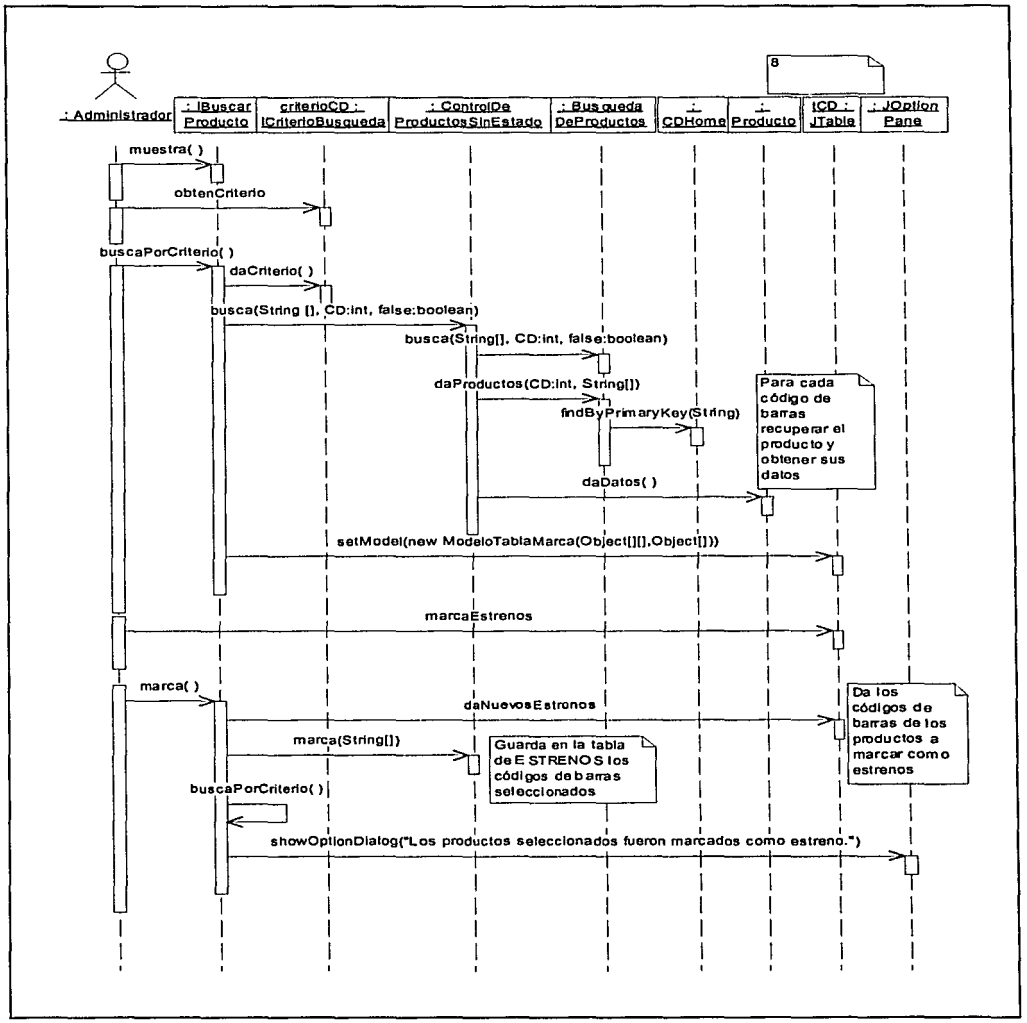


Figura A.18 Diagrama de secuencia para el caso de uso 8.Marcar Estrenos. El diagrama incluye la ejecución del caso de uso 10.Buscar Productos.

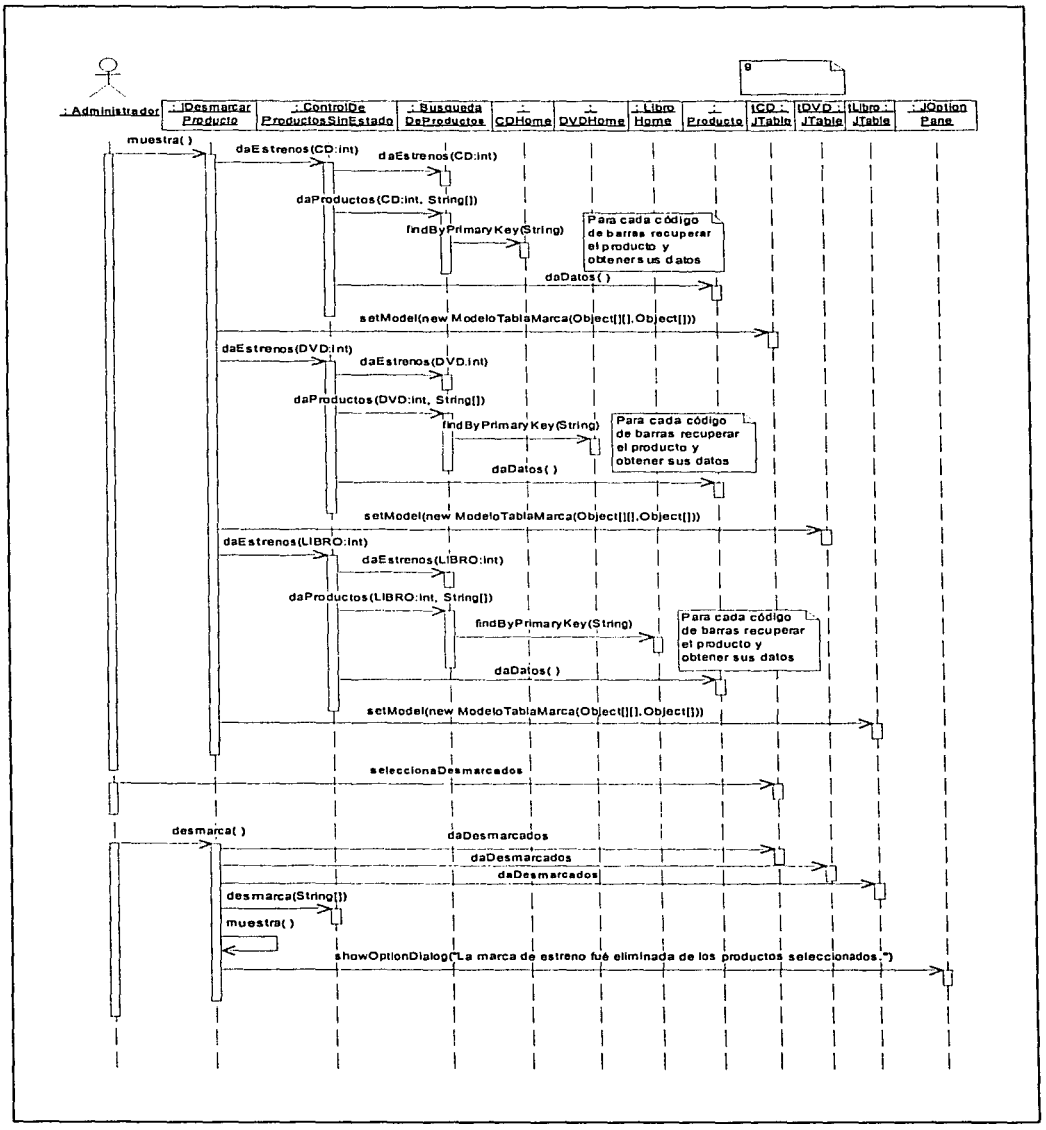


Figura A.19 Diagrama de secuencia para el caso de uso 9.Desmarcar estrenos.

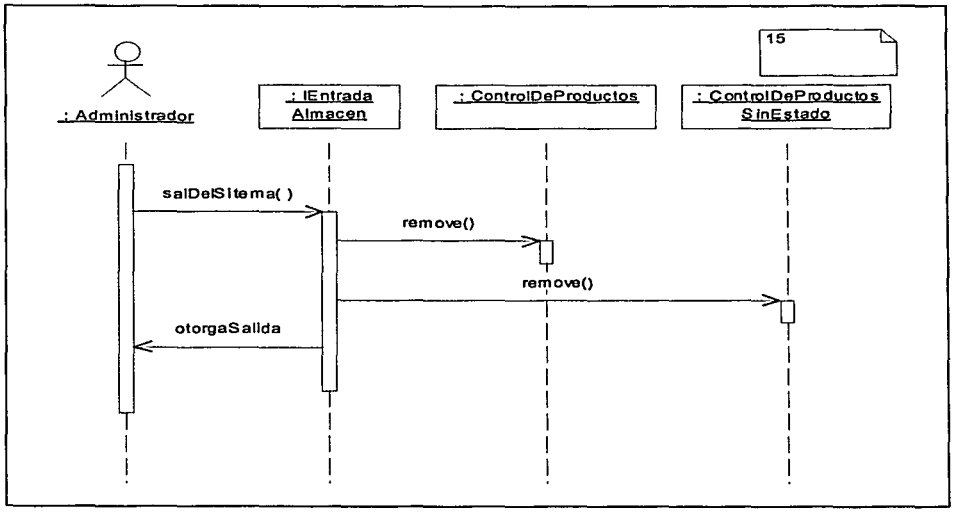


Figura A.20 Diagrama de secuencia para el caso de uso 15.Salir.

Sitio Web

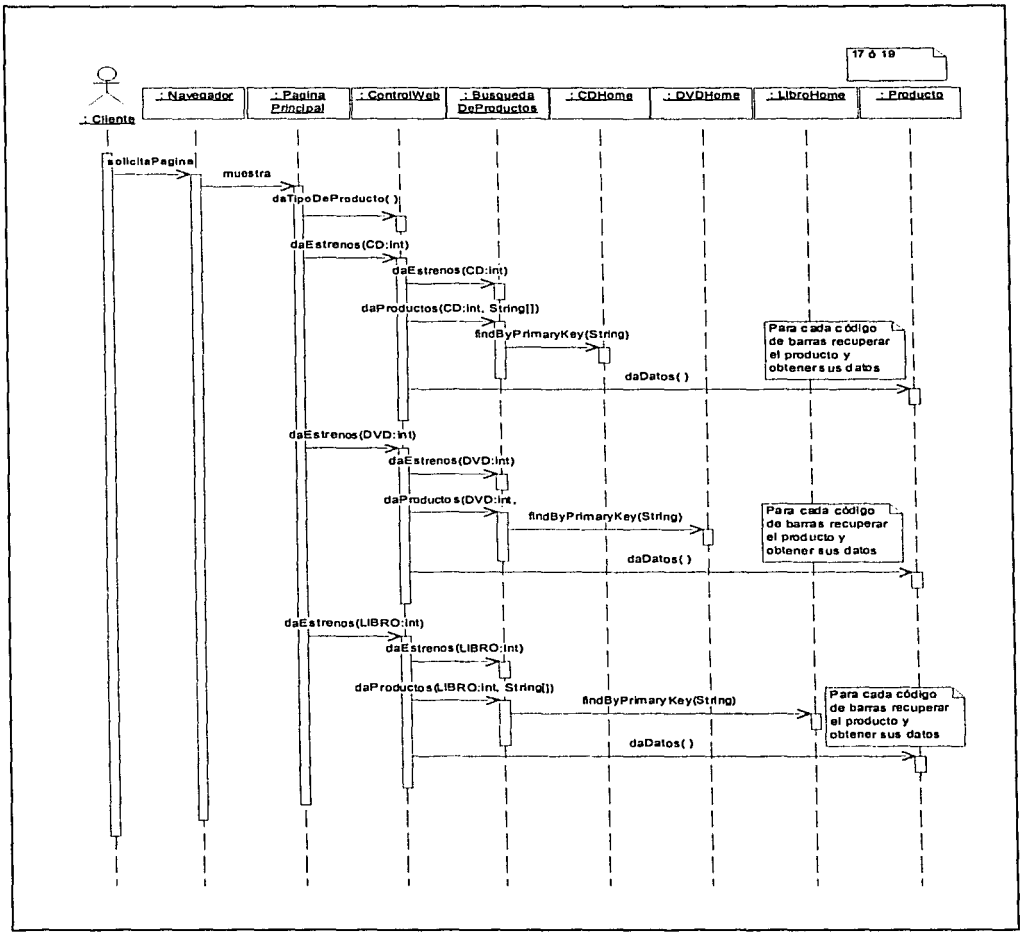


Figura A.21 Diagrama de secuencia para los casos de uso 17.Entrar a la página principal de la tienda o 19.Seleccionar home.

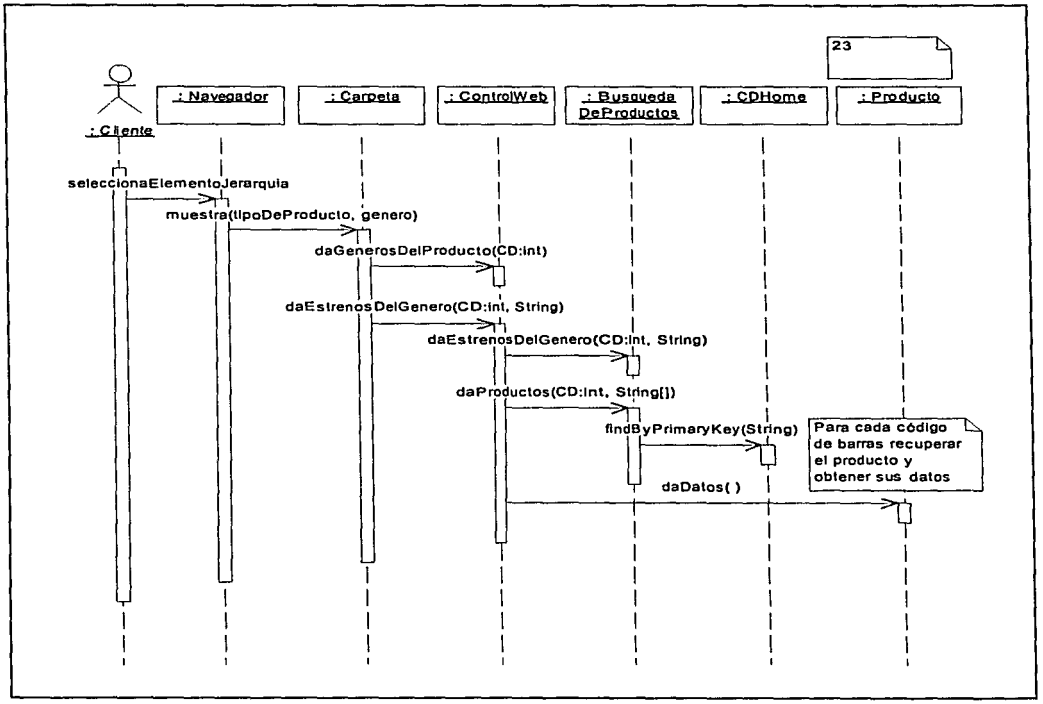


Figura A.22 Diagrama de secuencia para el caso de uso 23.Navegar la jerarquía de productos. El diagrama incluye la ejecución del caso de uso 22.Muestra estrenos.

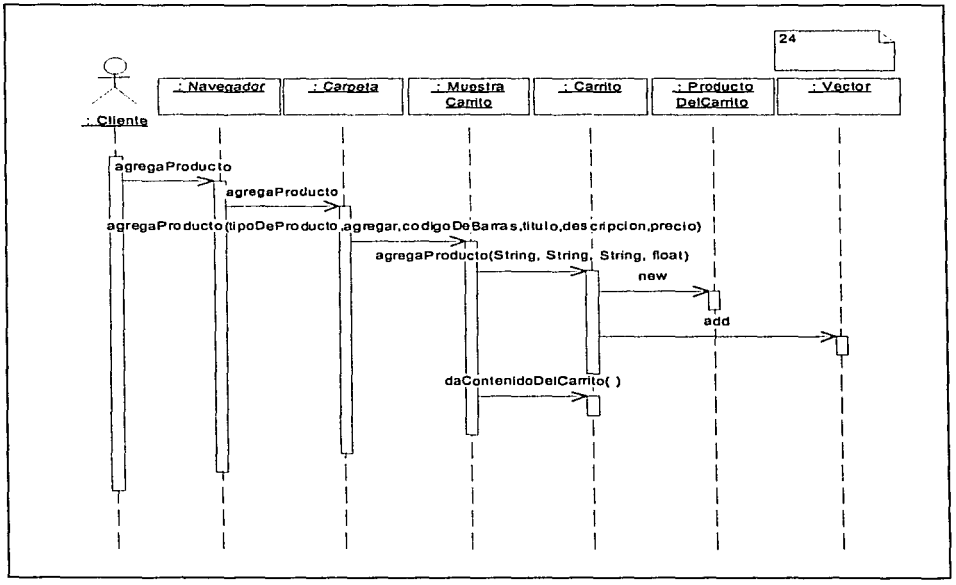


Figura A.23 Diagrama de secuencia para el caso de uso 24.Agregar producto al carrito. El diagrama incluye la ejecución del caso de uso 30.Ver información del carrito.

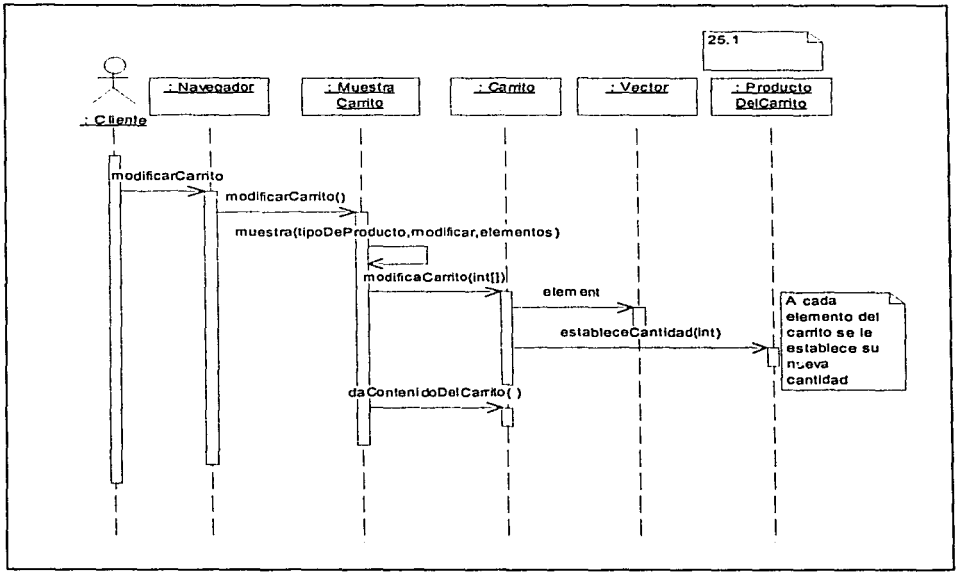


Figura A.24 Diagrama de secuencia para el caso de uso 25.Revisar el carrito, flujo alternativo 25.1.

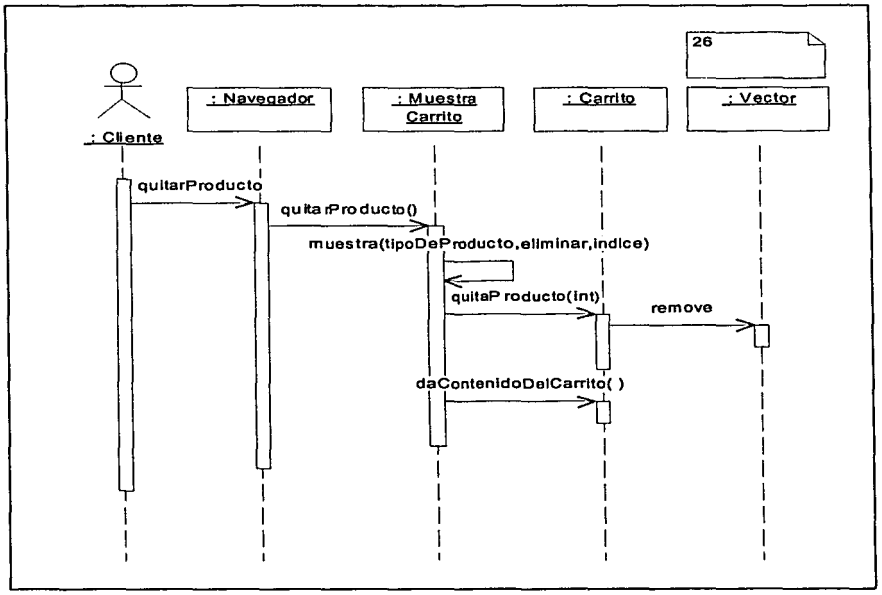


Figura A.25 Diagrama de secuencia para el caso de uso 26.Eliminar producto del carrito.

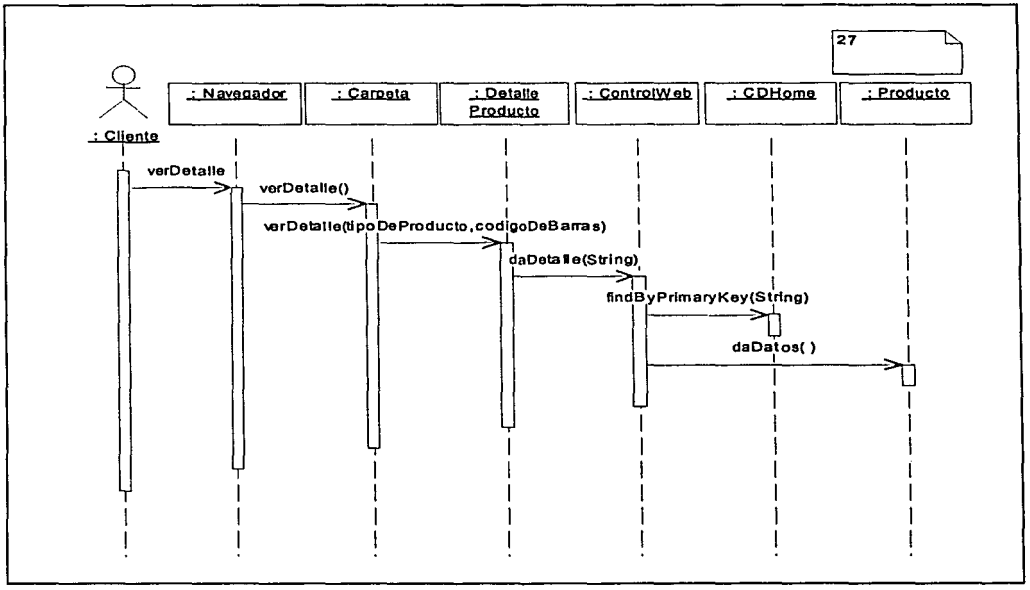


Figura A.26 Diagrama de secuencia para el caso de uso 27.Ver información detallada de un producto.

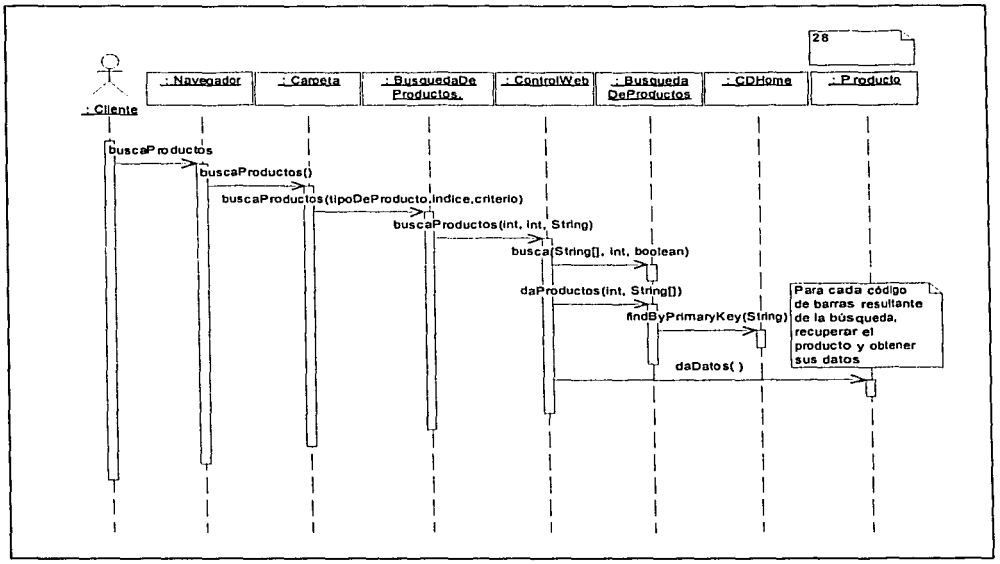


Figura A.27 Diagrama de secuencia para el caso de uso 28.Buscar producto(s). El diagrama incluye la ejecución del caso de uso 29.Ver información general de un producto.

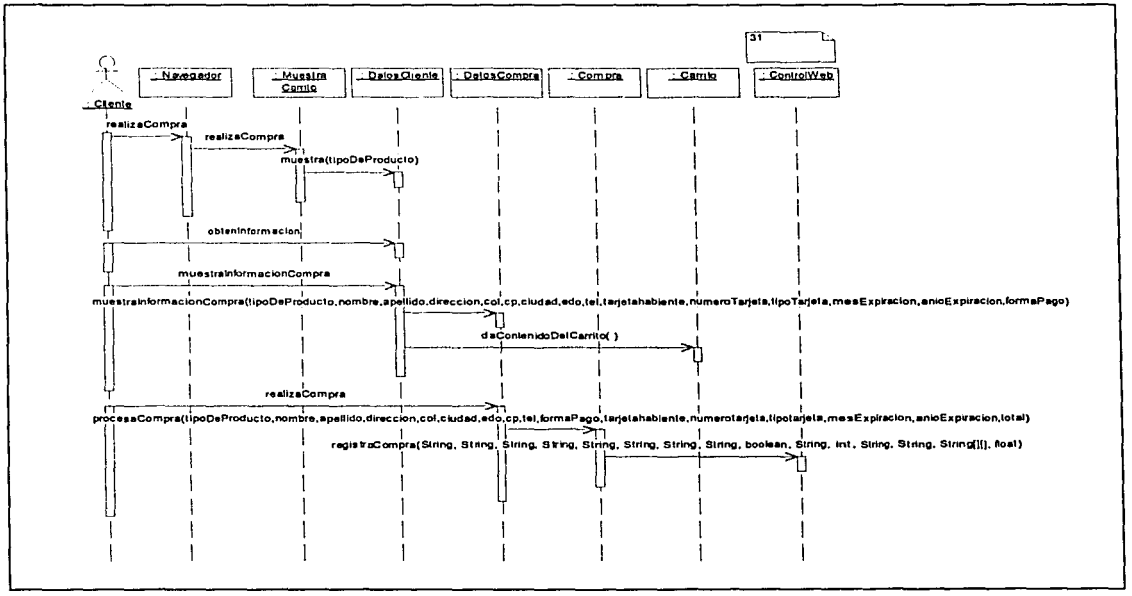


Figura A.28 Diagrama de secuencia para el caso de uso 31.Pasar a caja. El diagrama incluye la ejecución del caso de uso 30.Ver información del carrito.

A.3 Diagramas complementarios para el modelo de implementación

4.4.1.3 Diagramas de componentes para la dependencia de compilación entre componentes de distintos paquetes.

Administración

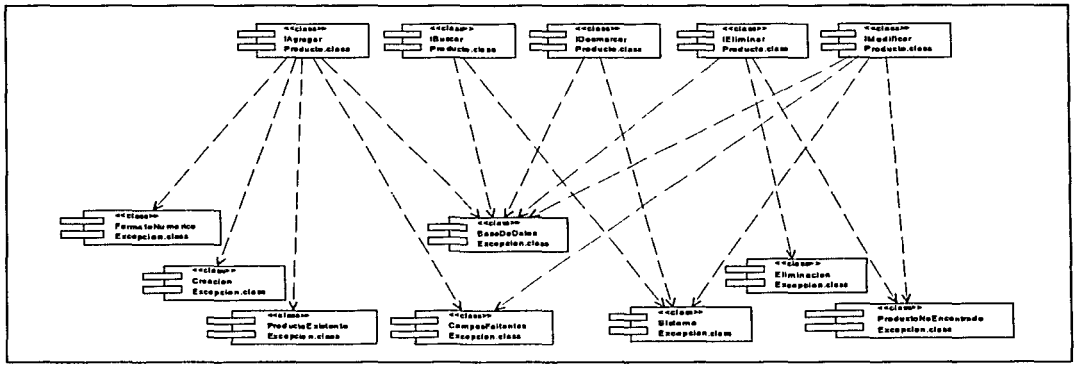


Figura A.29 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes del paquete interfazgraficadeadministrador y las excepciones definidas en el paquete controlador.

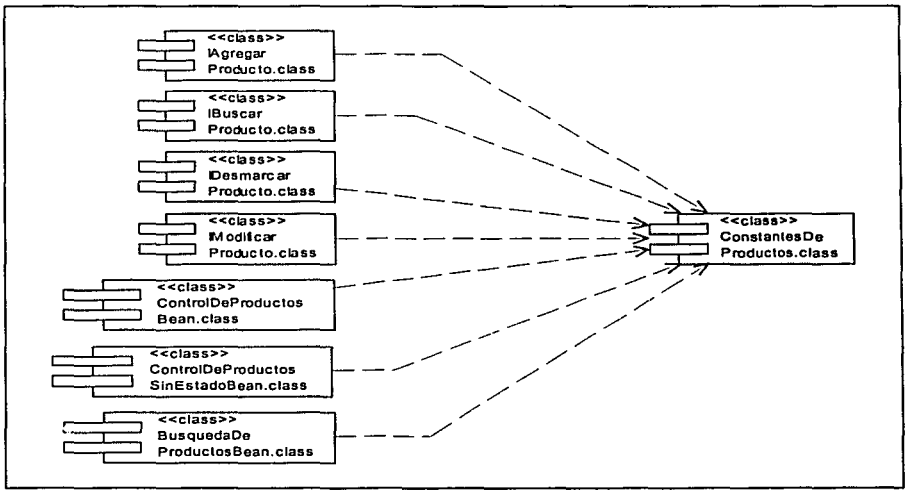


Figura A.30 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes de los paquetes interfazgraficadeadministrador y controlador con utileria.

Sitio Web

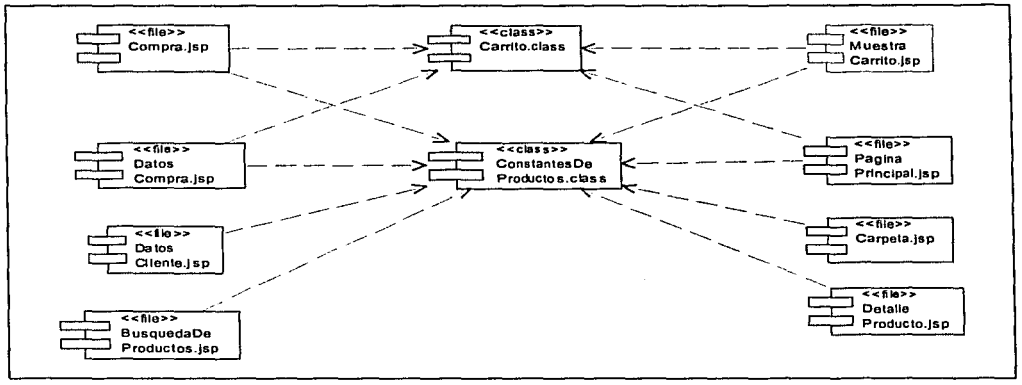


Figura A.31 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes de los paquetes interfazgraficaparainternet y utileria.

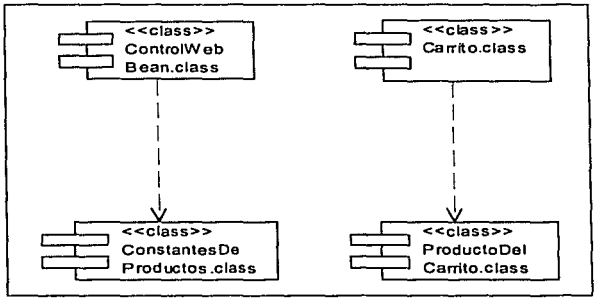


Figura A.32 Diagrama de componentes para enfatizar las dependencias de compilación entre los componentes de los paquetes web y utileria.

Bibliografía

[1] Anish Bhimani. *Securing the Commercial Internet*, Communications of the ACM, Junio 1996, Vol. 39, No. 6, p.29.

[2] Raymond Pyle. *Electronic Commerce and the Internet*, Communications of the ACM, Junio 1996, Vol. 39, No. 6, p.23.

[3] Gerhard Schmied. *High Quality Messaging and Electronic Commerce: technical fundations, standards, and protocols*, Springer-Verlag, 1999.

[4] Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Modeling Language. User Guide*, Addison-Wesley, USA, tercera edición, 1999. ISBN 0-201-57168-4.

[5] Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Software Development Process*, Addison-Wesley, USA, primera edición, 1999. ISBN 0-201-57169-2.

[6] Griselda González Rodríguez. *Tecnología J2EE aplicada a un Sistema de Comercio Electrónico*, Tesis de Licenciatura en Ciencias de la Computación, Facultad de Ciencias, Universidad Nacional Autónoma de México, 2002.

[7] Marty Hall. *Core Servlets & JavaServer Pages*, Prentice Hall, USA, primera edición, 2000.

[8] Simon Brown, Robert Burdick, Jayson Falkner. *Profesional JSP*, Wrox Press Inc., USA, segunda edición, 2001.

[9] http://javaboutique.internet.com, consultada en Diciembre de 2000.

[10] http://ute.edu.ec/~mjativa/ce/main.html, consultada en Diciembre de 2000.

[11] http://www.anu.edu.au/people/Roger.Clarke/EC/, consultada en Diciembre de 2000.

[12] http://www.theserverside.com, consultada en Diciembre de 2000.

[13] *What is E-commerce?*, http://www.careergun.com/WHAT.htm, consultada en Noviembre de 2000.