ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Implementación de un Website de Comercio Electrónico, utilizando una Infraestructura de Red Segura"

TÓPICO DE GRADUACIÓN

Previa la obtención del Titulo de:

INGENIERO EN COMPUTACIÓN

Presentada por:

Maria Auxiliadora Abarca Anormaliza
Fernando Costa Neumane
Douglas Bustos Mero

GUAYAQUIL – ECUADOR

AÑO

2006

A G R A D E C I M I E N T O

Este trabajo no pudo ser llevado a cabo sin el apoyo incondicional de nuestras familias y amigos.

Agradecemos también a todas las personas que de una u otra manera nos ayudaron a realizar este trabajo y nos transmitieron sus conocimientos durante nuestra carrera universitaria en especial a la Ingeniera Karina Astudillo.

D E D I C A T O R I A

A NUESTROS

PADRES

A NUESTRAS

ESPOSAS

TRIBUNAL DE GRADUACIÓN

Ing. Miguel Yapur

DECANO DE LA FIEC

Ing. Karina Astudillo

PROFESORA DEL TÓPICO

Ing. Lorena Carlo

VOCAL

Ing. Albert Espinal

VOCAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponden exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL).

María Auxiliadora Abarca

Fernando Costa Neumane

Douglas Bustos

RESUMEN

El presente trabajo está formado por el desarrollo de dos proyectos juntos. El primero consiste en el diseño e implementación de una infraestructura de red segura que servirá para el almacenamiento o hosting de un sitio web de comercio electrónico.

El segundo proyecto consiste en el diseño e implementación de un Mall virtual para venta de regalos *on-line,* con servicio de entrega en el Ecuador y aceptación de métodos de pago electrónicos.

En los primeros capítulos de este documento se plantean los objetivos generales y específicos del proyecto, la justificación del trabajo realizado y el análisis del negocio revisando sus ventajas y desventajas.

En los capítulos posteriores se procedió a diseñar e implementar la infraestructura de red y el sitio web de venta electrónica en base al análisis realizado en la primera parte.

ÍNDICE GENERAL

Pág.

RESUMEN VI

ÍNDICE GENERAL VII

ÍNDICE DE FIGURAS X

ÍNDICE DE TABLAS XI

INTRODUCCIÓN 1

1 ANTECEDENTES Y JUSTIFICACIÓN. 2
1.1 Antecedentes generales del proyecto 3
1.2 Definición del Problema y Objetivos del Proyecto 3
1.3 Justificación del Trabajo que se realizará como parte del Proyecto 5
2 PLAN DE MARKETING DE UN SITIO DE COMERCIO PARA LA VENTA DE REGALOS. 7
2.1 Descripción de los servicios de la empresa 8
2.1.1 Servicios especiales del sitio 9
2.2 Necesidad del Mercado 11
2.2.1 Desarrollo de una encuesta de mercado 12
2.2.2 Análisis de resultados de la encuesta 12
2.3 Análisis de la Competencia 14
2.3.1 Competencia actual del negocio 14
2.3.2 Ventajas y desventajas de la Competencia 15
2.4 Segmento de Mercado 21
2.5 Estrategia de Mercado 21
2.5.1 Orden de factores estratégicos del mercadeo 22
2.5.2 Precio adecuado de los productos ofrecidos en el sitio 23
3 DISEÑO DE UNA INFRAESTRUCTURA DE RED SEGURA PARA EL ALMACENAMIENTO DE UN SITIO DE COMERCIO ELECTRÓNICO. 25
3.1 Selección de Mecanismos de Seguridad 26
3.2 Selección de Herramientas de Seguridad 28
3.2.1 Microsoft Internet Security and Acceleration Server 30
3.2.2 Computer Associates E-Trust Intrusion Detection 35
3.2.3 Symantec Antivirus Corporate Edition 45
3.2.4 Sophos Antivirus MailMonitor. 47
3.2.5 Servicio de Directorio de Microsoft (Active Directory). 48

3.2.6 Servidor SMTP - Sendmail. 50
3.3 Definición de Políticas de Seguridad 52
3.3.1 Políticas de Tráfico en la Red 52
3.3.2 Políticas de Detección de Intrusos 57
3.3.3 Políticas de Acceso de Usuarios Internos 60
3.3.4 Políticas de Acceso al SMTP 61
3.3.5 Políticas de Servicio de Antivirus 62
3.3.6 Políticas de Seguridad de Base de Datos 63
3.3.7 Políticas de Seguridad de Servidor WEB 66
3.4 Diseño de la Contingencia de la Red 69
3.5 Diagrama de una Red Segura 70
3.5.1 Diagrama de Red Ideal 74
3.5.2 Diagrama de Red Implementada en el Proyecto 76
4 IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA DE RED SEGURA PARA EL ALMACENAMIENTO DE UN SITIO DE COMERCIO ELECTRÓNICO. 78
4.1 Implementación de la Estructura de Dominio de Red 79
4.1.1 Instalación de Servidores Windows 2000 y Windows 2003 79
4.1.2 Construcción del dominio Active Directory 82
4.2 Implementación de Políticas de Dominio 84
4.3 Implementación de Firewall Externo 85
4.3.1 Instalación de ISA Server como Firewall Externo 86
4.3.2 Configuración de Políticas en el Firewall Externo 90
4.4 Implementación de Firewall Interno 94
4.5 Implementación de Detección de Intrusos 96
4.5.1 Instalación y Configuración de eTrust Intrusion Detection 96
4.6 Implementación del Servicio de Antivirus para Estaciones y Servidores. 98
4.7 Implementación del Servicio de Antivirus para Correo Electrónico 100
4.8 Implementación del Servidor de Base de Datos 102
4.9 Implementación de Seguridades de la Base de Datos 103
4.10 Implementación de Seguridades del Servidor WEB 105
5 DISEÑO E IMPLEMENTACIÓN DE UN SITIO DE COMERCIO ELECTRÓNICO 109
5.1 Definición de los requerimientos del sitio de comercio electrónico. 110
5.1.1 Objetivos de la construcción del sitio de e-commerce. 111
5.1.2 Servicios que el sitio debe brindar. 111
5.1.3 Seguridad que el sitio debe ofrecer durante la compra. 113
5.1.4 Interface de usuario vendedora. 113
5.2 Desarrollo de una estrategia para la creación del sitio 114
5.2.1 Análisis de los Requerimientos 115
5.2.2 Diseño General del Sistema 118
5.2.3 Implementación 122
5.3 Selección de la Plataforma de desarrollo y ejecución del aplicativo 133
5.3.1 .NET Framework 133

5.3.2 ASP.NET 134
5.3.3 C# 135
5.3.4 ADO.NET 135
5.4 Modelo de Datos usado 136
5.4.1 Entidades de la Base de Datos 136
6 CONCLUSIONES Y RECOMENDACIONES 140

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE FIGURAS

Pág.

Figura 3.1	Políticas basadas en Paquetes	59
Figura 3.2	Políticas basadas en Estadísticas	60
Figura 3.3	Diagrama de Red Segura	72
Figura 3.4	Diagrama de Red Ideal	75
Figura 3.5	Diagrama de Red Implementada	77
Figura 4.1	Herramienta Domain Security Policy	85
Figura 4.2	Instalación de Isa Server	87
Figura 5.1	Diagrama de la Arquitectura del Sitio Web	122
Figura 5.2	Proceso de Venta de Regalos	124
Figura 5.3	CU1 Búsqueda de Regalos	125
Figura 5.4	CU2 Asistente de Selección de Regalos	126
Figura 5.6	CU4 Organizador Regalos en Grupo	128
Figura 5.7	CU5 Organizador Regalos en Grupo	129
Figura 5.8	CU6 Creación y Actualización Datos Usuarios	130
Figura 5.9	CU7 Login al Sistema	131

ÍNDICE DE TABLAS

Tabla I Herramientas de Seguridad Seleccionadas……. 29

INTRODUCCIÓN

El presente trabajo trata sobre el diseño y la implementación de un sitio de comercio electrónico utilizando una infraestructura propia de red segura con la finalidad de crear un negocio de venta de productos a través de la Internet y enfocado hacia el mercado Ecuatoriano.

Debido al tipo de negocio que hemos planteado en este proyecto hemos analizado todos los aspectos de seguridad necesarios para salvaguardar la información, siendo necesario diseñar una red segura que almacene el sitio web de compras y brinde el acceso a los clientes de la empresa.

Adicionalmente al diseño e implementación de la infraestructura de red, este trabajo comprende también la creación del sitio Web de comercio electrónico y se desarrollan las etapas de análisis, diseño y programación de las páginas Web que forman el sitio.

Capítulo 1

1 ANTECEDENTES Y JUSTIFICACIÓN.

El propósito de este proyecto es implementar un sitio de comercio electrónico de venta de regalos. La idea de crear un negocio virtual nace de la iniciativa de emprender una actividad comercial utilizando la tecnología informática y aprovechar los nuevos métodos de hacer negocio, que la Internet ofrece hoy en día.

Para desarrollar esta idea y convertir el proyecto en una empresa, hemos debido realizar un trabajo de investigación que nos permita contar con las herramientas necesarias para tener éxito en un negocio de comercio electrónico, tales como seguridad de la información, mercadeo de servicios en la Web y la creación de un diseño atractivo y amigable que facilite la venta de productos a través de Internet.

En este capítulo se describirán los antecedentes del proyecto y la justificación para su desarrollo.

1.1 Antecedentes generales del proyecto

Los avances actuales de la informática y la difusión global de la Internet han cambiado la manera en que se desarrollan las actividades de la sociedad en los ámbitos de la comunicación, la calidad de vida y el comercio. Hoy en día la Internet ofrece nuevas alternativas de negocio ya que esta nos permite llegar a una audiencia masiva y a un gran número de posibles clientes; podemos ofrecer nuestros servicios a un mercado mucho mayor ya que el tiempo y la distancia dejan de ser obstáculos.

Basados en este concepto hemos decidido crear una empresa que ofrezca productos a través de la Internet, cuyas operaciones estarán enfocadas al mercado ecuatoriano.

Siendo el mercado ecuatoriano relativamente novato en las transacciones electrónicas y en la compra de productos y servicios a través de la Internet, es necesario ofrecer un alto nivel de seguridad para ganar la confianza de los clientes potenciales.

1.2 Definición del Problema y Objetivos del Proyecto

La utilización de la Internet como medio de compra y venta de productos nos obliga a analizar de manera distinta muchos de los aspectos del comercio tradicional ya que el hecho de comercializar de una manera impersonal y a través de un computador genera algunos problemas de tipo social, cultural y económico.

El problema principal en el desarrollo de nuestro proyecto es vencer las barreras sociales, culturales y económicas que un negocio de comercio electrónico enfrenta; proveyendo al usuario la confianza, las facilidades e incentivos que este busca al momento de realizar una compra.

El objetivo principal de este proyecto es crear una empresa líder en el mercado de comercio electrónico ecuatoriano. Partiendo de este concepto surgen las siguientes metas para el proyecto:

- Proveer un medio transaccional seguro que le de la confianza al cliente en el momento de hacer sus compras. Para lograr esto debemos considerar no sólo la seguridad de las páginas del sitio sino también la infraestructura de red que utilizaremos para dar acceso a nuestro portal de compras.
- Utilizar una estrategia de mercadeo efectiva que acapare la atención de los clientes potenciales y que asegure la afluencia de compradores hacia el portal.
- Diseñar una interface amigable y atractiva del sitio de compras que facilite la navegación de los clientes y venta de los

productos y que además incite al usuario a regresar en un futuro cercano.

1.3 Justificación del Trabajo que se realizará como parte del Proyecto

Una vez planteados los objetivos y las metas de nuestro proyecto se requerirá un trabajo de investigación y desarrollo que haga posible el cumplimiento de estas metas.

La primera parte de este proyecto corresponde al análisis del negocio de venta de regalos a través de un sitio de comercio electrónico. Para formular una estrategia de marketing adecuada es primordial entender primero las necesidades del mercado, estudiar las acciones de los competidores y enfocar correctamente el segmento del mercado hacia el cual estará dirigida nuestra estrategia.

La segunda parte de este proyecto deberá incluir el diseño e implementación de la red de computadoras que se utilizará para hacer posible el acceso de nuestros clientes al portal de compra de regalos.

Lo más importante en esta etapa es alcanzar un alto nivel de seguridad aplicando los principios fundamentales de Seguridad de la Información. Otro aspecto importante en esta etapa es utilizar una infraestructura de red propia en la cual el nivel de seguridad esté controlado por nosotros y que la confiabilidad de nuestras transacciones no dependan de un tercero.

La última parte del proyecto corresponde al diseño e implementación del portal de venta de regalos. De nada nos servirá elaborar una estrategia de mercado acertada e implementar un alto nivel de seguridad en las transacciones si el cliente no se siente atraído a nuestro portal. Esta parte del proyecto consistirá en la creación de una interface amigable que atraiga a posibles compradores, facilite la búsqueda de un producto y la posterior compra del mismo.

Capítulo 2

2 PLAN DE MARKETING DE UN SITIO DE COMERCIO PARA LA VENTA DE REGALOS.

El objetivo fundamental de este proyecto es crear una nueva empresa orientada al negocio de compras en línea, proyectado al mercado ecuatoriano. Antes de desarrollar un proyecto de empresa, es necesario realizar un análisis del mercado y crear un plan de marketing que demuestre la factibilidad de la idea y su probabilidad de éxito en nuestro entorno comercial.

La misión de la empresa es brindar un servicio innovador que permita a los usuarios de Internet del Ecuador realizar compras de productos específicos sin necesidad de moverse de su casa, contando con un ambiente amigable y creativo para encontrar un regalo.

En este capítulo se desarrolla el plan de marketing de nuestra empresa, el cual pretende evaluar las condiciones actuales del mercado de comercio electrónico y las estrategias que se aplicarán para posicionar el sitio en el segmento de compradores en línea ecuatorianos.

El nombre comercial seleccionado para nuestro negocio es "REGALOS.COM".

2.1 Descripción de los servicios de la empresa

El sitio de compras www.regalos.com pretende ofrecer un servicio con el cual los clientes puedan encontrar artículos para regalar, realizar la compras en línea y recibir el producto en su casa o enviarlo directamente a la persona a quien se quiere regalar.

El sitio estará diseñado para ofrecer regalos en las categorías de: Bautizos, Cumpleaños, Aniversarios, Día de la Madre, Día del Padre, Navidad, Primera Comunión, Quince años, Graduación y Matrimonios.

El cliente podrá encontrar regalos para las siguientes ocasiones:

- Matrimonios (Línea Blanca, Cristalería, Vajillas, Cubiertos, Artículos de Cocina).
- Aniversarios (Arreglos Florales, Chocolates, Peluches).
- Día de la Madre (Agendas, Adornos variados, Lámparas, Perfumes).
- Día del Padre (Libros, Habanos, Licores, Corbatas).
- Cumpleaños (Reloj, CDs, DVDs, Artículos Electrónicos, Billeteras, Bolígrafos, Cinturones).

- Navidad, Primera Comunión, Quince años, Graduación y Bautizos

El usuario de nuestro sitio podrá buscar un regalo de acuerdo a las categorías o por un nombre de producto específico, de tal manera que el ambiente sea amigable y fácil de utilizar. El cliente podrá planificar su entrega en una fecha específica en la que desea que se entregue el regalo y en el lugar que desee.

2.1.1 Servicios especiales del sitio

La aceptación y el éxito de un sitio de comercio electrónico dependen mucho de la manera de atraer al usuario con servicios innovadores que faciliten la navegación y asesoren al usuario a tomar una decisión de compra. En la actualidad existen muchos sitios de comercio electrónico funcionando y por esto es necesario crear una ventaja sobre nuestra competencia con servicios que mejoren la experiencia al navegar y comprar para de esta manera satisfacer al 100% la necesidad de atención del usuario.

Los servicios especiales que el sitio brindará son los siguientes:

Organización de regalos en conjunto

Uno de los servicios que se ofrecerá en el sitio será la capacidad de crear regalos en conjunto de tal manera que una vez seleccionado el regalo varias personas puedan ingresar y realizar un abono a dicho regalo.

Recordatorio de cumpleaños, aniversarios y fechas especiales.

Los usuarios del sitio podrán suscribirse a un servicio en el cual podrán ingresar las fechas importantes que desean recordar como cumpleaños o aniversarios de familiares y amigos.

Asistente para la selección de un regalo

El sitio contará con un programa asistente (Wizard) que ayudará a los usuarios a seleccionar el regalo mas apropiado. El asistente realizará una búsqueda basado en los parámetros de la persona a quien se quiere regalar como edad, sexo, motivo del regalo y presupuesto.

Servicio de envoltura de regalos

El servicio de envoltura de regalos estará disponible para todos los artículos comprados en el sitio. Los usuarios podrán seleccionar entre una variedad de papel de regalo que vaya acorde a quien se quiere regalar y a la ocasión.

Beneficios para usuarios frecuentes

Usuarios frecuentes del sitio obtendrán beneficios adicionales como descuentos, envíos sin costo, obsequios especiales, etc.

2.2 Necesidad del Mercado

La idea de este sitio surgió de la necesidad de encontrar un regalo apropiado para diferentes ocasiones. Muchas veces las personas se topan con el problema de tener que regalar algo y se ven en la necesidad de salir a recorrer locales comerciales buscando un producto en especial que cumpla con el perfil de la ocasión y de la persona a quien se va a regalar. Existe la necesidad de un sitio Web que agrupe las distintas opciones de artículos a regalar.

Otra necesidad a la que apunta este proyecto es organizar regalos en conjunto entre varias personas. Puede darse el caso de que se quiera regalar un producto cuyo precio se desea compartir entre varias personas y en lugar de reunirlos a todos para que aporten con una cuota, el organizador de la compra podrá indicar a las demás personas que pueden entrar al sitio, ver el regalo seleccionado y abonar la cantidad requerida.

Adicionalmente es primordial crear una necesidad en el mercado, debido a que son pocas las personas en el Ecuador que ya realizan transacciones en línea y que se animan a comprar un producto directamente desde la Internet. Es necesario realizar una promoción adecuada que convenza al mercado de que tienen un medio alternativo de buscar sus regalos y comprarlos.

2.2.1 Desarrollo de una encuesta de mercado

Como parte del análisis de mercado hemos realizado una encuesta a un grupo de 30 individuos comprendidos entre edades de 25 a 40 años.

El cuestionario utilizado para la encuesta y la tabulación de los resultados de la misma se encuentran detallados en el Apéndice A.

2.2.2 Análisis de resultados de la encuesta

Los resultados obtenidos durante la encuesta mostraron datos interesantes sobre la posición del mercado ante el advenimiento de sitios de comercio electrónico; se ha podido percibir que a pesar de que la mayoría de los encuestados nunca han realizado compras en la Web estarían interesados en utilizar este mecanismo para realizar compras.

Menos de la mitad de los encuestados (43%) afirmaron haber comprado algún articulo en la Internet.

El 70% de los encuestados afirmo que utilizaría un servicio de venta de regalos en línea para adquirir obsequios de distintas ocasiones, siendo las mas preferidas Cumpleaños (50%), Navidad (33%) y Aniversarios (20%).

Con respecto a la pregunta de cual es el factor más importante al momento de comprar en línea el 40% de los encuestados considera la seguridad como el aspecto primordial a la hora de realizar una transacción en un sitio de e-commerce.

Adicionalmente la mayoría de los encuestados considera que los mayores beneficios de comprar regalos por Internet son la comodidad y la facilidad de envío / entrega de los obsequios.

Luego de haber analizado los resultados de la encuesta realizada podemos llegar a la conclusión de que nuestro negocio tiene buenas posibilidades de éxito al aprovechar una necesidad real del mercado.

2.3 Análisis de la Competencia

El servicio que se ofrece en regalos.com no es único en su género; debido a que en la gran red del Internet existen muchos sitios Web con orientación similar. Es necesario realizar un análisis de nuestros competidores, identificar sus debilidades y aprovecharlas como parte de nuestra estrategia de mercado.

2.3.1 Competencia actual del negocio

Pocas son las empresas que como nosotros se han planteado una idea de negocio de comercio electrónico, sin embargo cada día surgen nuevos sitios que representan una posible amenaza de competencia. A continuación mencionamos algunos sitios que actualmente ya están realizando ventas a través de sus páginas Web y que por ofrecer productos similares a los nuestros se consideran como competidores:

- Bragança
- GiftCompany
- Créditos Económicos

- Los Colores de mi Tierra

Los sitios analizados no cubren todo el mercado objetivo, pues nuestra presentación y clasificación es diferente al tratarse de artículos definidos por categorías de celebración en vez de clasificarlos por tipos de artículos como lo hace la competencia.

2.3.2 Ventajas y desventajas de la Competencia

Las diferencias de competencia más relevantes que hemos encontrado son: precios, opciones de facilidad al usuario y ofertas.

Precios: Al comparar este punto con la competencia encontramos que los nuestros serán más atractivos, y en aquellos que tengan un poco de variación elevada se deberá al constante cambio que sufre el mercado y a la calidad de nuestros productos.

Opciones de facilidad al usuario: Una de las facilidades para el usuario es encontrar los artículos clasificados por celebración, ya que se puede orientar directamente a su necesidad y el sitio también incluye sugerencias específicas a este punto.

Ofertas: Nuestras ofertas no solamente estarán dirigidas a ocasiones especiales sino que en cada categoría habrán ciertos artículos en promoción.

A continuación detallamos las ventajas y desventajas que hemos encontrado en las empresas consideradas como competencia:

braganca.com.ec

Ventajas:

✓ Reconocimiento de la marca. Bragança es un sitio de compras especializado en flores y chocolates con algunos años en el mercado, lo cual ha logrado que una buena parte de nuestros clientes objetivos lo conozcan.

✓ Imagen corporativa del sitio. El diseño de las páginas del sitio es atractivo y de fácil navegación ya que han aplicado las técnicas de diseño adecuadas como resolución de pantalla apropiada, páginas livianas, distribución de los elementos y selección de colores.

✓ Teléfono 1700. Bragança ofrece una alternativa de compra a las personas que visitan su página con un número telefónico 1700 a nivel nacional. Los usuarios interesados en alguno de los artículos que se exhiben, pueden realizar la compra por teléfono si no desean realizar la transacción en línea.

Desventajas:

✗ Precios Elevados. La principal desventaja de bragança son los elevados precios de sus productos.

✗ Poca variedad de productos. Otra desventaja de este sitio es la poca variedad de artículos de venta. Si bien es cierto, se han concentrado en ofrecer una pequeña categoría de productos, en comparación a nuestro negocio tienen poca variedad.

Gift Company (todo1plaza.com)

Ventajas:

✓ Publicidad de Todo1Plaza. El hecho de estar asociado al portal Todo1Plaza es una ventaja ya que este portal puede atraer tráfico hacia su página captando potenciales clientes.

Desventajas:

- ✗ Precios Elevados. La principal desventaja de Gift Company son los elevados precios de sus productos.
- ✗ Imagen corporativa del sitio. El diseño de la página es poco atractivo en cuanto a la selección de los colores. Las fotos de los productos mostrados son de pésima calidad y no llaman la atención de los compradores.
- ✗ Competencia en Todo1Plaza. El hecho de pertenecer a este conocido portal, no sólo se considera una ventaja sino también una desventaja debido a que la competencia del sitio se encuentra en el mismo portal y un cliente potencial podría ser atraído hacia estas páginas.
- ✗ Limitación del Sitio de Todo1Plaza. Los servicios ofrecidos por Todo1Plaza son muy inflexibles y no permiten que la página incluya opciones nuevas o servicios diferenciadores de la competencia.

Créditos Económicos

Ventajas:

- ✓ Reconocimiento de la empresa. Créditos Económicos es una empresa con muchos años en el mercado y una de las más conocidas en el Ecuador. Los usuarios que compran en esta página se sienten seguros de que es una empresa conocida y de gran respaldo.

✓ Imagen corporativa del sitio. El diseño de las páginas del sitio es atractivo y de fácil navegación. A pesar de mostrar una gran cantidad y variedad de productos, la estructura del sitio no permite que el usuario se confunda, sino que al contrario encuentre rápidamente el producto que desea.

✓ Infraestructura propia para entregas. Siendo una empresa comercial con muchos años en el Mercado, cuenta con su propia infraestructura de entregas que le permite reducir los costos finales para el usuario y asegurar la entrega sin depender de un tercero.

✓ Ofertas en el sitio. Créditos Económicos ha creado ofertas dirigidas a los usuarios que adquieran productos a través de su página Web. Estas ofertas no están disponibles en sus almacenes lo cual incentiva al usuario a realizar sus compras a través de la página Web.

Desventajas:

✗ Variedad en nuestro mercado objetivo. Debido a la orientación que le hemos dado a nuestro sitio y al mercado objetivo al que nos estamos enfocando, el hecho de que Créditos Económicos solo venda electrodomésticos llega a ser una desventaja al momento de comprar un regalo.

Los Colores de mi tierra

Ventajas:

✓ Publicidad de Todo1Plaza. El hecho de estar asociado al portal Todo1Plaza es una ventaja ya que este portal puede atraer tráfico hacia su página captando potenciales clientes.

Desventajas:

× Precios Elevados. La principal desventaja de este sitio son los elevados precios de sus productos.

× Imagen corporativa del sitio. El diseño de la página es poco atractivo en cuanto a la selección de los colores.

× Poca variedad de productos. Este sitio ofrece una muy limitada cantidad de artículos.

× Nombre del sitio. "Los colores de mi tierra" es un nombre que no llama la atención y al mismo tiempo no se asocia a la actividad de la página Web.

× Competencia en Todo1Plaza. El hecho de pertenecer a este conocido portal, no solo se considera una ventaja sino también una desventaja debido a que la competencia del sitio se encuentra en el mismo portal y un cliente potencial podría ser atraído hacia estas páginas.

- Limitación del Sitio de Todo1Plaza. Los servicios ofrecidos por Todo1Plaza son muy inflexibles y no permiten que la página incluya opciones nuevas o servicios diferenciadores de la competencia.

2.4 Segmento de Mercado

El segmento al que está dirigido nuestro negocio son hombres y mujeres entre los 18 y los 50 años, que posean la capacidad económica para adquirir los productos ofrecidos en el sitio y que sientan la necesidad de un lugar que les facilite la compra de regalos.

El segmento de mercado de nuestro negocio son los compradores tradicionales de bienes de Guayaquil y Quito.

2.5 Estrategia de Mercado

La principal estrategia de mercado que utilizaremos es la publicidad en modo de correo directo, banners en sitios Web locales de alto tráfico, ferias de computación y en las estaciones de radio. Utilizando estos medios nos daremos a conocer como un nuevo servicio en el mercado.

La publicidad en correo directo se realizará contratando a empresas de tarjetas de crédito para hacer llegar información del sitio, al segmento de mercado al que estamos apuntando. A través de las bases de datos de clientes de Pacificard, Diners Club, Bankard, entre otras, haremos llegar publicidad de nuestra empresa junto con los estados de cuenta de las tarjetas.

Otro método de hacernos conocer es contratar banners publicitarios en sitios Web locales de alto tráfico como portales de proveedores de Internet, diarios, etc.

Las ferias de computación permitirán promocionar nuestro sitio como una nueva alternativa entre los asistentes a las ferias y como último mecanismo utilizaremos publicidad radial que llegan en menor proporción a nuestro segmento del mercado.

2.5.1 Orden de factores estratégicos del mercadeo

La estrategia de mercadeo del sitio debe considerar distintos factores de los cuales depende la aceptación y éxito del sitio. Es necesario para elaborar una estrategia de mercado considerar la importancia de cada factor y asignarle la prioridad correcta en nuestro negocio.

A continuación se detalla en orden de mayor a menor la importancia de los distintos factores de mercado:

1. Costo
2. Creatividad
3. Tecnología
4. Calidad
5. Tiempo
6. Flexibilidad

2.5.2 Precio adecuado de los productos ofrecidos en el sitio

El precio de nuestros productos es uno de los factores más importantes para el éxito del sitio Web. La mayoría de personas que realizan compras por Internet, aprovechan esta tecnología para comparar los precios de los productos en distintos sitios Web similares. El precio que se coloque a los productos debe de ser muy cercano a los precios del mercado tradicional de manera que el usuario se decida por realizar sus compras en nuestro sitio Web y no prefiera salir a la calle a buscar un regalo.

Para poder ofrecer precios competitivos sobre nuestros productos, la empresa deberá buscar los proveedores adecuados y establecer las compras de tal manera que se reduzcan los costos para la empresa, esto permitirá que el sitio ofrezca precios adecuados sin sacrificar demasiado el margen de utilidad de la empresa.

La diferencia de precios entre nuestro sitio y los almacenes tradicionales no debe superar a un 10%.

Los precios de cada producto que se ofrecerá en el sitio serán analizados en el momento de buscar y contactar proveedores y se establecerán en base al margen deseado por la empresa 25 a 30 por ciento, y a los precios del mercado tradicional.

Capítulo 3

3 DISEÑO DE UNA INFRAESTRUCTURA DE RED SEGURA PARA EL ALMACENAMIENTO DE UN SITIO DE COMERCIO ELECTRÓNICO.

Una de las tareas principales en la construcción y puesta en producción de un negocio de comercio electrónico es la seguridad del sitio Web y de los datos que este va a manejar. El sitio Web interactúa con datos muy sensibles entre los que están información de clientes, productos de la empresa, transacciones realizadas e información de tarjetas de crédito; es vital para el éxito de este negocio que ese tipo de información permanezca secreta, ya que un robo de información podría ocasionar fraude electrónico y lo que es peor la pérdida de la confianza de nuestros clientes.

Con el fin de mantener el control de la seguridad de nuestro sitio Web y nuestros datos, se tomó la decisión de construir una infraestructura de red propia en la que se almacenará el sitio Web de comercio electrónico.

Esta infraestructura debe contar con los niveles de seguridad necesarios para lograr el objetivo de la seguridad de la información.

En este capítulo se explica el diseño y la implementación de una infraestructura de red segura que servirá para almacenar nuestro sitio de comercio electrónico.

Nuestro diseño de la red, debe considerar todos los riesgos de seguridad que pueden existir y por esta razón se ha diseñado una infraestructura ideal en la cual se incorporan todas las herramientas de seguridad necesarias y además se cuenta con elementos redundantes y de contingencia a fallos.

Adicionalmente a la infraestructura ideal, también se diseño una infraestructura mínima segura que se construirá como parte de este proyecto y en el cual se almacenará el sitio Web de comercio electrónico.

3.1 Selección de Mecanismos de Seguridad

Los mecanismos de seguridad que se utilizarán en el diseño y la implementación de la red segura son firewalls, sistemas de detección de intrusos, servidores Web seguros, sistemas antivirus y servicios de directorio de seguridad.

Los elementos de seguridad principal son los 2 firewalls que formarán parte de nuestra red, los firewalls tienen como función el bloqueo de tráfico de entrada hacia la red, de tal manera que solo se permita el paso de tráfico absolutamente necesario. El primer firewall protegerá a toda la red de posibles ataques desde la Internet. El segundo Firewall protegerá la subred más crítica es decir la subred de la base de datos. Este segundo Firewall es necesario para proteger la base de datos de posibles atacantes internos en la empresa.

El software de detección de intrusos es otro de los principales componentes de seguridad, este se encargará de capturar y analizar el tráfico de paquetes en la red, para detectar e impedir los ataques en caso de que tráfico malicioso penetre en la red o en caso de que se realice un ataque desde la red de administración de la empresa. Las estadísticas muestran que la mayoría de los ataques a las redes son realizados por personas con acceso físico a la red interna.

A pesar de los mecanismos de bloqueo de tráfico implementados, siempre el servidor de páginas Web estará expuesto a posibles ataques ya que hacia este servidor deberán llegar los requerimientos de usuarios en los puertos HTTP 80 y HTTPS 443. El software de servidor de páginas Web debe de tener la suficiente robustez para no procesar tráfico malicioso que penetre por dichos puertos. Adicionalmente el servidor Web debe tener características de seguridad que no otorguen permisos innecesarios a usuarios que no los requieren o que no han sido definidos como usuarios privilegiados ya que esto puede derivar en posibles ataques.

El tercer elemento de seguridad es un sistema de protección contra virus. Todas las estaciones y servidores deberán contar con un antivirus robusto que detecte y neutralice los posibles ataques de virus. La vía mas común de infección de virus es el correo electrónico, por tanto el antivirus seleccionado deberá asegurar que el correo electrónico que llegue a los usuarios este libre de virus.

El cuarto mecanismo de seguridad que debe implantarse en la red es un servicio de directorio seguro que permita asignar privilegios mínimos a los usuarios en la red, así como establecer políticas de acceso a los distintos recursos de la red. Este mecanismo ayudará a prevenir posibles ataques provenientes desde dentro de la empresa.

3.2 Selección de Herramientas de Seguridad

Para realizar el diseño de la infraestructura ideal de red y de la infraestructura mínima para el proyecto fue necesario analizar los distintos mecanismos y herramientas de seguridad disponibles en el mercado y seleccionar entre todos los fabricantes y proveedores de software el conjunto de herramientas que nos brinden el nivel de seguridad requerido y al mismo tiempo presenten una buena relación de costo beneficio. En base a este análisis hemos seleccionado algunos productos de seguridad entre todos los disponibles actualmente, los cuales se detallan en la TABLA 1.

TABLA I

HERRAMIENTAS DE SEGURIDAD SELECCIONADAS

Mecanismo de Seguridad	**Producto Seleccionado**
Firewalls	Microsoft ISA Server
Detección de Intrusos	CA E-Trust
Antivirus de Host	Symantec Antivirus Corporate
Antivirus de SMTP	Sophos MailMonitor
Servidor Web	Internet Information Server 6.0
Servidor de Base de Datos	Microsoft SQL Server
Servicio de Directorio	Microsoft Active Directory

3.2.1 Microsoft Internet Security and Acceleration Server

Durante la etapa de diseño del proyecto, seleccionamos como Firewall de nuestra red el producto Microsoft Internet Security and Acceleration Server 2000, también conocido como ISA Server. Para realizar esta selección de productos de Firewall, buscamos una herramienta que cumpla con los requisitos básicos de un sistema Firewall, que estuviera certificado por los laboratorios ICSA, que brinde la seguridad necesaria para la red y que al mismo tiempo sea fácil de administrar.

Microsoft ISA Server, es una solución de Firewall muy robusta que cumple con los estándares de los diferentes tipos de firewalls, proveyendo a nuestra red con un sistema de filtrado de paquetes (Packet Filtering – Stateful Inspection) y también es un Application Level Gateway es decir que permite establecer permisos de acceso de acuerdo al tipo de aplicación o servicio a la cual se quiere dar acceso. Uno de lo problemas de otras herramientas de firewall es que se limitan a cumplir las funciones de Packet Filtering y de Stateful Inspection, sin embargo hoy en día es necesario tener mucha más funcionalidad de un firewall para brindar seguridad a la red. ISA Server implementa un Gateway de aplicación que no solo filtra el tráfico de entrada y salida, sino que lo analiza y determina si el tipo de tráfico enviado a los puertos especificados es el que debería estarse enviando y no un tráfico malicioso, oculto en una transmisión de datos permitida.

Microsoft ISA Server es un producto ampliamente utilizado a nivel mundial para garantizar la seguridad de las redes informáticas en empresas e instituciones.

A continuación se detallan algunos de las características y beneficios que este producto ofrece.

Características de Microsoft ISA Server.

- ISA Server se caracteriza por bloquear todo el tráfico entrante y saliente a menos que el administrador haya establecido reglas de acceso que habiliten el tráfico a través del Firewall.

- Microsoft ISA Server cumple con el estándar establecido por los ICSA Labs. para la evaluación de firewalls disponibles en el mercado.

- La funcionalidad de filtrado a nivel de aplicación, permite una inspección profunda del contenido que atraviesa el Firewall. Es necesario para una correcta administración de la seguridad detectar y entender el tipo de información que esta entrando o saliendo a través del firewall. Microsoft ISA Server nos brinda esta capacidad que otras soluciones no poseen.

- Implementa filtrado de paquetes y stateful inspection, de tal manera que el firewall no mantiene puertos abiertos para el tráfico de respuesta de una comunicación, sino que solo permite el tráfico de respuesta en los puertos necesarios para conexiones establecidas usando una regla de acceso válida.

- La arquitectura avanzada de proxy de Microsoft ISA Server permite además de brindar un adecuado nivel de seguridad a la red, utilizar un servicio de cache de páginas Web y servidor proxy para el acceso de la red de la organización a través de una sola IP pública.

- Publicación Segura de servicios es una de las características de Microsoft ISA Server. Esta característica permite al administrador definir el acceso hacia servicios que corren en la red interna de la organización, a través de conversión de IPs (NAT). Esta característica permite acceder a servidores Web y servidores de Mail de manera segura desde la Internet, sin dejar de analizar si a través de los respectivos puertos 80 y 25, se están enviando instrucciones (comandos) autorizadas de http y smtp.

- Permite el escaneo del texto de los emails e incluso de los archivos adjuntos en los mensajes.

- ISA Server tiene la capacidad de examinar el tráfico SSL de entrada, buscando comandos o instrucciones no permitidas en los mensajes.

- Además de las características de firewall, Microsoft ISA Server es también un servidor de proxy y cache para el acceso al Internet. Acelera el acceso de usuarios a contenido Web, manteniendo las páginas en un cache de disco y de RAM. Con ISA Server es posible realizar una descarga proactiva de contenido al cache para que esté disponible rápidamente para el usuario.

- ISA Server permite priorizar el ancho de banda en función del tipo de tráfico que se esta transmitiendo.

- Administración de políticas de acceso basadas en Dirección IP, Usuario, Calendario y Grupo de Contenido. Para la creación de políticas por usuarios, ISA Server se integra con el servicio de directorio de Windows 2000 (Active Directory) permitiendo la asignación de accesos a los mismos usuarios definidos en la red de la organización.

- Creación de redes privadas virtuales entre dos oficinas o entre una oficina y un usuario.

- Facilidad para crear arreglos redundantes sin incrementar la administración de la seguridad.

- ISA Server genera Logs y Reportes de utilización, para auditar el tráfico que ha logrado penetrar a la red o que ha sido rechazado; y para auditar el tráfico Web que ha sido accedido a través del Firewall.

Estas y otras mas son las razones por las cuales hemos seleccionado Microsoft Internet Security and Acceleration Server 2000 como el producto a utilizar en nuestros firewalls de red.

3.2.2 Computer Associates E-Trust Intrusion Detection

Este sistema de detección de intrusos es una solución completa que abarca tres campos principales de seguridad en una sola herramienta: sistema comprensivo de la detección de la intrusión de la red, monitoreo de sesiones en tiempo real y bloqueo de contenido de Internet; de esta manera se forma una verdadera protección sobre la red a un bajo costo.

E-Trust es una solución escalable, de alto rendimiento diseñada para resolver las necesidades de seguridad de redes de cualquier tamaño, incluyendo redes empresariales del alto tráfico.

Las características de administración incluyen una consola centralizada que permite que los administradores supervisen sus redes enteras desde un solo sitio de trabajo; un motor integrado de antivirus que permite detectar un brote potencial de este tipo de ataques; actualizaciones automatizadas de direcciones de Internet que deberían bloquearse; y optimización del funcionamiento de la red trabajando en conjunto con soluciones de terceros.

Características de E-Trust Intrusion Detection.

- Bloqueo de tráfico en la red: en el cual se puede bloquear todo el tráfico de un servicio específico, tales como: http, smtp, pop3, etc.

- Exploración de patrones de palabra: los administradores pueden definir patrones de palabra que pueden indicar violaciones de políticas. Esto evita que los datos sensibles sean enviados sin autorización vía correo electrónico o por el Web.

- Verificación de actividades de ataques a la red como: abuso de ping o escaneo de puertos TCP, permitiendo tomar acciones inmediatas que son recomendadas por el software.

- Monitoreo de accesos a los recursos de la red definiendo criterios por usuario, dirección IP o nombre de estación.

- Definición de objetos de la red: usuarios, estaciones, rango de direcciones IP sobre los cuales se pueden aplicar reglas.

- Servicios de exclusión: Si se definen reglas con este parámetro, todo registro o matching es ignorado.

- Identificación de puertos abiertos en el firewall.

- Alarmas sobre el servicio SMTP al detectar:
    - Envío de correos con archivos adjuntos, virus, nombre especifico del adjunto o tamaño.
    - Palabras específicas en el asunto.
    - Palabras específicas en el cuerpo del mensaje.

- Personalización de reportes:
    - Por usuarios
    - Por nombre de PC
    - Por reglas
    - Por tipo de tráfico
    - Por dirección IP
    - Por dirección URL
    - Por reglas cumplidas
    - Estadísticos

- Personalización de mensajes de alerta: vía correo, mensajes NT, beeper, mensajes en la consola del Intrusion Detection.

- Análisis del tráfico la red en base a resultados obtenidos por E-Tust que permiten definir mejores reglas y bloquear accesos innecesarios o prohibidos.

- Análisis de utilización de la red por parte de usuarios y aplicaciones permite que los administradores realicen un mejor planeamiento de las políticas de red.

- Registro y análisis histórico de intrusiones disponible para posteriores análisis.

- Administración remota: La consola de administración de E-Trust permite a usuarios delegados conectarse remotamente al motor de Intrusion Detection para supervisar, cambiar reglas y crear informes, dependiendo de los permisos definidos por el administrador.

- Administración Centralizada: En un ambiente empresarial es posible tener diversas instancias de Intrusion Detection en varios segmentos de red. En este caso el administrador puede supervisar y controlar cada instancia desde una estación central. La verificación de alarmas y los informes estarán basados en información consolidada de todos los puntos donde esté instalado el motor.

Funcionamiento de E-Trust Intrusion Detection.

Cuando E-Trust Intrusion Detection está activo, escucha todo el tráfico TCP/IP que pasa a través de la red e identifica ataques contra servidores específicos o contra la red entera. Toda comunicación TCP/IP entre dos computadores es considerada una sesión y cada sesión se compone de distintos acontecimientos los cuales representan requerimientos hacia un servicio específico. Cada acontecimiento es comparado a las reglas establecidas y se toman las acciones configuradas en base al resultado de la comparación.

Cada acontecimiento se comprueba con todos los tipos de reglas en base a la siguiente prioridad:

- Intrusion Attempt Detection rules
- Content Inspection rules
- URL Access Monitoring and Control rules
- Monitor/Block/Alert rules
- Suspicious Network Activity Detection rules (procesado paralelamente con otras reglas y no tienen efecto las prioridades)

Si el acontecimiento coincide con algunas de las reglas se termina la sesión y se ejecutan las acciones definidas en dicha regla (registro, bloqueo o notificación).

Si la primera regla no se resuelve, el programa comprueba la segunda regla, etcétera, hasta que se resuelve alguna regla o se han comprobado todas las reglas.

Bloqueos del E-Trust

El mecanismo de bloqueo trabaja haciendo spoofing a la sesión y enviando un paquete de fin de sesión para desconectarla. El cliente recibe un mensaje de bloqueo dependiendo del protocolo y el servidor cree que la sesion ha sido desconectada.

E-Trust Intrusion Detection puede supervisar y bloquear cualquier sesión que viaja a través de un puerto de protocolos UDP o TCP; los criterios se diferencian dependiendo del protocolo y del tráfico, en base a estos puede bloquear sesiones, basado en los siguientes criterios:

- Contenido incluido en la sección de la cabecera del paquete
- Contenido activo, información de la cabecera y patrones de palabras definidos

Alertas y Respuestas de E-Trust

Cuando la solicitud de una sesión hace match con una de las reglas, el Intrusion Detection realiza una o más de las siguientes acciones:

- Envía un correo o fax.

- Añade una entrada al Log de Eventos del sistema operativo.
- Despliega un mensaje específico en la pantalla del operador del Intrusion Detection.
- Invoca un programa de Windows o DLL para crear una alerta customizada.
- Emite un sonido.
- Crea un trap SNMP.
- Envía mensajes a un beeper.
- Termina la sesión actual.
- Define una nueva regla de bloqueo.
- Exporta el registro de una sesión a cierto directorio.
- Configura un router Cisco para permitir o negar cierta clase de tráfico para ciertas direcciones IP.
- Configura las reglas de un sistema CheckPoint Firewall-1 para cerrar sesiones, o bloquear / desbloquear direcciones.
- Envía una alerta a la consola de administración del Unicenter TNG Management.
- Añade reglas al E-Trust Firewall.
- Envía mensajes al sistema de E-Trust audit.
- Envía una alerta al E-Trust Policy Compliance.

Configuración de Reglas de E-Trust

E-Trust Intrusion Detection incluye un conjunto de reglas predefinidas sobre la red siendo posible utilizar estas reglas o crear nuevas en función de las necesidades de la empresa. Es importante definir las reglas usando un criterio propio para bloquear, monitorear o alertar al administrador del sistema de las actividades indeseadas sobre la red.

La creación de una nueva regla consiste en definir:

- Las estaciones que supervisará
- El o los servicios que controlará
- La acción que se tomará
- El horario en que se aplicara la regla

Tipos de Reglas del E-Trust

- Monitoreo / Bloqueo / Alerta

  Estas reglas se utilizan al realizar un bloqueo por protocolo o por sitio Web específico. Es posible ver los acontecimientos registrados y bloqueados en la consola del software.
- Detección de Intento de Intrusión

Estas reglas detectan intentos de intrusión sobre la red basado en un número de actividades predefinidas que representan un posible ataque a un servicio; por ejemplo, un comando de ftp que utilice un puerto TCP distinto al estándar, podría indicar que alguien esta empleando mal el servicio o trata de confundir al servidor FTP.

De forma predeterminada, los detalles de los registros de la intrusión están en la ventana de la consola y en la caja de diálogo de los mensajes de alerta.

- Inspección de Contenido

  Estas reglas comprueban si hay componentes activos en archivos adjuntos de correo electrónico tales como código HTML, virus, transferencias directas de ftp o código binario. En la lista de las reglas de la inspección del contenido, se puede elegir que componentes buscará, y la acción que se tomará cuando se detecte uno de estos componentes.

- Detección de Actividad de Red Sospechosa

E-Trust utiliza estas reglas para comprobar si hay alguna rutina no considerada en las actividades de la LAN, por ejemplo, un ataque distribuido de negación de servicio.

- Monitoreo y Control de Acceso a URLs

  El monitoreo y registro de estas reglas tienen cabida sobre los sitios que son improductivos (no relacionado al trabajo) y tienen un grado específico. Es posible definir categorías (por ejemplo, juegos) para cada sitio Web.

3.2.3 Symantec Antivirus Corporate Edition

Como parte de la protección contra virus aplicada en la red, hemos decidido instalar Symantec Antivirus Corporate Edition en todas las estaciones y servidores de nuestra red. Esta solución corporativa permitirá mantener al día las actualizaciones de nuevos virus de tal manera que la información no corra riesgos por un ataque de virus.

Esta solución de antivirus corporativa, nos permitirá instalar de manera sencilla el antivirus en todas las estaciones y servidores de red, administrar y actualizar centralizadamente las definiciones de virus, establecer y forzar políticas de protección antivirus, verificar y controlar las revisiones de discos y los logs o historia generado por el antivirus desde una consola central.

Algunas de las características que respaldan nuestra selección de solución de antivirus son las siguientes:

- Symantec Antivirus es una de las soluciones de protección contra virus mas utilizada y cuenta con el respaldo de una empresa dedicada a la seguridad de las redes y la información.

- El tiempo de respuesta ofrecido por Symantec para encontrar una protección contra un nuevo virus es uno de los más rápidos entre otras compañías del mercado.

- Symantec Antivirus esta respaldado por el centro de investigaciones de virus Symantec Security Response, uno de las instituciones de investigación y soporte de seguridad en Internet más prestigiosas del mundo.

- Symantec Antivirus Corporate Edition es una solución escalable, con soporte multiplataforma, para estaciones y servidores a lo largo de toda la empresa. Las características de seguridad y la capacidad de habilitar políticas de protección centralizadas lo convierten en una herramienta poderosa para el administrador de seguridades.

3.2.4 Sophos Antivirus MailMonitor.

MailMonitor protege la red de posibles virus desde el servidor de correo, comprobando todo el mail que pasa a través del servidor SMTP. Es un sistema de reducción de amenazas incorporado en la suite para SMTP que mejora la protección ante potenciales portadores de virus.

MailMonitor incorpora detección y desinfección de virus en email, incluyendo los ocultos en archivos comprimidos (por ejemplo, en archivos ZIP). El sistema de reducción de amenazas disminuye el riesgo de entrada de posibles portadores de virus. Las opciones de notificación y configuración dan al administrador el control total sobre toda la red.

MailMonitor se basa en Sophos Anti-Virus para el escaneado de virus. Las versiones de Sophos Anti-Virus para Windows pueden actualizarse de forma automática con Sophos Enterprise Manager.

Para la construcción de nuestro ambiente hemos instalado Sophos MailMonitor sobre el servidor Linux de mail. Sophos MailMonitor escanea los mails enviados al SMTP de Linux antes de que lleguen a este, haciendo un relay de los mensajes no infectados.

3.2.5 Servicio de Directorio de Microsoft (Active Directory).

Un aspecto importante de las seguridades en la red es el nivel de acceso que se otorga a los usuarios internos de la misma. No es suficiente con provenir y evitar los ataques externos sino que es necesario también evaluar los riesgos de seguridad causados por usuarios internos. Las estadísticas muestran que la mayor parte de violaciones de seguridad de información provienen de personas con acceso físico a la red interna y se deben a una falta de políticas estrictas de acceso.

El directorio activo de Microsoft incluido en los productos Windows 2000 Server y Windows Server 2003 permiten mantener un ambiente de red seguro implementando políticas de acceso y niveles de privilegios al mismo tiempo que facilita la interacción de los usuarios con los recursos de la red, centraliza las información de cuentas de usuario y simplifica la administración de usuarios y estaciones de trabajo.

Beneficios del Directorio Activo de Microsoft

Algunos de los beneficios principales del directorio activo de Microsoft se detallan a continuación:

- Ofrece una de las soluciones mas flexibles del mercado en cuanto a productos para servicios de directorio.
- La administración de políticas de usuario y de grupos de usuarios facilita la asignación de privilegios en el acceso y la ejecución de software, así como la capacidad de instalar software en las estaciones o realizar tareas propias de un administrador.

- Permite a la organización administrar los recursos compartidos de red y los servicios a usuarios de manera centralizada y segura.
- Incluye la funcionalidad de directorio público de usuarios a través del cual una persona puede acceder al directorio interno de la empresa y a los recursos publicados en el.
- Microsoft Active Directory es compatible con el estándar propuesto de directorios LDAP v3 .
- La escalabilidad del producto permite que la red de la empresa se distribuya en distintos puntos geográficos sin incrementar la complejidad de la administración de la red.

3.2.6 Servidor SMTP - Sendmail.

Otro componente necesario para la operación de cualquier empresa hoy en día y más aún para un negocio de comercio electrónico es un sistema de correo electrónico confiable y seguro. Para cubrir esta necesidad se ha incluido como parte del diseño de red un servidor SMTP que envíe y reciba correo de Internet. El producto seleccionado debe de cumplir con los requerimientos de escalabilidad, confiabilidad y seguridad del negocio.

El producto seleccionado para realizar las tareas de servidor SMTP por sus características y funcionalidad es el SENDMAIL.

SENDMAIL permite habilitar algunas opciones de seguridad necesarias en toda red que se considere segura tales como Relay controlado y autenticación de SMTP.

Las características de relay controlado de SENDMAIL permiten al administrador especificar las condiciones bajo las cuales el SMTP server aceptará el relay de mensajes, es decir que dependiendo de los dominios de origen y destino, SENDMAIL aceptará o rechazará la operación de Relay. Esta característica de seguridad es crítica en toda red debido al riesgo de que usuarios o empresas externas utilicen nuestro SMTP Server para enviar mails masivos no solicitados también conocidos como SPAM.

Otras medida de seguridad incluida en SENDMAIL es la autenticación del SMTP. Esta funcionalidad permite restringir el relay de mensajes únicamente a aquellos usuarios que se encuentren en una red autorizada y que posean credenciales validas en el servidor SMTP. Esta característica de SENDMAIL restringirá el envío de correo únicamente a aquellos usuarios que estén autorizados para utilizar dicho servicio.

3.3 Definición de Políticas de Seguridad

Ninguna red puede estar protegida si sus administradores no definen las políticas de seguridad correctas. No basta con seleccionar e implementar los mejores mecanismos de seguridad, estos no servirán de nada sin una correcta configuración y administración.

Las políticas de seguridad en la red son las reglas que regirán el comportamiento de los usuarios y evitarán accesos maliciosos a recursos y servicios de la red.

Una correcta definición de políticas de seguridad estará dada en base al análisis de todos los riesgos posibles y es así que como parte de nuestro diseño se han definido las siguientes políticas de tráfico, detección de intrusos, privilegios de usuarios internos, acceso al SMTP, políticas del antivirus, de la base de datos y del servidor Web.

3.3.1 Políticas de Tráfico en la Red

Las políticas de tráfico en la red se pueden denominar también las políticas del firewall. Los sistemas de Firewall instalados en la red son quienes se encargarán de implementar las políticas de tráfico definidas y permitirán el paso únicamente de paquetes autorizados.

Como parte de las políticas de tráfico es necesario definir el tipo de tráfico que se autorizará en el firewall externo y en el firewall interno, es decir el tipo de paquetes que estarán autorizados para pasar por el firewall entre la red de administración y la Internet así como el tráfico entre la red de base de datos y la red de administración. Las políticas de tráfico deben de ser lo suficientemente restrictivas para asegurar la información y al mismo tiempo flexibles para dar acceso únicamente a lo requerido por el usuario interno y externo.

Cabe mencionar que estas políticas deberán considerar tanto tráfico entrante como tráfico saliente en ambos Firewalls.

3.3.1.1 Definición de Políticas del Firewall Externo

Estas son las reglas de acceso para el firewall que separa la red de administración de la red externa (Internet). A continuación se detalla el tipo de tráfico que estará autorizado en el firewall externo:

Red Externa a la Red Desmilitarizada (Inbound)

- HTTP (TCP 80), hacia la dirección IP del servidor Web.
- HTTPS (TCP 443), hacia la dirección IP del servidor Web.

Red Externa a la Red Interna (Inbound)

- SMTP (TCP 25), únicamente hacia el servidor de correo SENDMAIL ubicado en la red interna.

Red Desmilitarizada a la Red Interna (Inbound)

- SQL Server (TCP 1433) hacia el servidor de base de datos ubicado en la red interna.
- SMTP (TCP 25) desde el servidor Web ubicado en la DMZ hacia el servidor de correo corporativo.

Red Interna a la Red Desmilitarizada (Outbound)

- HTTP (TCP 80), hacia la dirección IP del servidor Web.
- HTTPS (TCP 443), hacia la dirección IP del servidor Web.

Red Interna a la Internet o red Externa (Outbound)

- HTTP (TCP 80) permitido para el grupo de estaciones de usuarios y servidor de antivirus corporativo.
- HTTPS (TCP 443) permitido para el grupo de estaciones de usuarios y servidor de antivirus corporativo
- FTP (TCP 21) permitido únicamente para usuarios administradores y servidor de antivirus corporativo
- SMTP (TCP 25) permitido únicamente para el servidor de correo corporativo
- POP3 (TCP 110) permitido para el servidor de correo corporativo y usuarios autorizados

- Tráfico de búsquedas y actualización de registros DNS únicamente para el servidor DNS corporativo

El resto de servicios estará restringido en el firewall externo, es decir que si un usuario o estación intenta enviar paquetes utilizando algún otro puerto no definido en las políticas, dicho paquete será descartado y no podrá atravesar hacia la red de destino.

La configuración del firewall externo está detallada en el apéndice B.

3.3.1.2 Definición de Políticas del Firewall Interno

Estas son las reglas de acceso para el firewall que separa la red de la Base de Datos de la red de administración, es decir que este firewall bloquea el tráfico indebido enviado hacia el servidor de base de datos.

Es importante aclarar que para el firewall interno, la red de administración de la empresa es considerada la red externa y la red de datos es considerada la red interna.

A continuación se detalla el tipo de tráfico que estará autorizado en el firewall interno:

Red Externa (Administración) a la Red Interna (Red de Datos) - (Inbound)

- Tráfico hacia la Base de Datos SQL desde la interface interna del firewall externo.
- Tráfico hacia la Base de Datos SQL desde la estación del Administrador de base de datos.

El resto de servicios estará restringido en el firewall interno.

La configuración del firewall interno está detallada en el apéndice B.

3.3.2 Políticas de Detección de Intrusos

El sistema de detección de intrusos es un componente de seguridad necesario en toda red que pretenda garantizar la confiabilidad, disponibilidad y privacidad de sus transacciones. El software de detección de intrusos tiene como finalidad detectar una posible intrusión o ataque en base al comportamiento de los usuarios y del tráfico generado por estos en la red. Debido a que la operación de estos sistemas se basa en patrones de tráfico identificados como anómalos e intrusivos el correcto funcionamiento del detector de intrusos dependerá de la política que se implemente sobre este. Una política mal diseñada tendrá como efecto la generación de alarmas innecesarias como Falsos Negativos, Falsos Positivos y Negativos Verdaderos. Estos tres tipos de falsas alarmas se producen cuando el sistema erróneamente no detecta una intrusión debido a que esta no es anómala, cuando el sistema detecta una intrusión por tráfico que en realidad no lo es pero que por ser anómalo es detectado como tal y cuando la actividad es no intrusiva por tanto el sistema lo muestra como tal. Ninguno de estos tres tipos de alarma es deseado porque dan una sensación de falsa seguridad y porque confunden el criterio con el que los administradores responden a las alarmas pudiendo en algunos casos ignorar verdaderos problemas.

Las políticas del detector de intrusos deben estar correctamente establecidas para que este solo genere alarmas al detectar una intrusión Positiva Verdadera, es decir una actividad de tráfico intrusivo y anómalo.

La política establecida en nuestro sistema de detección de intrusos E-Trust consiste de las siguientes reglas: Log de http, SMTP, ftp, telnet.

Las figuras 3.1 y 3.2 muestran las politicas que se han implementado en el ETrust Intrusion Detection.

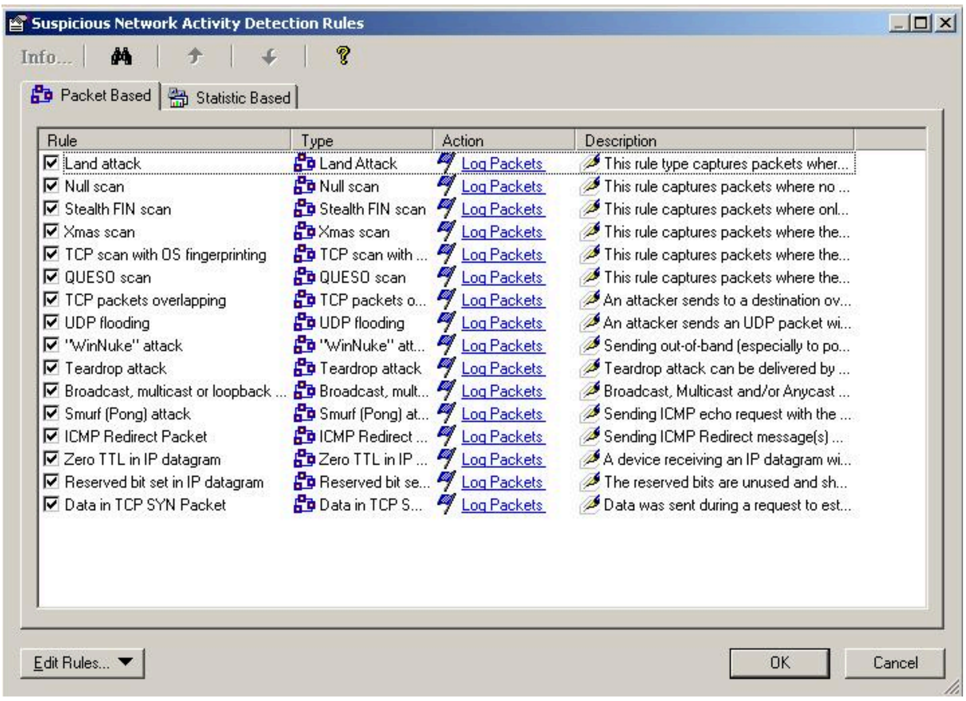


Figura 3.1 Políticas basadas en Paquetes

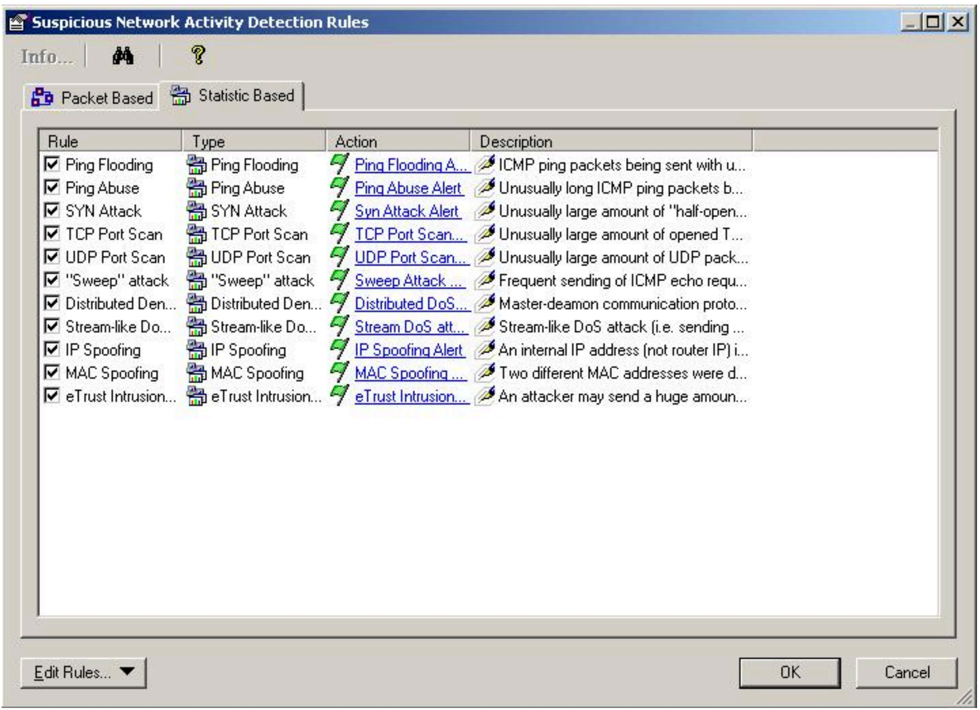


Figura 3.2 Políticas basadas en Estadísticas

3.3.3 Políticas de Acceso de Usuarios Internos

Haciendo caso a la estadística de que la mayoría de ataques provienen de usuarios internos de una red o por el mal manejo de los permisos internos de los usuarios, se han definido políticas que se aplicaran sobre las personas que laboren en la empresa.

Todo el personal interno de la empresa deberá poseer una credencial de usuario que le permitirá conectarse a la red y acceder a los recursos disponibles en la misma. Estos usuarios deberán tener privilegios restringidos sobre los recursos de la red e inclusive sobre sus propias estaciones.

Tipos de ataques comunes originados por un mal manejo de las políticas internas de acceso van desde instalaciones de programas troyanos e infección de virus hasta el robo de información utilizando programas sniffers.

Con el afán de eliminar en lo posible este tipo de riesgos hemos implementado el directorio activo de Windows 2000 (dominio de red) sobre el cual se han definido las políticas internas que se detallan en las tablas del Apéndice C.

3.3.4 Políticas de Acceso al SMTP

Uno de los problemas de seguridad más comunes es el Relay SMTP que realizan personas en la red para enviar correos electrónicos no solicitados mejor conocidos como SPAM. Este tipo de ataque aprovecha las vulnerabilidades y malas configuraciones realizadas a los servidores SMTP de otras empresas en Internet.

Para prevenir que esto ocurra es necesario mantener una correcta configuración de nuestro servidor SMTP implementando una política adecuada sobre este servicio.

Nuestra política de acceso al SMTP Sendmail consiste en realizar relay de correo únicamente para aquellos mensajes cuyo destino es nuestro dominio de correo electrónico, es decir todo mensaje recibido desde la Internet que tenga como destinatario una dirección de correo del dominio regalos.com será aceptado y procesado, caso contrario el mensaje no será aceptado por el SMTP. De igual manera se establecerá una restricción de acceso sobre este servicio de tal manera que únicamente se acepte el relay de mensajes provenientes de nuestra red interna.

Aplicando las políticas antes mencionadas evitaremos que personas no autorizadas utilicen nuestro SMTP para enviar SPAM hacia la Internet.

3.3.5 Políticas de Servicio de Antivirus

El antivirus para estaciones y servidores Symantec Corporate Edition debe estar instalado en todas las computadoras con sistema operativo Windows. Uno de los servidores de la red estará configurado como servidor central de antivirus y administrará el funcionamiento y las políticas del antivirus de las demás estaciones y servidores.

El servidor central de antivirus deberá estar programado para descargar diariamente las nuevas actualizaciones, tanto de las definiciones de virus como su motor.

Las opciones de configuración del antivirus no estarán disponibles desde cada estación o servidor, todas las configuraciones estarán bloqueadas y solo podrán ser modificadas desde la consola central del antivirus.

La protección en tiempo real para el sistema de archivos se habilitará para todas las estaciones y servidores.

Se establecerá una política de detección y desinfección de archivos mediante la cual se desinfecten los archivos con virus y en caso de no ser posible, se coloquen en cuarentena en la consola central hasta que una actualización logre limpiar el archivo o el administrador decida eliminarlo.

3.3.6 Políticas de Seguridad de Base de Datos

La seguridad de la base de datos es crítica en una empresa dedicada a la venta de productos a través de un sitio Web. De nada sirve implementar elementos de seguridad que garanticen la privacidad de la información durante la comunicación entre el cliente y el servidor Web si los datos una vez que sean almacenados no permanecen en un medio seguro. Es necesario además de asegurar los canales de comunicación, asegurar también la base de datos que almacenará la información de los clientes, de las transacciones y de los productos de la empresa.

Una parte fundamental de la seguridad de la base de datos es la protección que se implemente utilizando un firewall interno que permita el acceso únicamente del tráfico estrictamente necesario. El firewall interno de la red debe proteger el componente más sensible de nuestra infraestructura y este es la base de datos.

El segundo componente de la seguridad de la base de datos son las políticas de acceso que se implementen directamente en el motor de base de datos. Los usuarios creados en el sistema de bases de datos relacional deberán tener configurados los permisos adecuados de tal manera que no se otorguen privilegios innecesarios a usuarios no autorizados ya que esto podría dar paso a un ataque realizado por un usuario interno o a un daño no intencionado en los datos.

Microsoft SQL Server 2000 permite definir usuarios y roles para proveer accesos en base a roles o grupos de trabajo.

Nuestra política de seguridad para la base de datos implementará un rol de usuario con acceso a los datos dependiendo del tipo de aplicación que este utilice.

Otro elemento importantísimo de las políticas de seguridad de la base de datos son los service packs y parches de seguridad publicados por Microsoft. La mejor manera de prevenir un ataque que explote una vulnerabilidad del motor de base de datos es manteniendo actualizada la base con los últimos paquetes de servicio liberados.

Las políticas de auditoria de SQL Server estarán habilitadas para registrar todas las conexiones realizadas a la base y los comandos SQL ejecutados.

La política de backup de datos implementará respaldos diarios, semanales y mensuales, en modo incremental y en modo total respectivamente.

Los usuarios GUEST se deshabilitarán para evitar accesos no autorizados que aprovechen estos usuarios.

Periódicamente se ejecutarán los siguientes procedimientos de seguridad sobre la base de datos SQL Server:

- Microsoft Baseline Security Analyzer.
- Escaneo de logins para verificar contraseñas en blanco.
- Verificación de stored procedures de inicio.

Seguridad de la información sensible de clientes almacenada en la Base de Datos.

Todas las transacciones de compra que realicen nuestros clientes se registrarán en la base de datos. La información sensible del cliente tales como el número de tarjeta de crédito, usuario y contraseña del sitio deberán almacenarse en un formato encriptado que asegure la confidencialidad de la información. Esto ayudará a proteger los datos críticos del cliente de un posible ataque interno o externo a la base de datos.

El método de encriptación que se utilizará para almacenar la información sensible del cliente en la Base de Datos es una función HASH de una vía. El algoritmo HASH utilizado en nuestro trabajo es el SHA1 (Secure Hash Algorithm 1) implementado como parte de las librerías de clases .NET Framework.

3.3.7 Políticas de Seguridad de Servidor WEB

La seguridad de un servidor Web depende en gran parte del tipo de tráfico que se permita pasar hacia el servidor, únicamente los puertos necesarios 80 y 443 estarán permitidos desde otras redes hacia la dirección IP del servidor Web en la red desmilitarizada, de esta manera se evitarán ataques en otros puertos no autorizados.

Este bloqueo de tráfico no autorizado no garantiza del todo la seguridad del servidor Web ya que dicho servicio seguirá escuchando requerimientos en los puertos habilitados y el atacante puede tratar de utilizar estos puertos para causar daños en el servidor. Debido a esto, es sumamente importante que el servicio de páginas Web este implementado sobre una solución robusta que detecte el tráfico malicioso pasando a través de los puertos 80 y 443, de tal manera que se deseche este tipo de requerimientos.

Los principales ataques a servidores Web han ocurrido por vulnerabilidades de los servidores en detectar este tipo de tráfico malicioso, debido a esto es muy importante que el servidor Web que se utilice este siempre actualizado con los últimos parches de seguridad liberados por el fabricante.

El servidor Web seleccionado por nosotros para el proyecto es el Internet Information Server 6.0 que es la última versión del servidor Web de Microsoft. Este servidor es un producto maduro y robusto que nos permitirá eliminar los riesgos de caídas del servicio Web causadas por un ataque. Una de las características del Internet Information Server 6.0 es la capacidad de analizar el tráfico de URLs antes de pasar el requerimiento al motor del servidor Web y solamente acepta los comandos permitidos que no pueden causar daño. Esta funcionalidad se logra instalando la herramienta de seguridad URLScan de Microsoft sobre el servidor Web.

Adicionalmente al análisis de URLs realizado por IIS 6.0, el firewall ISA Server 2000 de nuestra red, tendrá habilitada la funcionalidad de Intrusion Detection que detectará también cualquier comando malicioso incluido en el tráfico HTTP.

Seguridad del tráfico entre el servidor Web y el browser del cliente.

Uno de los riesgos que los clientes tienen al realizar compras en la Web es que su número de tarjeta de crédito y sus datos personales sean atrapados en el trayecto entre su Web browser y el servidor Web del sitio de compras. Para eliminar este riesgo es necesario implementar el envío de tráfico cifrado desde y hacia nuestro servidor Web.

El mecanismo de cifrado que se utilizará en la implementación de nuestro sitio es protocolo SSL y HTTPS.

El protocolo SSL (Secure Sockets Layer) utiliza un algoritmo de cifrado de clave pública y privada mediante el cual la información es cifrada en la estación del usuario usando la clave pública del servidor y sólo puede ser descifrada por el servidor Web utilizando la clave privada que el posee.

El intercambio de claves públicas y privadas se realiza a través de un certificado digital que será emitido por una entidad certificadora autorizada, este certificado será instalado en nuestro servidor Web.

El nivel de cifrado que se utilizará es de 128 bits por lo cual el cliente requerirá un browser que soporte este nivel de cifrado (Internet Explorer 5.0 o superior).

3.4 Diseño de la Contingencia de la Red

Para poder considerar a una red como segura, es necesario que durante su diseño se hayan analizado todos las posibles fallas de la misma. Uno de los mayores riesgos en una infraestructura de red es el daño físico de uno de sus componentes críticos, es por esto que es necesario que se planifique la contingencia de la red y que un diseño adecuado considere la necesidad de elementos redundantes.

Para nuestro caso específico en el cual se pretende almacenar un sitio Web de comercio electrónico, los elementos críticos que podrían generar una caída general del servicio son: los firewalls, la base de datos del sitio y el servidor de páginas Web. Como parte de nuestro análisis hemos determinado necesario que los elementos antes mencionados posean equipos redundantes para que en caso de fallas físicas en alguno, la atención general de la red y del sitio Web no se vea afectada.

Todos los elementos redundantes en una red significan un costo operativo muy alto ya que en muchas ocasiones se cuenta con una capacidad instalada no utilizada en la cual se ha invertido gran cantidad de dinero. Es importante siempre analizar la relación costo beneficio de incorporar elementos redundantes en la red; una hora de interrupción del servicio a nuestros clientes puede significar un costo muy elevado para la empresa por las ventas que no se pudieron realizar y por la desconfianza generada en nuestros clientes.

3.5 Diagrama de una Red Segura

Los diagramas mostrados corresponden al diseño de la red segura que almacenará el sitio Web de comercio electrónico. Como parte de este proyecto se han realizado dos diseños, el primero corresponde a una red segura ideal que será necesario implementar en caso de formalizar un proyecto de negocio y poner en producción el sitio de comercio electrónico, el segundo diseño corresponde a una red segura con una configuración mínima que ha sido implementada como parte de este proyecto para el desarrollo del sitio Web y las pruebas de seguridad. El siguiente gráfico muestra un diagrama general de una red segura para un sitio de comercio electrónico.

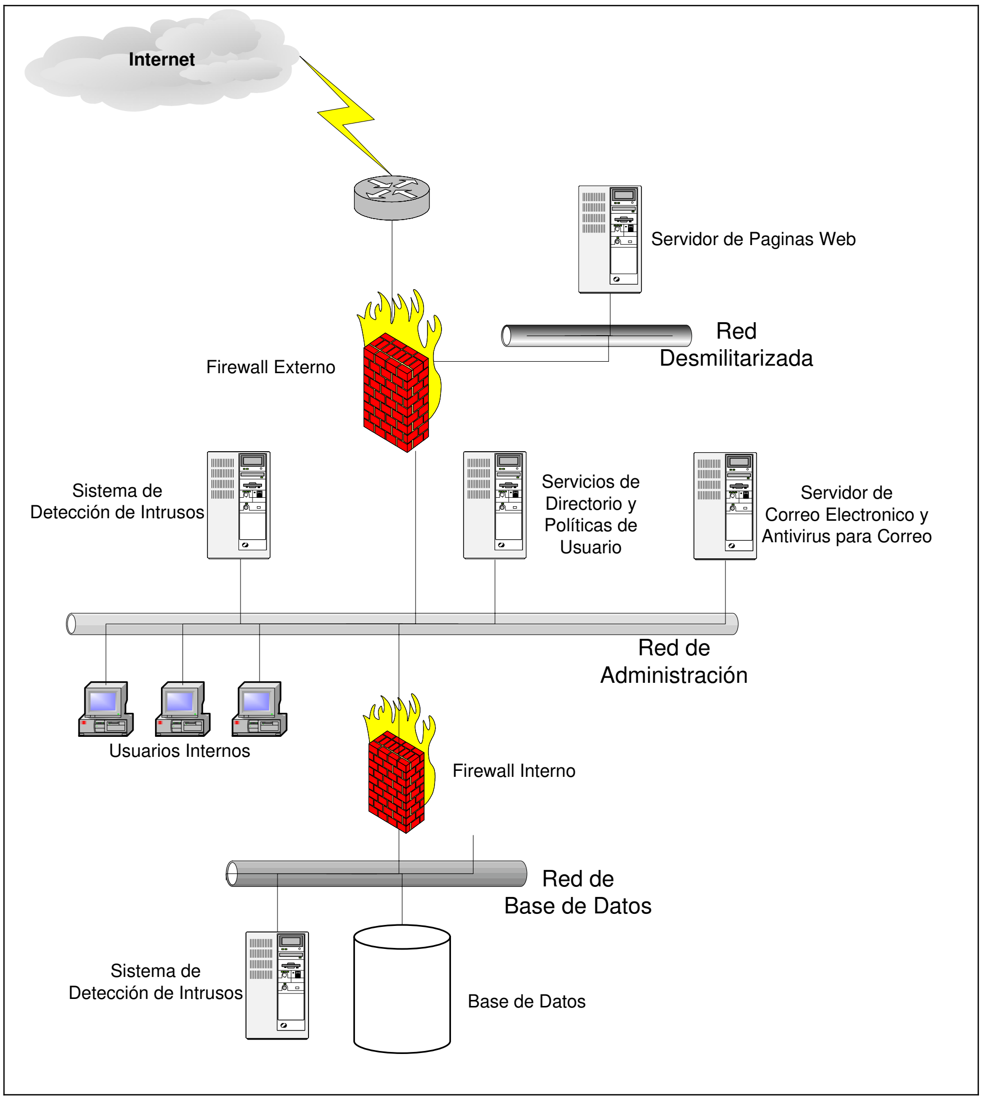


Figura 3.3 Diagrama de Red Segura

La red diseñada esta formada por 3 subredes independientes separadas por 2 arreglos de firewall. A continuación se explicarán cada una de las redes.

Red Desmilitarizada

La primera red es una zona desmilitarizada del arreglo de firewalls que asegurará la conexión hacia Internet. En esta red estarán conectados 2 servidores Web en modo de balanceo de red, que alojarán las páginas del sitio Web. Ambos servidores Web atenderán requerimientos de usuario en paralelo brindando un servicio ágil y redundante.

En la red desmilitarizada se ubicará también un servidor con el software de detección de intrusos. Este servidor analizará todo el tráfico dirigido a los servidores Web y reportará a los administradores cualquier intento de ataque al servicio Web o cualquier tráfico malicioso enviado desde las otras redes.

Red de Administración.

La red de administración es la red en la cual estarán ubicados los usuarios internos y los administradores de la red. Adicionalmente estarán conectados en esta red, los servidores de dominio y el servidor mail. Estos servicios a pesar de ser críticos para establecer las políticas de red y para la comunicación vía mail, no son servicios críticos que afecten la operación del negocio.

Red de Bases de Datos

La tercera red es la red más importante de todo el ambiente ya que en esta estará conectado el clúster de base de datos, que alojará la información de compras del sitio y las transacciones realizadas por los clientes a través de los servidores Web. Esta red esta protegida por un segundo arreglo de Firewalls que bloquea cualquier acceso que no haya sido autorizado por los administradores. Únicamente los usuarios internos y los servidores Web podrán enviar tráfico al servicio SQL del Clúster. El resto del tráfico hacia esta red estará restringido para todo el mundo.

3.5.1 Diagrama de Red Ideal

El diagrama de red ideal considera todos los elementos de contingencia descritos en este capítulo y que son necesarios para eliminar riesgos de disponibilidad del sitio Web. En toda red de datos existe el riesgo de que una falla física del hardware cause una caída general del sistema y se afecte la disponibilidad del sitio Web, debido a esto es necesario mantener un buen nivel de redundancia en los componentes de manera que se pueda eliminar en lo posible este riesgo.

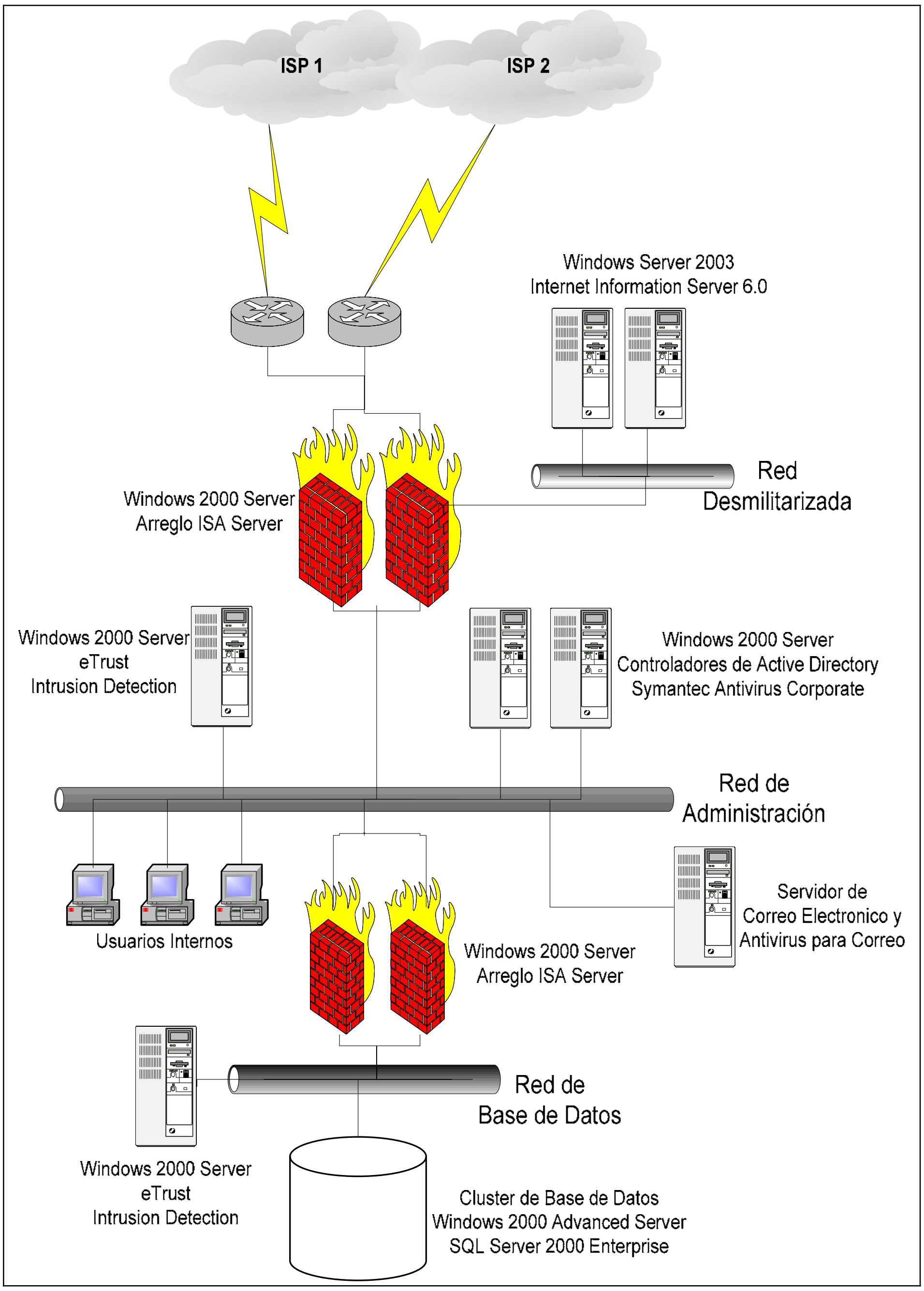


Figura 3.4 Diagrama de Red Ideal

3.5.2 Diagrama de Red Implementada en el Proyecto

El diagrama de red implementada corresponde a la red de computadoras que se implementó como parte de este proyecto. La finalidad de este diagrama es mostrar una configuración segura mínima utilizando los mecanismos de seguridad necesarios. Esta configuración de red se utilizó para el desarrollo del sitio Web de comercio electrónico y para las pruebas de penetración y seguridad. El diagrama de red implementado no incluye ningún elemento redundante ni de contingencia a fallos debido a la inversión económica necesaria para implementar dicha configuración.

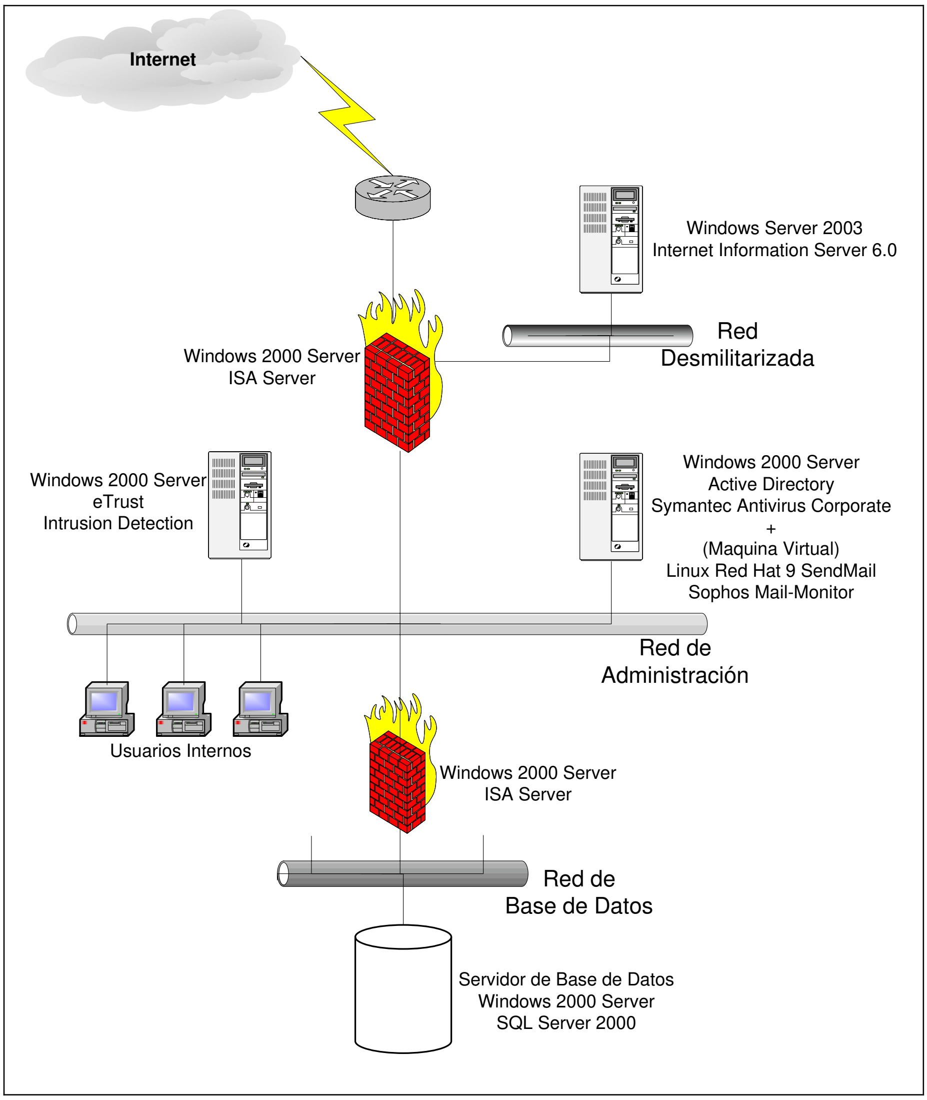


Figura 3.5 Diagrama de Red Implementada

Capítulo 4

4 IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA DE RED SEGURA PARA EL ALMACENAMIENTO DE UN SITIO DE COMERCIO ELECTRÓNICO.

Una vez que se han explicado las bondades de los productos seleccionados para asegurar nuestra infraestructura de red y que se han definido las políticas de seguridad que se configurarán en cada uno de los elementos de la red, hemos procedido a construir un ambiente de trabajo inicial con la finalidad de implementar el sitio de comercio electrónico.

En este capítulo se detallan cada una de las configuraciones realizadas en los productos de seguridad y el procedimiento para la implementación de cada elemento de la red. El propósito de este capítulo es mostrar una referencia de cómo se deben configurar los firewalls de la red, el servicio de detección de intrusos, las políticas de usuarios internos, los servicios de protección contra virus y el resto de elementos necesarios para construir una infraestructura de red segura que almacene el sitio de comercio electrónico.

4.1 Implementación de la Estructura de Dominio de Red

La implementación de la infraestructura de dominio consiste de dos partes, la instalación de los servidores de Windows 2000 y Windows 2003; y la construcción del dominio Active Directory.

4.1.1 Instalación de Servidores Windows 2000 y Windows 2003

La mayoría de los servidores de nuestra red utilizan como sistema operativo base el Windows 2000 Server. Uno de los procesos más importantes de la implementación de la infraestructura es instalar el sistema operativo adecuadamente para cada servidor. La instalación de componentes o servicios innecesarios en los servidores por omisión o desconocimiento del producto, puede ocasionar vulnerabilidades ya que estos servicios no estarán protegidos por mecanismos de seguridad definidos en la etapa de diseño.

En esta sección del capítulo se han documentado los puntos más importantes que deben ser considerados al instalar los sistemas operativos Windows 2000 Server y Windows Server 2003.

Recomendaciones al instalar Windows 2000 Server y Windows Server 2003.

- Particionar los discos duros de tal manera que el disco C tenga al menos 8 GigaBytes de espacio disponible para el sistema operativo y el archivo de paginación.
- En caso de contar con un segundo disco físico en el servidor, el archivo de paginación deberá ubicarse en el segundo disco.
- Todos los discos del servidor deberán utilizar el sistema de archivos NTFS. Esto es necesario por las características de seguridad que puede brindar NTFS.
- Al momento de la instalación se debe especificar la contraseña para el administrador local, la cual debe cumplir con las políticas de seguridad del dominio.
- Se deben instalar únicamente los componentes de Windows 2000 necesarios para las funciones del servidor. Durante la instalación se deberán deseleccionar componentes que no formen parte del rol del servidor.

- El sistema operativo Windows 2000 instala de manera predeterminada el protocolo TCP/IP. No deberán instalarse protocolos adicionales ya que no son requeridos para ninguna de las funciones del servidor.
- El sistema operativo Windows 2000 Server deberá tener instalado el último service pack disponible (actualmente Service Pack 4) y todos los parches de seguridad liberados por Microsoft.
- Como regla general no deberán existir cuentas de usuario locales en los servidores Windows 2000 a excepción de la cuenta de administrador local. Cualquier cuenta creada localmente en los servidores deberá cumplir con las políticas de contraseñas definidas, entre las que deberán estar la expiración de contraseñas.
- Definir la política “RestrictAnonymous” de tal manera que un usuario firmado como Invitado (Guest) no pueda tener acceso a listar los usuarios del dominio, recursos compartidos y políticas del sistema.
- La configuración de Autologon de Windows 2000 deberá estar deshabilitada ya que esta permite que el usuario y clave de logon queden almacenadas en el disco del servidor.

- No deberá existir más de un administrador local en cada servidor Windows 2000.
- Deberán desactivarse todos los servicios del sistema que no se requieran para el funcionamiento del servidor. Servicios innecesarios instalados de manera predeterminada como Telnet, SNMP o SMTP deberán deshabilitarse para eliminar cualquier vulnerabilidad de estos servicios.
- Los recursos compartidos en los servidores tales como carpetas e impresoras deberán tener permisos específicos por usuario o grupo y no deberán incluir el grupo Todos (Everyone).

4.1.2 Construcción del dominio Active Directory

La arquitectura del directorio de seguridad Active Directory consiste de servidores conocidos como controladores de dominio los cuales realizan las tareas de autenticación de usuarios, ejercen las políticas de seguridad y mantienen sincronizada la información del directorio a través de la red. Los servicios del dominio están basados en la arquitectura de resolución de nombres DNS y utilizan los servicios de este protocolo para enumerar y descubrir la ubicación de los servidores de dominio.

Una parte fundamental de la implementación de Active Directory es la configuración de un servidor DNS sobre Windows 2000 o 2003, ya que el DNS registrará la ubicación de los distintos servicios del directorio en la red.

Para la implementación de nuestro proyecto hemos dedicado un servidor para realizar las tareas de controlador de dominio y servidor DNS a la vez.

Pasos utilizados para la construcción del directorio activo.

- La instalación del directorio activo se inicia ejecutando el comando DCPROMO en el primer servidor controlador de dominio.
- Es necesario definir durante la instalación el nombre del dominio que se utilizará. El nombre de dominio utilizado en Active Directory debe constar de dos partes, el nombre completo basado en el estándar DNS también conocido como FQDN (Fully Qualified Domain Name) y el nombre de dominio en formato NETBIOS. Para

nuestro proyecto los nombres seleccionados respectivamente son REGALOS.COM.EC y REGALOS.

- Durante la instalación del directorio activo el asistente pedirá identificar el servidor DNS de la red. En este paso se deberá especificar un servidor DNS con soporte de registros dinámicos. En caso de no contar con este servicio en la red, Windows 2000 instalará el servicio de DNS Server en el mismo controlador de dominio.
- Una vez terminada la instalación del directorio, se deberán crear las cuentas de usuario y de grupo con los permisos adecuados de acuerdo a su perfil de trabajo.

4.2 Implementación de Políticas de Dominio

La implementación de las políticas de dominio definidas en el capítulo 3 se debe realizar utilizando las herramientas de administración que Windows ofrece una vez instalado el directorio activo.

La principal herramienta utilizada para la configuración de políticas del dominio es “Domain Security Policy”. Esta herramienta contiene una plantilla de políticas disponibles que se aplicarán sobre estaciones, servidores, usuarios o grupos de usuarios, dependiendo del tipo de política configurada.

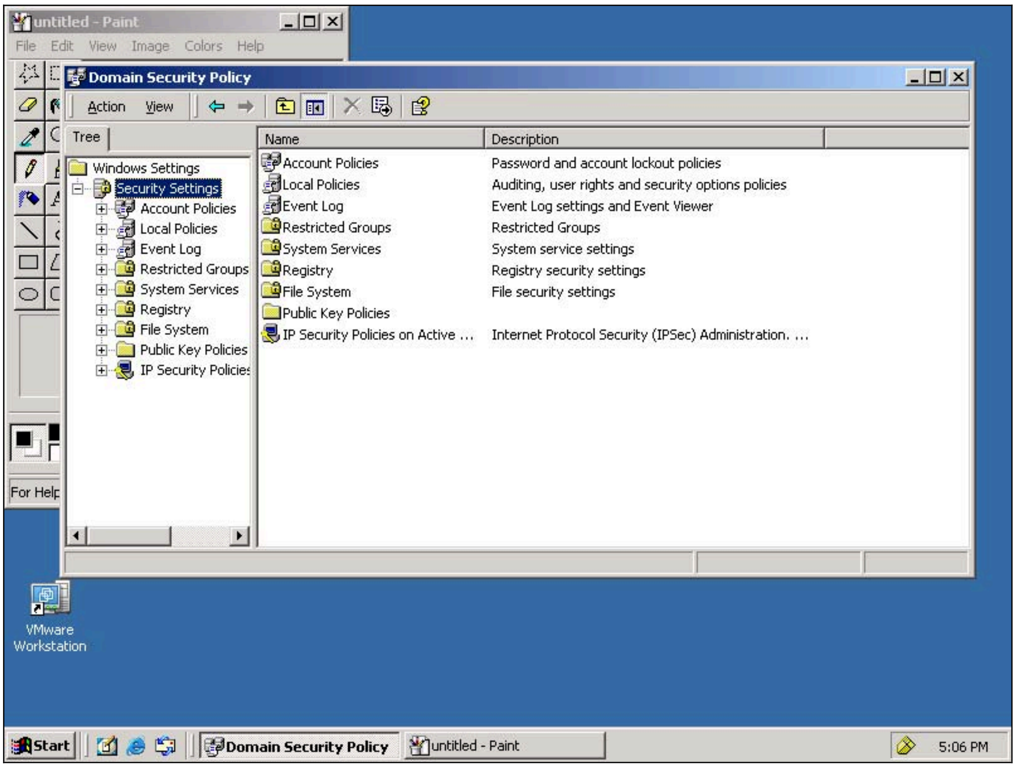


Figura 4.1 Herramienta Domain Security Policy

La asignación de políticas se debe realizar utilizando unidades organizacionales del dominio, las cuales serán creadas en base al tipo de usuario y tipo de máquina y que a su vez agruparán a usuarios, grupos, estaciones y servidores de la red.

4.3 Implementación de Firewall Externo

El firewall externo de nuestra red fue implementado utilizando Microsoft Internet Security and Acceleration Server 2000. El firewall ISA Server se debe instalar sobre un servidor Windows 2000 y una vez instalado es necesario configurar las reglas de tráfico permitido. Microsoft ISA Server bloquea de manera predeterminada todo el tráfico de entrada y salida; en caso de querer permitir el paso de cierto tráfico, se deberán definir las reglas adecuadas de acceso.

4.3.1 Instalación de ISA Server como Firewall Externo

En esta sección del documento se detallan las opciones de instalación de Microsoft ISA Server y su configuración para el caso de nuestro firewall de red externo.

El proceso de instalación del ISA Server se realiza ejecutando el instalador el cual invoca a un asistente que facilita el proceso de construcción de ISA Server.

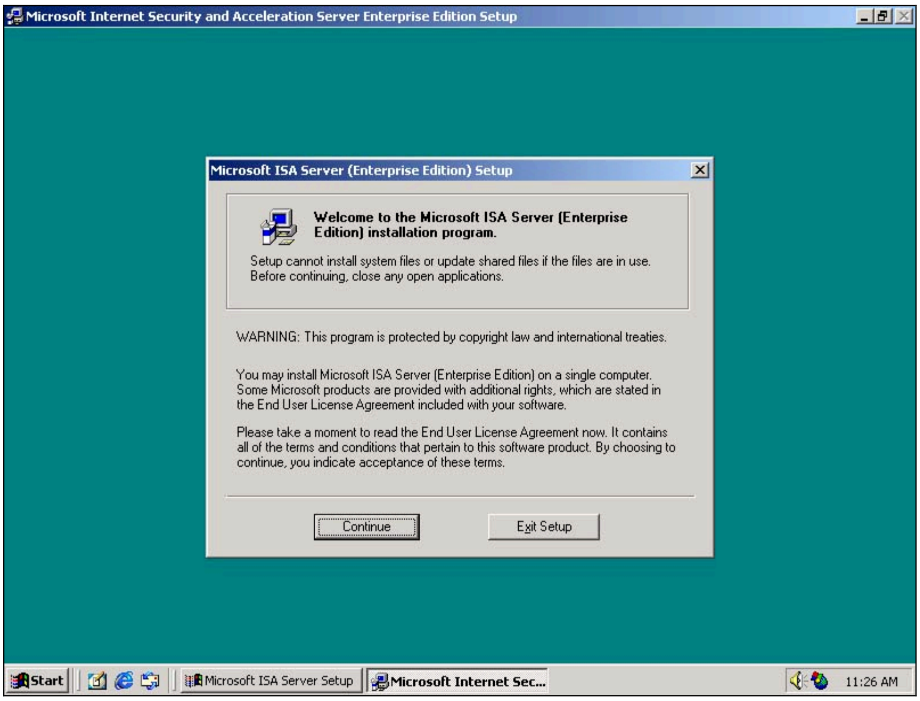


Figura 4.2 Instalación de Isa Server

Uno de los puntos más importantes de la instalación consiste en seleccionar los componentes del software que se desean instalar. Las tres opciones que se muestran son la instalación típica, la customizada y la completa, y dependiendo de la opción seleccionada se instalarán algunos o todos los componentes de ISA Server. La instalación típica de ISA Server instala las opciones de componentes más utilizadas. La instalación customizada permite seleccionar cuales de los componentes del software se copiaran. La opción completa instala el Microsoft ISA

Server con todos los componentes disponibles en el software.

Durante la instalación el asistente requerirá que se defina si el firewall ISA Server pertenecerá a un arreglo de firewalls ISA configurado en la red o funcionará como un firewall stand-alone. La modalidad de arreglo de ISA Server permite crear una solución de firewall redundante en la cual 2 o más servidores ISA Server trabajan bajo la misma configuración de manera que se tiene un backup del firewall y una distribución de la carga.

Para realizar esta configuración de arreglo es necesario que los ISA Server se encuentren configurados como miembros de un dominio Active Directory. En caso de no ser así, el servidor deberá instalarse en modo Stand-Alone. El firewall externo de nuestra red fue configurado como el primer servidor de un arreglo de firewalls externos de tal manera que es posible adicionar un segundo ISA Server a esta configuración y obtener una solución de firewalls redundantes.

Otro beneficio de la modalidad de arreglo consiste en que la configuración de los firewall se centraliza en la estructura del directorio activo de Windows y no se almacena en cada firewall, impidiendo de esta manera que un intruso obtenga acceso a esta configuración.

Otro punto muy importante de la instalación corresponde a la selección del modo en el que se instalará el Firewall ISA Server. Los modos de trabajo disponibles del ISA Server son 3: Modo de Firewall, Modo de Cache y Modo Integrado.
El modo de Firewall es el modo más seguro del ISA Server, con esta configuración únicamente funcionara como firewall y no habilitara las funciones de proxy server o Web hosting.
El modo de Cache es una opción menos segura en la cual el servidor ISA Server puede operar también como servidor Web. Este modo no es recomendable para firewalls conectados directamente a la Internet.
El modo integrado habilita las opciones de Proxy server y cache de páginas además de los servicios de firewall. Es una opción más segura que el modo cache pero que a su vez brinda la funcionalidad de servidor proxy para la red interna.
El modo seleccionado para nuestro firewall externo es el modo integrado.

El último paso para la instalación de Microsoft ISA Server consiste en la definición del segmento de red que ISA Server considerará como su red interna (segura).

En base al rango de direcciones IP que se defina como red interna para Microsoft ISA Server, este construirá una tabla de ruteo interna con la cual administrará las políticas de tráfico que sean definidas posteriormente en el firewall.

Una vez instalado ISA Server como firewall externo, el siguiente paso de esta implementación consiste en configurar todas las políticas definidas para el firewall externo en el capítulo 3.

4.3.2 Configuración de Políticas en el Firewall Externo

La configuración de políticas en Microsoft ISA Server esta subdividida de acuerdo al tipo de acceso que se quiere otorgar. A continuación se explican cada unas de las opciones que deben ser configuradas en las políticas de ISA Server.

Política de Acceso – Reglas de Sitios y Contenido

Este tipo de políticas se definen para otorgar acceso a contenido Web para las estaciones y servidores ubicados en la red interna. ISA Server permite otorgar permisos de acceso en base al tipo de contenido, el sitio destino y la estación o usuario que originan el requerimiento.

En esta sección hemos configurado las políticas de acceso de nuestros usuarios internos. Todos los usuarios tendrán acceso a sitios en la Internet, pero solo usuarios autenticados en el dominio tendrán acceso al sitio de e-commerce.

Política de Acceso – Reglas de Protocolo

Las reglas de protocolo de ISA Server permiten definir el tipo de tráfico TCP o UDP que está autorizado en el firewall desde la red interna hacia la red pública y hacia la red desmilitarizada. En esta sección de la consola de administración de ISA Server se deben configurar las reglas para tráfico SMTP de salida, FTP para administradores y servidor antivirus y consultas del DNS Server hacia los servidores DNS raíz en la Internet. Adicionalmente deben configurarse los protocolos de navegación que estarán permitidos para el grupo de usuarios y para el administrador.

Política de Acceso – Filtrado de Paquetes IP

En esta sección de la herramienta de configuración se deben definir las reglas de acceso de entrada hacia la red interna y hacia la red desmilitarizada.

Se utilizó el filtrado de paquetes IP para bloquear tráfico ICMP y UDP dirigido hacia el servidor Web. De la misma manera se creó una política que permite el paso de paquetes HTTP y HTTPS hacia el servidor Web.

Una política adicional de filtrado de paquetes fue definida para permitir la comunicación por protocolo SMTP desde la Internet hacia el servidor de correo.

Publicación – Reglas de Publicación Web

Este tipo de políticas se utiliza para establecer el acceso a servidores Web ubicados en la red interna (segura) del firewall. En caso de colocar un servicio Web en la red interna, se deberá utilizar este tipo de política para proveer acceso desde la Internet utilizando conversión NAT de la dirección IP del servicio Web. En la implementación de nuestra red no hemos ubicado ningún servicio Web público en la red interna y no se utilizarán este tipo de políticas.

Publicación – Reglas de Publicación de Servidores

Este tipo de política es similar a la de publicación de servidores Web pero es utilizada para establecer acceso a otro tipo de servicios desde la red pública y desmilitarizada a servicios en la red interna.

Este tipo de políticas se utiliza en nuestra red para permitir el acceso al servidor SMTP interno desde la Internet y desde el servidor Web ubicado en la red desmilitarizada. Adicionalmente se utiliza este tipo de políticas para permitir el acceso del servidor Web hacia la base de datos SQL Server ubicada en la red interna.

Elementos de políticas

ISA Server provee la definición de elementos de políticas en la cual el administrador puede definir tipos de protocolo de acuerdo al puerto que utilizan, nombres definidos para grupos de estaciones y servidores en las redes interna, pública y desmilitarizada. Como parte de la configuración del firewall externo hemos definido algunos elementos que facilitan la administración de nuestras reglas en el firewall. Algunos de esos elementos son Administradores, Usuarios, Servidor de Base de Datos, Servidor Web, Servidor Antivirus, Servidor DNS y Servidor SMTP.

Otro servicio habilitado a través de la configuración de ISA Server es la capacidad de detección de intrusos y de posibles ataques hacia el firewall externo. ISA Server provee esta funcionalidad a través de la cual el firewall examina el contenido del tráfico y logra detectar código malicioso oculto en un tráfico de red autorizado. En caso de detectar una amenaza en el tráfico de entrada ISA Server bloquea este tipo de tráfico y alerta al administrador de un posible ataque.

4.4 Implementación de Firewall Interno

La implementación del firewall interno ISA Server es muy similar a la realizada en la implementación del firewall externo. Existen algunas diferencias que se basan en el rol que cada firewall cumple y en las reglas que se deben definir.

La instalación del ISA Server interno es una instalación Stand-Alone a diferencia de la del firewall externo en el cual se creo un arreglo. Esta configuración se debe a que para formar un arreglo hace falta definir un servicio de directorio en la red de la base de datos el cual no estará disponible para la construcción de nuestra red.

Otra diferencia es el modo de firewall seleccionado para el ISA Server interno, a diferencia de la instalación del firewall externo, el modo seleccionado es el de Firewall Seguro, es decir que este equipo no proveerá servicios de cache de páginas Web y de servidor proxy si no que realizará las funciones exclusivas de un firewall. Este es el modo más seguro y por tanto el recomendado para la protección de la base de datos.

La definición de políticas del firewall interno es más simple que la del firewall externo debido al tipo de tráfico que deberá pasar por este. Para la configuración de la política de acceso a la base de datos SQL Server, se utilizó una regla de publicación de servidores en el cual se establece el acceso a la base de datos utilizando conversión NAT de una dirección IP interna oculta a la dirección IP externa del firewall interno.

4.5 Implementación de Detección de Intrusos

El software de detección de intrusos utilizado en nuestro proyecto es el CA eTrust Intrusion Detection. Este producto se especializa en descubrir posibles ataques hacia la red desde la Internet o desde la red interna. Es recomendable que la implementación de eTrust se realice sobre un servidor Windows 2000 ya que esta plataforma ofrece mayores seguridades y características de estabilidad.

El proceso de implementación de CA eTrust consiste de la instalación del producto y de la configuración de políticas definidas para la red.

4.5.1 Instalación y Configuración de eTrust Intrusion Detection

La instalación de eTrust se inicia ejecutando el asistente instalador desde el CD proporcionado por el fabricante. El primer paso de la instalación consiste en una verificación de los requerimientos del software sobre el servidor en el que se va a instalar. Algunos de los requerimientos revisados son sistema operativo, memoria RAM, procesador y tarjetas de red compatibles. Es muy importante contar en el servidor con tarjetas de red compatibles con eTrust ya que sin estas el producto funcionará de manera errada. Se recomienda revisar el listado provisto por el fabricante antes de instalar el producto.

Una consideración importante antes de instalar el eTrust es definir si el equipo sobre el que se instalará será un controlador de un nuevo dominio o un servidor miembro del dominio de red. Esto es necesario ya que en lo posible se debe mantener el servidor de eTrust como un equipo independiente de las demás estaciones. Para nuestro proyecto hemos instalado eTrust sobre un servidor standalone sin dominio.

El último paso importante durante la instalación de eTrust es seleccionar los componentes que se instalarán. La instalación consta de 3 componentes: Intrusion Detection, Central Agent y Data Client. Intrusion Detection es el componente principal del producto y debe ser instalado necesariamente. Los otros dos componentes son útiles en ambientes en los que se quiere centralizar la información recolectada por varios servidores eTrust. Para nuestro ambiente el único componente instalado es el Intrusion Detection.

La configuración de eTrust consiste en especificar las políticas definidas en los capítulos anteriores. CA eTrust viene configurado de manera predeterminada con ciertas políticas de seguridad propias del producto, basadas en tipo de paquete y basadas en información estadística, de tal manera que se verifica cualquier actividad sospechosa en la red. Para nuestro proyecto se utilizaron todas las políticas predeterminadas del producto y se configuraron las alarmas para que sean enviadas a los administradores de la red.

Otra configuración que debe realizarse en el servicio de Intrusion Detection es la actualización de la base de reglas y de la base de direcciones Web a ser bloqueadas. Estas dos actualizaciones pueden realizarse de manera periódica automáticamente a través de Internet.

4.6 Implementación del Servicio de Antivirus para Estaciones y Servidores

El servicio de antivirus para estaciones y servidores que hemos utilizado en el proyecto es el Symantec Antivirus Corporativo, el cual nos permite de manera centralizada mantener las estaciones y servidores protegidos y actualizados.

La implementación del servicio de antivirus consiste de varios componentes de software, el primero que se debe de instalar es el Symantec System Center el cual consta de la consola de administración del antivirus y la consola de administración de alarmas y mensajes. Symantec System Center es la herramienta que nos permitirá instalar desde la consola central los antivirus hacia las demás estaciones y servidores de la red. Adicionalmente este servicio se encargará de descargar desde el sitio Web del fabricante las actualizaciones de definiciones de virus y las replicará hacia los demás equipos de la red.

Otro componente importante de la instalación del servidor de antivirus de Symantec es la herramienta LiveUpdate Administration la cual se encarga de actualizar el servidor en intervalos definidos por el administrador.

Una vez instalado y actualizado el servidor de antivirus el siguiente paso es instalar desde la consola central el cliente antivirus para las demás estaciones y servidores de la red. Esta instalación remota utiliza los servicios de llamadas de procedimientos de Windows RPC para desplegar el software a través de la red.

4.7 Implementación del Servicio de Antivirus para Correo Electrónico

El servicio de antivirus para el SMTP utilizado en nuestro proyecto es el Sophos Antivirus MailMonitor. Este servicio consiste de un monitor para el SMTP Server que analiza todos los mensajes enviados y recibidos en la red. La instalación de Sophos Mailmonitor se debe realizar en un servidor que reciba todos los mensajes para luego hacer un relay de estos hacia el SMTP, es decir que el servicio SMTP esta detrás del MailMonitor y solo recibirá mensajes que provengan de este.

En nuestro proyecto hemos realizado una implementación de MailMonitor sobre Linux, y es la que se detalla a continuación.

La instalación de Sophos MailMonitor sobre Linux comprende dos pasos: instalación de Sophos Antivirus sobre Linux y la instalación de Sophos Mailmonitor.

La instalación de Sophos Antivirus para Linux se inicia desempaquetando el producto en formato TAR. Una vez que se ha descomprimido el instalador se debe ejecutar el asistente de instalación con el comando ./install.sh –ni.

Una vez instalado Sophos Antivirus se deberán copiar las actualizaciones de virus provenientes del fabricante. Estas actualizaciones se descargan de Internet en un formato propietario de extensión .IDE. Los archivos IDE deberán ser copiados en la carpeta usr/local/sav.

El siguiente paso de esta implementación consiste en instalar Sophos Mailmonitor sobre el servidor Linux. Esta instalación se inicia de la misma manera que el Sophos Antivirus es decir se descomprime y se ejecuta el archivo ./install.sh.

Algunos de los pasos de instalación de mailmonitor consisten en definir las carpetas sobre las que se instalará, las carpetas que almacenarán temporalmente los mensajes entrantes y salientes; y la configuración de la dirección IP y puerto que utilizará el monitor SMTP. Durante esta etapa se deberá especificar en que puerto escucha el mailmonitor (TCP25) y el puerto y dirección al que se enviarán los mensajes salientes.

Otro parámetro importante de la instalación es definir el servidor SMTP que se utilizará para enviar las alertas al administrador.

Los comandos utilizados para iniciar y terminar el servicio MailMonitor son:

./mmsmtpd –terminate

./mmsmtpd -start

4.8 Implementación del Servidor de Base de Datos

La base de datos seleccionada para almacenar la información de nuestro sitio de comercio electrónico es Microsoft SQL Server 2000. La implementación de la base de datos SQL Server requiere que se definan ciertos parámetros importantes durante la instalación.

La instalación del motor de base de datos se inicia ejecutando el asistente instalador y seleccionando los componentes de servidor de base de datos. Durante la instalación deben seleccionarse los módulos de cliente y servidor SQL Server, ya que los primeros se utilizarán para administrar la base de datos. Uno de los puntos importantes que se deben definir durante la instalación es la cuenta de usuario con la que se iniciarán los servicios de SQL Server. Hemos creado una cuenta llamada servicios que será utilizada para realizar estas funciones.

El modo de autenticación de la base de datos debe ser modo mixto, es decir que SQL Server podrá autenticar a usuarios del dominio o a usuarios creados en la base de datos. El tipo de licenciamiento que se debe utilizar para Microsoft SQL Server es por sitio ya que el único cliente de la base será el servidor Web y solo para este se requerirá una licencia de conexión (CAL).

Una vez instalado el servidor Microsoft SQL Server, el siguiente paso para la implementación de la base de datos del sitio Web es crear la base con sus tablas de acuerdo al diseño realizado durante la etapa de análisis previo al desarrollo del sitio.

La definición de las tablas puede realizarse manualmente utilizando las herramientas de administración de SQL Server o también pueden crearse procedimientos almacenados que al ser llamados desde un script creen automáticamente la estructura de tablas de la base.

4.9 Implementación de Seguridades de la Base de Datos

La base de datos es el elemento más crítico de nuestro ambiente ya que en esta residirá información confidencial de nuestros clientes como dirección, teléfono, correo electrónico y tarjetas de crédito. Debido a lo sensible de la información es necesario implementar un buen nivel de seguridad para este servidor. A continuación se detallan los puntos que se deben considerar para mantener segura la base de datos:

- La clave del administrador (sa) deberá seguir las políticas de passwords definidas, de tal manera que ningún usuario no autorizado obtenga privilegios de administrador y ponga en riesgo la información almacenada en la base.

- La ubicación física y lógica del servidor deberá considerar los riesgos de un ataque sobre el hardware o a través de la red para lo cual, este servidor deberá encontrarse dentro del centro de computo de la empresa y protegido por un firewall interno que permita la conexión del Web server y de los administradores.
- El acceso que realiza el Web server hacia la base de datos deberá ser autenticado utilizando un usuario creado en la base de datos y que tenga los suficientes permisos para poder realizar esta conexión. Jamás deberá considerarse el uso del usuario SA para este tipo de conexión del sitio.
- El servidor de base de datos deberá tener instalado todos los parches de seguridad liberados por Microsoft tanto para el sistema operativo Windows 2000 como para SQL Server. Actualmente el último service pack de SQL Server 2000 es el Service Pack 1.
- Otro elemento importante de la seguridad de la base de datos son los respaldos diarios que se realicen a la información almacenada. Es necesario crear usuarios no administradores que posean los suficientes privilegios para poder realizar esta actividad sin poner en riesgo la seguridad de la base.

- Nuestro diseño ideal de la infraestructura de red considera la implementación de un clúster de base de datos utilizando SQL Server Enterprise Edition. El clúster de base de datos ofrece un nivel adicional de seguridad ya que en caso de un fallo del hardware de uno de los servidores, el clúster seguirá funcionando con el segundo servidor.
- La encriptación de datos sensibles del cliente (número de tarjeta de crédito, usuario y contraseña) se realizará en la programación del sitio utilizando una función hash. La lógica del sitio deberá obtener el valor hash para cada uno de los datos sensibles e incluir la información encriptada en el stored procedure que guarde el registro en la base.

4.10 Implementación de Seguridades del Servidor WEB

El servidor Web es el servicio que mas riesgos corre en la red debido a que es quien permite el acceso de usuarios externos hacia el sitio y para esto debe estar expuesto a tráfico proveniente de la Internet. El servidor Web realiza las funciones de alojar las páginas del sitio y responder a los requerimientos de los clientes de la empresa.

La seguridad de este servidor es crítica para el éxito de nuestro negocio ya que una falla en el servicio Web ocasionará pérdidas económicas directas al no poder vender nuestros productos y a la vez generará desconfianza y una mala imagen a nuestra empresa.

Las políticas de seguridad implementadas sobre el servicio Web deberán permitir a los usuarios interactuar con las páginas y al mismo tiempo proteger al servidor de accesos maliciosos de un atacante, es decir debe minimizar el riesgo de una falla operativa del sistema causada por un intruso. La seguridad del sitio no solo dependerá de las políticas implementadas sino también de la capacidad del servidor Web a responder ante un ataque, es por esto que hemos seleccionado como servidor Web de nuestra red al producto Internet Information Server 6.0 de Microsoft el cual nos brinda la confiabilidad que requerimos para alojar nuestro sitio de comercio electrónico.

A continuación se detallan las consideraciones de seguridad que deberán analizarse al implementar el servicio de páginas Web:

- La ubicación física y lógica del servidor deberá considerar los riesgos de un ataque sobre el hardware o a través de la red. El servidor Web deberá colocarse dentro del centro de cómputo de la empresa y conectado a la red desmilitarizada siendo separado de la red interna y de la red publica por un firewall externo.

- El firewall externo de la red deberá permitir el tráfico http y https hacia este servidor y deberá bloquear cualquier otro tipo de protocolo que se intente enviar hacia este equipo. El firewall deberá también permitir que el servidor Web se comunique con la base de datos a través del firewall interno de la red utilizando el puerto TCP de SQL Server. Como medida de seguridad adicional hemos habilitado el escaneo de paquetes http y https en el firewall externo ISA Server de tal manera que este bloquee comandos maliciosos enviados hacia el servidor Web utilizando este tipo de tráfico.
- El servidor Web deberá tener instalado todos los parches de seguridad liberados por Microsoft para el sistema operativo Windows 2003 y para el servicio Internet Information Server 6.0.
- El acceso a páginas Web durante la compra de productos se realizará utilizando el protocolo HTTPS, para esto es necesario instalar un certificado digital sobre el servidor Web. Este proceso de instalación de certificados se realizará utilizando las herramientas que el software provee para este fin.
- Las páginas del sitio almacenadas en el servidor deberán poseer los permisos adecuados de tal manera que los usuarios puedan acceder a dichas páginas (permisos de lectura) y

además ejecutar los scripts almacenados en páginas aspx del sitio.

- El usuario y contraseña que utiliza el sitio para la conexión hacia la base de datos estará almacenada en una página especial de configuración la cual no podrá ser accedida por usuarios comunes. La página que almacena esta información en nuestro sitio es Web.config la cual posee un formato XML.
- Deberá estar instalada la ultima actualización de Microsoft .NET Framework ya que esta corrige una falla en la seguridad del producto.
- Se deberá instalar un certificado digital para el servidor Web, este procedimiento se realiza generando un archivo con un requerimiento de certificado digital y enviando este archivo a la entidad certificadora junto con los datos de nuestra empresa y servidor Web. Una vez aprobado y emitido el certificado se instalará en el servicio Internet Information Services y se configurará el acceso a las páginas seguras a través de encriptación de 128 bits.

Capítulo 5

5 DISEÑO E IMPLEMENTACIÓN DE UN SITIO DE COMERCIO ELECTRÓNICO

En capítulos anteriores hemos explicado la infraestructura de red segura que se debe utilizar para alojar un sitio de comercio electrónico, los mecanismos de seguridad que se deben implementar, la arquitectura de la red y las políticas de acceso que permiten asegurar un servicio de acceso público.

En este capítulo se ha documentado el procedimiento utilizado para el diseño e implementación de nuestro sitio de comercio electrónico. El desarrollo del sitio Web en el cual se ofrecerán y comercializarán productos ha seguido un proceso de análisis y diseño en base a nuestra necesidad y al problema que se intenta resolver, es decir que se han considerado los aspectos de diseño necesarios para la creación de las páginas y de la lógica que hace posible ofrecer y vender productos a través de un sitio Web.

5.1 Definición de los requerimientos del sitio de comercio electrónico.

Antes de desarrollar cualquier sistema de información es necesario definir cual es el problema que se intenta resolver y que es lo que se quiere lograr. Una vez definidos todos los aspectos de negocio que el sistema de información considerará y los procesos que el sistema implementará podremos realizar un análisis y un diseño especifico que nos muestre como será el producto final antes de comenzar con su desarrollo.

Como primer paso en la construcción de nuestro sitio de comercio electrónico hemos definido los problemas que el sistema deberá resolver y para esto nos hemos planteado las siguientes preguntas:

- ¿Qué es lo que se quiere lograr con el sitio?
- ¿Cuáles son los servicios que el sitio debe brindar?
- ¿Qué seguridad debe ofrecer el sitio durante una compra?
- ¿Cómo debe ser la interface de tal manera que impulse al usuario a comprar?

5.1.1 Objetivos de la construcción del sitio de e-commerce.

El objetivo principal de nuestro sitio de comercio electrónico es comercializar productos variados. El tipo de producto que se ofrecerá son artículos que un comprador adquiriría para regalar a otra persona por motivo de un acontecimiento importante como es un cumpleaños, un matrimonio, un aniversario, etc.

Este tipo de negocio en una manera tradicional requeriría de un almacén en el cual los posibles compradores pudieran entrar, examinar los productos exhibidos en los estantes de la tienda, seleccionar los artículos que les gustaría comprar y finalmente acercarse a un cajero para cancelar su compra. Este mismo funcionamiento de compra en un almacén es el que requerimos implementar en el sitio, los usuarios deben poder entrar a nuestra tienda virtual, deben poder recorrer la tienda, seleccionar uno o varios productos y finalmente pagar su compra.

5.1.2 Servicios que el sitio debe brindar.

En un mercado tan pequeño y competitivo como es el ecuatoriano es necesario para todo negocio lograr una diferenciación de su competencia que asegure el éxito de la empresa al atraer y mantener clientes y finalmente incrementar las ventas.

Existen en el mercado otras opciones de sitios de compra en línea y por supuesto existen también los almacenes y comercios tradicionales, parte de nuestra estrategia comercial es brindar facilidades y servicios adicionales a nuestros clientes de tal manera que ellos prefieran comprar en nuestro sitio antes que ir a recorrer las calles en busca de un regalo.

Los servicios adicionales que el sitio debería implementar son:

- Un asistente para la búsqueda de regalos en el cual el usuario pueda ubicar un artículo basado en las características de la persona a quien se regalará.
- Una herramienta de organización de regalos en grupos orientada a regalos para matrimonios.
- Un sistema de recordatorio de fechas importantes que le avise al cliente cuando se acerca un evento importante para el cual debería adquirir un regalo.

5.1.3 Seguridad que el sitio debe ofrecer durante la compra.

Una de las desventajas del tipo de negocio en el cual nos hemos enfocado es la desconfianza del mercado a realizar una compra en Internet utilizando una tarjeta de crédito. No importa cuanto gusten los artículos o lo agradable de la compra en el sitio, si no se ofrecen los niveles de seguridad adecuados nuestro negocio puede fracasar.

El sitio de compras desarrollado deberá contar con un sistema de transacciones seguras en el cual el cliente pueda enviar de manera cifrada la información de su tarjeta de crédito sin riesgos de que una persona no autorizada la intercepte y la utilice.

5.1.4 Interface de usuario vendedora.

Se mencionó en párrafos anteriores que nuestro sistema debe brindar una experiencia similar a la que una persona obtiene cuando entra a un centro comercial o a un almacén. El cliente debe sentirse a gusto dentro del almacén y los artículos deben estar ordenados de tal manera que se llame la atención del usuario hacia ciertos productos específicos.

Este mismo concepto de mercadeo debe aplicarse a la interfaz de usuario del sitio de compras. El usuario que ingrese a nuestra página Web deberá sentirse a gusto en ella gracias a un manejo correcto de los colores y de otros componentes gráficos. Adicionalmente debemos atraer la atención del usuario hacia productos especiales en los que queremos que se fije.

Hacer posible esta estrategia de mercadeo dependerá de un correcto diseño y de un manejo adecuado de los elementos gráficos del sitio.

5.2 Desarrollo de una estrategia para la creación del sitio

Una vez definido de manera general el problema debemos seguir una metodología o estrategia para la construcción del sitio Web. Como metodología de desarrollo hemos utilizado algunas de las recomendaciones definidas en el estándar UML (Unified Modeling Language) para el desarrollo de aplicaciones orientado a objetos.

Los pasos de la estrategia utilizada son 4:

- Análisis de los Requerimientos.
- Diseño General del Sistema.
- Diseño Detallado.
- Implementación.

5.2.1 Análisis de los Requerimientos

Los requerimientos tecnológicos del sitio estarán dirigidos al desarrollo de una aplicación amigable que permita buscar y comprar productos.

Esta sección de la estrategia detalla las especificaciones necesarias con que debe contar la aplicación y que servirán para realizar un diseño inicial del sitio y posteriormente su desarrollo.

Validaciones Generales

- El sistema debe permitir que un usuario navegue a través de las distintas categorías de productos.
- El sistema debe mostrar la información disponible de cada producto junto con una foto del mismo.
- El sitio debe contar con un carrito de compras en el cual el usuario podrá agregar los distintos productos que piensa comprar para luego realizar la compra de los artículos que haya agregado al carrito.
- El sistema debe tener la capacidad de mostrar y vender los productos que hayan sido ingresados en una base de datos.

- El sistema debe ejecutarse sobre un servidor de páginas Web y acceder a un servidor de base de datos en el que se almacenará la información de productos, clientes y transacciones.
- Debe ser posible aplicar descuentos a ciertos productos.
- El sistema debe mostrar en su página inicial ciertos productos que se encuentren en oferta.
- El sitio debe contar con una herramienta recordatoria de eventos, en el cual el usuario ingrese las fechas especiales de sus familiares y amigos.
- El sitio debe avisar al usuario que esas fechas especiales se acercan.
- El sistema debe poder almacenar la información de los usuarios que se registren durante su primera compra. Para el ingreso de esta información se le debe solicitar al usuario una cuenta de correo electrónica valida, así como una clave de acceso que asegure los datos del usuario.
- La compra de productos en el sitio debe contar con un sistema de seguridad que envíe de manera cifrada la información del usuario.

- La información de usuarios ingresada en el sitio debe almacenarse de manera segura en una base de datos.
- El sitio debe contar con un asistente en el cual el usuario defina las características de la persona a la que se regalará el producto, y el asistente le sugerirá ciertos productos basándose en esas categorías.
- El sistema debe contar con una página en la cual los usuarios puedan separar regalos de matrimonio y luego entre varios clientes realizar abonos y adquirir el o los regalos en conjunto.
- El usuario debe poder desde cualquier página y en cualquier momento regresar al menú inicial del sitio.
- El sistema debe registrar el número de intentos fallidos de logon que el usuario realiza cada vez que este ingrese incorrectamente sus credenciales del sitio. El sistema debe permitir al usuario equivocarse hasta 10 veces con la clave, en caso de que se exceda este límite, la cuenta quedará bloqueada y el usuario deberá comunicarse con el administrador del sitio para pedir el desbloqueo de su cuenta.
- El sistema deberá validar el formato de las contraseñas de los usuarios. Las contraseñas deberán poseer

caracteres numéricos y alfabéticos; y la longitud mínima de la contraseña será de 8 caracteres.

5.2.2 Diseño General del Sistema

A continuación se detalla la arquitectura que hemos desarrollado para la implementación del sitio Web.

Arquitectura del Sitio Web

La arquitectura del sitio esta formada por 6 capas lógicas que interactúan entre si, cada una de las capas implementa las funciones necesarias para manejar una porción específica del proyecto.

Cada capa lógica ha sido implementada como una librería de funciones (.dll) o Class Library de Visual Studio.NET, también conocida como “ensamblado”. El conjunto de estas librerías forman una solución de Visual Studio.NET denominada PRYECOMMERCE.

A continuación se detallan cada una de las capas de la solución:

- **Capa Web**

La capa Web está formada por formularios Web Forms ASP.NET y archivos de código de niveles inferiores. Esta capa es la que permite que el usuario de Internet acceda al sitio Web. Esta capa esta implementada como proyecto con el nombre “Web” dentro del archivo de solución Pryecommerce.sln. Con los formularios Web Forms, el usuario funciona en HTML, mientras que los archivos de código implementan el control de eventos para los distintos controles.

- **Capa CapaNegocio**

La capa CapaNegocio funciona como una capa de aislamiento que separa la interfaz del usuario de la lógica implementada para realizar las distintas funciones del negocio. La capa CapaNegocio posee las interfaces hacia la capa Web que permiten el control de cuentas de usuario, la búsqueda por categorías y la compra de productos. Esta capa esta implementada como proyecto con el nombre “CapaNegocio” dentro del archivo de solución Pryecommerce.sln.

Todas las llamadas a la bases de datos que se generan en nuestro proyecto se realizan a través de este ensamblado.

- **Capa ReglasNegocio**

La capa ReglasNegocio contiene la implementación de las reglas y lógica del negocio tales como la validación de las cuentas de usuario y los pedidos de productos realizados en el sitio.

La capa ReglasNegocio está implementada como proyecto con el nombre “CapaNegocio” dentro del archivo de solución Pryecommerce.sln.

- **Capa AccesoDatos**

La capa AccesoDatos es la encargada de brindar todos los servicios de datos para la capa ReglasNegocio. Esta capa está implementada como proyecto con el nombre “AccesoDatos” dentro del archivo de solución Pryecommerce.sln.

- **Capa SystemFramework**

La capa SystemFramework proporciona las funciones necesarias para cargar las configuraciones iniciales del sitio, es decir las variables iniciales del sistema. Adicionalmente esta capa se encarga de habilitar el rastreo y el registro de eventos durante el funcionamiento del sitio. Esta capa se encuentra implementada como proyecto con el nombre “SystemFramework” dentro del archivo de solución Pryecommerce.sln.

La implementación de esta capa se realiza siguiendo las recomendaciones encontradas en la metodología de desarrollo de aplicaciones Web.

- **Capa Común**

La capa Común contiene cierta lógica que se utiliza para pasar información entre las distintas capas. La capa Común contiene también clases necesarias para la configuración y el seguimiento de las aplicaciones, realizado por la capa SystemFramework. La capa Común está implementada como proyecto con el nombre “Común” dentro del archivo de solución Pryecommerce.sln.

A continuación se muestra el diagrama de la arquitectura del sitio.

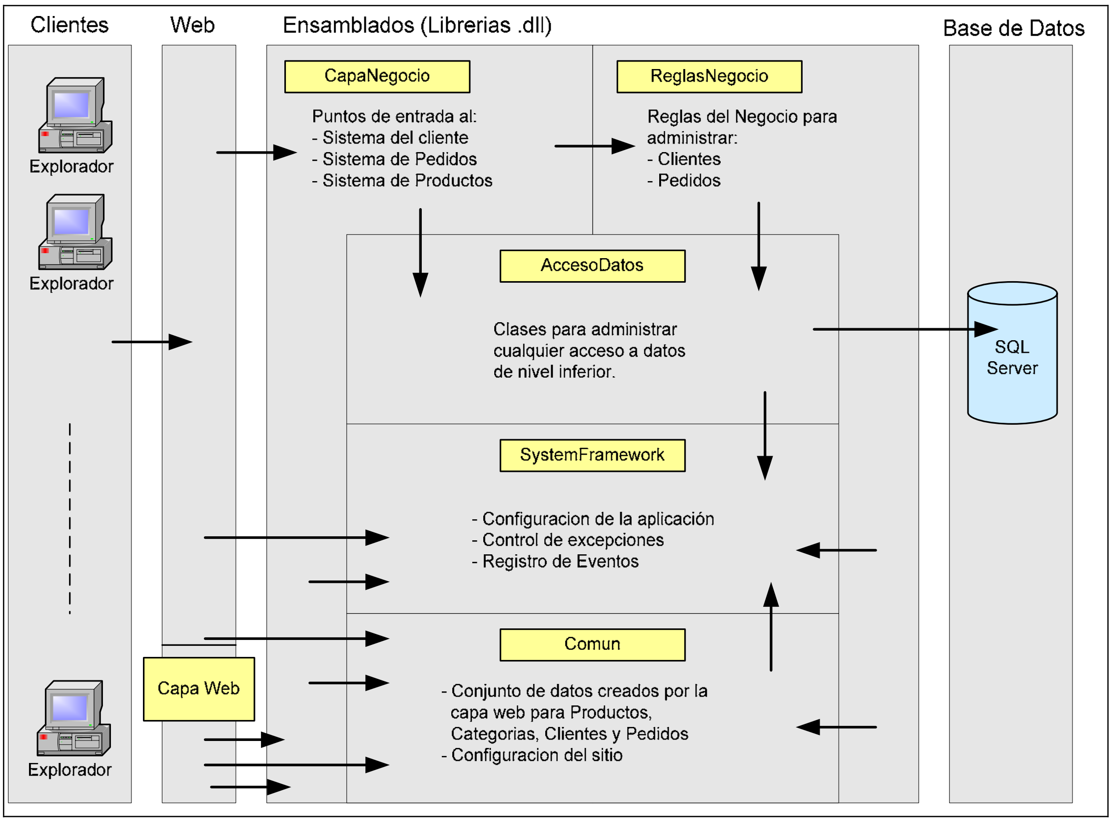


Figura 5.1 Diagrama de la Arquitectura del Sitio Web

Hemos incluido el diagrama de la estructura estática del proyecto en el Apéndice D de este documento.

5.2.3 Implementación

Para realizar la implementación del sitio Web hemos analizado y esquematizado los procesos de nuestro negocio, identificando todos los posibles casos de uso que pueden existir en el momento de la venta de regalos.

Los distintos casos de uso corresponden a las posibles acciones que un usuario puede tomar durante la interacción con el sitio y la manera en que el sistema deberá responderle.

A continuación se incluyen los diagramas del proceso de venta de regalos y de los distintos casos de uso utilizando la metodología UML para la representación del problema.

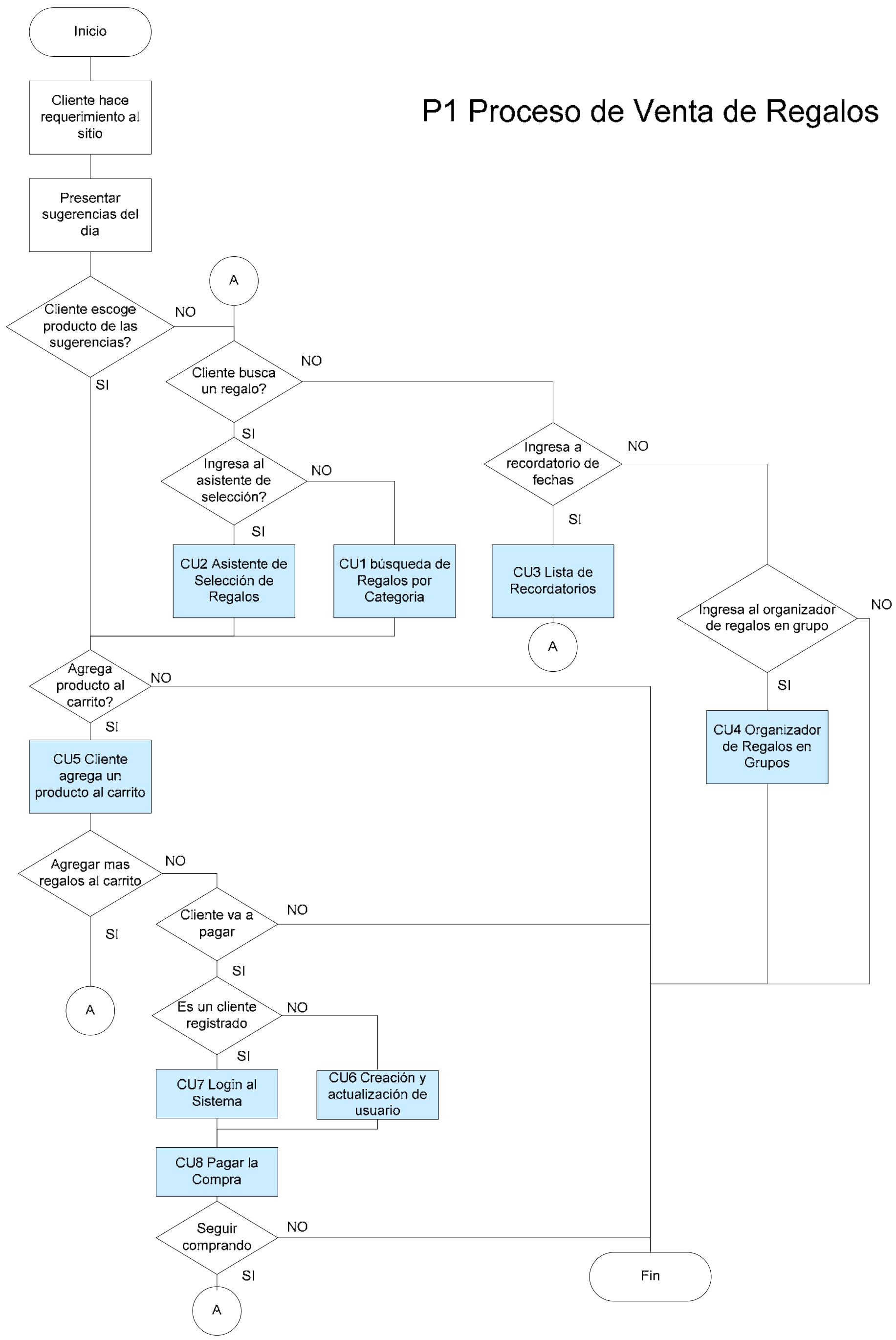


Figura 5.2 Proceso de Venta de Regalos

CU1 Búsqueda de Regalos

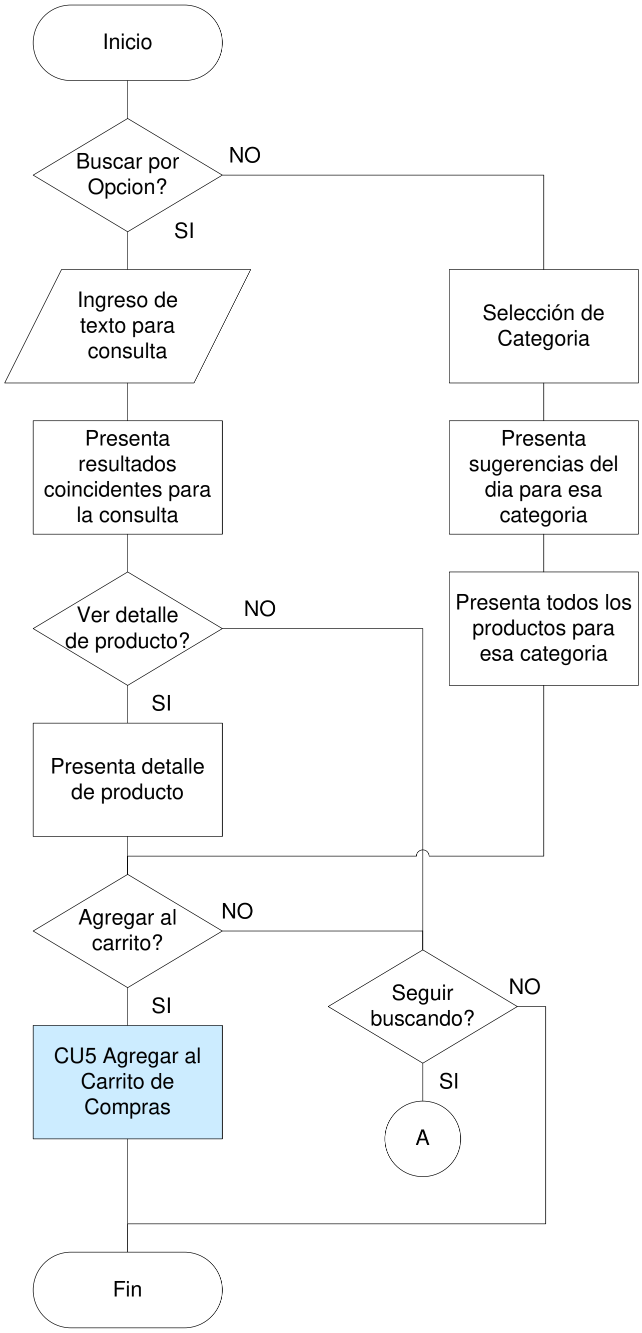


Figura 5.3 CU1 Búsqueda de Regalos

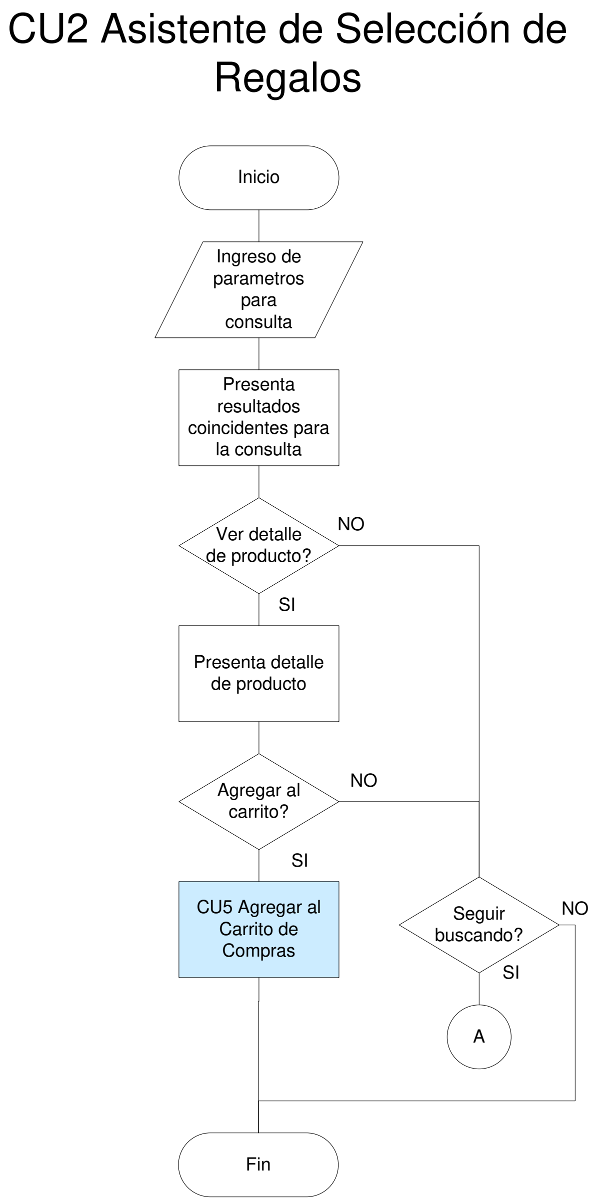


Figura 5.4 CU2 Asistente de Selección de Regalos

CU3 Lista de Recordatorios

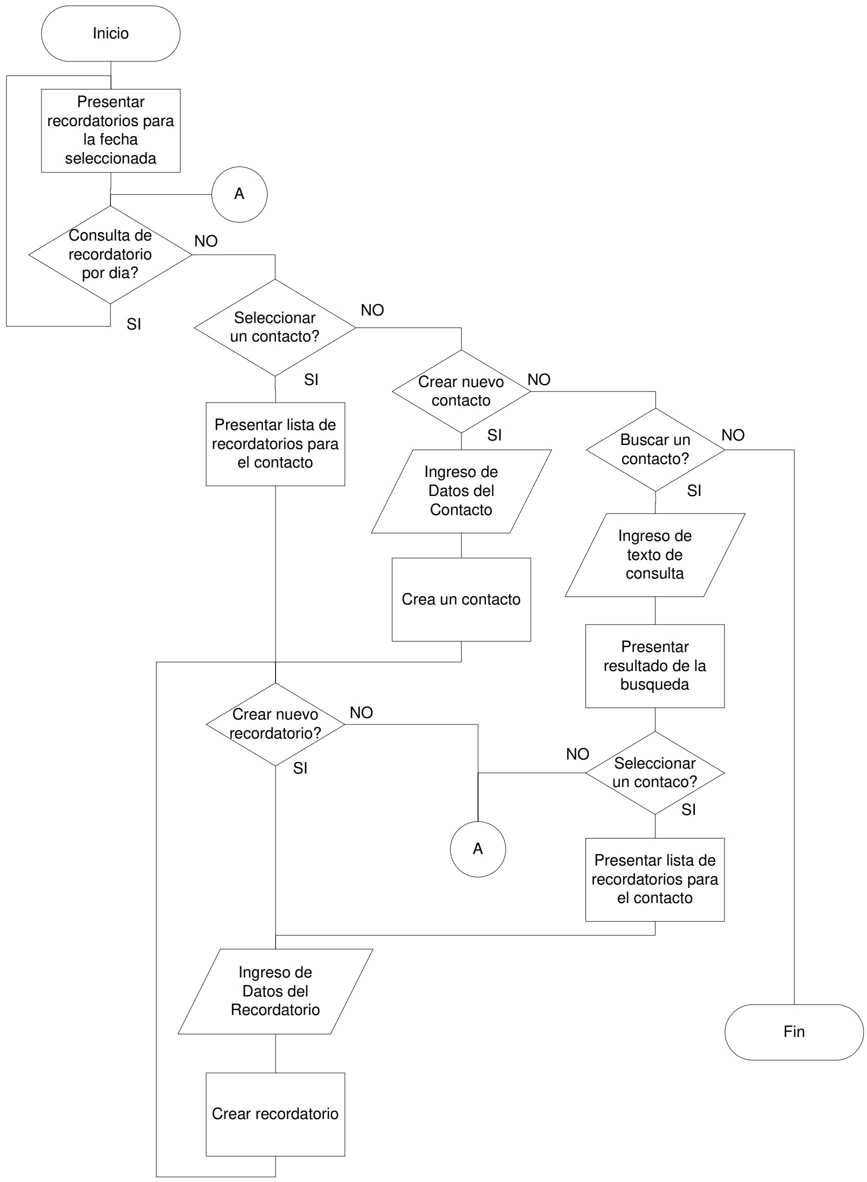


Figura 5.5 CU3 Lista de Recordatorios

CU4 Organizador de Regalos en Grupo

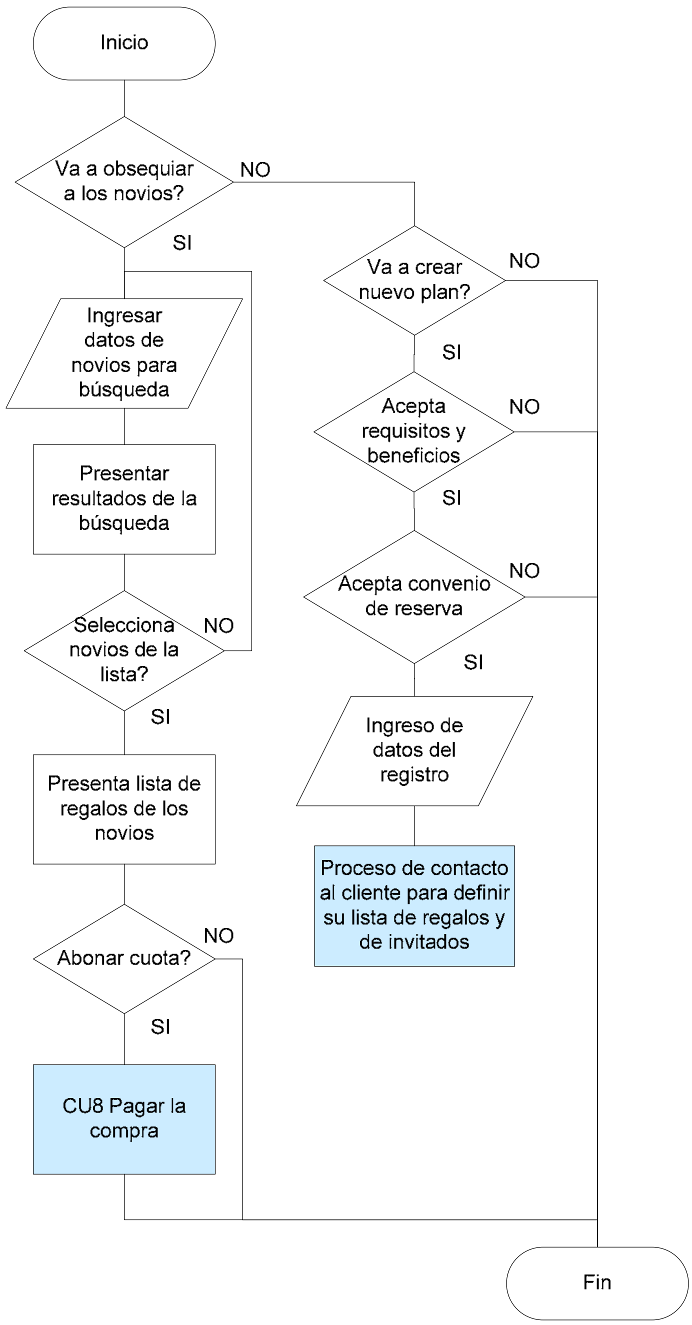


Figura 5.6 CU4 Organizador Regalos en Grupo

CU5 Carrito de Compras

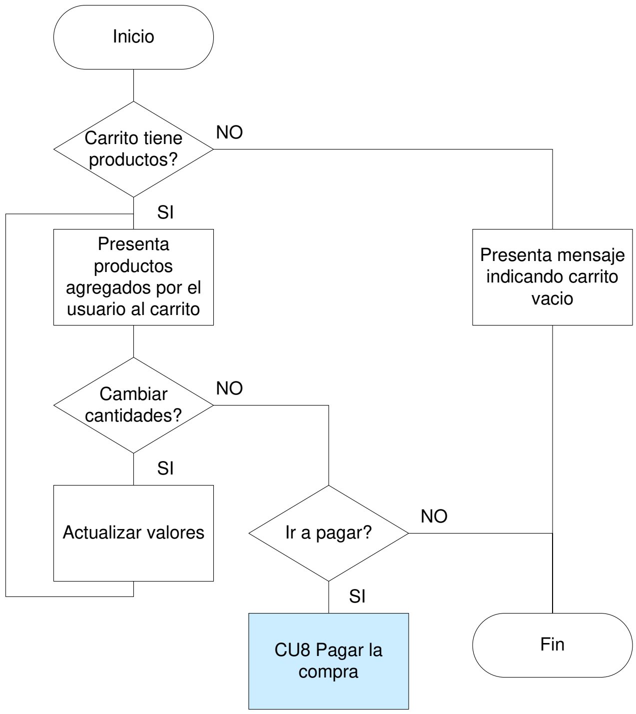


Figura 5.7 CU5 Organizador Regalos en Grupo

CU6 Creación y Actualización datos
de Usuario

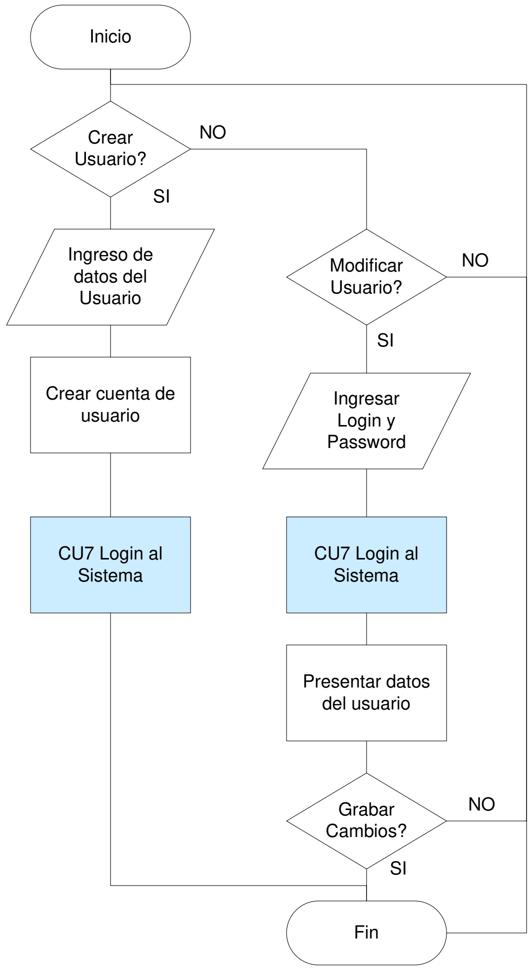


Figura 5.8 CU6 Creación y Actualización Datos Usuarios

CU7 Login al Sistema

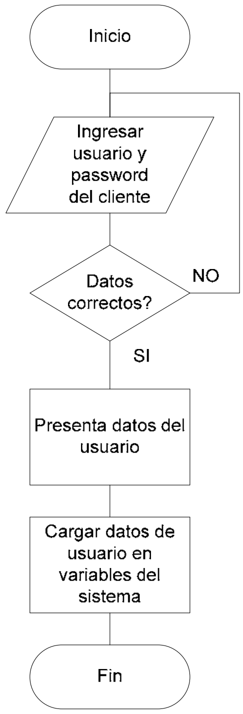


Figura 5.9 CU7 Login al Sistema

CU8 Pagar la Compra

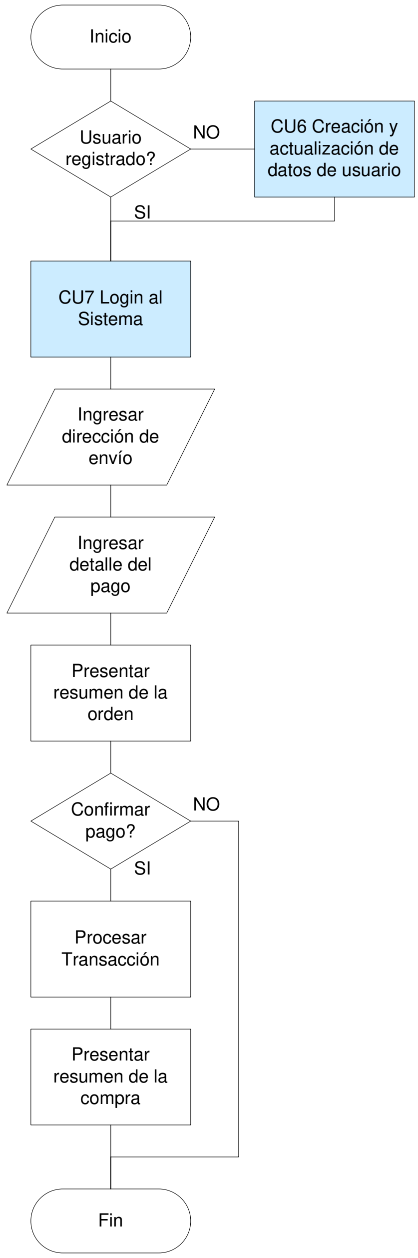


Figura 5.10 CU8 Pagar la Compra

Los diagramas de secuencia y los diagramas de clases que se derivan de los distintos casos de uso han sido incluidos en el Apéndice D.

5.3 Selección de la Plataforma de desarrollo y ejecución del aplicativo

Antes de comenzar a desarrollar el sitio de compras hemos realizado un análisis de las herramientas de desarrollo que se orientaban más hacia el tipo de solución que queríamos implementar.

Como parte de ese análisis hemos incluido una breve explicación de cada una de las tecnologías que se utilizaron.

5.3.1 .NET Framework

.NET Framework es el fundamento y marco de trabajo de Windows para la creación y ejecución de una nueva generación de software y servicios Web. Está formada por 2 componentes principales, “Common Language Runtime” y la “Biblioteca de clases de .NET Framework”.

El “Common Language Runtime” también conocido como Motor de tiempo de ejecución, es un agente que administra el código durante la ejecución del software y provee algunos servicios como la administración de memoria y la administración de subprocesos.

La biblioteca de clases de .NET Framework como su nombre lo indica es el conjunto de clases predefinidas que proveen tipos de objetos reutilizables y con los cuales se pueden desarrollar cualquier clase de aplicación, ya sea una interfaz de línea de comando o una aplicación innovadora que utilice ASP.NET, WebForms o Web XML.

5.3.2 ASP.NET

ASP.NET proporciona un modelo de desarrollo Web que es una mejora de la tecnología ASP (Active Server Page) y que permite la creación de aplicaciones más escalables y estables, desarrollándolas en cualquier lenguaje compatible con .NET como Visual Basic .NET, C#, JScript.NET y la biblioteca de clases de .NET Framework.

Otra de las características de ASP.NET es que permite el desarrollo utilizando formularios WebForms y Web XML facilitando el uso de esquemas de autenticación y el almacenamiento en cache de datos frecuentemente usados.

ASP.NET brinda también muchas facilidades para el acceso a bases de datos desde aplicaciones desarrolladas con esta tecnología, siendo esta una de las principales razones de su constante utilización en aplicaciones de comercio electrónico.

5.3.3 C#

Microsoft C# es un nuevo lenguaje de programación que ha evolucionado de Microsoft C y Microsoft C++ y permite crear un amplio número de aplicaciones que se ejecutan en .NET Framework.

C# facilita la llamada a subprocesos almacenados en librerías dinámicas (dll's) que pueden interactuar con ASP.NET y con .NET Framework.

Una de las ventajas principales de C# es que permite crear porciones del aplicativo utilizando la misma interface de desarrollo conocida en las versiones anteriores para empaquetar porciones de código que serán llamadas con total compatibilidad desde las páginas ASP.NET.

5.3.4 ADO.NET

La tecnología ADO.NET provee las facilidades necesarias para el acceso a datos contenidos en Microsoft SQL Server y en orígenes de datos como OLE DB y XML, permitiendo a las aplicaciones conectarse a las bases de datos para recuperar, manipular y actualizar la información.

ADO.NET separa el acceso a datos de la manipulación y crea componentes que se pueden usar por separado o conjuntamente. Los resultados del acceso o la manipulación de datos se procesan directamente o pueden colocarse en un objeto conocido como “DataSet” de ADO.NET para finalmente exponerlos al usuario con un propósito específico.

5.4 Modelo de Datos usado

El modelo de datos que hemos usado para almacenar la información es el Modelo de Entidad – Relación, ya que aunque el sitio en su totalidad se basa en el paradigma orientado a objetos, la base de datos no se fundamenta en esta tecnología por múltiples razones que se escapan del alcance de este documento.

Detallamos a continuación el modelo de datos Entidad Relación utilizado en la implementación de la solución.

5.4.1 Entidades de la Base de Datos

El diagrama entidad relacion de nuestra base de datos ha sido incluido en el Apéndice D de este documento. A continuación se explican cada una de las entidades diseñadas.

Customers

Guarda los datos de los clientes.

Addresses

Almacena la dirección del cliente.

Categories

Guarda las categorías de los productos como las siguientes:

- Aniversarios
- Cumpleaños
- Graduación
- Matrimonios
- Sentimientos

ItemCategory

Guarda la clasificación de cada ítem.

Items

Guarda los datos de los ítems de venta.

ItemType

Guarda el tipo de ítems.

ContactReminder

Almacena los datos a recordar de los contactos del cliente.

DailyPick

Registra los ítems que son sugeridos en la pantalla principal.

Dealers

Guarda los datos de los proveedores de los ítems de venta.

OrderItems

Guarda el detalle de códigos y cantidades de los productos de una orden o venta.

Orders

Guarda los datos de la cabecera de la orden o venta.

Origins

Guarda la información de los orígenes de los productos.

ProductOrigin

Almacena la información de origen de los productos.

Products

Guarda la información de los productos como los campos de origen, proveedor, breve descripción, etc.

RelationAsistant

Guarda la información de las combinaciones de los ítems con las clasificaciones establecidas.

GroupsGifs

Guarda la información de los regalos en grupos generados por los clientes de este servicio.

Capítulo 6

6 CONCLUSIONES Y RECOMENDACIONES

1. La mayor barrera para una empresa que pretende comercializar productos a través de la Internet es el grado de confianza que actualmente tiene el mercado en este tipo de negocios. Ya sea por el miedo a ser estafado por el sitio o a que un hacker obtenga su información personal y tarjeta de crédito a través de un ataque al sitio, el mercado se abstiene de realizar compras en línea a pesar de la comodidad y facilidad que esta actividad ofrece al comprador.

   El ser humano es desconfiado por naturaleza, una frase común es "tener temor a lo desconocido" por consiguiente la mayoría de los usuarios tiene desconfianza de la tecnología y necesita más que un candadito en la parte inferior de la pantalla para confiar en una empresa con la que nunca ha hecho negocios y a la que no conoce.

2. Ya que el mercado exige confianza y seguridad debemos escuchar esta demanda y enfocar nuestra estrategia en estos importantes aspectos.

   La única manera de vencer este paradigma es mejorando los niveles de seguridad que se ofrecen al comprador y dando a conocer la inversión que se hace en recursos y conocimientos con el fin de que las transacciones sean seguras.

   Muchas empresas de comercio electrónico no han tenido el éxito que esperaban debido a centrar su estrategia en el mercadeo de sus productos y no en la seguridad; y el resultado es un fracaso debido al miedo del comprador, inseguridad del sitio y errores tecnológicos de la empresa.

3. La seguridad es el aspecto más importante en un negocio de comercio electrónico pero no es el único, también es necesario contar con un buen diseño gráfico del sitio y con herramientas de ayuda al usuario. Es por esto que el equipo de trabajo de la empresa deberá tener a la seguridad como un parte integral durante el diseño e implementación de las páginas y de la red, pero además deberá tener buenos conocimientos sobre programación e interacción hombre – máquina.

4. Durante el desarrollo de este trabajo hemos intentado maximizar la seguridad de la infraestructura de red utilizando herramientas confiables y acordes a los requerimientos de nuestro proyecto, sin embargo la seguridad nunca será absoluta. Todos los elementos de seguridad como los firewalls, servidores Web, sistemas de detección de intrusos, bases de datos, antivirus y directorio de red son propensos a sufrir ataques malintencionados y nuestra tarea constante será evitarlos, detectarlos y corregirlos.

5. El desarrollo de la solución fue guiado por la metodología de desarrollo de Microsoft, bajo las recomendaciones de arquitectura de capas lógicas que dividen la funcionalidad de usuario de la implementación.

   En la actualidad existen varios Frameworks (Marcos de Trabajo) de desarrollo rápido como lo son “EDRA”, “We Fly”, “We Rock”, “IDEA BLADE”, etc., la selección de algunos de ellos dependerá mucho del tipo y alcance del proyecto. Microsoft ha tomado la iniciativa de difundir su tecnología para desarrollo web, y ha liberado bastante material para la rápida construcción de soluciones funcionales, escalables, de alto rendimiento y de elevada interoperabilidad.

6. Para la construcción del sitio decidimos usar esta metodología pues era la más viable y la que más se ajustaba al tipo de proyecto académico que se nos planteó, pues debemos recordar que el presente trabajo de software es un componente secundario del proyecto ya que el objetivo del Tópico tenía mas relación con los esquemas de seguridad de redes aplicados al comercio electrónico.

   Este proyecto queda abierto a posteriores desarrollos que complementen su funcionalidad, donde en su diseño mismo se puede observar que puede servir como marco referencial a futuros proyectos sobre tecnología de desarrollo web utilizando las herramientas de Microsoft .Net. Se deja abierto a un abanico de posibilidades como son los Servicios Web (WS y Enhancement WS 2.0), y toda la tecnología de Indigo que está próxima a liberarse con la nueva plataforma Windows Longhorn.

7. La recomendación más importante para toda empresa que quiera incursar en el comercio electrónico es no sólo realizar un diseño e implementación segura sino planificar un mantenimiento continuo de la seguridad. Esto significa investigar día a día nuevas vulnerabilidades, mantener contacto permanente con los fabricantes de software y sus proveedores y crear una cultura de seguridad en la empresa de tal manera que toda actividad realizada por el equipo de

trabajo tenga siempre presente los principios básicos de la seguridad informática.

APENDICE A

ENCUESTA DE MERCADO

1- Ha realizado compras a través de Internet alguna vez?

sì ☐ no ☐

2- Si respondió SI a la pregunta 1, indique con que frecuencia realiza compras por Internet?

____

3- Si respondió SI a la pregunta 1, indique en que 2 sitios en los que haya comprado?

____

4- Cuando compra por Internet, en qué se fija?
(anotar las dos primeras respuestas, en el orden en que sean dadas)

- ☐ En el precio
- ☐ En la seguridad del sitio
- ☐ En el tiempo de entrega
- ☐ En la facilidad de navegación del sitio
- ☐ En la variedad de articulos de venta
- ☐ Otra variable. Cuál? ________

5- Compraría regalos por Internet?

sì ☐ no ☐

6- Si respondió SI a la pregunta 5, Que tipo de regalos compraría en un sitio de e-commerce Ecuatoriano?

- ☐ Cumpleaños
- ☐ Aniversario
- ☐ Graduación
- ☐ Navidad
- ☐ Dia de la Madre/Padre
- ☐ Otra variación. Cuál? ________

7- Si respondió NO a la pregunta 5, Indique por que no compraría un regalo por Internet?

- ☐ Ha tenido malas experiencias
- ☐ No confia en la seguridad de las transacciones
- ☐ Prefiere recorrer los centros comerciales
- ☐ No tiene acceso a Internet
- ☐ Realiza sus compras en efectivo
- ☐ Otra variación. Cuál? ________

8- Que beneficio lograría usted si comprara un regalo por Internet?

- ☐ Facilidad de búsqueda
- ☐ Comodidad
- ☐ Facilidad de envío/entrega
- ☐ Mejores precios
- ☐ Asesoría en la selección del regalo
- ☐ Otra variación. Cuál? ________

9- Qué nombre le daría a un sitio de venta de regalos en Internet?

____

Resultados de la Encuesta de Mercado - Preguntas 1 al 6

				Aspectos mas importantes al comprar							Tipo de regalos que compraria					
	Ha realizado compras en Internet	Frecuencia con que compra en Internet	Sitios en que ha comprado	Precio	Seguridad del Sitio	Tiempo de Entrega	Facilidad de Navegacion del sitio	Variedad de articulos de venta	Otro	Compraria regalos por internet	Cumpleaños	Aniversario	Graduacion	Navidad	Dia de la Madre - Padre	Otros
1	1	cada 3 meses	No recuerda	1	1					1	1	1		1		
2	0									1	1			1		
3	0									1	1					celebracion
4	0			1				1		1	1					
5	1	2 veces / año	amazon			1		1		1				1		
6	1	esporadico	plantretail paquetevote		1					1	1				1	
7	1	cada mes	amazon petco	1	1					1	1	1		1	1	
8	1	casi nunca	airfrance alitalia	1	1					0						
9	1	hace un año	www.creative.com					1		0						
10	0						1			1	1			1		
11	0				1					1				1		
12	0									1	1					
13	0									1					1	
14	0									1	1					
15	0									1	1				1	
16	0									1				1	1	
17	1	2 veces / año	amazon	1	1					1	1			1		
18	1	una sola vez	No recuerda					1		1		1			1	
19	0			1						1	1	1				
20	1	esporadico	braganca			1				1	1				1	
21	0			1	1					0						
22	0			1	1					0						
23	1	1 vez / año	bestbuy.com	1	1			1		1	1			1	1	
24	1	poco	ebay	1			1			0						
25	0			1		1				0						
26	0				1					0						
27	0						1	1		1		1		1		
28	1	cada 4 meses	deremate.com	1	1					1	1	1				
29	0									0						
30	1	una sola vez	braganca		1	1				0						
Si	43%			40%	40%	13%	10%	20%	0%	70%	50%	20%	0%	33%	27%	0%

Resultados de la Encuesta de Mercado - Preguntas 7 al 9

	Por que no compraria regalo por Internet						Beneficio que lograria comprando un regalo en Internet						
	Mala experiencia anterior	No confia en la seguridad	Prefiere ir al centro comercial	No tiene internet en casa	Comprar en efectivo	Otro	Facilidad de Busqueda	Comodidad	Facilidad de Envio - Entrega	Mejores Precios	Asesoria al comprar regalos	Otro	Nombre que le daria a un sitio de venta de regalos
1							1						the-gift.com
2								1	1				www.obsequios.com
3													cajitadesorpresas
4								1	1				www.buscando.com
5								1					amazon
6								1	1				regalink.com
7								1		1			www.queteregalo.com
8			1									Articulos que regularmente no se encuentran en el mercado local	www.regalaseguro.com
9	1						1						www.altavista.com
10								1	1				easy-purchase.com
11								1	1				www.comprame.com
12							1	1	1				www.buyhouse.com
13							1						turegalo.com
14							1	1					www.comprafacil.com
15							1	1	1				www.turegalo.com
16								1		1			www.oportunidadeahorro.com
17							1	1					www.regaladito.com
18								1		1			campraselo.com
19		1						1					deregalo.com
20							1			1			gifts.com
21			1										
22													quetecompro
23								1	1				elregalo.com
24							1	1	1				www.regalos.com
25				1					1				
26					1								
27													www.aniversario.com
28								1	1				floreriaonline
29							1						
30								1		1			www.ocasiones.com
Si	3%	3%	7%	3%	3%	0%	33%	60%	37%	17%	0%		

APENDICE B

POLITICAS DEL FIREWALL EXTERNO

IP Origen	IP Destino	Puerto	Acción
0.0.0.0	192.168.30.2	TCP 80	Allow
0.0.0.0	192.168.30.2	TCP 443	Allow
0.0.0.0	192.168.10.3	TCP 25	Allow
192.168.30.2	192.168.10.4	TCP 1433	Allow
192.168.30.2	192.168.10.3	TCP 25	Allow
192.168.10.0	192.168.30.2	TCP 80	Allow
192.168.10.0	192.168.30.2	TCP 443	Allow
192.168.10.0	0.0.0.0	TCP 80	Allow
192.168.10.0	0.0.0.0	TCP 443	Allow
192.168.10.2	0.0.0.0	TCP 21	Allow
Administradores	0.0.0.0	TCP 21	Allow
192.168.10.3	0.0.0.0	TCP 25	Allow
192.168.10.0	0.0.0.0	TCP 110	Allow
192.168.10.2	0.0.0.0	TCP 53 UDP 53	Allow

POLITICAS DEL FIREWALL INTERNO

IP Origen	IP Destino	Puerto	Acción
192.168.10.1	192.168.20.2	TCP 1433	Allow
Administradores	192.168.20.2	TCP 1433	Allow

DIRECCIONES IP ASIGNADAS A LOS SERVIDORES DE LA RED IMPLEMENTADA

Servidor	Nombre	Dirección IP	
Firewall Externo	SRVFW01	Interface Externa	192.168.0.2
		Interface DMZ	192.168.30.1
		Interfase Interna	192.168.10.1
Firewall Externo	SRVFW02	Interfase Externa	192.168.10.4
		Interfase Interna	192.168.20.1
Controlador de Dominio	SRVDC01	Interfase Única	192.168.10.2
Servidor SMTP Sendmail	SRVMAIL	Interfase Única	192.168.10.3
Base de Datos	SRVDB01	Interfase Única	192.168.20.2
Servidor Web	SRVWEB01	Interfase Única	192.168.30.2
Servidor de Detección de Intrusos	SRVIDS01	Interfase Única	192.168.10.5

APENDICE C

POLÍTICAS DE SEGURIDAD LOCAL – DERECHOS DE USUSARIO

Politica	Usuarios Autorizados
Access this computer from the network	Everyone, Administrators, Authenticated Users, REGALOS\IUSR_SRVDC01
Add workstations to domain	Authenticated Users
Back up files and directories	Administrators,Backup Operators,Server Operators
Bypass traverse checking	Everyone,Administrators,Authenticated Users
Change the system time	Administrators,Server Operators
Create a pagefile	Administrators
Create global objects	Administrators,SERVICE
Debug programs	Administrators
Enable computer and user accounts to be trusted for delegation	Administrators
Force shutdown from a remote system	Administrators,Server Operators
Impersonate a client after authentication	Administrators,SERVICE
Increase quotas	Administrators
Increase scheduling priority	Administrators
Load and unload device drivers	Administrators
Log on as a batch job	REGALOS\IUSR_SRVDC01
Log on locally	TsInternetUser,Administrators,Backup Operators,Account Operators,Server Operators,Print Operators,REGALOS\IUSR_SRVDC01
Manage auditing and security log	Administrators
Modify firmware environment values	Administrators
Profile single process	Administrators
Profile system performance	Administrators
Remove computer from docking station	Administrators
Restore files and directories	Administrators,Backup Operators,Server Operators
Shut down the system	Administrators,Backup Operators,Account Operators,Server Operators,Print Operators
Take ownership of files or other objects	Administrators

POLÍTICAS DE SEGURIDAD LOCAL – OPCIONES DE SEGURIDAD

Politica	Configuración
Additional restrictions for anonymous connections	No access without explicit anonymous permissions
Allow server operators to schedule tasks	Disabled
Allow system to be shut down without having to log on	Disabled
Allowed to eject removable NTFS media	Administrators
Amount of idle time required before disconnecting session	15 minutes
Audit the access of global system objects	Disabled
Audit use of Backup and Restore privilege	Disabled
Automatically log off users when logon time expires	Enabled
Automatically log off users when logon time expires (local)	Enabled
Clear virtual memory pagefile when system shuts down	Enabled
Digitally sign client communication (always)	Enabled
Digitally sign client communication (when possible)	Enabled
Digitally sign server communication (always)	Enabled
Digitally sign server communication (when possible)	Enabled
Disable CTRL+ALT+DEL requirement for logon	Disabled
Do not display last user name in logon screen	Enabled
LAN Manager Authentication Level	Send NTLMv2 response only\refuse LM & NTLM
Number of previous logons to cache in case DC not available	10 logons
Prevent system maintenance of computer account password	Disabled
Prevent users from installing printer drivers	Enabled
Prompt user to change password before expiration	14 days
Recovery Console: Allow automatic administrative logon	Disabled
Recovery Console: Allow floppy copy and access to all drives	Disabled
Rename administrator account	giftmanager
Restrict CD-ROM access to locally logged-on user only	Enabled
Restrict floppy access to locally logged-on user only	Enabled
Secure channel: Digitally encrypt or sign secure channel data	Enabled
Secure channel: Digitally encrypt secure channel data	Enabled
Secure channel: Digitally sign secure channel data	Enabled
Secure channel: Require strong session key	Enabled
Send unencrypted password to connect to third-party servers	Disabled
Shut down system immediately if unable to log security audits	Disabled
Smart card removal behavior	Force Logoff
Strengthen default permissions of global system objects	Enabled
Unsigned driver installation behavior	Do not allow installation
Unsigned non-driver installation behavior	Silently succeed

POLÍTICAS DE SEGURIDAD LOCAL – POLÍTICAS DE AUDITORIA

Politica	Configuración
Audit account logon events	Success, Failure
Audit account management	Success, Failure
Audit directory service access	Success, Failure
Audit logon events	Success, Failure
Audit object access	Success, Failure
Audit policy change	Success, Failure
Audit privilege use	Success, Failure
Audit process tracking	No auditing
Audit system events	Success, Failure

POLÍTICAS DE SEGURIDAD DE DOMINIO – POLÍTICAS DE CUENTAS

Políticas de Kerberos	**Configuración**
Enforce user logon restrictions	Enabled
Maximum lifetime for service ticket	600 minutes
Maximum lifetime for user ticket	10 hours
Maximum lifetime for user ticket renewal	7 days
Maximum tolerance for computer clock synchronization	5 minutes
Políticas de Bloqueos de Cuentas	**Configuración**
Account lockout duration	0
Account lockout threshold	5 invalid logon attempts
Reset account lockout counter after	30 minutes
Políticas de Passwords	**Configuración**
Enforce password history	24 passwords remembered
Maximum password age	42 days
Minimum password age	2 days
Minimum password length	8 characters
Passwords must meet complexity requirements	Enabled
Store password using reversible encryption for all users in the domain	Disabled

POLÍTICAS DE SEGURIDAD DE DOMINIO – OPCIONES DE SEGURIDAD

Política	Configuración
Additional restrictions for anonymous connections	No access without explicit anonymous permissions
Allow server operators to schedule tasks	Disabled
Allow system to be shut down without having to log on	Disabled
Allowed to eject removable NTFS media	Administrators
Amount of idle time required before disconnecting session	15 minutes
Audit the access of global system objects	Disabled
Audit use of Backup and Restore privilege	Disabled
Automatically log off users when logon time expires	Enabled
Automatically log off users when logon time expires (local)	Enabled
Clear virtual memory pagefile when system shuts down	Enabled
Digitally sign client communication (always)	Enabled
Digitally sign client communication (when possible)	Enabled
Digitally sign server communication (always)	Enabled
Digitally sign server communication (when possible)	Enabled
Disable CTRL+ALT+DEL requirement for logon	Disabled
Do not display last user name in logon screen	Enabled
LAN Manager Authentication Level	Send NTLMv2 response only\refuse LM & NTLM
Number of previous logons to cache in case DC not available	10 logons
Prevent system maintenance of computer account password	Disabled
Prevent users from installing printer drivers	Enabled
Prompt user to change password before expiration	14 days
Recovery Console: Allow automatic administrative logon	Disabled
Recovery Console: Allow floppy copy and access to all drives	Disabled
Rename administrator account	giftmanager
Restrict CD-ROM access to locally logged-on user only	Enabled
Restrict floppy access to locally logged-on user only	Enabled
Secure channel: Digitally encrypt or sign secure channel data	Enabled
Secure channel: Digitally encrypt secure channel data	Enabled
Secure channel: Digitally sign secure channel data	Enabled
Secure channel: Require strong session key	Enabled
Send unencrypted password to connect to third-party servers	Disabled
Shut down system immediately if unable to log security audits	Disabled
Smart card removal behavior	Force Logoff
Strengthen default permissions of global system objects	Enabled
Unsigned driver installation behavior	Do not allow installation
Unsigned non-driver installation behavior	Silently succeed

POLÍTICAS DE SEGURIDAD DE DOMINIO – POLÍTICAS DE AUDITORIA

Políticas	Configuración
Policy	Computer Setting
Audit account logon events	Success, Failure
Audit account management	Success, Failure
Audit directory service access	Success, Failure
Audit logon events	Success, Failure
Audit object access	Success, Failure
Audit policy change	Success, Failure
Audit privilege use	Success, Failure
Audit process tracking	No auditing
Audit system events	Success, Failure

POLÍTICAS DE SEGURIDAD DE DOMINIO – VISOR DE EVENTOS

Políticas	Configuración
Maximum application log size	Not defined
Maximum security log size	10240 kilobytes
Maximum system log size	2048 kilobytes
Restrict guest access to application log	Enabled
Restrict guest access to security log	Enabled
Restrict guest access to system log	Enabled
Retain application log	Not defined
Retain security log	Not defined
Retain system log	Not defined
Retention method for application log	As needed
Retention method for security log	As needed
Retention method for system log	As needed
Shut down the computer when the security audit log is full	Not defined

APENDICE D

ESTRUCTURA ESTÁTICA DEL PROYECTO REGALOS.COM

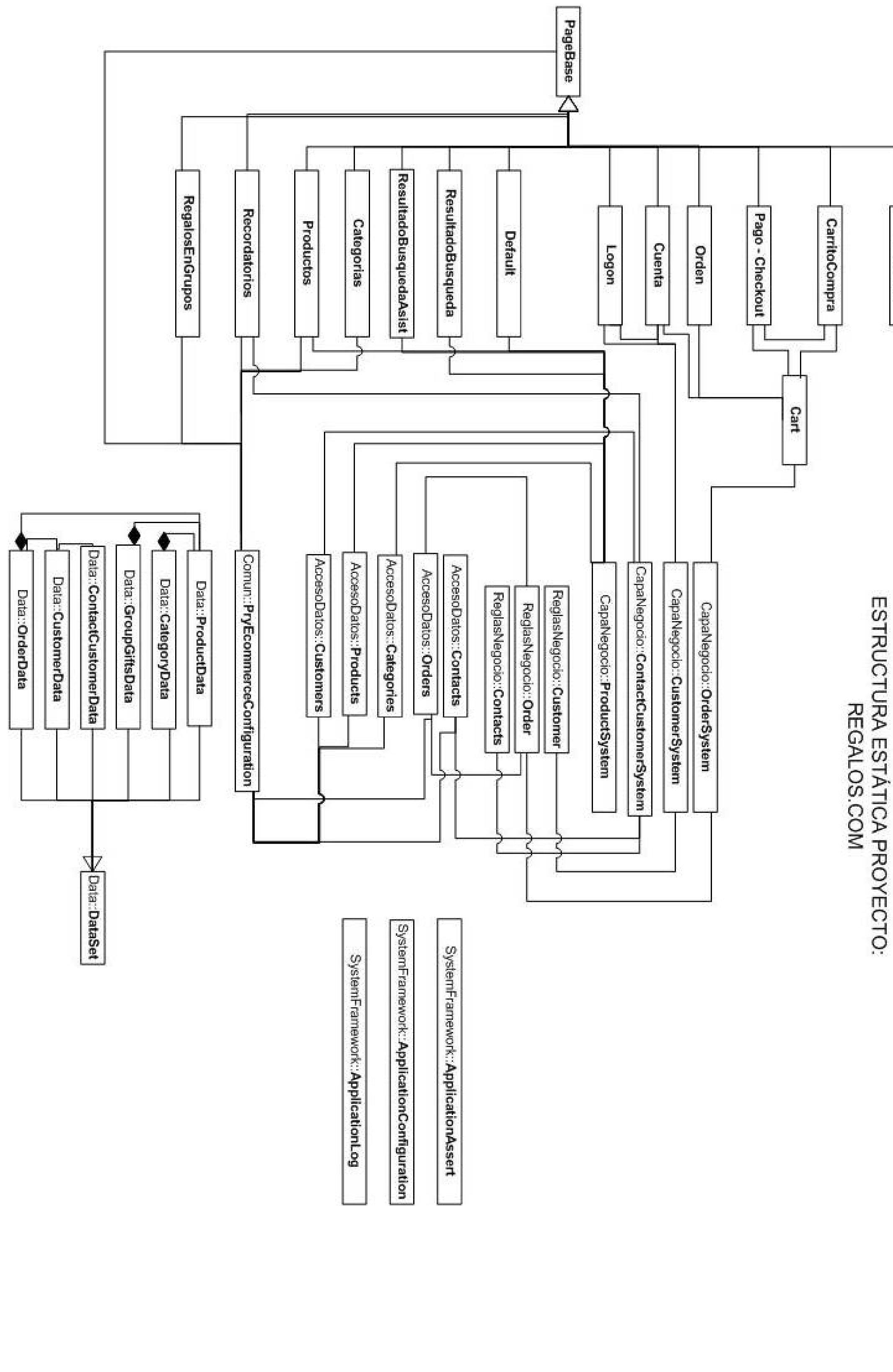
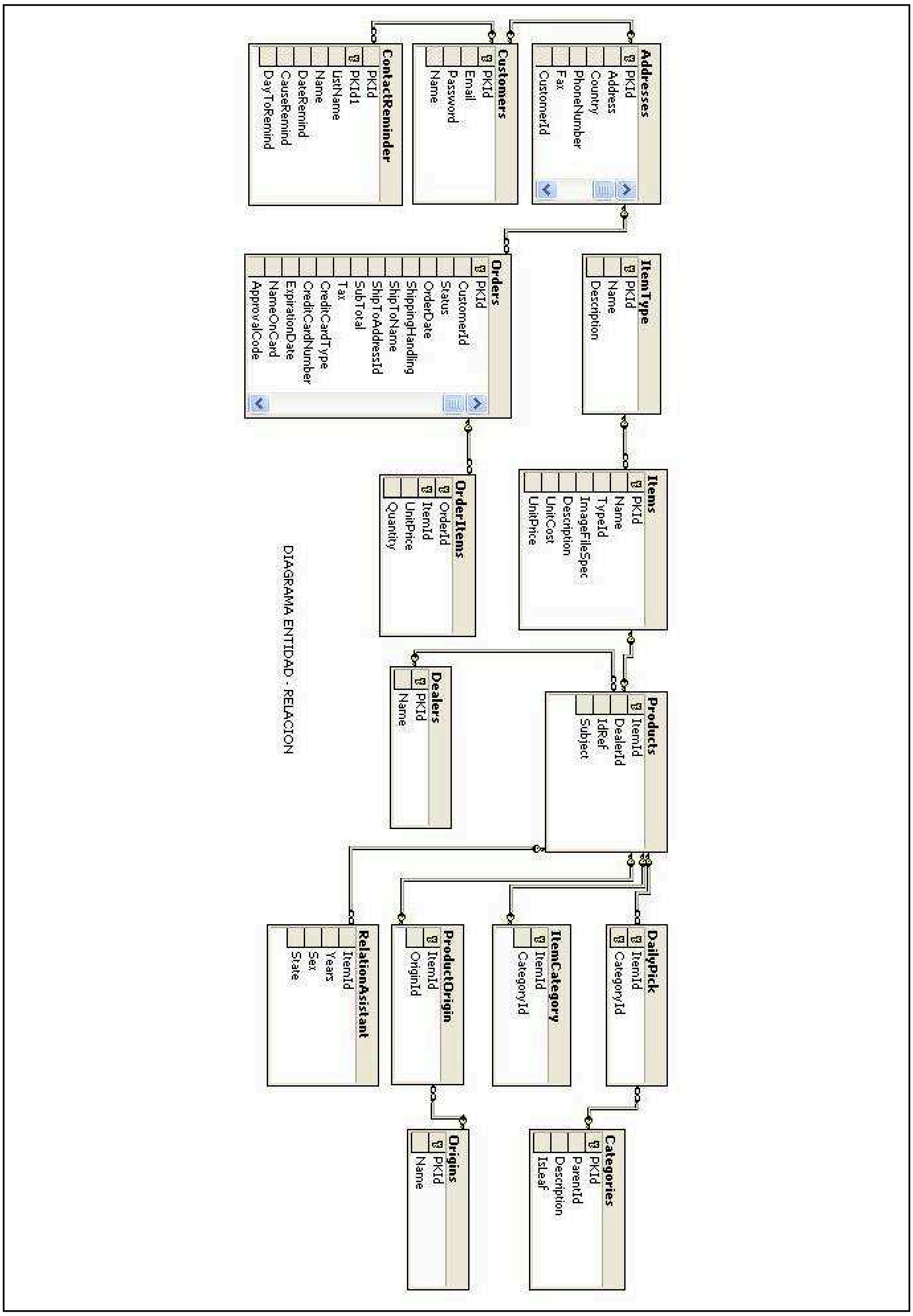
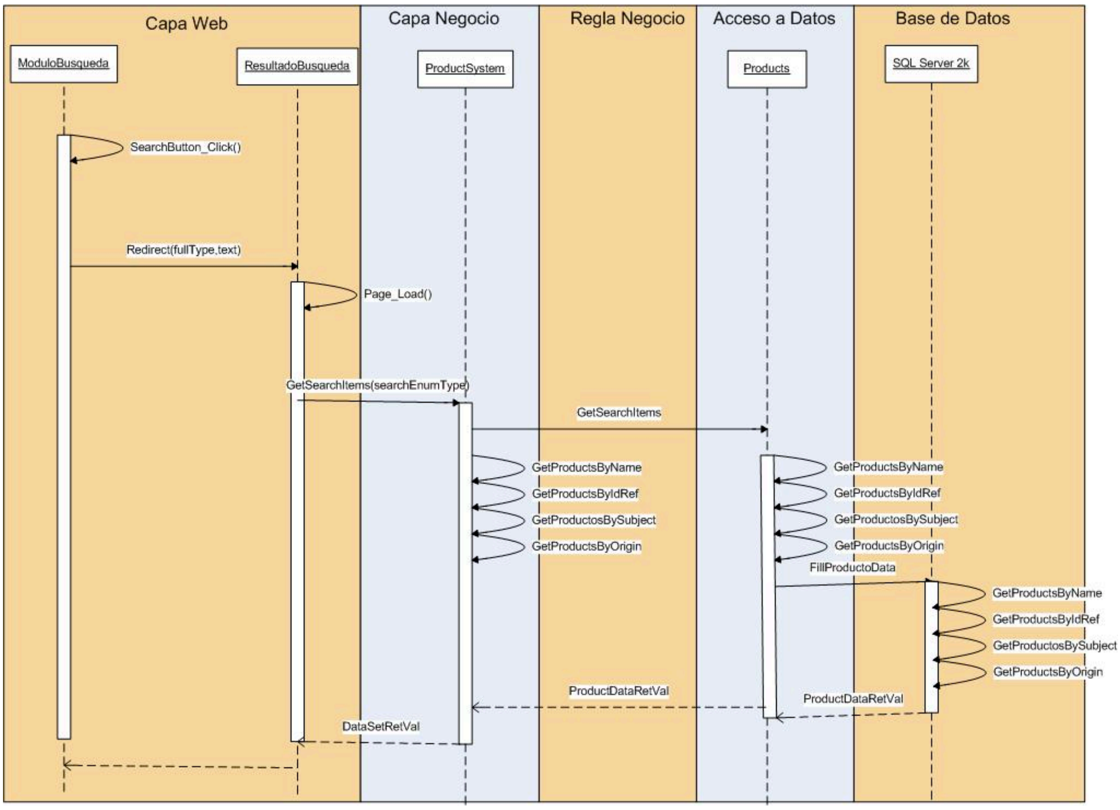


DIAGRAMA ENTIDAD RELACIÓN

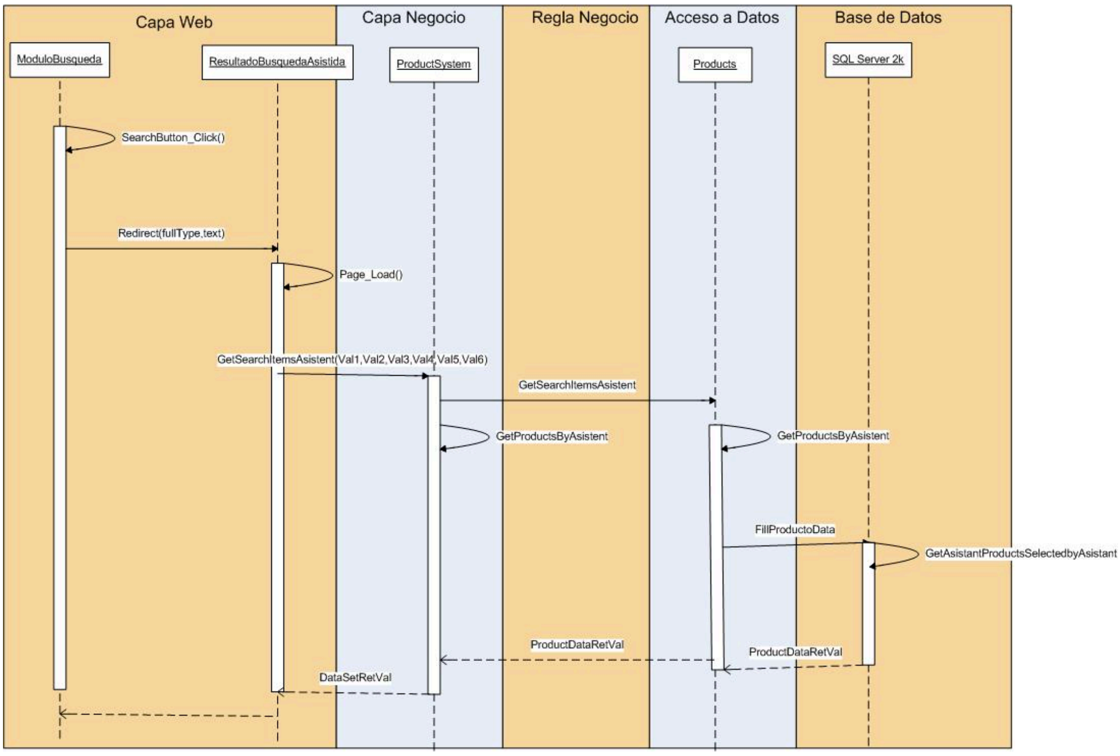


DIAGRAMAS DE SECUENCIA

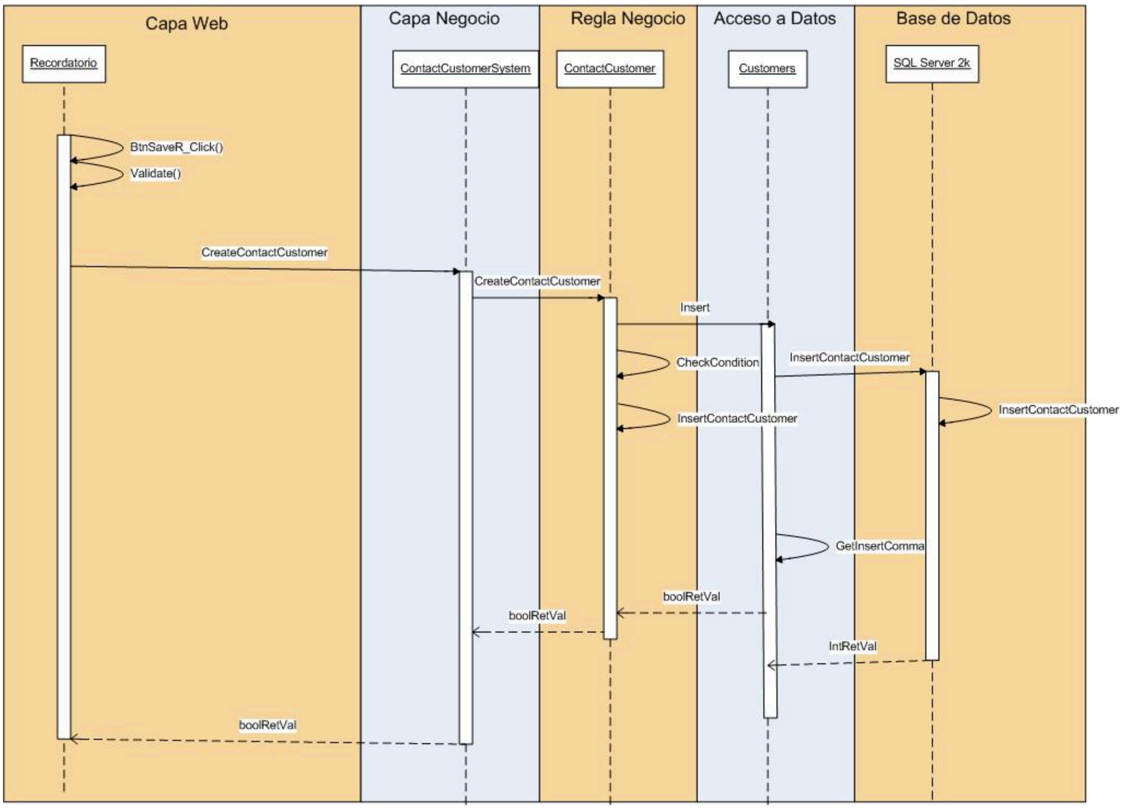
SECUENCIA BUSQUEDA DE REGALOS



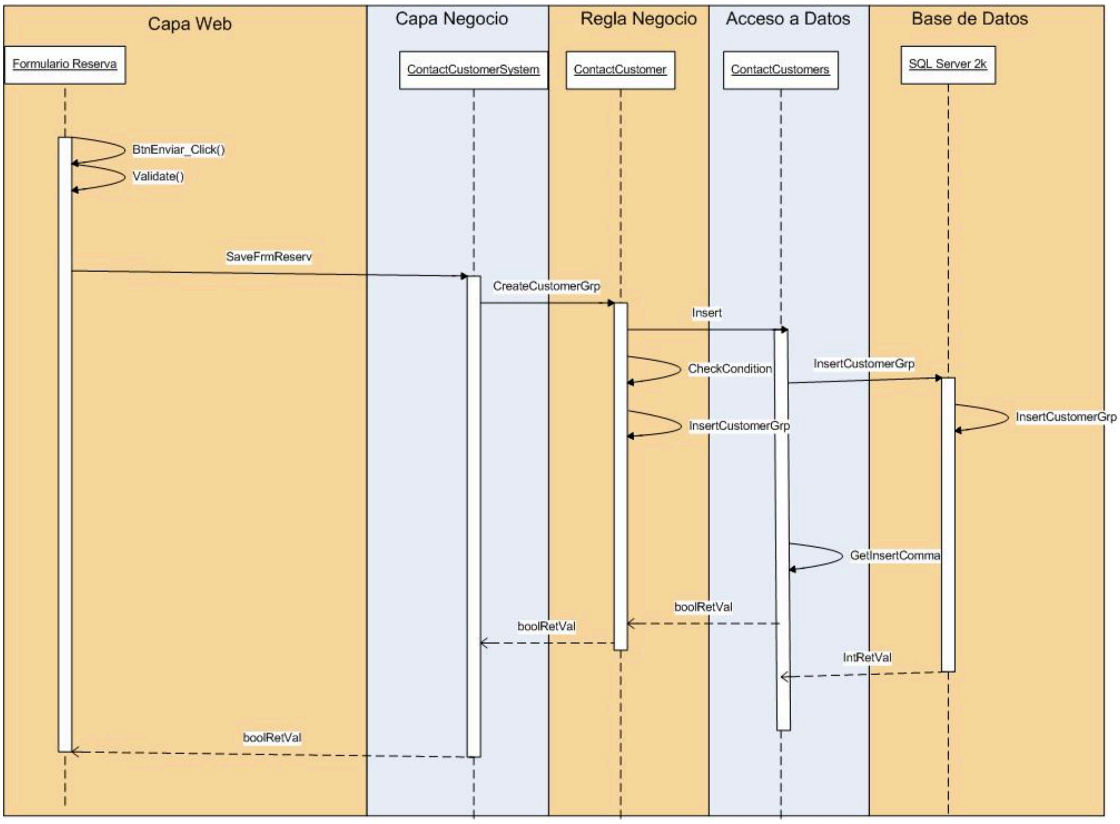
SECUENCIA ASISTENTE DE SELECCION DE REGALOS



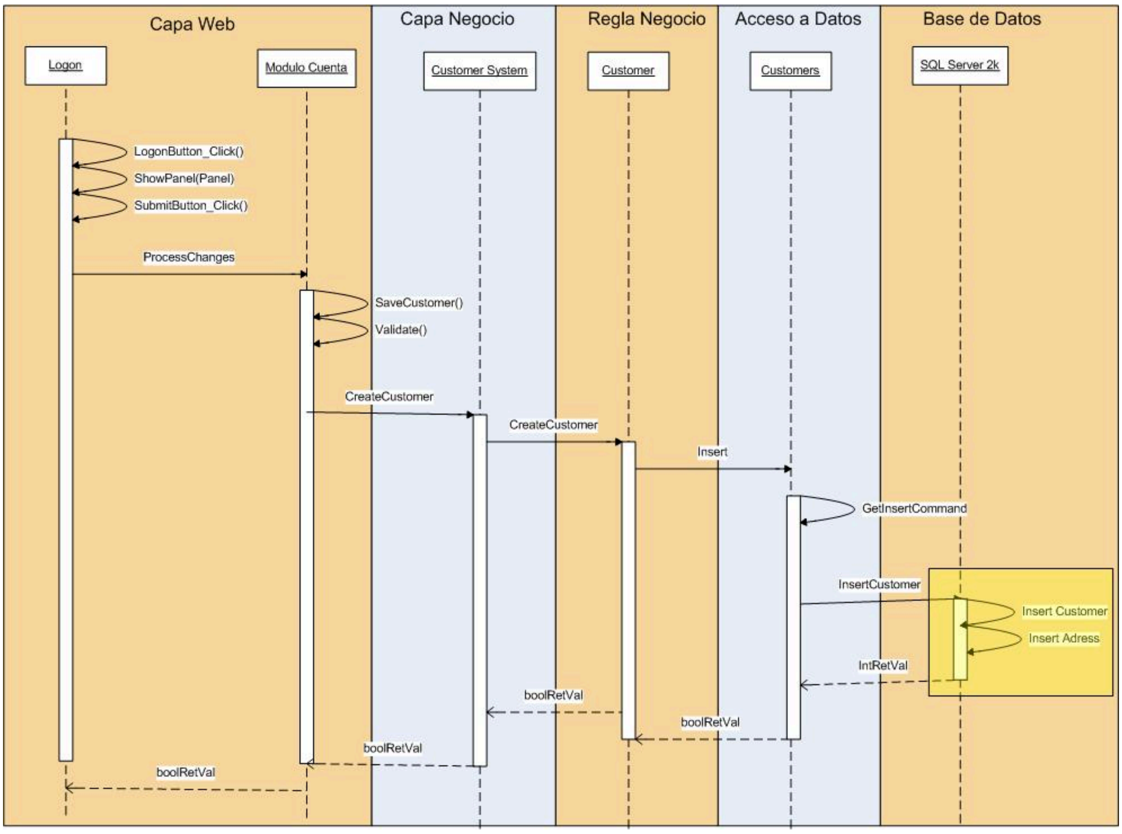
SECUENCIA LISTA DE RECORDATORIOS



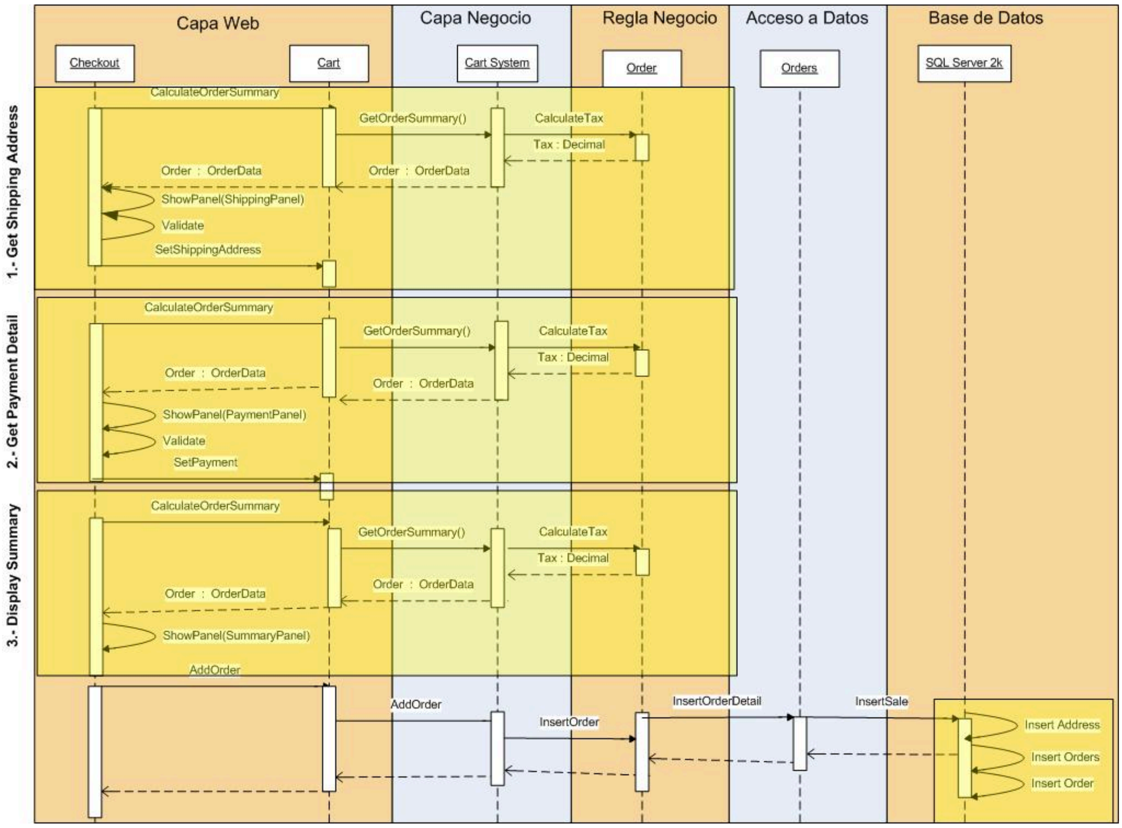
SECUENCIA ORGANIZADOR DE REGALOS EN GRUPO



SECUENCIA CREACION Y ACTUALIZACION DE DATOS DE USUARIO

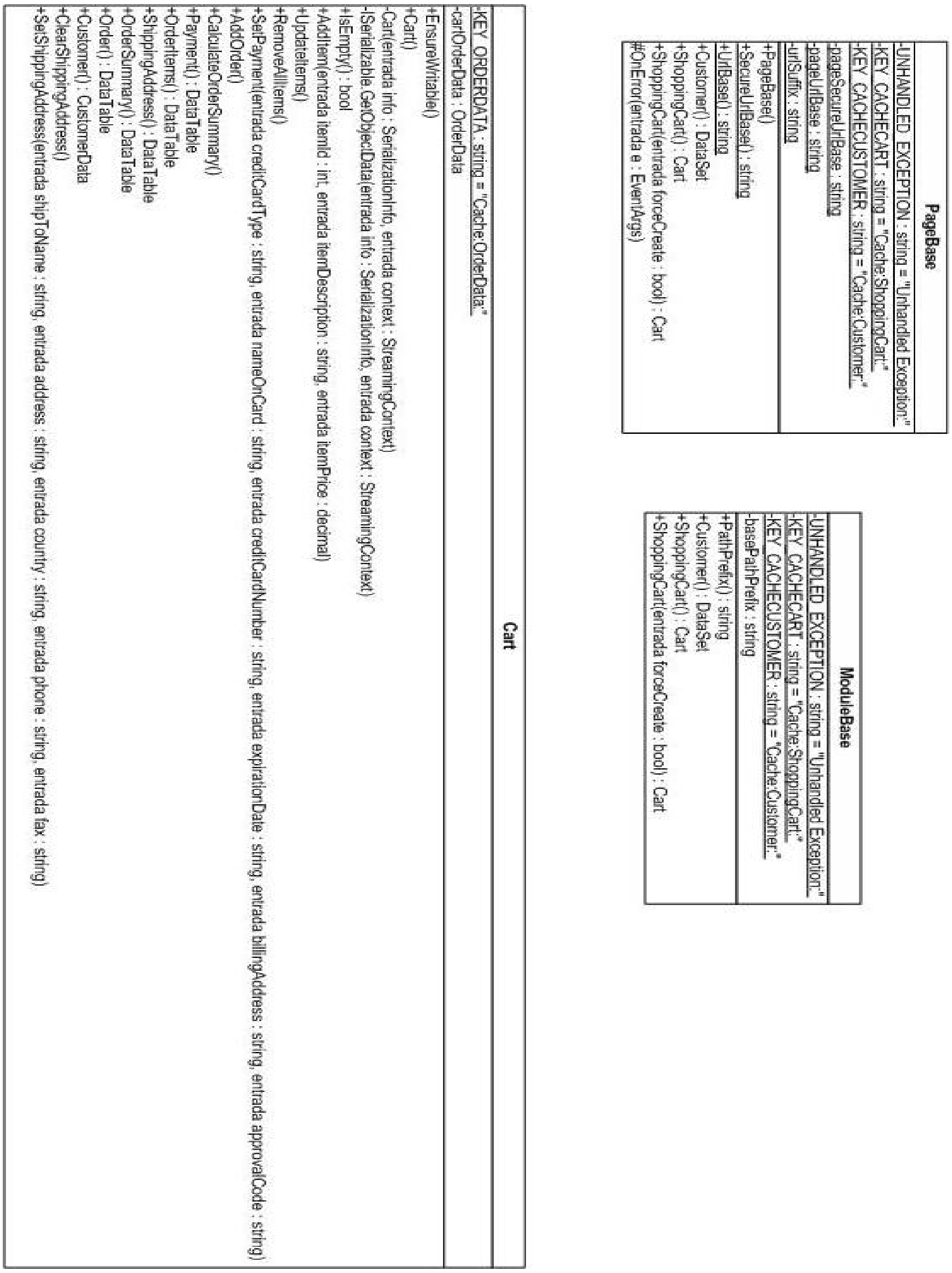


SECUENCIA CREACION PAGAR LA COMPRA

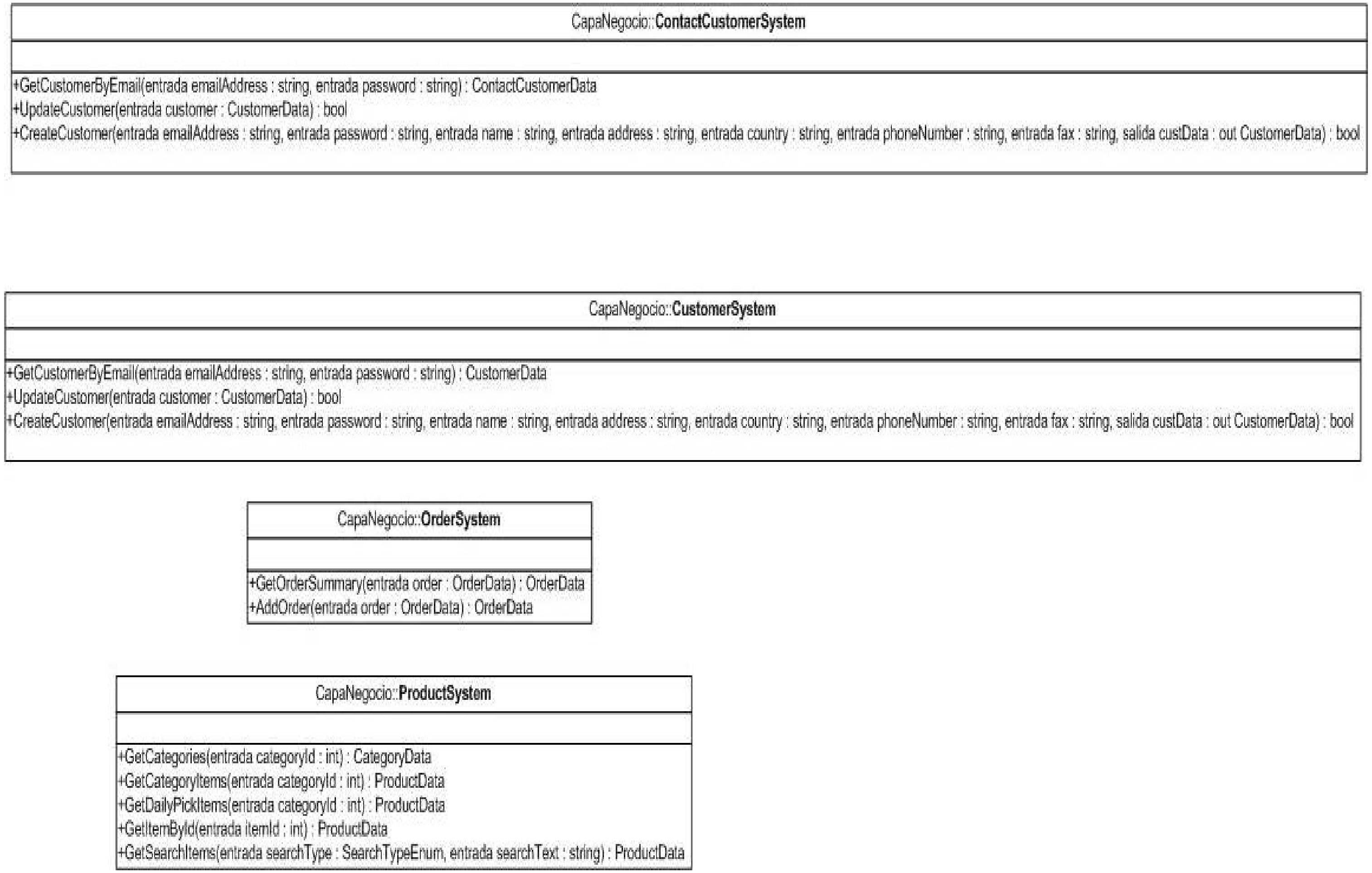


DIAGRAMAS DE CLASES

CLASES CAPA WEB



CLASES CAPA NEGOCIO



CLASES REGLA NEGOCIO

ReglasNegocio::Contacts
-REGEXP_ISVALIDEMAIL : string = @"^\w+((-\w+)|(\.\w+))*\@\w+((\.|-)\w+)*\.\w+$"
+Insert(entrada customer : CustomerData) : bool +Update(entrada customer : CustomerData) : bool -GetCustomerByEmail(entrada emailAddress : string) : ContactCustomerData -Validate(entrada customerRow : DataRow) : bool -IsValidField(entrada customerRow : DataRow, entrada fieldName : string, entrada maxLen : short) : bool -IsValidEmail(entrada customerRow : DataRow) : bool

ReglasNegocio::Customer
-REGEXP_ISVALIDEMAIL : string = @"^\w+((-\w+)|(\.\w+))*\@\w+((\.|-)\w+)*\.\w+$"
+Insert(entrada customer : CustomerData) : bool +Update(entrada customer : CustomerData) : bool -GetCustomerByEmail(entrada emailAddress : string) : CustomerData -Validate(entrada customerRow : DataRow) : bool -IsValidField(entrada customerRow : DataRow, entrada fieldName : string, entrada maxLen : short) : bool -IsValidEmail(entrada customerRow : DataRow) : bool

ReglasNegocio::Order
-MINIMUM_SHIPPING_CHARGE : short = 5 -STANDARD_ITEM_COUNT : short = 5
+CalculateTax(entrada order : OrderData) : decimal +CalculateShipping(entrada order : OrderData) : decimal +InsertOrder(entrada order : OrderData) : bool -IsValidField(entrada order : OrderData, entrada tableName : string, entrada fieldName : string, entrada maxLen : short) : bool -IsValidField(entrada OrderRow : DataRow, entrada fieldName : string, entrada maxLen : short) : bool

CLASES CAPA ACCESO A DATOS

```
AccesoDatos::Categories
-dsCommand : SqlDataAdapter
+Categories()
+Dispose()
#Dispose(entrada disposing : bool)
+GetCategories(entrada categoryId : int) : CategoryData
-FillCategoryData(entrada commandText : string) : CategoryData
```

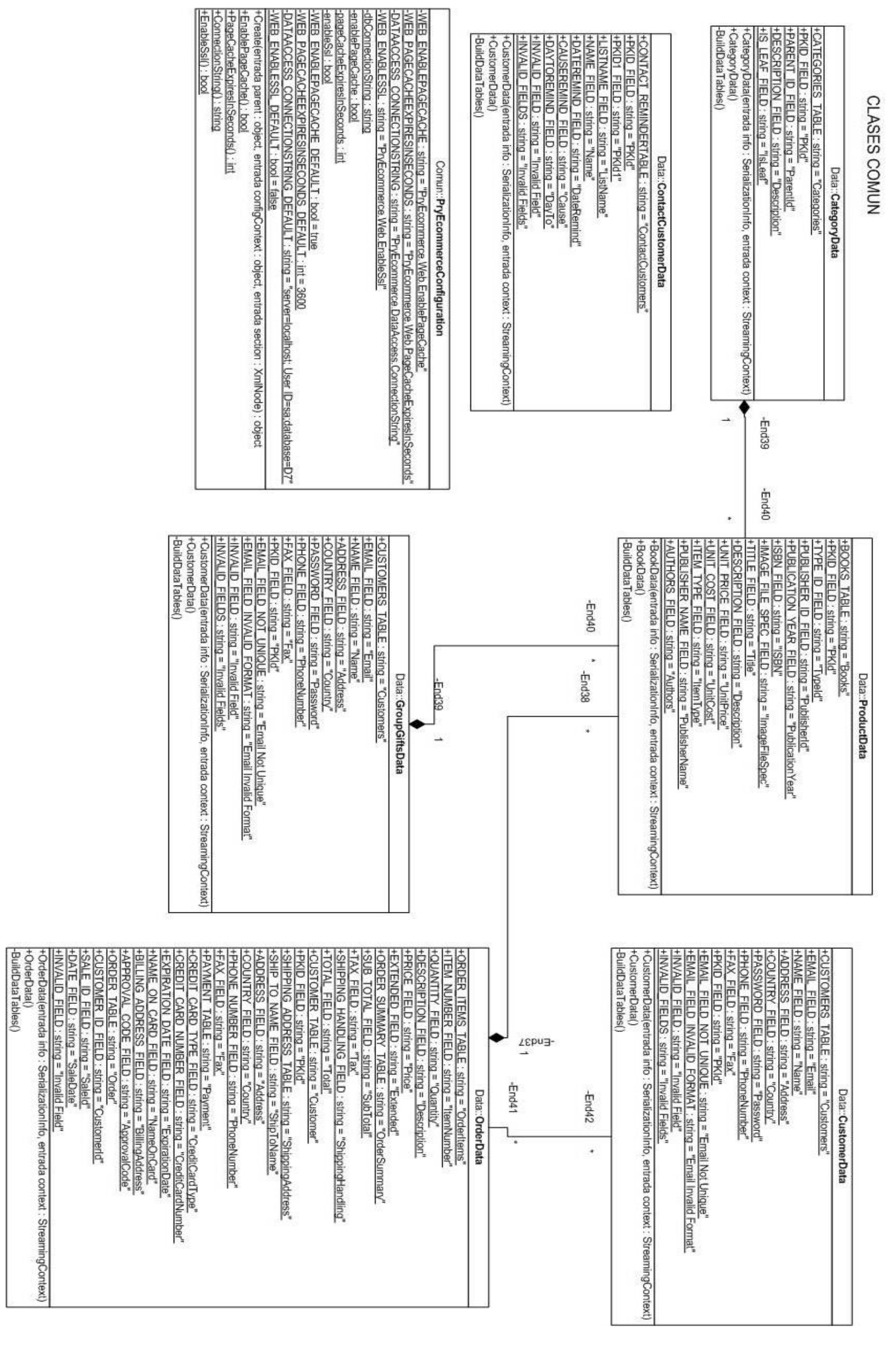
```
AccesoDatos::Customers
-dsCommand : SqlDataAdapter
-PKID_PARM : string = "@PKId"
-EMAIL_PARM : string = "@Email"
-NAME_PARM : string = "@Name"
-ADDRESS_PARM : string = "@Address"
-COUNTRY_PARM : string = "@Country"
-PASSWORD_PARM : string = "@Password"
-PHONE_PARM : string = "@PhoneNumber"
-FAX_PARM : string = "@Fax"
-loadCommand : SqlCommand
-insertCommand : SqlCommand
-updateCommand : SqlCommand
+Customers()
+Dispose()
#Dispose(entrada disposing : bool)
-GetLoadCommand() : SqlCommand
-GetInsertCommand() : SqlCommand
-GetUpdateCommand() : SqlCommand
+LoadCustomerByEmail(entrada emailAddress : string) : CustomerData
+UpdateCustomer(entrada customer : CustomerData) : bool
+InsertCustomer(entrada customer : CustomerData) : bool
```

```
AccesoDatos::Contacts
-dsCommand : SqlDataAdapter
-PKID_PARM : string = "@PKId"
-PKID1_PARM : string = "@PKId1"
-LISTNAME_PARM : string = "@ListName"
-NAME_PARM : string = "@Name"
-DATEREMIND_PARM : string = "@DateRemind"
-CAUSEREMIND_PARM : string = "@Cause"
-DAYTOREMIND_PARM : string = "@DayToRemind"
-loadCommand : SqlCommand
-insertCommand : SqlCommand
-updateCommand : SqlCommand
+Contacts()
+Dispose()
#Dispose(entrada disposing : bool)
-GetLoadCommand() : SqlCommand
-GetInsertCommand() : SqlCommand
-GetUpdateCommand() : SqlCommand
+LoadCustomerByEmail(entrada emailAddress : string) : ContactCustomerData
+UpdateContactCustomer(entrada customer : CustomerData) : bool
+InsertContactCustomer(entrada customer : CustomerData) : bool
+LoadSearchCustomer() : ContactCustomerData
```

```
AccesoDatos::Orders
-insertCommand : SqlCommand
+Orders()
+Dispose()
#Dispose(entrada disposing : bool)
-GetInsertCommand() : SqlCommand
+InsertOrderDetail(entrada order : OrderData) : int
```

```
AccesoDatos::Products
-dsCommand : SqlDataAdapter
+Products()
+Dispose()
#Dispose(entrada disposing : bool)
+GetProductsByCategoryId(entrada categoryId : int) : ProductData
+GetDailyPickProductsByCategoryId(entrada categoryId : int) : ProductData
+GetProductById(entrada bookId : int) : ProductData
+GetProductsByAuthor(entrada searchText : string) : ProductData
+GetProductsByOrigin(entrada searchText : string) : ProductData
+GetBooksBySubject(entrada searchText : string) : ProductData
+GetProductsByName(entrada searchText : string) : ProductData
-FillProductData(entrada commandText : string) : ProductData
```

CLASES CAPA COMUN



CLASES CAPA SYSTEM FRAMEWORK

```
SystemFramework::ApplicationAssert
----------------------------------
+LineNumber() : int
+Check(entrada condition : bool, entrada errorText : string, entrada lineNumber : int)
+CheckCondition(entrada condition : bool, entrada errorText : string, entrada lineNumber : int)
-GenerateStackTrace(entrada lineNumber : int, salida currentTrace : string)
```

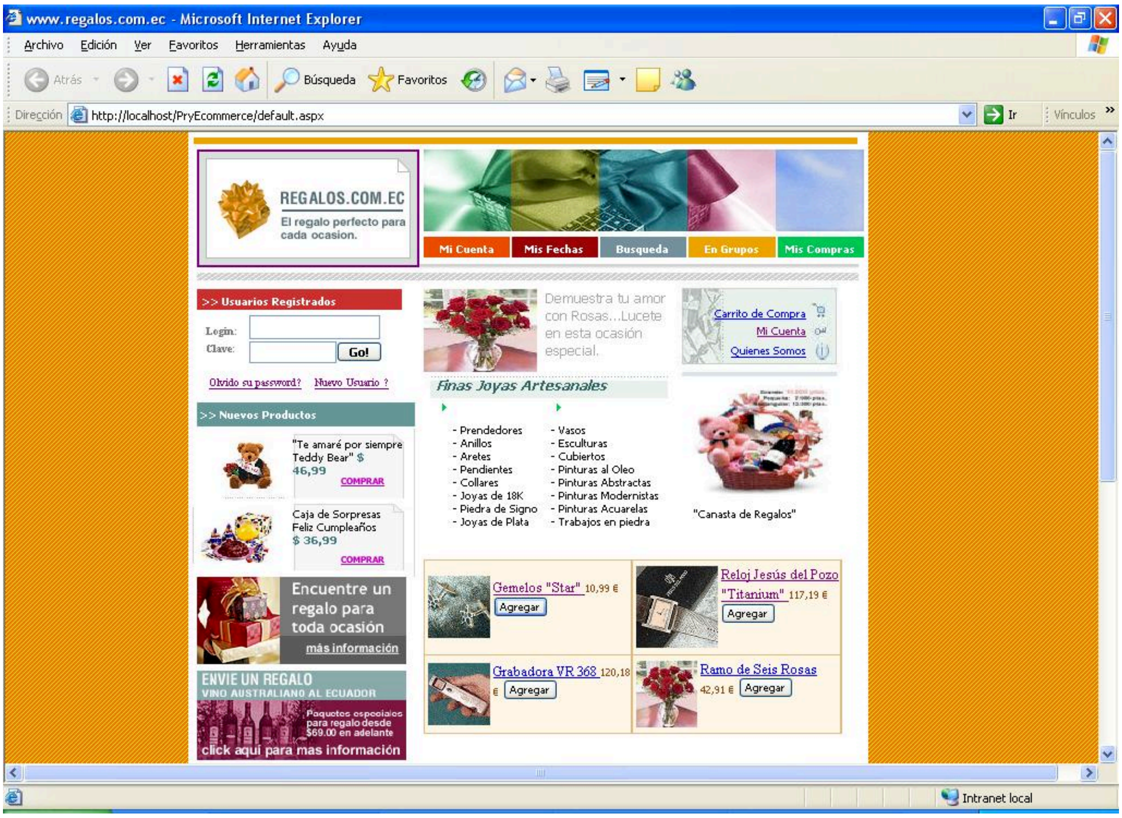
```
SystemFramework::ApplicationConfiguration
----------------------------------
-TRACING_ENABLED : string = "SystemFramework.Tracing.Enabled"
-TRACING_TRACEFILE : string = "SystemFramework.Tracing.TraceFile"
-TRACING_TRACELEVEL : string = "SystemFramework.Tracing.TraceLevel"
-TRACING_SWITCHNAME : string = "SystemFramework.Tracing.SwitchName"
-TRACING_SWITCHDESCRIPTION : string = "SystemFramework.Tracing.SwitchDescription"
-EVENTLOG_ENABLED : string = "SystemFramework.EventLog.Enabled"
-EVENTLOG_MACHINENAME : string = "SystemFramework.EventLog.Machine"
-EVENTLOG_SOURCENAME : string = "SystemFramework.EventLog.SourceName"
-EVENTLOG_TRACELEVEL : string = "SystemFramework.EventLog.LogLevel"
-tracingEnabled : bool
-tracingTraceFile : string
-tracingTraceLevel : TraceLevel
-tracingSwitchName : string
-tracingSwitchDescription : string
-eventLogEnabled : bool
-eventLogMachineName : string
-eventLogSourceName : string
-eventLogTraceLevel : TraceLevel
-appRoot : string
-TRACING_ENABLED_DEFAULT : bool = true
-TRACING_TRACEFILE_DEFAULT : string = "ApplicationTrace.txt"
-TRACING_TRACELEVEL_DEFAULT : TraceLevel = TraceLevel.Verbose
-TRACING_SWITCHNAME_DEFAULT : string = "ApplicationTraceSwitch"
-TRACING_SWITCHDESCRIPTION_DEFAULT : string = "Application error and tracing information"
-EVENTLOG_ENABLED_DEFAULT : bool = true
-EVENTLOG_MACHINENAME_DEFAULT : string = ""
-EVENTLOG_SOURCENAME_DEFAULT : string = "WebApplication"
-EVENTLOG_TRACELEVEL_DEFAULT : TraceLevel = TraceLevel.Error
----------------------------------
+Create(entrada parent : object, entrada configContext : object, entrada section : XmlNode) : object
+ReadSetting(entrada settings : NameValueCollection, entrada key : string, entrada defaultValue : string) : string
+ReadSetting(entrada settings : NameValueCollection, entrada key : string, entrada defaultValue : bool) : bool
+ReadSetting(entrada settings : NameValueCollection, entrada key : string, entrada defaultValue : int) : int
+ReadSetting(entrada settings : NameValueCollection, entrada key : string, entrada defaultValue : TraceLevel) : TraceLevel
+OnApplicationStart(entrada myAppPath : string)
+AppRoot() : string
+TracingEnabled() : bool
+TracingTraceFile() : string
+TracingTraceLevel() : TraceLevel
+TracingSwitchName() : string
+TracingSwitchDescription() : string
+EventLogEnabled() : bool
+EventLogMachineName() : string
+EventLogSourceName() : string
+EventLogTraceLevel() : TraceLevel
```

```
SystemFramework::ApplicationLog
----------------------------------
-debugSwitch : TraceSwitch
-debugWriter : StreamWriter
-eventLog : EventLog
-eventLogTraceLevel : TraceLevel
----------------------------------
+WriteError(entrada message : string)
+WriteWarning(entrada message : string)
+WriteInfo(entrada message : string)
+WriteTrace(entrada message : string)
+FormatException(entrada ex : Exception, entrada catchInfo : string) : string
-WriteLog(entrada level : TraceLevel, entrada messageText : string)
-ApplicationLog()
```

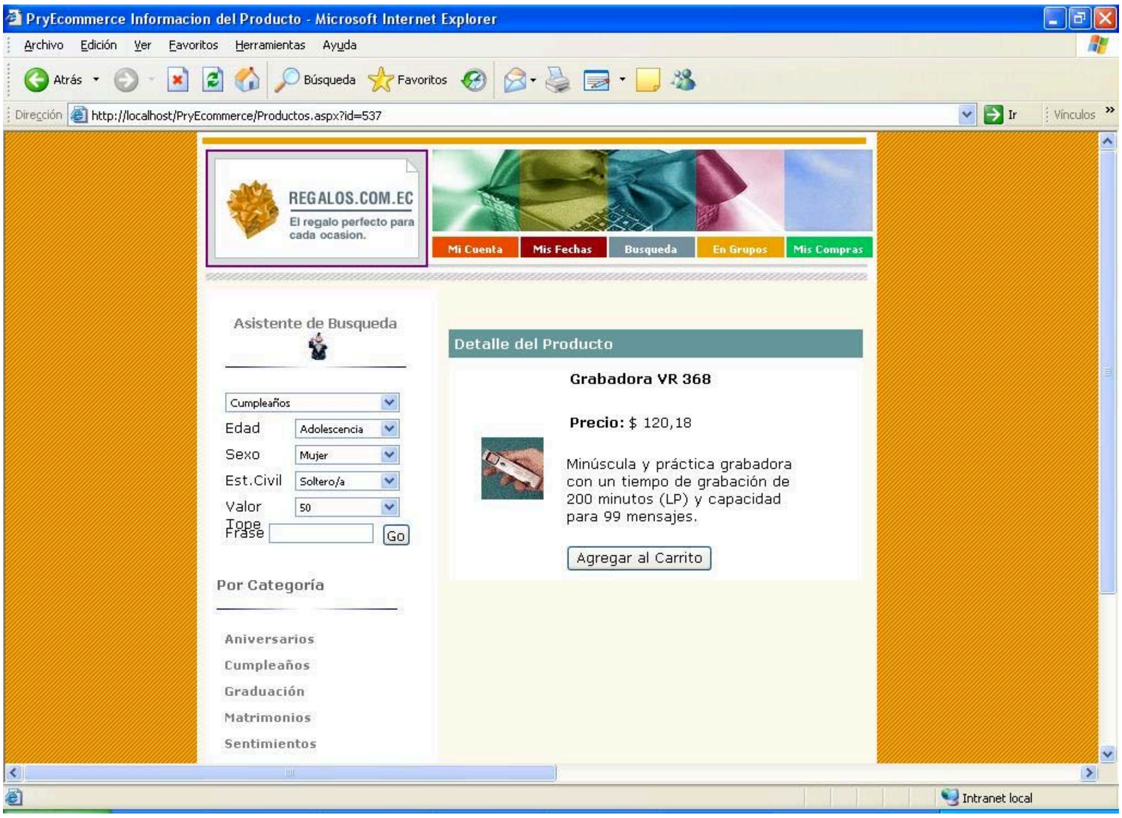
APENDICE E

PANTALLAS DEL SITIO E-COMMERCE

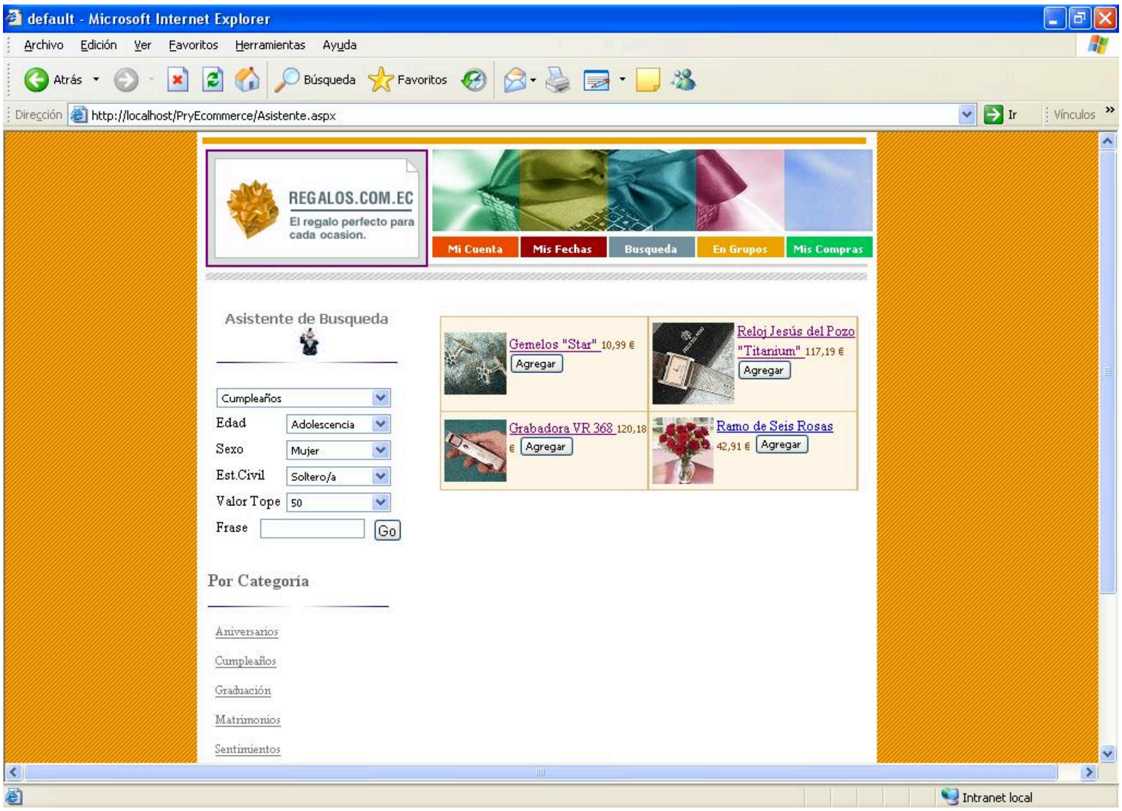
PANTALLA PRINCIPAL



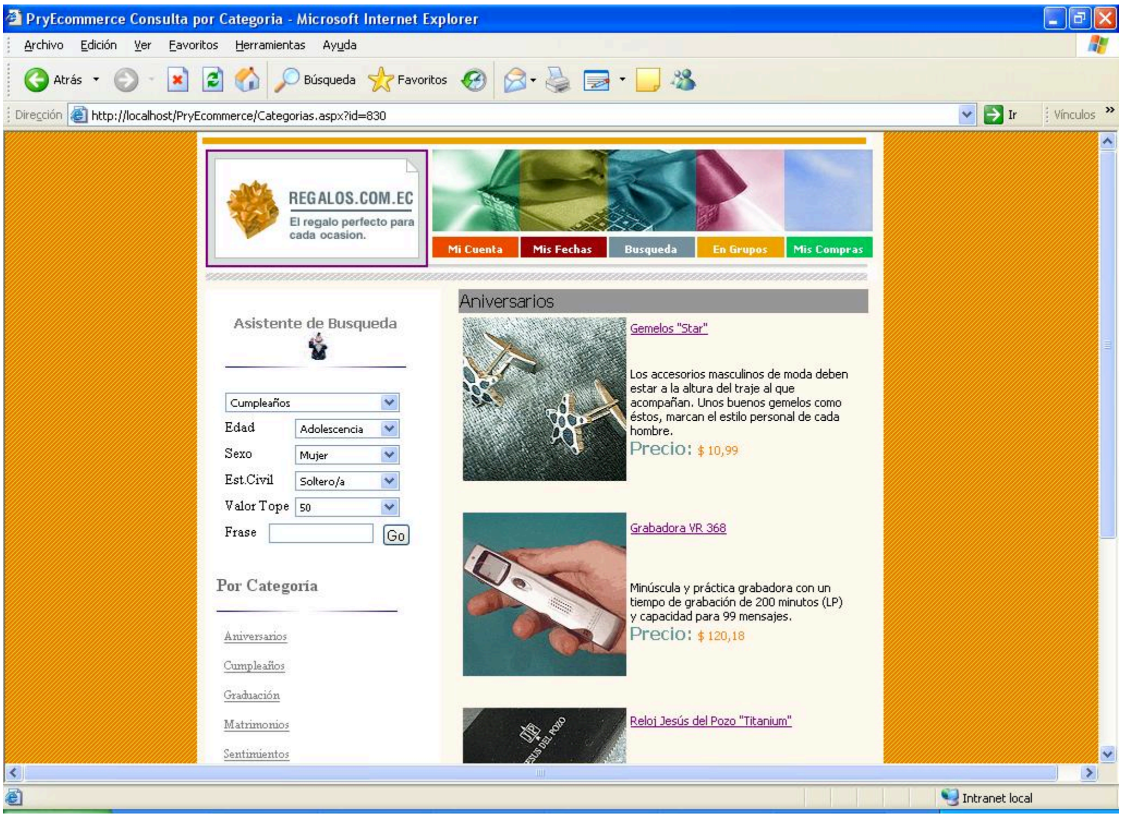
DETALLE DE PRODUCTO SELECCIONADO



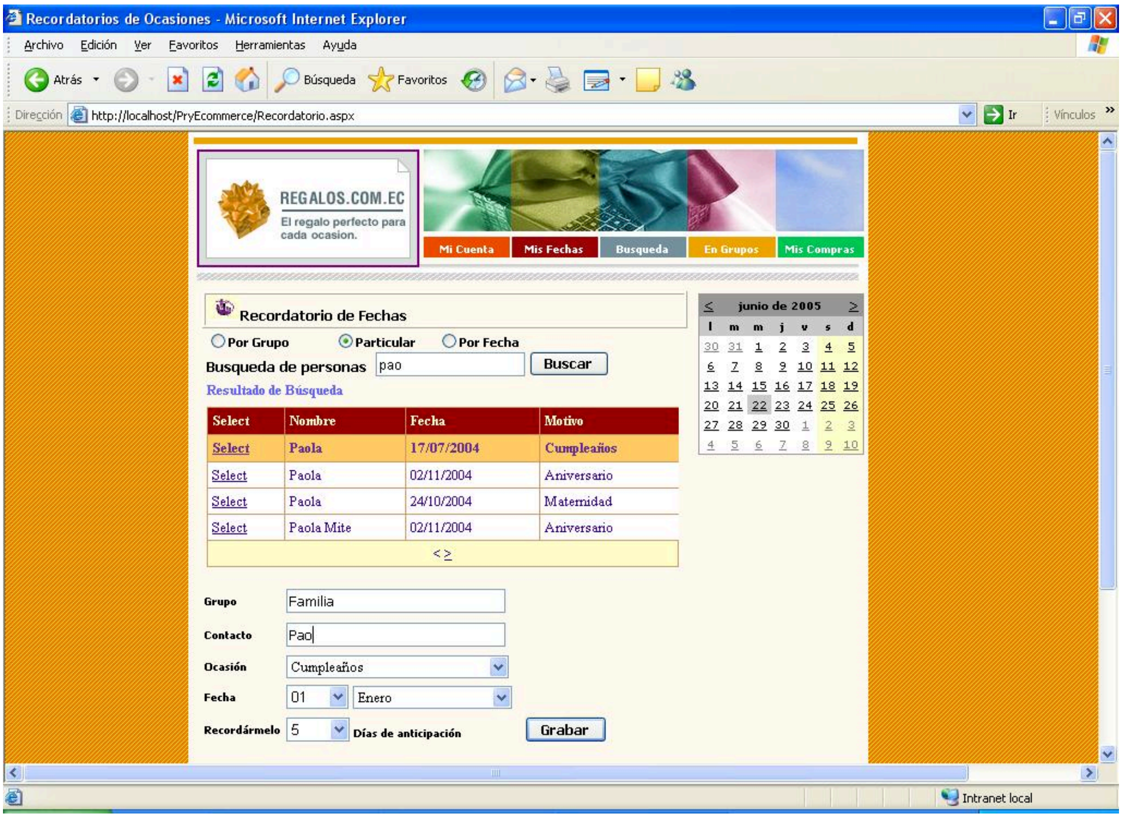
ASISTENTE DE SELECCIÓN DE REGALOS



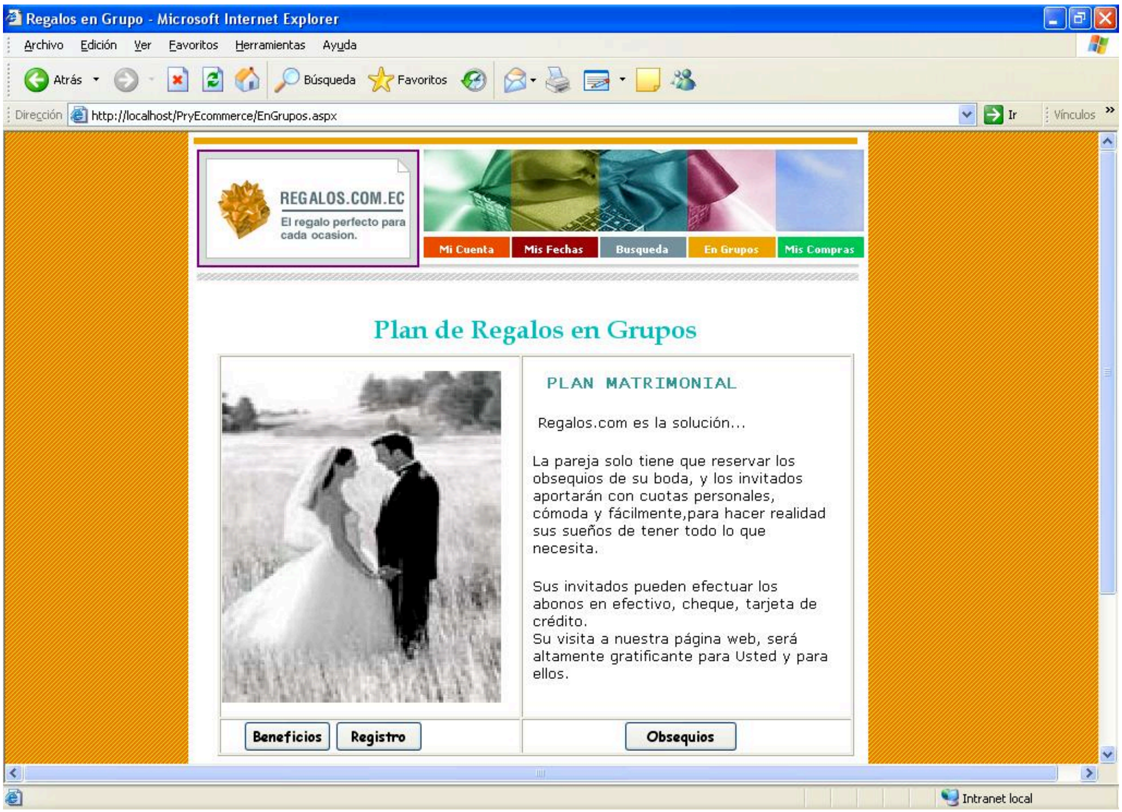
BUSQUEDA DE REGALOS



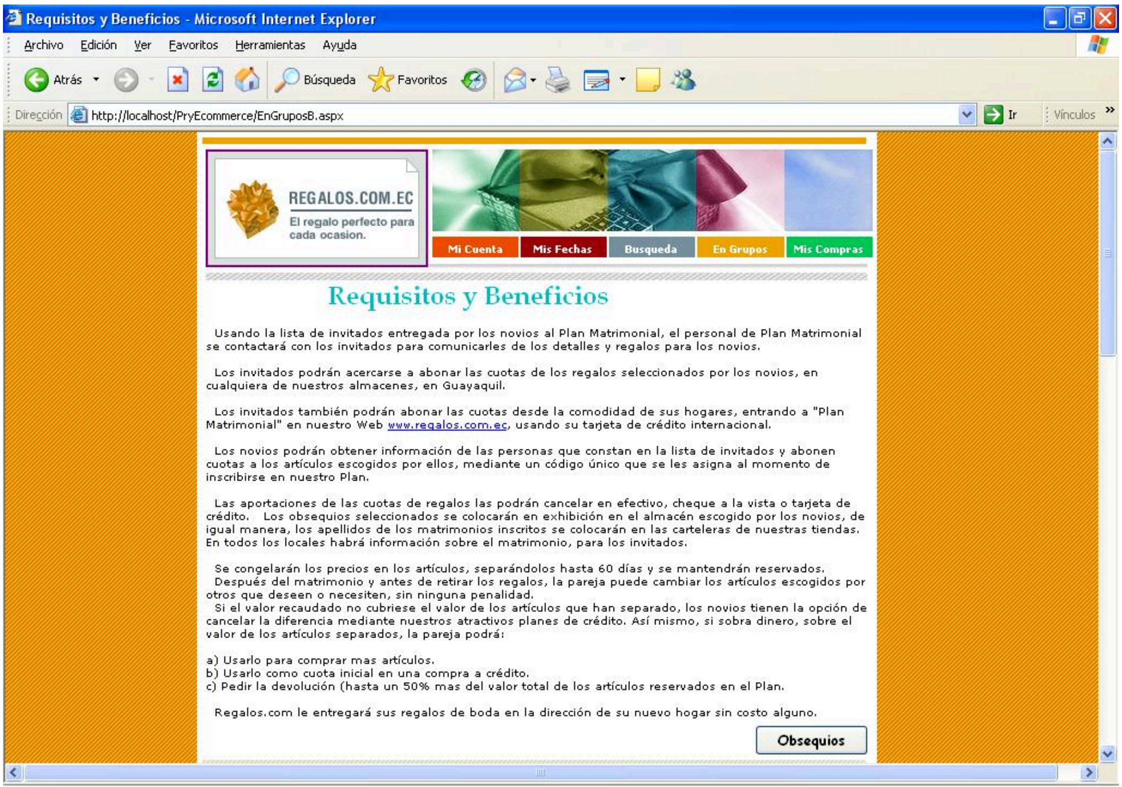
SERVICIO DE RECORDATORIOS



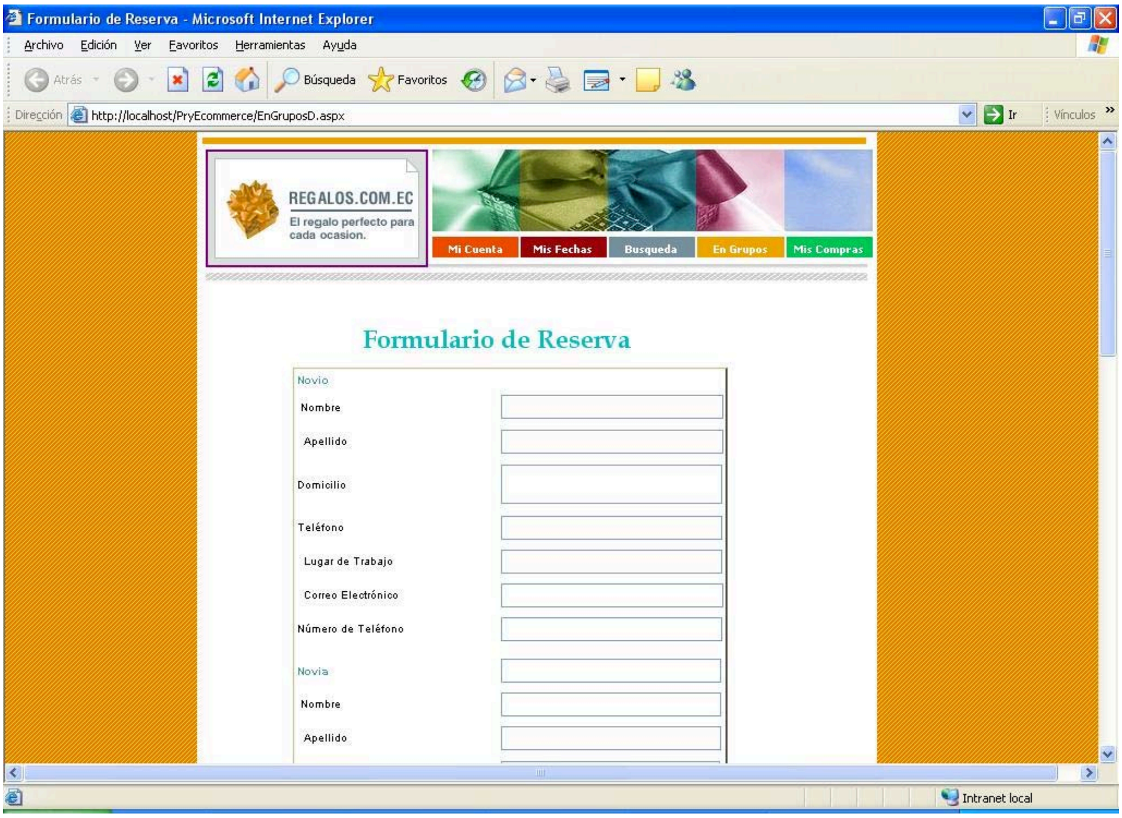
PANTALLA 1 DE REGALOS EN GRUPO



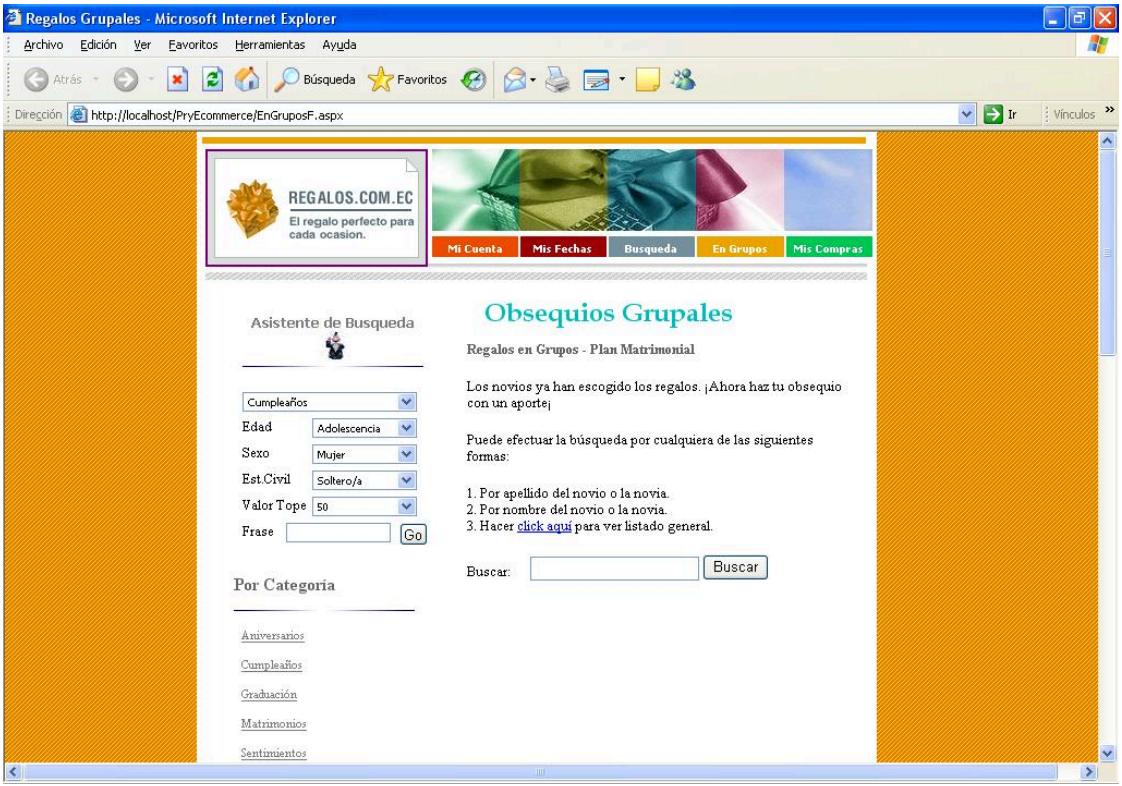
PANTALLA 2 DE REGALOS EN GRUPO



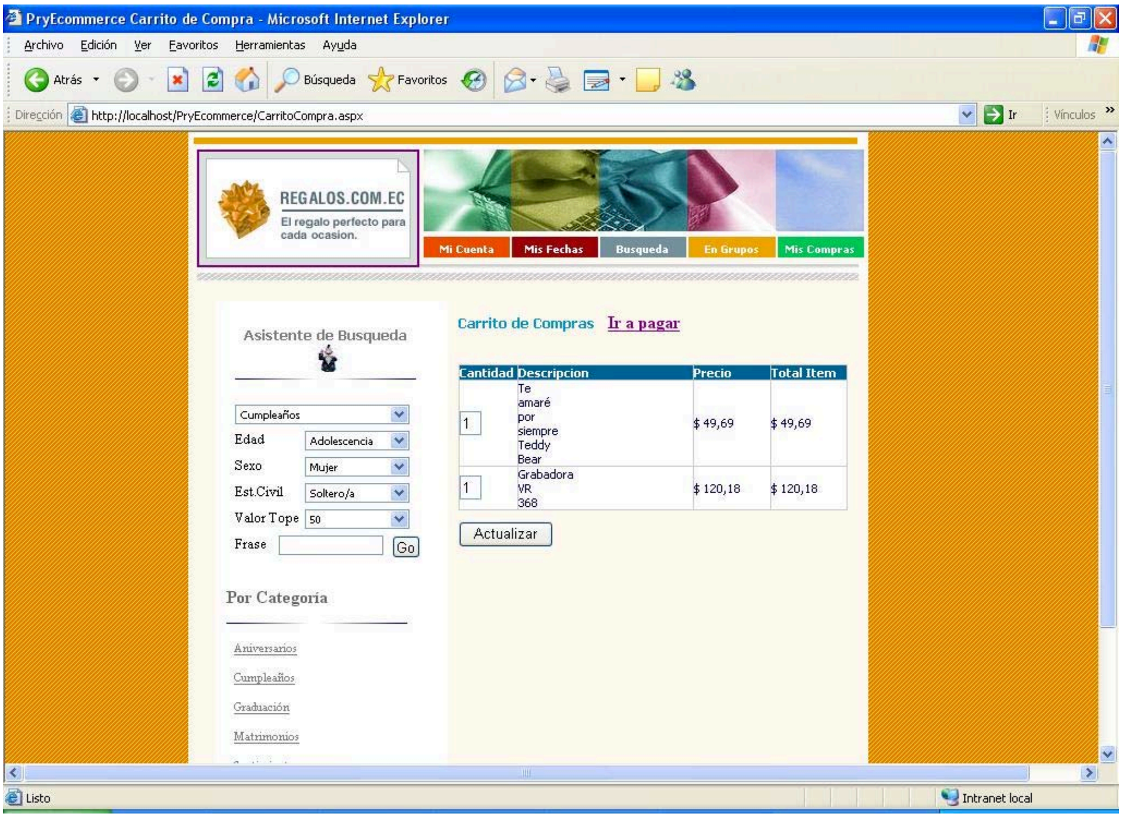
PANTALLA 3 DE REGALOS EN GRUPO



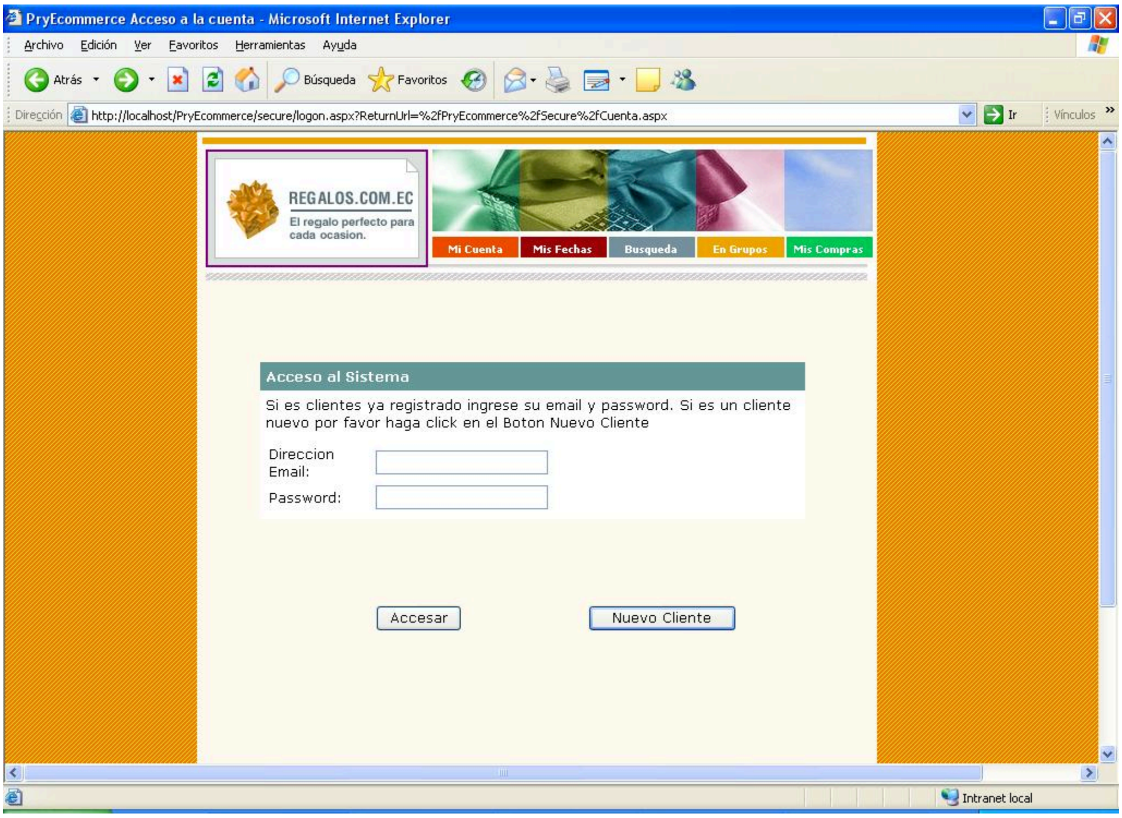
PANTALLA 4 DE REGALOS EN GRUPO



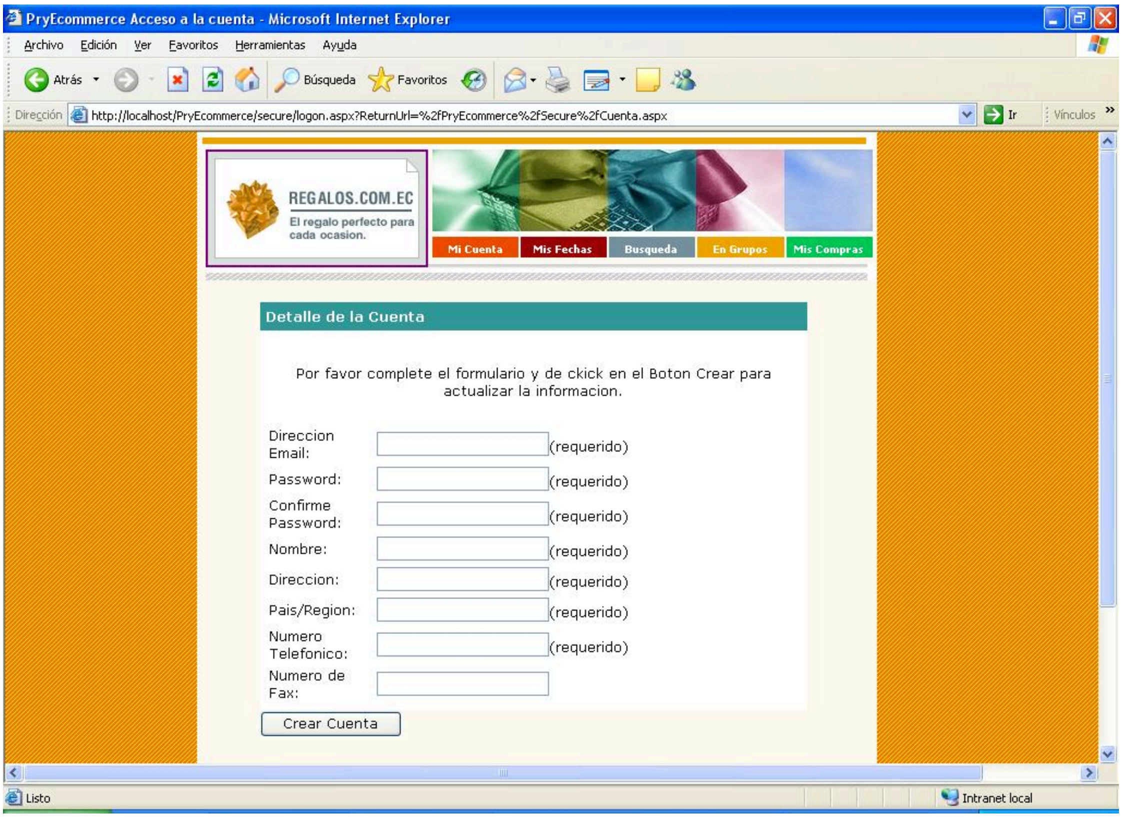
CARRITO DE COMPRAS



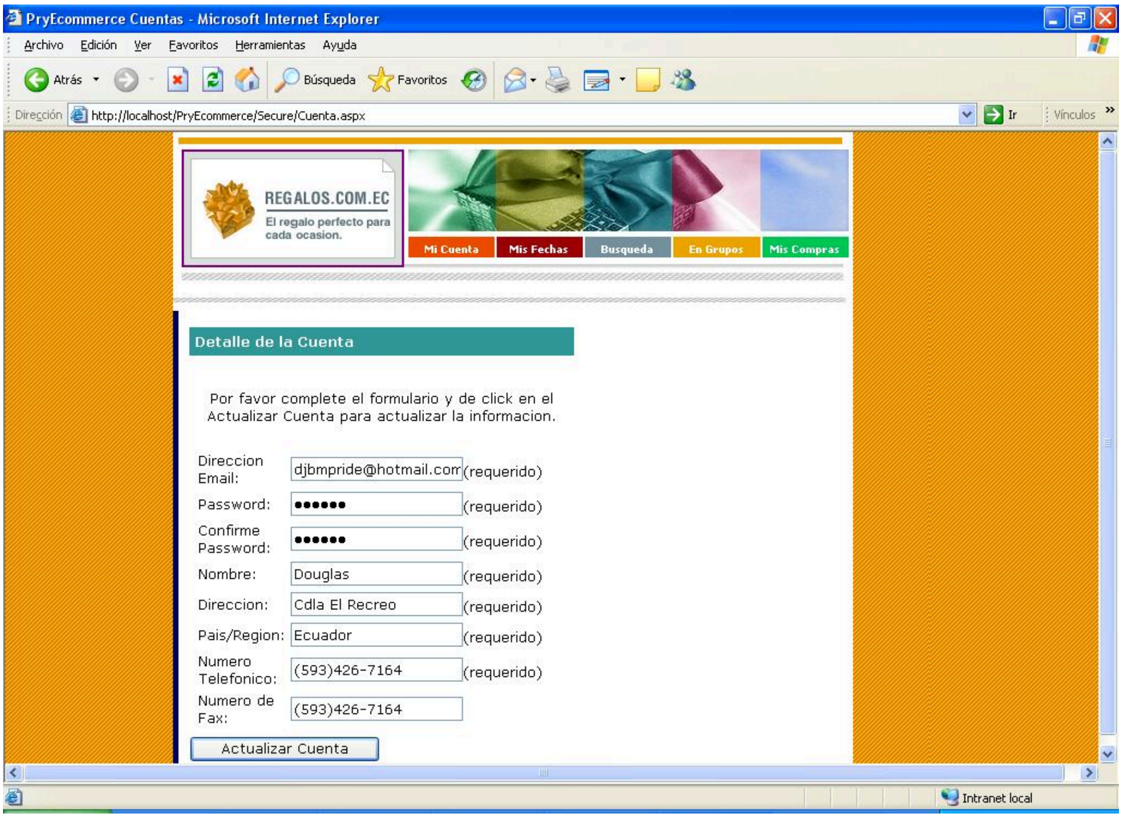
PANTALLA DE ACCESO DE USUARIO



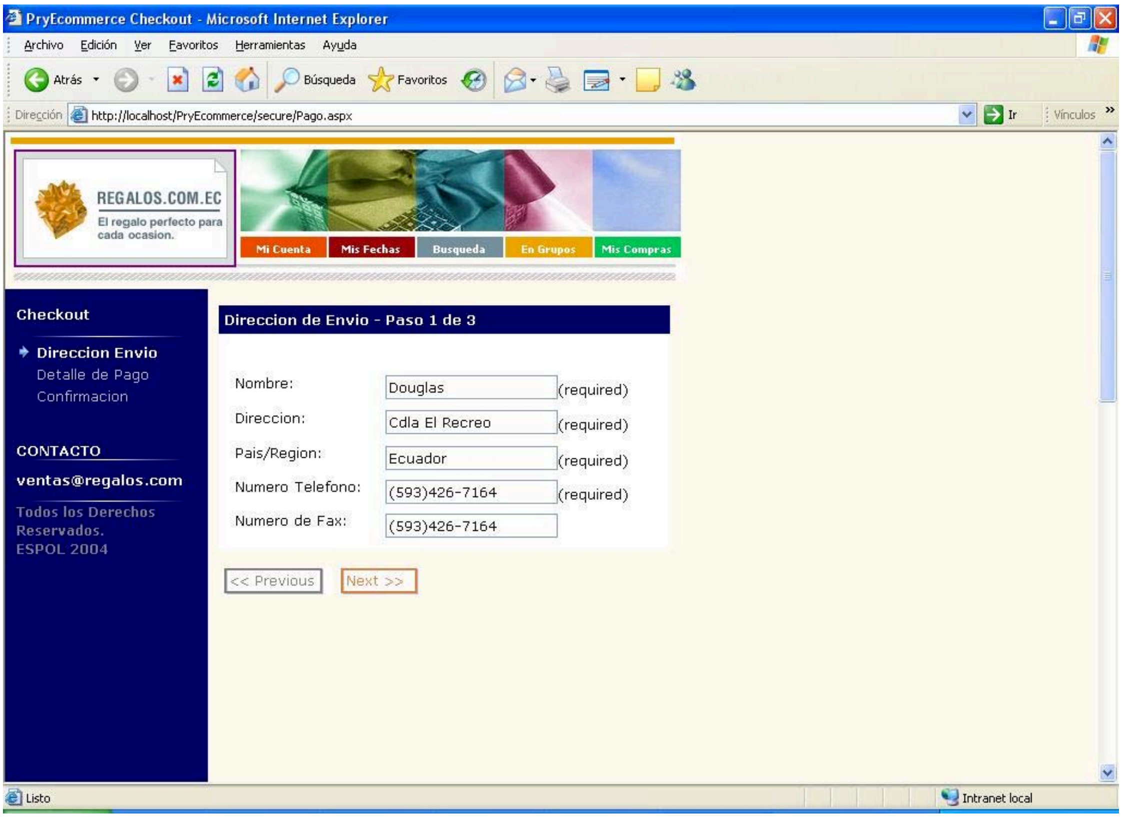
CREACION DE NUEVA CUENTA DE USUARIO



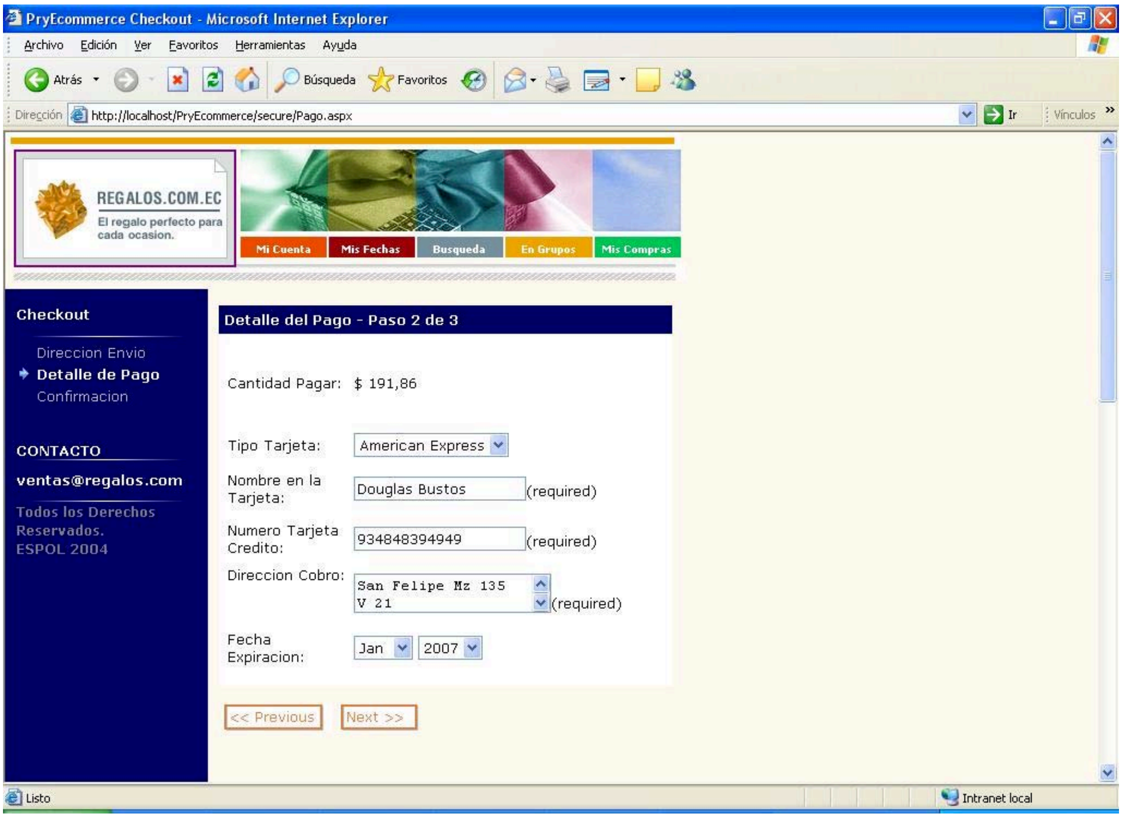
DATOS DEL CLIENTE



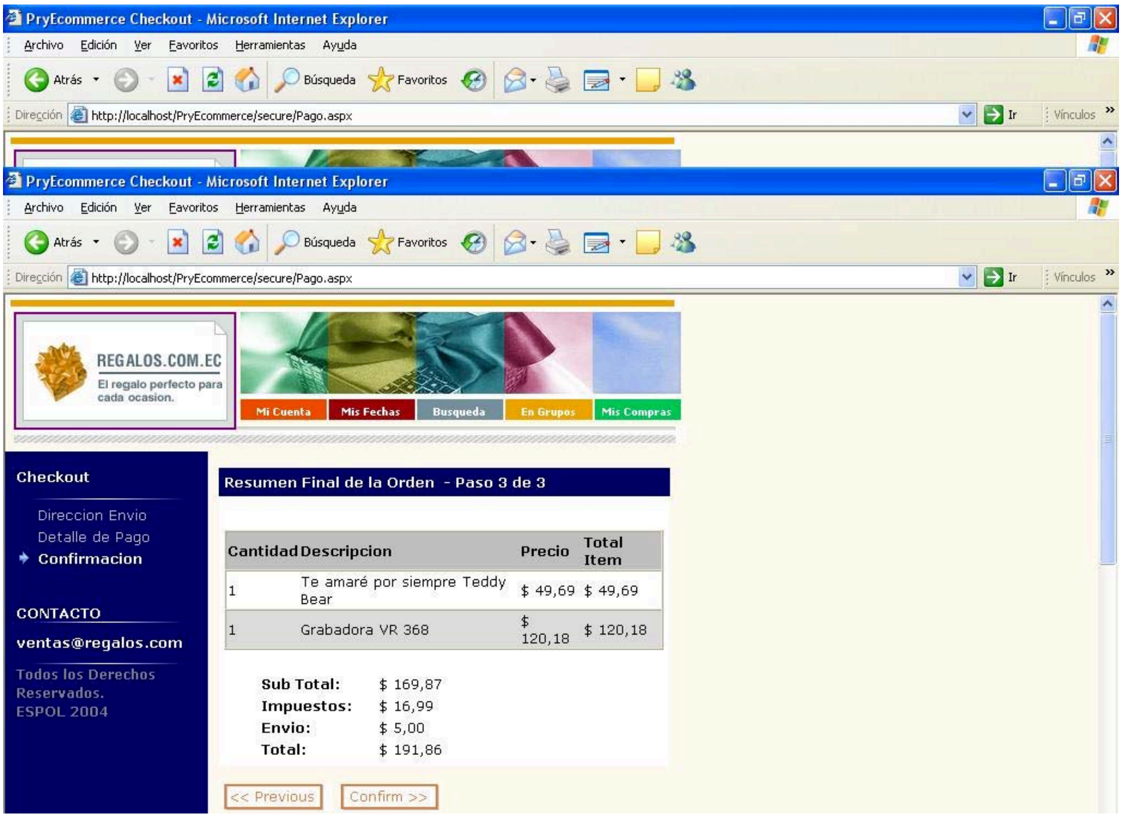
PROCESO DE CHECKOUT – PASO 1



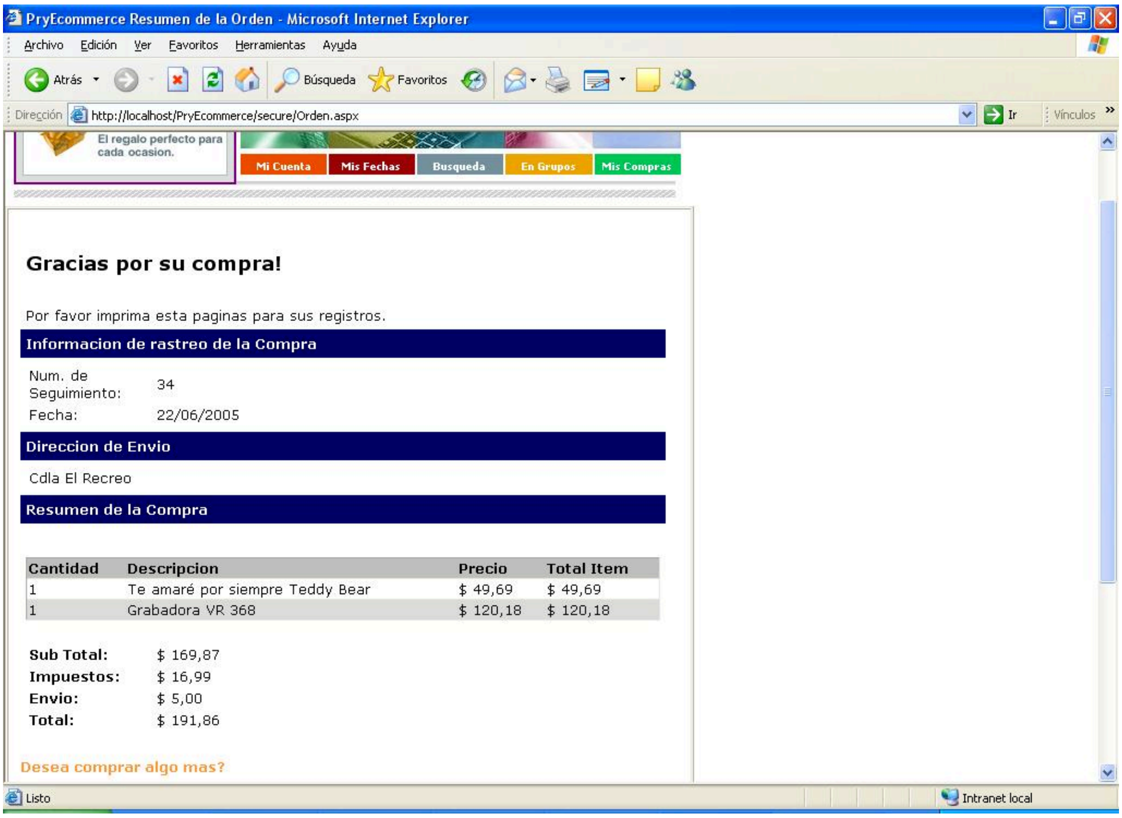
PROCESO DE CHECKOUT – PASO 2



PROCESO DE CHECKOUT – PASO 3



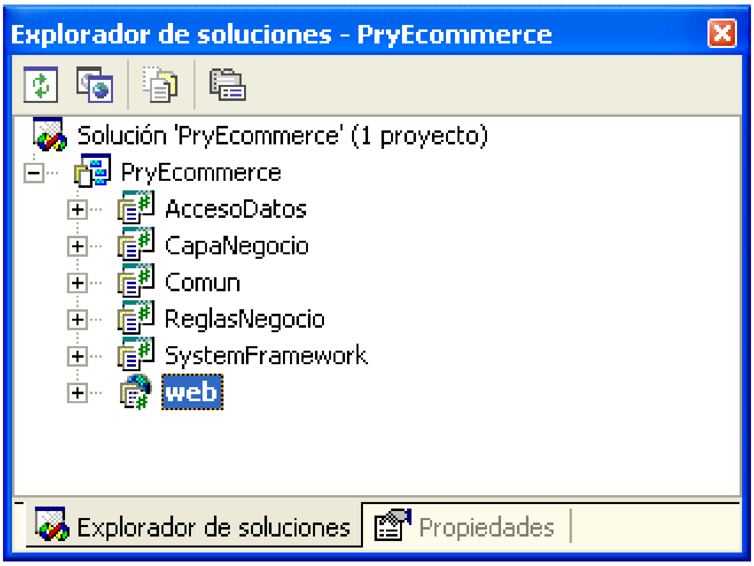
PROCESO DE CHECKOUT – PASO 4



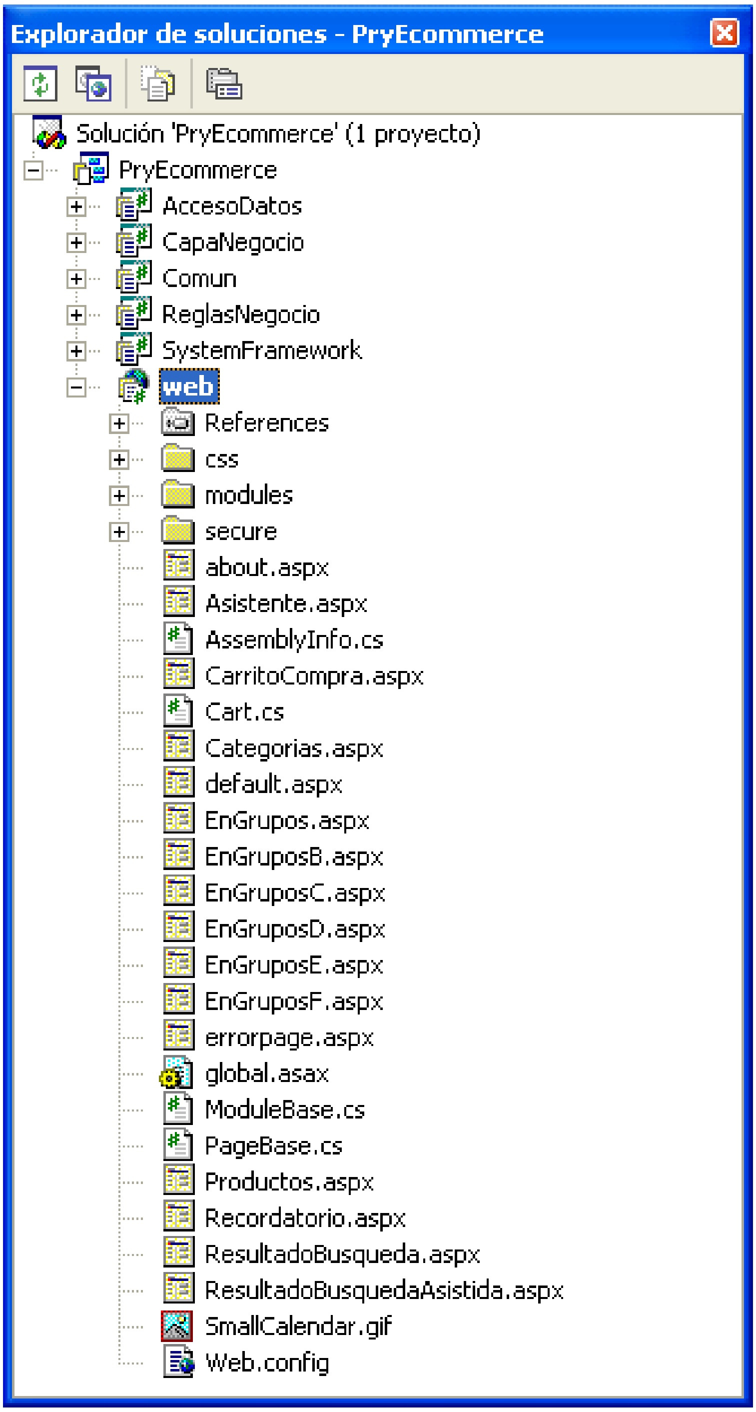
APENDICE F

EXTRACTO DEL CODIGO FUENTE DEL SITIO

Componentes de la Solución



Capa de Interfaz de Usuario - Web



Capa de Interfaz de Usuario Web

Referencias a Librerías:

Librerías Componentes de Proyecto:
- Comun.dll
- SystemFramework.dll
- CapaNegocio.dll

Librerías .Net :
- MsCorLib.dll
- System.dll
- System.Data.dll
- System.Design.dll
- System.Drawing.dll
- System.Web.dll
- System.Web.Service.dll
- System.Xml

Archivos de Clases:

Css
- PryEcommerce.css

Modules
- ModuloAcercaDe.ascx
- ModuloBanner.ascx
- ModuloBusqueda.ascx
- ModuloBusquedaAsistida.ascx
- ModuloCategorias.ascx
- ModuloCuenta.ascx
- ModuloPago.ascx
- ModuloSeleccionDia.ascx
- ModuloServicios.ascx

Secure
Cuenta.aspx
logon.aspx
Orden.aspx
Pago.aspx
Web.config

default.aspx
about.aspx

Asistente.aspx
CarritoCompra.aspx
Cart.cs
Categorias.aspx
EnGrupos.aspx
EnGruposA.aspx
EnGruposB.aspx
EnGruposC.aspx
EnGruposD.aspx
EnGruposE.aspx
EnGruposF.aspx
errorpage.aspx
global.asax
ModuleBase.cs
PageBase.cs
Productos.aspx
Recordatorio.aspx
ResultadoBusqueda.aspx
ResultadoBusquedaAsistida.aspx
Web.config
AssemblyInfo.cs

default.cs - default.aspx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;


using PryEcommerce.CapaNegocio;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{
	/// <summary>
	/// Summary description for default2.
	/// </summary>
	public class Default: PageBase //: System.Web.UI.Page
	{
		protected System.Web.UI.WebControls.Button BtnLogInOut;
		protected System.Web.UI.WebControls.Button AddToCartButton;
		protected System.Web.UI.WebControls.Button Button2;
```

```
			protected System.Web.UI.WebControls.TextBox LogonEmailTextBox;
			protected System.Web.UI.WebControls.Label Label1;
			protected System.Web.UI.WebControls.TextBox LogonPasswordTextBox;
			protected System.Web.UI.WebControls.Label Label2;
			protected System.Web.UI.WebControls.Button Button1;

			//
			// private member variables
			//
			private CustomerData logonCustomerData;  //used for the current
Customer

			private void Page_Load(object sender, System.EventArgs e)
			{
				// Put user code to initialize the page here

				string Lang =  Request.UserLanguages[0];

				System.Threading.Thread.CurrentThread.CurrentCulture =
					new System.Globalization.CultureInfo(Lang) ;

			}

			#region Web Form Designer generated code
			override protected void OnInit(EventArgs e)
			{
				//
				// CODEGEN: This call is required by the ASP.NET Web Form
Designer.
				//
				InitializeComponent();
				base.OnInit(e);
			}

			/// <summary>
			/// Required method for Designer support - do not modify
			/// the contents of this method with the code editor.
			/// </summary>
			private void InitializeComponent()
			{
				this.Load += new System.EventHandler(this.Page_Load);

			}
			#endregion

			private void BtnLogInOut_Click(object sender, System.EventArgs e)
			{
				CustomerData custData;
				//
				// Check the Email and Password combination
				//
				custData = (new
CustomerSystem()).GetCustomerByEmail(LogonEmailTextBox.Text,
LogonPasswordTextBox.Text);

				if (custData != null)    //were they valid?
				{
					//
					// 1. Update customer in session.
					// 2. Update customer in cart.
```

```
                                //
                                base.Customer = custData;
                                base.CarritoCompra().Customer = custData;
                              //FormsAuthentication.RedirectFromLoginPage("*",
false);

                        }
                        else
                        {

                        }
                        }

                public void AddToCartButton_Click(Object sender, EventArgs e)
                {
                        Button itemButton = (Button)sender;
                        string arguments = itemButton.CommandArgument;
                        char [] parsChar = {'|'};
                        string[] addArgs = arguments.Split(parsChar);
                        CarritoCompra().AddItem(Int32.Parse(addArgs[0]), addArgs[1]
, Decimal.Parse(addArgs[2]));
                        Response.Redirect("CarritoCompra.aspx", false);

                }

                private void itemButton_Click(object sender, System.EventArgs e)
                {

                }


        }
}
```

Carts.cs - Carts.aspx

```
using System;
using System.Web;
using System.Collections;
using System.Data;
using System.Runtime.Serialization;

using PryEcommerce.CapaNegocio;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{

    /// <summary>
    ///     Helper class used to manage shopping cart data and session state.
```

```
    ///     <remarks>
    ///         This class derives off of ISerializable.  It's private
serializable constructor
    ///         is called by the runtime.
    ///     </remarks>
    /// </summary>
    public class Cart : ISerializable
    {

        private const String KEY_ORDERDATA = "Cache:OrderData:";

        //shopping cart data variable
        private Comun.Data.OrderData cartOrderData;

        /// <summary>
        ///     Make sure that the cart can actually be written to.
        ///     <remarks>
        ///         The cart may still be empty at this point, all methods will
succeed.
        ///         The cart is initially created without data so that we don't
actually
        ///         create an OrderData until we need to populate it.
        ///     </remarks>
        /// </summary>
        public void EnsureWritable()
        {
            if (null == cartOrderData)
            {
                cartOrderData = new OrderData();
            }
        }

        /// <summary>
        ///     Constructor for Cart.
        ///     <remarks>Create a new object. Included to expose default
constructor.</remarks>
        /// </summary>
        public Cart()
        {
        }

        /// <summary>
        ///     Depersist the Cart object from serialized data.
        ///     <remarks>Constructor that supports serialization.</remarks>
        ///     <remarks>
        ///         This is called by the runtime. It is private so that no
        ///         one else can call it.
        ///     </remarks>
        ///     <param name="info">The SerializationInfo object to read
from.</param>
        ///     <param name="context">Information on who is calling this
method.</param>
        /// </summary>
        private Cart(SerializationInfo info, StreamingContext context)
        {
            try
            {
                cartOrderData =
(Comun.Data.OrderData)info.GetValue(KEY_ORDERDATA,
typeof(Comun.Data.OrderData));
```

```
            }
            catch
            {
                // Leave cartOrderData null if it hasn't been serialized
            }
        }


        //--------------------------------------------------------------
        // Sub GetObjectData:
        //   Serialize the cart class.
        // Parameters:
        //   [in] info: The SerializationInfo object to write to
        //   [in] context: Information on who is calling this method
        //--------------------------------------------------------------
        void ISerializable.GetObjectData(SerializationInfo info,
StreamingContext context)
        {
            if (cartOrderData != null)
            {
                info.AddValue(KEY_ORDERDATA, cartOrderData);
            }
        }

        /// <value>
        ///     Property IsEmpty is used to get whether the cart is empty.
        ///     <remarks> Returns true if the cart is empty, false
otherwise.</remarks>
        /// </value>
        public bool IsEmpty
        {
            get
            {
                return (null == cartOrderData) ?
                         true :
                         (0 == OrderItems.Rows.Count);
            }
        }

        /// <summary>
        ///     Adds an Item to the shopping cart.
        ///     <remarks>Constructor that supports serialization.</remarks>
        ///     <param name="itemId">The Id (PKId) of a shopping cart
item.</param>
        ///     <param name="itemDescription">The Description of a shopping
cart item.</param>
        ///     <param name="itemPrice">The price of a shopping cart
item.</param>
        /// </summary>
        public void AddItem(int itemId, String itemDescription, Decimal
itemPrice)
        {
            DataTable itemTable = OrderItems;
            DataView itemSource = new DataView(itemTable);

            //search for item, check to see if the item has already been
ordered
            itemSource.RowFilter = "ItemNumber = " + itemId.ToString();

            if (itemSource.Count > 0)
```

```
            {
                DataRowView sourceRow = itemSource[0];
                short count = (short)(sourceRow[OrderData.QUANTITY_FIELD]);
                //maximum allowed for any specific item is 50
                if (count < 50)
                {
                    //bump the quantity by one
                    count += 1;
                    sourceRow[OrderData.QUANTITY_FIELD] = count;
                    sourceRow[OrderData.EXTENDED_FIELD] =
(Decimal)sourceRow[OrderData.PRICE_FIELD] * count;
                }
            }
            else
            {

                //It's a new item
                DataRow itemRow = itemTable.NewRow();
                itemRow[OrderData.ITEM_NUMBER_FIELD] = itemId;
                itemRow[OrderData.QUANTITY_FIELD] = 1;
                itemRow[OrderData.DESCRIPTION_FIELD] = itemDescription;
                itemRow[OrderData.PRICE_FIELD] = itemPrice;
                itemRow[OrderData.EXTENDED_FIELD] = itemPrice;

                //Add it to the table
                itemTable.Rows.Add(itemRow);
            }
        }

        /// <summary>
        ///     Updates the Items in the shopping cart.
        /// </summary>
        public void UpdateItems()
        {
            int quantity;
            int itemsCount = OrderItems.Rows.Count;
            DataRow row;

            for (int i = 0; i < itemsCount; i++)
            {
                row = OrderItems.Rows[i];
                quantity =
Int32.Parse(row[OrderData.QUANTITY_FIELD].ToString());
                if (quantity < 1)
                {
                    row.Delete();
                    itemsCount--;
                    i--;
                }
                else
                {
                    row[OrderData.EXTENDED_FIELD] =
(Decimal)row[OrderData.PRICE_FIELD] * quantity;
                }
            }
        }

        /// <summary>
        ///     Removes all items from the cart.
        /// </summary>
```

```
        public void RemoveAllItems()
        {
            OrderItems.Rows.Clear();
        }

        /// <summary>
        ///     Saves the payment information to the Cart.
        ///     <remarks>Constructor that supports serialization.</remarks>
        ///     <param name="creditCardType">Name of credit card type.</param>
        ///     <param name="nameOnCard">Name of credit card holder.</param>
        ///     <param name="creditCardNumber">16 digit credit card number
[xxxx xxxx xxxx xxxx].</param>
        ///     <param name="expirationDate">Date credit card expires
[mm/yy].</param>
        ///     <param name="billingAddress">The billing address for the credit
card.</param>
        ///     <param name="approvalCode">Placeholder for the approval code
for this transaction.</param>
        /// </summary>
        public void SetPayment(String creditCardType, String nameOnCard,
                               String creditCardNumber, String expirationDate,
                               String billingAddress, String approvalCode)
        {
            ApplicationAssert.Check(cartOrderData != null, "Don't call if
IsEmpty is True", ApplicationAssert.LineNumber);
            DataTable paymentTable =
cartOrderData.Tables[OrderData.PAYMENT_TABLE];
            paymentTable.Clear();    //flush old data

            DataView paymentSource = new DataView(paymentTable);

            DataRow paymentRow = paymentTable.NewRow();

            paymentRow[OrderData.CREDIT_CARD_TYPE_FIELD] = creditCardType;
            paymentRow[OrderData.CREDIT_CARD_NUMBER_FIELD] = creditCardNumber;
            paymentRow[OrderData.EXPIRATION_DATE_FIELD] = expirationDate;
            paymentRow[OrderData.NAME_ON_CARD_FIELD] = nameOnCard;
            paymentRow[OrderData.BILLING_ADDRESS_FIELD] = billingAddress;
            paymentRow[OrderData.APPROVAL_CODE_FIELD] = approvalCode;

            paymentTable.Rows.Add(paymentRow);
        }

        /// <summary>
        ///     Add the order in OrderData and returns the transaction id.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </summary>
        public void AddOrder()
        {
            ApplicationAssert.CheckCondition(cartOrderData != null, "Order
requires data", ApplicationAssert.LineNumber);

            //Write trace log.

ApplicationLog.WriteTrace("PryEcommerce.Web.Cart.AddOrder:\r\nCustomerId: " +

cartOrderData.Tables[OrderData.CUSTOMER_TABLE].Rows[0][OrderData.PKID_FIELD].To
String());
```

```
            cartOrderData = (new OrderSystem()).AddOrder(cartOrderData);
        }

        /// <summary>
        ///     Calls OrderSystem to calculate the order summery in OrderData.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </summary>
        public void CalculateOrderSummary()
        {
            ApplicationAssert.Check(cartOrderData != null, "Don't call if
IsEmpty is True", ApplicationAssert.LineNumber);

            cartOrderData = (new OrderSystem()).GetOrderSummary(cartOrderData);
        }

        /// <value>
        ///     Property Payment is used to get the Payment datatable from
OrderData.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </value>
        public DataTable Payment
        {
            get
            {
                ApplicationAssert.CheckCondition(cartOrderData != null,
"OrderData not set", ApplicationAssert.LineNumber);
                return cartOrderData.Tables[OrderData.PAYMENT_TABLE];
            }
        }

        /// <value>
        ///     Property OrderItems is used to get the OrderItems datatable
from OrderData.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </value>
        public DataTable OrderItems
        {
            get
            {
                ApplicationAssert.Check(cartOrderData != null, "Don't call if
IsEmpty is True", ApplicationAssert.LineNumber);
                return cartOrderData.Tables[OrderData.ORDER_ITEMS_TABLE];
            }
        }

        /// <value>
        ///     Property ShippingAddress is used to get the ShippingAddress
datatable from OrderData.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </value>
        public DataTable ShippingAddress
        {
```

```
            get
            {
                ApplicationAssert.CheckCondition(cartOrderData != null,
"OrderData not set", ApplicationAssert.LineNumber);
                return cartOrderData.Tables[OrderData.SHIPPING_ADDRESS_TABLE];
            }
        }

        /// <value>
        ///     Property OrderSummary is used to get the OrderSummary datatable
from OrderData.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </value>
        public DataTable OrderSummary
        {
            get
            {
                ApplicationAssert.Check(cartOrderData != null, "Don't call if
IsEmpty is True", ApplicationAssert.LineNumber);
                return cartOrderData.Tables[OrderData.ORDER_SUMMARY_TABLE];
            }
        }

        /// <value>
        ///     Property Order is used to get the Order datatable from
OrderData.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </value>
        public DataTable Order
        {
            get
            {
                ApplicationAssert.CheckCondition(cartOrderData != null, "Don't
call if IsEmpty is True", ApplicationAssert.LineNumber);
                return cartOrderData.Tables[OrderData.ORDER_TABLE];
            }
        }

        /// <value>
        ///     Property Customer is used to set customer info into the
shopping cart.
        ///     <exception> class='System.ApplicationException'>
        ///         The cartOrderData is null.
        ///     </exception>
        /// </value>
        public CustomerData Customer
        {
            set
            {
                ApplicationAssert.Check(cartOrderData != null, "Don't call if
IsEmpty is True", ApplicationAssert.LineNumber);
                DataTable customerTable =
cartOrderData.Tables[OrderData.CUSTOMER_TABLE];
```

```
                customerTable.Rows.Clear();          //Clear previous customer
info, if any exists.
                ClearShippingAddress();              //Clear previous shipping
address

                //
                // Watch out for set to nothing
                //
                if (value != null)
                {
                    DataRow customersRow =
value.Tables[CustomerData.CUSTOMERS_TABLE].Rows[0];
                    DataRow row = customerTable.NewRow();
                    row.BeginEdit();
                    row[OrderData.PKID_FIELD] =
customersRow[CustomerData.PKID_FIELD];
                    row.EndEdit();

                    customerTable.Rows.Add(row);
                    customerTable.AcceptChanges();


SetShippingAddress((String)customersRow[CustomerData.NAME_FIELD],

(String)customersRow[CustomerData.ADDRESS_FIELD],

(String)customersRow[CustomerData.COUNTRY_FIELD],

(String)customersRow[CustomerData.PHONE_FIELD],

(String)customersRow[CustomerData.FAX_FIELD]);
                }
            }
        }


        /// <summary>
        ///     Clear the shipping address from the cart.
        /// </summary>
        public void ClearShippingAddress()
        {
            ShippingAddress.Rows.Clear();
        }

        /// <summary>
        ///     Save a customer's shipping address in the OrderData for there
order.
        ///     <remarks>Constructor that supports serialization.</remarks>
        ///     <param name="shipToName">The name of the person the order will
be shipped to.</param>
        ///     <param name="address">The address where the order will be
shipped to.</param>
        ///     <param name="country">The country where the order will be
shipped..</param>
        ///     <param name="phone">The phone number of the person the order
will be shipped to.</param>
        ///     <param name="fax">The fax number of the person the order will
be shipped to.
        ///         This field is optional.
        ///     </param>
```

```
        /// </summary>
        public void SetShippingAddress(String shipToName, String address,
String country, String phone, String fax)
        {
            DataTable addressTable = ShippingAddress;
            addressTable.Rows.Clear();         //clear former address
            DataRow row = addressTable.NewRow();

            row.BeginEdit();
            row[OrderData.SHIP_TO_NAME_FIELD] = shipToName;
            row[OrderData.ADDRESS_FIELD] = address;
            row[OrderData.COUNTRY_FIELD] = country;
            row[OrderData.PHONE_NUMBER_FIELD] = phone;
            row[OrderData.FAX_FIELD] = fax;
            row.EndEdit();

            addressTable.Rows.Add(row);
            addressTable.AcceptChanges();
        }

    } // class Cart
} // namespace PryEcommerce.Web
```

ModuleBase.cs

```
using System;
using System.Web;
using System.Web.UI;
using System.ComponentModel;
using System.Data;

using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;


namespace PryEcommerce.Web
{

    /// <summary>
    ///     The code-behind base class for all user control modules.
    ///     <remarks>
    ///         This class derives off of UserControl.
    ///     </remarks>
    /// </summary>
    public class ModuleBase : UserControl
    {
        //
        // Exception Logging constant
        //
       ///    PageBase XYZ = new PageBase();
        private const String UNHANDLED_EXCEPTION = "Unhandled Exception:";
        //
        // Session Key Constants
        //
```

```
        private const String KEY_CACHECART = "Cache:CarritoCompra:";
        private const String KEY_CACHECUSTOMER = "Cache:Customer:";

        private String basePathPrefix;

        /// <value>
        ///     Property PathPrefix is used to get or set the file path prefix
to be used by the control.
        ///     <remarks>
        ///         Sets the value PathPrefix.
        ///         Gets the value PathPrefix.
        ///     </remarks>
        /// </value>
        public  String PathPrefix
        {
            get
            {
                if (null == basePathPrefix)
                {
                                 basePathPrefix = PageBase.UrlBase;

                }

                return basePathPrefix;
            }
            set
            {
                basePathPrefix = value;
            }
        }

        /// <value>
        ///     Property Customer is used to get or set the data for the logged
on customer.
        ///     <remarks>
        ///         Sets the value Customer.
        ///         Gets the value Customer.
        ///     </remarks>
        /// </value>
        public DataSet Customer
        {
            get
            {
                try
                {
                    return (DataSet)(Session[KEY_CACHECUSTOMER]);
                }
                catch
                {
                    return (null);  // for design time
                }
            }
            set
            {
                if ( null == value )
                {
                    Session.Remove(KEY_CACHECUSTOMER);
                }
                else
                {
```

```
                    Session[KEY_CACHECUSTOMER] = value;
                }
            }
        }

        /// <summary>
        ///     Retrieves the Cart for the session, forcing it to be created
        ///     if it does not already exist.
        /// </summary>
        public Cart CarritoCompra()
        {
            return CarritoCompra(true);
        }

        /// <summary>
        ///     Retrieves the Cart for the session, optionally forcing it to
        ///     be created if it does not already exist.
        ///     <param name="forceCreate">Create the shopping cart if it does
not exist.</param>
        /// </summary>
        public Cart CarritoCompra(bool forceCreate)
        {
            //
            // Try to get the cart from the Session
            //
            Cart returnValue = (Cart)(Session[KEY_CACHECART]);

            if (null == returnValue)
            {
                //
                // If there is no cart, create it now
                //
                returnValue = new Cart();

                //
                // Save it for later
                //
                Session.Add(KEY_CACHECART, returnValue);
            }

            if (forceCreate) returnValue.EnsureWritable();

            return returnValue;
        }

    } // class ModuleBase

} // namespace PryEcommerce.Web
```

PageBase.cs

```
namespace PryEcommerce.Web
{
    using System;
    using System.Web;
    using System.Web.UI;
    using System.ComponentModel;
```

```
using System.Data;

using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

/// <summary>
///     The code-behind base class for all pages.
///     <remarks>
///         This class derives off of System.Web.UI.Page.
///     </remarks>
/// </summary>
public class PageBase : System.Web.UI.Page
{
    //
    // Exception Logging constant
    //
    private const String UNHANDLED_EXCEPTION = "Unhandled Exception:";
    //
    // Session Key Constants
    //
    private const String KEY_CACHECART = "Cache:CarritoCompra:";
    private const String KEY_CACHECUSTOMER = "Cache:Customer:";
    //
    // SSL
    //
    private  static String pageSecureUrlBase;
    private  static String pageUrlBase;
    private  static String urlSuffix;


    /// <summary>
    ///     Constructor for PageBase.
    ///     <remarks>Initialize private members.</remarks>
    /// </summary>
    public PageBase ()
    {
        try
        {
                        urlSuffix =  Context.Request.Url.Host +
Context.Request.ApplicationPath;

            pageUrlBase = @"http://" + urlSuffix;
        }
        catch
        {
            // for design time
        }
    }

    /// <value>
    ///     Property SecureUrlBase is used to get the prefix for URLs in
the Secure directory.
    /// </value>
    public  static String SecureUrlBase
    {
        get
        {
            if (pageSecureUrlBase == null )
            {
```

```
                    pageSecureUrlBase = (PryEcommerceConfiguration.EnableSsl ?
@"https://": @"http://") + urlSuffix;
                }
                    return pageSecureUrlBase;
            }
        }

        /// <value>
        ///     Property UrlBase is used to get the prefix for URLs.
        /// </value>
        ///
        public static String UrlBase
        {

            get
            {

                    return pageUrlBase;


                }
        }


        /// <value>
        ///     Property Customer is used to get or set the data for the logged
on customer.
        ///     <remarks>
        ///         Sets the value Customer.
        ///         Gets the value Customer.
        ///     </remarks>
        /// </value>
        public DataSet Customer
        {
            get
            {
                try
                {
                    return (DataSet)(Session[KEY_CACHECUSTOMER]);
                }
                catch
                {
                    return (null);  // for design time
                }
            }
            set
            {
                if ( null == value )
                {
                    Session.Remove(KEY_CACHECUSTOMER);
                }
                else
                {
                    Session[KEY_CACHECUSTOMER] = value;
                }
            }
        }

        /// <summary>
        ///     Retrieves the Cart for the session, forcing it to be created
```

```
        ///     if it does not already exist.
        /// </summary>
        public Cart CarritoCompra()
        {
            return CarritoCompra(true);
        }


        /// <summary>
        ///     Retrieves the Cart for the session, optionally forcing it to
        ///     be created if it does not already exist.
        ///     <param name="forceCreate">Create the shopping cart if it does
not exist.</param>
        /// </summary>
        public Cart CarritoCompra(bool forceCreate)
        {
            //
            // Try to get the cart from the Session
            //
            Cart returnValue = (Cart)(Session[KEY_CACHECART]);

            if (null == returnValue)
            {
                //
                // If there is no cart, create it now
                //
                returnValue = new Cart();

                //
                // Save it for later
                //
                Session.Add(KEY_CACHECART, returnValue);
            }

            if ( forceCreate ) returnValue.EnsureWritable();

            return returnValue;
        }

        /// <summary>
        ///     Handles errors that may be encountered when displaying this
page.
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected override void OnError(EventArgs e)
        {

ApplicationLog.WriteError(ApplicationLog.FormatException(Server.GetLastError(),
UNHANDLED_EXCEPTION));
            base.OnError(e);
        }
    } // class PageBase

} // namespace PryEcommerce.Web
```

Productos.cs - Productos.aspx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Collections;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.CapaNegocio;
    using PryEcommerce.SystemFramework;
    using PryEcommerce.Comun;
    using PryEcommerce.Comun.Data;

    /// <summary>
    ///     The code-behind base class for the page that displays information
    ///     about a single product.
    ///     <remarks>This page is derived from PageBase.</remarks>
    ///     <remarks>This page is explicitly cached for each individual
product.</remarks>
    /// </summary>
    public class Product : PageBase
    {
        //
        // QueryString identifiers
        //
        private const String KEY_ID = "id";
             protected System.Web.UI.WebControls.Button AddToCartButton;
             protected System.Web.UI.WebControls.Label DescriptionLabel;
             protected System.Web.UI.WebControls.Label UnitPriceLabel;
             protected System.Web.UI.WebControls.Label TitleLabel;
             protected System.Web.UI.WebControls.Image FileSpecImage;
             protected System.Web.UI.HtmlControls.HtmlForm Form1;
             protected System.Web.UI.HtmlControls.HtmlForm frmproduct;


        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Constructor for Product.
        /// </summary>
        public Product()
       {
              Page.Init += new System.EventHandler(Page_Init);
        }

        /// <summary>
        ///     Populate the page with information for the product specified by
        ///     the incoming id parameter, then mark the page as being cached.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
```

```
            if ( !IsPostBack )
            {
                int itemId;
                //
                // Read and validate the QueryString information. If this fails
to validate
                // in any way then it will throw an exception, which will
transfer control
                // to the error page.
                //
                ReadQueryString(out itemId);


                //
                DataTable productTable = (new
ProductSystem()).GetItemById(itemId).Tables[ProductData.PRODUCTS_TABLE];
                //
                // We now have a datatable, but it may still be empty.
                //
                ApplicationAssert.CheckCondition(productTable.Rows.Count == 1,
"Id Producto Incorrecto", ApplicationAssert.LineNumber);

                DataRow ProductRow = productTable.Rows[0];
                //
                // Always HTMLEncode any strings from the database
                //
                TitleLabel.Text          =
Server.HtmlEncode(ProductRow[ProductData.NAME_FIELD].ToString());
                DescriptionLabel.Text    =
Server.HtmlEncode(ProductRow[ProductData.DESCRIPTION_FIELD].ToString());
                //
                // Display the currency in locale specific manner
                //
                UnitPriceLabel.Text      =
System.String.Format("{0:C}",(Decimal)ProductRow[ProductData.UNIT_PRICE_FIELD])
;
                FileSpecImage.ImageUrl   = "images/Products/" +
ProductRow[ProductData.IMAGE_FILE_SPEC_FIELD];
                FileSpecImage.Visible    =
System.IO.File.Exists(Server.MapPath(FileSpecImage.ImageUrl));

                //
                AddToCartButton.CommandArgument = itemId + "|" +
TitleLabel.Text + "|" + ProductRow[ProductData.UNIT_PRICE_FIELD];
                ///



                // by the current application configuration.
                //
                if ( PryEcommerceConfiguration.EnablePageCache )
                {
                    //Enable  Page Caching...
                    Response.Cache.SetExpires (
DateTime.Now.AddSeconds(PryEcommerceConfiguration.PageCacheExpiresInSeconds));
                    Response.Cache.SetCacheability(HttpCacheability.Public);

                }
            }
        }
```

```
        //-----------------------------------------------------------------
        // This code was automatically generated
        //-----------------------------------------------------------------

        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
            //
            InitializeComponent();
        }

        //-----------------------------------------------------------------
        // This code was automatically generated
        //-----------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
              {
                    this.AddToCartButton.Click += new
System.EventHandler(this.AddToCartButton_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

              }

        /// <summary>
        ///     Add the item to the shopping cart and then display the contents
        ///     of the cart.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void AddToCartButton_Click(Object sender, EventArgs e)
        {
            //
            // Get the info from the button's CommandArguments
            //
            string arguments = AddToCartButton.CommandArgument;
            char [] parsChar = {'|'};
            string[] addArgs = arguments.Split(parsChar);
            //
            // Add the item to the shoppingcart
            //
            CarritoCompra().AddItem(Int32.Parse(addArgs[0]), addArgs[1] ,
Decimal.Parse(addArgs[2]));

            Response.Redirect("CarritoCompra.aspx", false);
        }

             public void AddToCartButtonFlash()
```

```
                {
                        //
                        // Get the info from the button's CommandArguments
                        //
                        string arguments = AddToCartButton.CommandArgument;
                        char [] parsChar = {'|'};
                        string[] addArgs = arguments.Split(parsChar);
                        //
                        // Add the item to the shoppingcart
                        //
                        CarritoCompra().AddItem(Int32.Parse(addArgs[0]), addArgs[1]
, Decimal.Parse(addArgs[2]));


                        Response.Redirect("CarritoCompra.aspx", false);
                }

        //----------------------------------------------------------------
        // Sub ReadQueryString:
        //   Validate the incoming QueryString information. If the query
        //   string is malformed, return false.
        // Parameters:
        //   [out] Id: The product identifier
        //----------------------------------------------------------------
        private bool ReadQueryString(out int Id)
        {
            Id = -1;
            //
            // Let this function throw an exception if the QueryString contains
            // out of range or poorly formed data.
            //
            String IdString = Request.QueryString[KEY_ID].ToString();

            if ( IdString != String.Empty )
            {
                Id = Int32.Parse(IdString);
            }

            ApplicationAssert.CheckCondition( Id >= 0, "Invalid Item Id
Number", ApplicationAssert.LineNumber);

            return true;
        }

    } // class Product

} // namespace PryEcommerce.Web
```

CarritoCompra.cs - CarritoCompra.aspx

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Data;
```

```
using PryEcommerce.Comun;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{

    /// <summary>
    ///     The code-behind base class for the page that displays the
    ///     current contents of the shopping cart.
    ///     <remarks>
    ///         Special considerations:
    ///         This class requires read/write session state support
    ///         to enable in-place editing of the shopping cart contents
    ///         using the WebDataGrid control. This class is a view only;
    ///         the Cart class (Cart.cls) is responsible for managing
    ///         the actual shopping cart data.
    ///     </remarks>
    ///     <remarks>This page is derived from PageBase.</remarks>
    /// </summary>
    public class CarritoCompra : PageBase
    {
        /// <value>The range validator used to limit the number of each item
that you can have in the cart.</value>
        protected System.Web.UI.WebControls.RangeValidator
QuantityRangeValidator;
        /// <value>The required validator used ensure that the QuantityTextBox
contains a value.</value>
        protected System.Web.UI.WebControls.RequiredFieldValidator
QuantityRequiredFieldValidator;
              protected System.Web.UI.HtmlControls.HtmlForm Form1;
              protected System.Web.UI.WebControls.Label EmptyCartLabel;
              protected System.Web.UI.WebControls.Button UpdateButton;
              protected System.Web.UI.WebControls.ValidationSummary
QuantityValidationSummary;
              protected System.Web.UI.WebControls.DataGrid CartItemsDataGrid;
              protected System.Web.UI.WebControls.HyperLink CheckOutHyperLink;
        /// <value>The text box used to allow editing of the quantity per
item.</value>
        protected System.Web.UI.WebControls.TextBox QuantityqextBox;

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Constructor for ShoppingCart.
        /// </summary>
        public CarritoCompra ()
       {
               Page.Init += new System.EventHandler(Page_Init);
        }

        /// <summary>
        ///     Retrieve the current shopping cart information from the session
        ///     and populate the grid.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
```

```
        {
            bool cartHasItems = false;  //Set if the cart contains items


       if (!IsPostBack)
            {
                //
                // Retrieve the shopping cart information, open a DataSource on
the
                //   table and bind the DataGrid to the DataSource.
                //
                Cart shoppingCart = CarritoCompra(false);
                if (!shoppingCart.IsEmpty)
                {
                    CartItemsDataGrid.DataSource =
(ICollection)shoppingCart.OrderItems.DefaultView;
                    CartItemsDataGrid.DataBind();
                    cartHasItems = true;
                }

                //
                // Set visibility of displayed items to correspond to
                //   whether or not we have items in the cart.
                //
               // ShoppingCartPanel.Visible = cartHasItems;
                CartItemsDataGrid.Visible = cartHasItems;
                CheckOutHyperLink.NavigateUrl = PageBase.SecureUrlBase +
"/secure/Pago.aspx";
                CheckOutHyperLink.Visible= cartHasItems;
                EmptyCartLabel.Visible = !cartHasItems;
            }
        }

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
            //
            InitializeComponent();
        }

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
```

```
        private void InitializeComponent()
              {
                    this.UpdateButton.Click += new
System.EventHandler(this.UpdateButton_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

              }

        /// <summary>
        ///     Update the quantity of item(s) in the shopping cart and then
display the contents
        ///     of the cart.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void UpdateButton_Click(Object sender, EventArgs e)
        {
            bool cartHasItems = false;       //Set if the cart contains items

            Cart shoppingCart = CarritoCompra(false);
            if (!shoppingCart.IsEmpty)
            {
                if (!Page.IsValid)
                {
                    return;
                }

                //
                // Validators have run so we can assume valid input
                //

                //
                // reset the quanities for all the items in cart
                //
                DataRowCollection orderRows = shoppingCart.OrderItems.Rows;
                int i = 0;
                foreach (DataGridItem item in CartItemsDataGrid.Items)
                {
                    //update the quantity
                    orderRows[i][1] =
Int32.Parse(((TextBox)item.FindControl("QuantityTextBox")).Text);
                    ++i;
                }

                shoppingCart.UpdateItems();

                cartHasItems = !shoppingCart.IsEmpty;
                if (cartHasItems)
                {
                    //Bind the DataGrid to the items
                    CartItemsDataGrid.DataSource =
(ICollection)shoppingCart.OrderItems.DefaultView;
                    CartItemsDataGrid.DataBind();
                }
            }

            //
            // Set visibility of displayed items to correspond to
            //   whether or not we have items in the cart.
```

```
            //
            //ShoppingCartPanel.Visible = cartHasItems;
            CartItemsDataGrid.Visible = cartHasItems;
            CheckOutHyperLink.Visible= cartHasItems;
            EmptyCartLabel.Visible = !cartHasItems;
        }

             private void CartItemsDataGrid_SelectedIndexChanged(object sender,
System.EventArgs e)
             {

             }

    } // class ShoppingCart
} // namespace PryEcommerce.Web
```

Categorias.cs - Categorias.aspx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Collections;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.CapaNegocio;
    using PryEcommerce.Comun;
    using PryEcommerce.Comun.Data;
    using PryEcommerce.SystemFramework;

    /// <summary>
    ///     The code-behind base class for the page that displays items
    ///     for a category.
    ///     <remarks>This page is derived from PageBase.</remarks>
    ///     <remarks>This page is explicitly cached for each
category.</remarks>
    /// </summary>
    public class Categorias : PageBase
    {

        //
        // QueryString identifiers
        //
        private const String KEY_ID = "id";
             protected System.Web.UI.HtmlControls.HtmlForm Form1;
             protected System.Web.UI.WebControls.Label CategoryDesciptionLabel;
             protected System.Web.UI.WebControls.DataList CategoryList;
        //
        // DataView for the items in the selected category
        //
        private DataView itemView;

        //-----------------------------------------------------------------
```

```
        // This code was automatically generated
        //-----------------------------------------------------------------
        /// <summary>
        ///     Constructor for Categories.
        /// </summary>
        public Categorias()
       {
               Page.Init += new System.EventHandler(Page_Init);
        }


        /// <value>
        ///     Property ItemList is used to get the DataView that contains the
list of products for the
        ///     currently selected category.
        /// </value>
        public DataView ItemList
        {
            get
            {
                return itemView;
            }
        }

        /// <summary>
        ///     Populate the page with the category information then mark
        ///     the page as cached.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            int categoryID = -1;    // Default value
            if (!IsPostBack)
            {
                //
                // Read and validate the QueryString information. If this fails
to validate
                // in any way then it will throw an exception, which will
transfer control
                // to the error page.
                //
                ReadQueryString(out categoryID);

                ProductSystem productSystem;  // Facade layer object
                CategoryData  categorySet;    // Category DataSet, returned
from facade layer
                DataView      categoryView;

                //
                // Retrieve a category, it's parent and children from the
database.
                // Note: at present Duwamish does not support nested
categories. As a result,
                //       code is used to retrieve the name of the selected
category.
                //
                productSystem = new ProductSystem();
```

```
                categorySet   = productSystem.GetCategories(categoryID);
                //
                // Retrieve information from the CategoryData
                //
                categoryView =
categorySet.Tables[CategoryData.CATEGORIES_TABLE].DefaultView;
                categoryView.RowFilter = CategoryData.PKID_FIELD + " = " +
categoryID.ToString();
                //
                // Check that we actually have category data
                //
                ApplicationAssert.CheckCondition(categoryView.Count != 0,
"Invalid categoryID", ApplicationAssert.LineNumber);
                //
                // Read the description for the Category
                //
                CategoryDesciptionLabel.Text =
Server.HtmlEncode((String)categoryView[0][CategoryData.DESCRIPTION_FIELD]);
                //
                // Read the products in the category
                //
                ProductData itemSet =
productSystem.GetCategoryItems(categoryID);
                //
                // Save the view for data binding
                //
                itemView =
itemSet.Tables[ProductData.PRODUCTS_TABLE].DefaultView;
                //
                // Set the DailyPick category
                //


              /////  ModuloSeleccionDia.CategoryID = categoryID;
                //
                // If everything succeeded, then enable page caching as
indicated
                // by the current application configuration.
                //
                if (PryEcommerceConfiguration.EnablePageCache)
                {
                    //Enable  Page Caching...
                    Response.Cache.SetExpires (
DateTime.Now.AddSeconds(PryEcommerceConfiguration.PageCacheExpiresInSeconds));
                    Response.Cache.SetCacheability(HttpCacheability.Public);
                }
                //
                // Remind the page that it needs bind its controls to the data.
                //
                DataBind();
            }
        }


        //------------------------------------------------------------------
        // This code was automatically generated
        //------------------------------------------------------------------
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
```

```
        ///     <param name="e">An EventArgs that contains the event 
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form 
Designer.
            //
            InitializeComponent();
        }

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
               {
                        this.CategoryList.SelectedIndexChanged += new 
System.EventHandler(this.CategoryList_SelectedIndexChanged);
                        this.Load += new System.EventHandler(this.Page_Load);

               }


        //----------------------------------------------------------------
        // Function ReadQueryString:
        //   Retrieves the query string information and validates it.
        //     currently selected category.
        // Parameters:
        //   out int id - the id value from the query string
        // Returns:
        //   bool - true for success, false for failure
        //----------------------------------------------------------------
        private bool ReadQueryString(out int id)
        {
            String idString;  //The requested category id in string format

            id = -1;
            //
            // Get the category id from the query string and validate the data.
            // Let this function throw an exception if the QueryString contains
            // out of range or poorly formed data.
            //
            idString = Request.QueryString[KEY_ID].ToString();

            if (idString != String.Empty)
            {
                id = Int32.Parse(idString);
            }

            return true;
        }

              private void CategoryList_SelectedIndexChanged(object sender, 
System.EventArgs e)
               {
```

```
            }

    } // class Categories

} // namespace PryEcommerce.Web
```

ResultadoBusqueda.cs - ResultadoBusqueda.aspx

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Data;

using PryEcommerce.CapaNegocio;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{

    /// <summary>
    ///     The code-behind base class for the page that displays product
    ///     search results, based on the parameters passed to it from
    ///     search.aspx.
    ///     <remarks>This page is derived from PageBase.</remarks>
    /// </summary>
    public class ResultadoBusqueda : PageBase
    {

              protected  PryEcommerce.Web.ModuloBusqueda ModuloBusqueda;

        //member variables
        private String searchText;
        private String searchType;
        private DataView resultsDataView;
        /// <value>The label for the displayed result's search type.</value>
        protected System.Web.UI.WebControls.Label SearchTypeLabel;
        /// <value>The label for the displayed result's search text.</value>
        protected System.Web.UI.WebControls.Label SearchTextLabel;
        /// <value>The label for the displayed search result's items.</value>
        protected System.Web.UI.WebControls.DataList ResultsList;
        /// <value>The label for the number of items found.</value>
        protected System.Web.UI.WebControls.Label ResultCountLabel;

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Constructor for SearchResults.
        /// </summary>
        public ResultadoBusqueda()
```

```
      {
              Page.Init += new System.EventHandler(Page_Init);
        }

        /// <value>
        ///     Property SearchResultsList is used to get the DataView that
contains the list of items for the
        ///     search.
        ///     <remarks>A DataList loaded with search results.</remarks>
        /// </value>
        public DataView SearchResultsList
        {
            get
            {
                return resultsDataView;
            }
        }

        /// <summary>
        ///     Performs search and loads datalist with results for display.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            if (IsPostBack)
            {

                //
                // Avoid a refresh problem if the user has left the search
                // string empty by rerunning last search.
                //

                           if ( ((TextBox)
ModuloBusqueda.FindControl("SearchTextBox")).Text.Trim() != String.Empty )
                {
                    return;
                }
             }

            PryEcommerce.Comun.Data.ProductData.SearchTypeEnum searchEnumType;
//type of search used
            DataSet resultsSet = null;                            //search results
returned from facade

            searchEnumType =
(PryEcommerce.Comun.Data.ProductData.SearchTypeEnum)Int32.Parse(Request.QuerySt
ring["type"]);
            searchType = Request.QueryString["fullType"].ToString();
            searchText = Request.QueryString["text"].ToString();

            if (searchText != String.Empty)
            {
                try
                {
                    resultsSet = (new
ProductSystem()).GetSearchItems(searchEnumType, searchText);
                }
```

```
                    catch (ApplicationException)
                    {
                        // Fall into next if block for remaining settings
                    }
                    // Other exceptions throw
                }

                //handle error case where resultsSet is Nothing
                if (null == resultsSet)
                {
                    SearchTypeLabel.Text = Server.HtmlEncode(searchType);
                    ResultCountLabel.Text = Server.HtmlEncode("0");
                    SearchTextLabel.Text = Server.HtmlEncode(searchText);

                    return;
                }

                DataTable table = resultsSet.Tables[0];
                resultsDataView = table.DefaultView;

                SearchTypeLabel.Text = Server.HtmlEncode(searchType);
                ResultCountLabel.Text =
Server.HtmlEncode(table.Rows.Count.ToString());
                SearchTextLabel.Text = Server.HtmlEncode(searchText);
                //
                // Remind the page that it needs bind its controls to there data.
                //
                DataBind();
            }

            //----------------------------------------------------------------
            // This code was automatically generated
            //----------------------------------------------------------------
            /// <summary>
            ///     Used to wireup this page's event handlers.
            ///     <param name="sender">The source of the event.</param>
            ///     <param name="e">An EventArgs that contains the event
data.</param>
            /// </summary>
            protected void Page_Init(object sender, EventArgs e)
            {
                //
                // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
                //
                InitializeComponent();
            }

            //----------------------------------------------------------------
            // This code was automatically generated
            //----------------------------------------------------------------
            /// <summary>
            ///    Required method for Designer support - do not modify
            ///    the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
                  {
                        this.ResultsList.SelectedIndexChanged += new
System.EventHandler(this.ResultsList_SelectedIndexChanged);
                        this.Load += new System.EventHandler(this.Page_Load);
```

```
            }

            private void ResultsList_SelectedIndexChanged(object sender,
System.EventArgs e)
            {

            }

    } // class SearchResults
} // namespace PryEcommerce.Web
```

ResultadoBusquedaAsistida.cs - ResultadoBusquedaAsistida.aspx

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Data;

using PryEcommerce.CapaNegocio;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{

    /// <summary>
    ///     The code-behind base class for the page that displays product
    ///     search results, based on the parameters passed to it from
    ///     search.aspx.
    ///     <remarks>This page is derived from PageBase.</remarks>
    /// </summary>
    public class ResultadoBusquedaAsistida : PageBase
    {

             protected  PryEcommerce.Web.ModuloBusquedaAsistida
ModuloBusquedaAsistida;

        //member variables


            private String Valor1 ="";
            private int Valor2 =0;
            private int Valor3 =0;
            private int Valor4 =0;
            private int Valor5 =0;
            private int Valor6 =0;
            protected System.Web.UI.WebControls.DataList ResultsList;
            protected System.Web.UI.WebControls.Label ResultCountLabel;
            protected System.Web.UI.WebControls.Label SearchTextLabel;
            protected System.Web.UI.WebControls.Label SearchTypeLabel;
```

```
        private DataView resultsDataView;

        //------------------------------------------------------------------
        // This code was automatically generated
        //------------------------------------------------------------------
        /// <summary>
        ///     Constructor for SearchResults.
        /// </summary>
        public ResultadoBusquedaAsistida()
       {
               Page.Init += new System.EventHandler(Page_Init);
        }

        /// <value>
        ///     Property SearchResultsList is used to get the DataView that
contains the list of items for the
        ///     search.
        ///     <remarks>A DataList loaded with search results.</remarks>
        /// </value>
        public DataView SearchResultsList
        {
            get
            {
                return resultsDataView;
            }
        }

        /// <summary>
        ///     Performs search and loads datalist with results for display.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            if (IsPostBack)
            {

                //
                // Avoid a refresh problem if the user has left the search
                // string empty by rerunning last search.
                //


             }


            DataSet resultsSet = null;                         //search results
returned from facade


                    Valor1 = Request.QueryString["Textobusqueda"].ToString();
                    Valor2 =  Convert.ToInt16(Request.QueryString["Category"])
;
                    Valor3 =  Convert.ToInt16(Request.QueryString["Edad"]);
                    Valor4 =  Convert.ToInt16(Request.QueryString["Sexo"]);
                    Valor5 =  Convert.ToInt16(Request.QueryString["Estado"]);
                    Valor6 =  Convert.ToInt16(Request.QueryString["Monto"]);
```

```
                    // @Name,@CategoryId,@Years,@Sex,@State,@Cost

            if (Valor1 != String.Empty)
            {
                try
                {
                    resultsSet = (new
ProductSystem()).GetSearchItemsAsistent(Valor1,Valor2,Valor3,Valor4,Valor5,Valo
r6);

                }

                catch (ApplicationException)
                {
                    // Fall into next if block for remaining settings
                }
                // Other exceptions throw
            }

            //handle error case where resultsSet is Nothing
            if (null == resultsSet)
            {
               //SearchTypeLabel.Text = Server.HtmlEncode(searchType);
                ResultCountLabel.Text = Server.HtmlEncode("0");
                SearchTextLabel.Text = Valor1;

                return;
            }

            DataTable table = resultsSet.Tables[0];
            resultsDataView = table.DefaultView;

        //    SearchTypeLabel.Text = Server.HtmlEncode(searchType);
            ResultCountLabel.Text =
Server.HtmlEncode(table.Rows.Count.ToString());
            SearchTextLabel.Text = Valor1;
            //
            // Remind the page that it needs bind its controls to there data.
            //
            DataBind();
        }

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
            //
            InitializeComponent();
```

```
        }

        //-----------------------------------------------------------------
        // This code was automatically generated
        //-----------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
              {
                    this.ResultsList.SelectedIndexChanged += new
System.EventHandler(this.ResultsList_SelectedIndexChanged);
                    this.Load += new System.EventHandler(this.Page_Load);

              }

              private void ResultsList_SelectedIndexChanged(object sender,
System.EventArgs e)
              {

              }

    } // class SearchResults
} // namespace PryEcommerce.Web
```

Recordatorio.cs - Recordatorio.aspx

```
namespace PryEcommerce.Web
{

       using System;
       using System.Collections;
       using System.ComponentModel;

       using System.Drawing.Drawing2D;

       using System.Data;
       using System.Drawing;
       using System.Web;
       using System.Web.SessionState;
       using System.Web.UI;
       using System.Web.UI.WebControls;
       using System.Web.UI.HtmlControls;
       using System.Web.Security;


       using PryEcommerce.CapaNegocio;
       using PryEcommerce.Comun;
       using PryEcommerce.Comun.Data;
       using PryEcommerce.SystemFramework;

       /// <summary>
       /// Descripción breve de _default.
       /// </summary>
       public class Recordatorio : PageBase
       {
```

```
		private DataView resultsDataView;


		private bool         moduleEditMode;         //edit or create mode
- default is create
		private ContactCustomerData moduleContactCustomerInfo;

		private int VarPkId;
		protected System.Web.UI.HtmlControls.HtmlForm Form1;
		protected System.Web.UI.HtmlControls.HtmlTableCell Td1;
		protected System.Web.UI.WebControls.Button BtnSaveR;
		protected System.Web.UI.WebControls.DropDownList Ddldaysbefore;
		protected System.Web.UI.WebControls.DropDownList DdlMont;
		protected System.Web.UI.WebControls.DropDownList DdlDay;
		protected System.Web.UI.WebControls.DropDownList DdlCause;
		protected System.Web.UI.WebControls.TextBox TxtNameContact;
		protected System.Web.UI.WebControls.TextBox TxtID1;
		protected System.Web.UI.WebControls.TextBox txtNameList;
		protected System.Web.UI.WebControls.Calendar CldDates;
		protected System.Web.UI.WebControls.DataGrid DgContacts;
		protected System.Web.UI.WebControls.Button BtnBuscar;
		protected System.Web.UI.WebControls.TextBox SearchTextBox;
		protected System.Web.UI.WebControls.RadioButton RBPorFecha;
		protected System.Web.UI.WebControls.RadioButton RBParticular;
		protected System.Web.UI.WebControls.RadioButton RBGrupo;
	    private String VarEmail;



		public Recordatorio()
		{
			Page.Init += new System.EventHandler(Page_Init);
		}

		//protected System.Web.UI.HtmlControls.HtmlForm Form1;

		public DataView SearchResultsList
		{
			get
			{
				return resultsDataView;
			}
		}


		private void Page_Load(object sender, System.EventArgs e)
		{

		//	string emailcustomer;
		//	string passwordcustomer;
			string datetoday;
			string txtsearch;


			//
		   // Ensure that the customer data is still in the session.
		   //
		   ApplicationAssert.Check(Customer != null, "No Customer at Account
Edit", ApplicationAssert.LineNumber);
```

```
                    if (null == Customer)
                    {
                          //
                          // Session has probably timed out
                          //
                          CarritoCompra().Customer = null;
                          //
                          // Clear the logon cookie
                          //
                          FormsAuthentication.SignOut();
                          //
                          // Try to get back here and force reauthentication
                          //


                          Response.Redirect("Secure/Cuenta.aspx", false);

                          VarEmail ="";
                          VarPkId  =0;
                          txtsearch="";


                    }
                    else
                    {

                          DataRow row =
Customer.Tables[CustomerData.CUSTOMERS_TABLE].Rows[0];

                          VarEmail = (string)row[CustomerData.EMAIL_FIELD];
                          VarPkId  = (int)row[CustomerData.PKID_FIELD];
                          txtsearch="";

                    }






              DataSet resultsSet = null;  //resultado de la busqueda desde la
capa negocio



              //      passwordcustomer="123456";
                    datetoday= System.DateTime.Now.ToShortDateString()  ;//
"17/07/2004";


                    if (IsPostBack)
                    {
                datetoday=CldDates.SelectedDate.Date.ToShortDateString()  ;

                    }

                    if (VarEmail != String.Empty)
                    {
```

```
                try
                {
                    resultsSet = (new
ContactCustomerSystem()).GetContactCustomerByEmail(
VarEmail,txtsearch,datetoday);


                }
                catch (ApplicationException)
                {
                    // Fall into next if block for remaining
settings
                }
                // Other exceptions throw
            }

            //handle error case where resultsSet is Nothing
            if (null == resultsSet)
            {
                ////SearchTypeLabel.Text =
Server.HtmlEncode(searchType);
                ////ResultCountLabel.Text = Server.HtmlEncode("0");
                ////SearchTextLabel.Text =
Server.HtmlEncode(searchText);


                return;
            }



            DataTable table = resultsSet.Tables[0];





            resultsDataView = table.DefaultView;





            ////SearchTypeLabel.Text = Server.HtmlEncode(searchType);
            ///ResultCountLabel.Text =
Server.HtmlEncode(table.Rows.Count.ToString());
            ///SearchTextLabel.Text = Server.HtmlEncode(searchText);
            //
            // Remind the page that it needs bind its controls to there
data.
            //




            DgContacts.DataSource =resultsDataView;



            DataBind();
```

```
                        DgContacts.Columns[1].Visible =false;
                        DgContacts.Columns[5].Visible =false;
                        DgContacts.Columns[6].Visible =false;




        }

                protected void Page_Init(object sender, EventArgs e)
                {
                        //
                        // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
                        //
                        InitializeComponent();
                }

                #region Web Form Designer generated code
                override protected void OnInit(EventArgs e)
                {
                        //
                        // CODEGEN: llamada requerida por el Diseñador de Web Forms
ASP.NET.
                        //
                        InitializeComponent();
                //      base.OnInit(e);
                }

                /// <summary>
                /// Método necesario para admitir el Diseñador, no se puede
modificar
                /// el contenido del método con el editor de código.
                /// </summary>
                private void InitializeComponent()
                {
                        this.BtnBuscar.Click += new
System.EventHandler(this.BtnBuscar_Click);
                        this.Load += new System.EventHandler(this.Page_Load);

                }
                #endregion

                private void DataList1_SelectedIndexChanged(object sender,
System.EventArgs e)
                {

                }

                private void BtnBuscar_Click(object sender, System.EventArgs e)
                {

                        //      string passwordcustomer;
                        string TxtSearch;
                        string TxtDateRemind;

                        int ValOpcion;
```

```
                    DataSet resultsSet = null;  //resultado de la busqueda
desde la capa negocio

                    ValOpcion=0;

                    if(RBGrupo.Checked==true)
                          ValOpcion=1;

                    if(RBParticular.Checked==true)
                          ValOpcion=2;

                    if(RBPorFecha.Checked==true)
                          ValOpcion=3;

                    TxtSearch="%"+SearchTextBox.Text.Trim()+"%";

                    TxtDateRemind=CldDates.SelectedDate.ToShortDateString() ;

                    if (IsPostBack)
                    {


                    }


                    DgContacts.Style.Clear();
                    DgContacts.ItemStyle.Reset();

                    if (VarEmail != String.Empty)
                    {
                          try
                          {
                                resultsSet = (new
ContactCustomerSystem()).GetSearchContactCustomer(
VarEmail,TxtSearch,TxtDateRemind,ValOpcion);


                          }
                          catch (ApplicationException)
                          {
                                // Fall into next if block for remaining
settings
                          }
                          // Other exceptions throw
                    }

                    //handle error case where resultsSet is Nothing
                    if (null == resultsSet)
                    {
                          ////SearchTypeLabel.Text =
Server.HtmlEncode(searchType);
                          ////ResultCountLabel.Text = Server.HtmlEncode("0");
                          ////SearchTextLabel.Text =
Server.HtmlEncode(searchText);


                          return;
                    }
```

```
                    DataTable table = resultsSet.Tables[0];

                    resultsDataView = table.DefaultView;

                    ////SearchTypeLabel.Text = Server.HtmlEncode(searchType);
                    ///ResultCountLabel.Text =
Server.HtmlEncode(table.Rows.Count.ToString());
                    ///SearchTextLabel.Text = Server.HtmlEncode(searchText);
                    //
                    // Remind the page that it needs bind its controls to there
data.
                    //

                    DgContacts.DataSource =resultsDataView;
                    DataBind();

                     DgContacts.Columns[1].Visible =false;
                  DgContacts.Columns[5].Visible =false;

             }

             private void Txtargsearch_TextChanged(object sender,
System.EventArgs e)
             {

             }

             private void DgContacts_SelectedIndexChanged(object sender,
System.EventArgs e)
             {
                    String VarDia;
                    String VarMes;
                    String VarDiasAntes;

                    String VarMotivo;


      txtNameList.Text=DgContacts.SelectedItem.Cells[1].Text.ToString();

      TxtNameContact.Text=DgContacts.SelectedItem.Cells[2].Text.ToString();

                    TxtID1.Text
=DgContacts.SelectedItem.Cells[6].Text.ToString();


      VarDia=DgContacts.SelectedItem.Cells[3].Text.ToString().Substring(0,2) ;

      VarMes=DgContacts.SelectedItem.Cells[3].Text.ToString().Substring(3,2);

                    VarMotivo=DgContacts.SelectedItem.Cells[4].Text.ToString();


      VarDiasAntes=DgContacts.SelectedItem.Cells[5].Text.ToString();


                    //DdlDay.Items.FindByText(VarDia);



                    switch (VarMes)
```

```
{
case "01":
        VarMes="Enero";
        break;
case "02":
        VarMes="Febrero";
        break;
case "03":
        VarMes="Marzo";
        break;
case "04":
        VarMes="Abril";
        break;
case "05":
        VarMes="Mayo";
        break;
case "06":
        VarMes="Junio";
        break;
case "07":
        VarMes="Julio";
        break;
case "08":
        VarMes="Agosto";
        break;
case "09":
        VarMes="Septiembre";
        break;
case "10":
        VarMes="Octubre";
        break;
case "11":
        VarMes="Noviembre";
        break;
case "12":
        VarMes="Diciembre";
        break;
default:
        VarMes="";
        break;
}

moduleEditMode=true;

for(int i=0;i<DdlDay.Items.Count;i++)
{
        if(VarDia==DdlDay.Items[i].Text.ToString())
        {
                DdlDay.SelectedIndex=i;
                break;
        }

}

for(int i=0;i<DdlMont.Items.Count;i++)
{
        if(VarMes==DdlMont.Items[i].Text.ToString())
        {
                DdlMont.SelectedIndex=i;
                break;
```

```
                }
        }


        for(int i=0;i<DdlCause.Items.Count;i++)
        {
                if(VarMotivo==DdlCause.Items[i].Text.ToString())
                {
                        DdlCause.SelectedIndex=i;
                        break;
                }
        }

        for(int i=0;i<Ddldaysbefore.Items.Count;i++)
        {
if(VarDiasAntes.Trim()==Ddldaysbefore.Items[i].Text.ToString().Trim())
                {
                        Ddldaysbefore.SelectedIndex=i;
                        break;
                }
        }


}

private void BtnSaveR_Click(object sender, System.EventArgs e)
{

String VarMotivo="";
        String VarFecha="";
        String VarDiasantes="";
        int VarPkId1=0;
        String VarMes;


        //VarPkId=3;

        DataRow         contactcustomerRow;

        bool            retVal          = false;


        txtNameList.Text=txtNameList.Text.Trim();
        TxtNameContact.Text=TxtNameContact.Text.Trim();
        VarMotivo = DdlCause.SelectedItem.Text.Trim();
        VarDiasantes= Ddldaysbefore.SelectedItem.Text.Trim();
        VarPkId1=Convert.ToInt16(TxtID1.Text.Trim());



        VarMes=DdlMont.SelectedItem.Text.Trim();

        switch (VarMes)
        {
                case "Enero":
                        VarMes="01";
                        break;
                case "Febrero":
                        VarMes="02";
```

```
                    break;
                case "Marzo":
                    VarMes="03";
                    break;
                case "Abril":
                    VarMes="04";
                    break;
                case "Mayo":
                    VarMes="05";
                    break;
                case "Junio":
                    VarMes="06";
                    break;
                case "Julio":
                    VarMes="07";
                    break;
                case "Agosto":
                    VarMes="08";
                    break;
                case "Septiembre":
                    VarMes="09";
                    break;
                case "Octubre":
                    VarMes="10";
                    break;
                case "Noviembre":
                    VarMes="11";
                    break;
                case "Diciembre":
                    VarMes="12";
                    break;
                default:
                    VarMes="";
                    break;
            }

            VarFecha = DdlDay.SelectedItem.Text.Trim()+"/"+
VarMes.Trim() +"/2004";


            foreach(IValidator val in Page.Validators)
            {
                val.Validate();
            }


            if(VarPkId1!=0)
                moduleEditMode=true;

            if ( Page.IsValid )
            {
                if ( moduleEditMode )
                {
                    //
                    // WORKAROUND: Create and Update the dataset.
Ideally we would do Customer.Clone()
                    //          here and update the row,
however DataSet.Clone() is broken. Instead,
                    //          we create a new dataset, add a
row, accept the changes then update
```

```
                                //              the row before passing it to
the facade layer. This ensures that
                                //              the row is marked as dirty when
the data access layer does the
                                //              update call.
                                //


                                moduleContactCustomerInfo = new
ContactCustomerData();
                                contactcustomerRow    =
moduleContactCustomerInfo.Tables[ContactCustomerData.CONTACT_REMINDERTABLE
].NewRow();


contactcustomerRow[ContactCustomerData.PKID_FIELD]    =     VarPkId;

       contactcustomerRow[ContactCustomerData.PKID1_FIELD]    =     VarPkId1;

       contactcustomerRow[ContactCustomerData.LISTNAME_FIELD ]    =
txtNameList.Text;

       contactcustomerRow[ContactCustomerData.NAME_FIELD] =
TxtNameContact.Text  ;

       contactcustomerRow[ContactCustomerData.DATEREMIND_FIELD]   = VarFecha ;

       contactcustomerRow[ContactCustomerData.CAUSEREMIND_FIELD]  = VarMotivo ;

       contactcustomerRow[ContactCustomerData.DAYTOREMIND_FIELD]  = VarDiasantes
;

                                // Add the row to the dataset
                                //


       moduleContactCustomerInfo.Tables[ContactCustomerData.CONTACT_REMINDERTABL
E].Rows.Add(contactcustomerRow);

                                moduleContactCustomerInfo.AcceptChanges();

                         //
       contactcustomerRow[ContactCustomerData.PKID1_FIELD] =
Customer.Tables[ContactCustomerData.CONTACT_REMINDERTABLE].Rows[0][ContactCusto
merData.PKID1_FIELD];
                                //
                                // WORKAROUND: End of workaround.
                                //
                                // Save changes to the database
                                //
                                retVal = (new
ContactCustomerSystem()).UpdateContactCustomer(moduleContactCustomerInfo);

                         }
                         else
                         {
                                retVal = (new
ContactCustomerSystem()).CreateContactCustomer(VarPkId
,txtNameList.Text,TxtNameContact.Text,VarFecha,VarMotivo,VarDiasantes,out
moduleContactCustomerInfo);
```

```
				}
			}

			TxtNameContact.Text="";

			//
			// Store customer in the Session
			//
////////				if ( retVal && ( moduleContactCustomerInfo != null )
)
////////				{
////////					//
////////					// 1. update customer in session
////////					// 2. update customer in cart
////////					//
////////////////						ContactCustomer			=
moduleContactCustomerInfo;
////////////////						CarritoCompra().Customer =
moduleContactCustomerInfo;
////////				}
////////
////////				return retVal;
		}

		private void CldDates_SelectionChanged(object sender,
System.EventArgs e)
		{

		}
	}
}
```

EnGrupos.cs - EnGrupos.aspx

```
namespace PryEcommerce.Web
{

	using System;
	using System.Collections;
	using System.ComponentModel;
	using System.Data;
	using System.Drawing;
	using System.Web;
	using System.Web.SessionState;
	using System.Web.UI;
	using System.Web.UI.WebControls;
	using System.Web.UI.HtmlControls;
	using System.Text;



	/// <summary>
	/// Descripción breve de _default.
	/// </summary>
	public class EnGrupos : PageBase
	{
		protected System.Web.UI.WebControls.Button Button3;
```

```
		protected System.Web.UI.WebControls.Button BtnRegistrar;
		protected System.Web.UI.HtmlControls.HtmlGenericControl DIV1;
		protected System.Web.UI.WebControls.Button Button2;
		protected System.Web.UI.WebControls.Label Label1;
		protected System.Web.UI.WebControls.Button BtnRqBn;


		public EnGrupos()
		{
			Page.Init += new System.EventHandler(Page_Init);
		}

		//protected System.Web.UI.HtmlControls.HtmlForm Form1;

		private void Page_Load(object sender, System.EventArgs e)
		{
			// Introducir aquí el código de usuario para inicializar la
página

			if (!IsPostBack)
			{
				//
				// Evals true first time browser hits the page
				//
			}

		}

		protected void Page_Init(object sender, EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP+ Windows Form
Designer.
			//
			InitializeComponent();
		}

		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: llamada requerida por el Diseñador de Web Forms
ASP.NET.
			//
			InitializeComponent();
		//	base.OnInit(e);
		}

		/// <summary>
		/// Método necesario para admitir el Diseñador, no se puede
modificar
		/// el contenido del método con el editor de código.
		/// </summary>
		private void InitializeComponent()
		{
			this.BtnRqBn.Click += new
System.EventHandler(this.BtnRqBn_Click);
			this.Button2.Click += new
System.EventHandler(this.Button2_Click);
```

```
                    this.Button3.Click += new
System.EventHandler(this.Button3_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

             }
             #endregion

             private void BtnRqBn_Click(object sender, System.EventArgs e)
             {

                    // Redirect to page...

                    Response.Redirect(PageBase.UrlBase + "/EnGruposB.aspx");


             }

             private void Button2_Click(object sender, System.EventArgs e)
             {

             Response.Redirect(PageBase.UrlBase + "/EnGruposD.aspx");
             }

             private void Button3_Click(object sender, System.EventArgs e)
             {
                    Response.Redirect(PageBase.UrlBase + "/EnGruposF.aspx");

             }

             private void BtnRegistrar_Click(object sender, System.EventArgs e)
             {
             Response.Redirect(PageBase.UrlBase + "/EnGruposC.aspx");
             }

       }
}
```

Cuenta.cs - Cuenta.aspx

```
using System;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Data;

using PryEcommerce.CapaNegocio;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{
```

```
    /// <summary>
    ///     The code-behind base class used to edit a customer's
    ///     account information.
    ///     <remarks>This class is derived from PageBase.</remarks>
    /// </summary>
    public class Cuenta : PageBase
    {
        /// <value>The Button to save off customer information.</value>
        protected  PryEcommerce.Web.ModuloCuenta ModuloCuenta;
             protected System.Web.UI.WebControls.Image Image2;
             protected System.Web.UI.HtmlControls.HtmlTableCell Td1;
       //     protected PryEcommerce.Web.AccountModule ModuleAccount ;
        protected System.Web.UI.WebControls.Button SubmitButton;

        //------------------------------------------------------------------
        // This code was automatically generated
        //------------------------------------------------------------------
        /// <summary>
        ///     Constructor for Account.
        /// </summary>
        public Cuenta()
       {
               Page.Init += new System.EventHandler(Page_Init);
        }

        /// <summary>
        ///     Prepopulates the fields if editing.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            //
            // Ensure that the customer data is still in the session.
            //
            ApplicationAssert.Check(Customer != null, "No Customer at Account
Edit", ApplicationAssert.LineNumber);

            if (null == Customer)
            {
                //
                // Session has probably timed out
                //
                CarritoCompra().Customer = null;
                //
                // Clear the logon cookie
                //
                FormsAuthentication.SignOut();
                //
                // Try to get back here and force reauthentication
                //
                Response.Redirect("Cuenta.aspx", false);
            }
        }

        //------------------------------------------------------------------
        // This code was automatically generated
        //------------------------------------------------------------------
```

```
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event 
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form 
Designer.
            //
            InitializeComponent();
        }

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
              {
                      this.SubmitButton.Click += new 
System.EventHandler(this.SubmitButton_Click);
                      this.Load += new System.EventHandler(this.Page_Load);

              }

        /// <summary>
        ///     Validates then saves off the customer account information.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event 
data.</param>
        /// </summary>
        public void SubmitButton_Click(Object sender, EventArgs e)
        {
            //
            // Process the changes to the form
            //
            ModuloCuenta.ProcessChanges();
        }

    } // class Account
} // namespace PryEcommerce.Web
```

logon.cs

```
using System;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using System.ComponentModel;
using System.Data;

using PryEcommerce.CapaNegocio;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{

    /// <summary>
    ///     The code-behind base class for the Checkout.aspx page. This is
    ///     used to logon, or create a customer's account information.
    ///     <remarks>This class is derived from PageBase.</remarks>
    ///     <remarks>
    ///         Special considerations:
    ///         This class used for both logon to or create a customer's
account.
    ///     </remarks>
    /// </summary>
    public class Logon : PageBase
    {

      //     protected  PryEcommerce.Web.ModuloCuenta ModuloCuenta;
        /// <value>The HtmlGenericControl used to dispaly the title for the
page.</value>
        protected System.Web.UI.HtmlControls.HtmlGenericControl
TitleHtmlGenericControl;

        //
        // Controls used on the Logon panel
        //
        /// <value>The Panel that contains all the controls for user
logon.</value>
        protected System.Web.UI.WebControls.Panel LogonPanel;
        /// <value>The TextBox used to collect the logon email address.</value>
        protected System.Web.UI.WebControls.TextBox LogonEmailTextBox;
        /// <value>The RequiredFieldValidator used ensure the logon email
address contains a value.</value>
        protected System.Web.UI.WebControls.RequiredFieldValidator
LogonEmailRequiredFieldValidator;
        /// <value>The TextBox used to collect the logon password.</value>
        protected System.Web.UI.WebControls.TextBox LogonPasswordTextBox;
        /// <value>The RequiredFieldValidator used ensure the logon password
contains a value.</value>
        protected System.Web.UI.WebControls.RequiredFieldValidator
LogonPasswordRequiredFieldValidator;
        /// <value>The ValidationSummary used display logon input
errors.</value>
        protected System.Web.UI.WebControls.ValidationSummary
LogonValidationSummary;
        /// <value>The Label used display an email address and password
mismatch.</value>
        protected System.Web.UI.WebControls.Label MismatchLabel;
        /// <value>The Button used to attempt logon.</value>
        protected System.Web.UI.WebControls.Button LogonButton;
        /// <value>The Button used to attempt to create a new user.</value>
        protected System.Web.UI.WebControls.Button CreateButton;
```

```
        //
        // Controls used on the LoggedOn panel
        //
        /// <value>The Panel that contains all the controls to display that the
user is already loged on.</value>
        protected System.Web.UI.WebControls.Panel LoggedOnPanel;
        /// <value>The Button used to attempt to logon to different
account.</value>
        protected System.Web.UI.WebControls.Button ContinueButton;
        /// <value>The Button used to remain logged on as the currently logged
on user.</value>
        protected System.Web.UI.WebControls.Button CancelButton;

        //
        // Controls used on the Details panel
        //
        /// <value>The Panel that contains all the controls to create a new
user.</value>
        protected System.Web.UI.WebControls.Panel DetailsPanel;
        /// <value>The Label used to display to the user that they are creating
a new account.</value>
        protected System.Web.UI.WebControls.Label CreateTitleLabel;
        /// <value>The Button used to create a new account.</value>
        ///

              protected  ModuloCuenta ModuloCuenta  ;// AccountModule
ModuleAccount;

        protected System.Web.UI.WebControls.Button SubmitButton;

        //
        // Controls used on the Create panel
        //
        /// <value>The Panel that contains all the controls to show the user
that their account has been created.</value>
        protected System.Web.UI.WebControls.Panel CreatedPanel;
        /// <value>The Button used to allow the user to continue in the
application after account has been created.</value>
        protected System.Web.UI.WebControls.Button CreateContinueButton;


        //
        // private member variables
        //
        private CustomerData logonCustomerData;  //used for the current
Customer

        //-----------------------------------------------------------------
        // This code was automatically generated
        //-----------------------------------------------------------------
        /// <summary>
        ///     Constructor for Logon.
        /// </summary>
        public Logon()
       {
               Page.Init += new System.EventHandler(Page_Init);
        }


        /// <summary>
```

```
        ///     Display logon with the correct information.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            //
            // Get the current customer from Session (if any)
            //
            logonCustomerData = (CustomerData) Customer;

            //
            // if we are here; a result of a postback (form submit) we know
            // what action was in progress. Proceed on that basis
            //
            if (IsPostBack)
            {
                //
                // Intentionally empty
                //
            }
            else
            {
                //
                // We're here to do a logon
                //
                // First, show the logon panel
                //
                ShowPanel(LogonPanel, true);
                MismatchLabel.Visible = false;
                //
                // And hide the logged on panel
                //
                ShowPanel(LoggedOnPanel, false);
                //
                // And hide the account details module
                //
                ShowPanel(DetailsPanel, false);
                //
                // And hide the created panel
                //
                ShowPanel(CreatedPanel, false);
            }

        }

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
```

```
            // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
            //
            InitializeComponent();
        }

        //---------------------------------------------------------------
        // This code was automatically generated
        //---------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
              {
                    this.LogonButton.Click += new
System.EventHandler(this.LogonButton_Click);
                    this.CreateButton.Click += new
System.EventHandler(this.CreateButton_Click);
                    this.ContinueButton.Click += new
System.EventHandler(this.ContinueButton_Click);
                    this.CancelButton.Click += new
System.EventHandler(this.CancelButton_Click);
                    this.SubmitButton.Click += new
System.EventHandler(this.SubmitButton_Click);
                    this.CreateContinueButton.Click += new
System.EventHandler(this.CreateContinueButton_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

              }

        /// <summary>
        ///     Validates a logon attempt  saves off the customer account
information.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void LogonButton_Click(Object sender, EventArgs e)
        {
            CustomerData custData;

            //
            // Check the Email and Password
            //
            MismatchLabel.Visible = false;

            //Validator controls make sure Email and Password exist
            if (!Page.IsValid)
            {
                return;
            }

            //
            // Check for already logged on
            //
            if (logonCustomerData != null)
            {
                ShowPanel(LogonPanel, false);
                ShowPanel(DetailsPanel, false);
```

```
                ShowPanel(CreatedPanel, false);
                ShowPanel(LoggedOnPanel, true);
                ViewState["target"] = "logon";
                return;
            }

            //
            // Ensure the right panel == visible
            //
            ShowPanel(LogonPanel, true);
            ShowPanel(LoggedOnPanel, false);
            ShowPanel(DetailsPanel, false);
            ShowPanel(CreatedPanel, false);

            //
            // Check the Email and Password combination
            //
            custData = (new
CustomerSystem()).GetCustomerByEmail(LogonEmailTextBox.Text,
LogonPasswordTextBox.Text);

            if (custData != null)   //were they valid?
            {
                //
                // 1. Update customer in session.
                // 2. Update customer in cart.
                //
                base.Customer = custData;
                base.CarritoCompra().Customer = custData;
                FormsAuthentication.RedirectFromLoginPage("*", false);

            }
            else
            {
                MismatchLabel.Visible = true;
            }
        }

        /// <summary>
        ///     Validates  saves off the customer account information.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void SubmitButton_Click(Object sender, EventArgs e)
        {
            //
            // Create the account and trigger logon if it's good
            //
           if (ModuloCuenta.ProcessChanges())
                 {

                        // Did we create on a redirect from account.
                //
                if
(FormsAuthentication.GetRedirectUrl("*",false).EndsWith("Cuenta.aspx") )
                {
                    //
                    // Tell authentication that we're good.
                    //
```

```
                    FormsAuthentication.SetAuthCookie("*",false);
                    //
                    // Display proper panels
                    //
                    ShowPanel(CreatedPanel, true);
                    ShowPanel(LogonPanel, false);
                    ShowPanel(DetailsPanel, false);
                    ShowPanel(LoggedOnPanel, false);
                }
                else
                {
                    //
                    // Tell authentication that we're good to go and redirect.
                    //
                    FormsAuthentication.RedirectFromLoginPage("*",false);
                }
            }
        }

        /// <summary>
        ///     Redirect the newly created user to the shopping cart.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        ///
        public void CreateContinueButton_Click(Object sender, EventArgs e)
        {
            // Redirect to shopping cart
            Response.Redirect(PageBase.UrlBase + @"/CarritoCompra.aspx",
false);
        }

        /// <summary>
        ///     Redisplays the page so that the user can create a new account.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void CreateButton_Click(Object sender, EventArgs e)
        {
            //
            // Check for already logged on
            //
            if (logonCustomerData != null)
            {
                ShowPanel(LogonPanel, false);
                ShowPanel(DetailsPanel, false);
                ShowPanel(CreatedPanel, false);
                ShowPanel(LoggedOnPanel, true);
                ViewState["target"] = "create";
            }
            else
            {
                ShowPanel(LogonPanel, false);
                ShowPanel(LoggedOnPanel, false);
                ShowPanel(CreatedPanel, false);
                ShowPanel(DetailsPanel, true);
                //
                // Change the page for create account
```

```
                //
                //TitleHtmlGenericControl.InnerText = CreateTitleLabel.Text;
            }
        }

        /// <summary>
        ///     Continue with logoff/logon or logoff/create.
        ///     <remarks>
        ///         The continue action clears the customer from the session
and cart
        ///         and  calls the appropriate button handler; if the user
        ///         had just clicked from the main logon panel.
        ///     </remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void ContinueButton_Click(Object sender, EventArgs e)
        {
            //
            // Clear the customer data from the session and the cart
            //
            base.Customer = null;
            base.CarritoCompra().Customer = null;

            //
            // Clear the property
            //
            logonCustomerData = null;

            //
            // Clear the authentication token
            //
            FormsAuthentication.SignOut();

            //
            // Do the work
            //
            if (0 == String.Compare(ViewState["target"].ToString(), "create"))
            {
                CreateButton_Click(sender, e);
            }
            else
            {
                LogonButton_Click(sender, e);
            }

            //
            // Clear the view state
            //
            ViewState["target"] = null;
        }

        /// <summary>
        ///     Cancel the logoff/logon or logoff/create action by redirecting
        ///     to the checkout page.
        ///     <remarks>
        ///         The continue action clears the customer from the session
and cart
        ///         and  calls the appropriate button handler; if the user
```

```
        ///         had just clicked from the main logon panel.
        ///     </remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void CancelButton_Click(Object sender, EventArgs e)
        {
            //Tell authentication that we're not good to go.
            FormsAuthentication.RedirectFromLoginPage("*",false);
        }

        //----------------------------------------------------------------
        // Sub ShowPanel:
        //   Helper sub used to make certain that the validators do not
        //   fire when their parent panel is not visible.
        //----------------------------------------------------------------

        private void ShowPanel(Panel panel, bool visible)
        {
            IValidator validator;

            foreach (Control ctrl in panel.Controls)
            {
                //check to see if its a validator
                if ( ctrl is IValidator)
                {
                    validator = (IValidator)ctrl;
                    ctrl.Visible = visible;
                    if (!visible)
                    {
                        validator.Validate();
                    }
                }
            }
            panel.Visible = visible;
        }

    } // class Logon
} // namespace PryEcommerce.Web
```

Orden.cs - Orden.aspx

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Data;

using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

namespace PryEcommerce.Web
{
```

```
    /// <summary>
    ///     The code-behind base class for the Order.aspx page.
    ///     <remarks>
    ///         Displays order information such as tracking number
    ///         and empties the shopping cart.
    ///     </remarks>
    ///     <remarks>This class is derived from PageBase.</remarks>
    /// </summary>
    public class Orden : PageBase
    {

        //QueryString identifier
        private const String KEY_ID = "id";

        //
        // Declarations for controls on the WebForm
        //
        /// <value>The Panel that contains all the controls for displaying an
order's information.</value>
        protected System.Web.UI.WebControls.Panel OrderDataPanel;
        /// <value>The DataGrid used to dispaly the items for the current
order.</value>
        protected System.Web.UI.WebControls.DataGrid ShoppingCartDataGrid;
        /// <value>The Label for the displayed order's time and date.</value>
        protected System.Web.UI.WebControls.Label TimeDateLabel;
        /// <value>The Label for the displayed order's tracking number.</value>
        protected System.Web.UI.WebControls.Label TrackingNumberLabel;
        /// <value>The Label for the displayed order's shipping
address.</value>
        protected System.Web.UI.WebControls.Label ShippingAddressLabel;
        /// <value>The Label for the displayed order's sub total.</value>
        protected System.Web.UI.WebControls.Label SubTotalLabel;
        /// <value>The Label for the displayed order's tax.</value>
        protected System.Web.UI.WebControls.Label TaxLabel;
        /// <value>The Label for the displayed order's shipping and handling
charges.</value>
        protected System.Web.UI.WebControls.Label ShippingHandlingLabel;
        /// <value>The Label for the displayed order's total.</value>
        protected System.Web.UI.WebControls.Label TotalLabel;
        /// <value>The Label used to display an error with an attempt to
display this page.</value>
        protected System.Web.UI.WebControls.Label OrderErrorLabel;

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Constructor for Order.
        /// </summary>
        public Orden()
       {
               Page.Init += new System.EventHandler(Page_Init);
        }

        /// <summary>
        ///     Display the current orders information.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
```

```
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            int transactionId;

            //
            // Get the cart, but don't populate if it isn't already filled
            //
            Cart orderShoppingCart = base.CarritoCompra(false);

            if (orderShoppingCart.IsEmpty)   //make sure we have items to
display
            {
                //
                // if we are a postback, the user has probably clicked "GO" on
                // the search panel. We shouldn't override this and force them
to
                // the shopping cart !!
                //
                if (!IsPostBack)
                {
                    //
                    // Check to make sure the correct state exists to display
this page
                    //
                    ApplicationAssert.Check(false, "Unexpected empty Shopping
Cart at Order Display", ApplicationAssert.LineNumber);
                    Response.Redirect(PageBase.UrlBase +
@"/CarritoCompra.aspx", false);
                }
                return;
            }

            DataRow row = orderShoppingCart.Order.Rows[0];
            transactionId = (int)row[OrderData.SALE_ID_FIELD];
            TimeDateLabel.Text =
Server.HtmlEncode((String)row[OrderData.DATE_FIELD]);

            //Check for an error with the order
            if (transactionId == 0)
            {
                OrderDataPanel.Visible = false;
                OrderErrorLabel.Visible = true;
                return;
            }

            //
            // Open a DataSource against the cart items and attach it to the
grid
            //
            ShoppingCartDataGrid.DataSource =
(ICollection)orderShoppingCart.OrderItems.DefaultView;

            //
            // Fill in the tracking number and shipping address fields
            //
            TrackingNumberLabel.Text =
Server.HtmlEncode(transactionId.ToString());
```

```
            ShippingAddressLabel.Text =
Server.HtmlEncode(orderShoppingCart.ShippingAddress.Rows[0][OrderData.ADDRESS_F
IELD].ToString());

            //
            // Fill in the order summary fields.
            //
            row = orderShoppingCart.OrderSummary.Rows[0];
            SubTotalLabel.Text =
Server.HtmlEncode(System.String.Format("{0:C}",row[OrderData.SUB_TOTAL_FIELD]))
;
            TaxLabel.Text = Server.HtmlEncode(System.String.Format("{0:C}",
row[OrderData.TAX_FIELD]));
            ShippingHandlingLabel.Text =
Server.HtmlEncode(System.String.Format("{0:C}",
row[OrderData.SHIPPING_HANDLING_FIELD]));
            TotalLabel.Text = Server.HtmlEncode(System.String.Format("{0:C}",
row[OrderData.TOTAL_FIELD]));


            ShoppingCartDataGrid.DataBind();
            //
            // Clean up the cart
            //
            orderShoppingCart.RemoveAllItems();
        }


        //--------------------------------------------------------------------
        // This code was automatically generated
        //--------------------------------------------------------------------
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
            //
            InitializeComponent();
        }

        //--------------------------------------------------------------------
        // This code was automatically generated
        //--------------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
             {
                    this.Load += new System.EventHandler(this.Page_Load);

             }
```

```
    } // class Order
} // namespace PryEcommerce.Web
```

Pago.cs - Pago.aspx

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.Security;
using System.ComponentModel;
using System.Data;

using PryEcommerce.SystemFramework;
using PryEcommerce.Comun;
using PryEcommerce.Comun.Data;

namespace PryEcommerce.Web
{

    /// <summary>
    ///     The code-behind base class for the Checkout.aspx page. This is
    ///     used to step the user through the checkout process.
    ///     <remarks>This class is derived from PageBase.</remarks>
    /// </summary>
    public class Pago : PageBase
    {

        private const String KEY_STAGE = "stage";

        //
        //Control definitions corresponding to items on the page
        //


        //
        // START SHIPPING CONTROLS
        //
        /// <value>The Panel that contains the controls for the shipping step
of checkout.</value>
        protected System.Web.UI.WebControls.Panel ShippingPanel;
        /// <value>The TextBox used to collect the recipients name.</value>
        protected System.Web.UI.WebControls.TextBox ShipToNameTextBox;
        /// <value>The TextBox used to collect the recipients address.</value>
        protected System.Web.UI.WebControls.TextBox AddressTextBox;
        /// <value>The TextBox used to collect the recipients country.</value>
        protected System.Web.UI.WebControls.TextBox CountryTextBox;
        /// <value>The TextBox used to collect the recipients phone
number.</value>
        protected System.Web.UI.WebControls.TextBox PhoneNumberTextBox;
        /// <value>The TextBox used to collect the recipients fax
number.</value>
        protected System.Web.UI.WebControls.TextBox FaxTextBox;
        //
        // END SHIPPING CONTROLS
```

```
        //

        //
        // START PAYMENT CONTROLS
        //
        /// <value>The Panel that contains the controls for the payment step of
checkout.</value>
        protected System.Web.UI.WebControls.Panel PaymentPanel;
        /// <value>The Label used to display the amount of the
purchase.</value>
        protected System.Web.UI.WebControls.Label AmountLabel;
        /// <value>The DropDownList used to collect the type credit
card.</value>
        protected System.Web.UI.WebControls.DropDownList CartTypeDropDownList;
        /// <value>The TextBox used to collect the name on the credit
card.</value>
        protected System.Web.UI.WebControls.TextBox NameOnCardTextBox;
        /// <value>The TextBox used to collect the credit card number.</value>
        protected System.Web.UI.WebControls.TextBox CardNumberTextBox;
        /// <value>The TextBox used to collect the credit card's billing
information.</value>
        protected System.Web.UI.WebControls.TextBox BillingInfoTextBox;
        /// <value>The CustomValidator used to display errors with the billing
information.</value>
        protected System.Web.UI.WebControls.CustomValidator
BillingInfoCustomValidator;
        /// <value>The DropDownList used to collect the credit card's
expiration month.</value>
        protected System.Web.UI.WebControls.DropDownList ExpMonthDropDownList;
        /// <value>The DropDownList used to collect the credit card's
expiration year.</value>
        protected System.Web.UI.WebControls.DropDownList
ExpYearDropDownListBox;
        //
        // END PAYMENT CONTROLS
        //

        //
        // START SUMMARY CONTROLS
        //
        /// <value>The Panel that contains the controls for the confirmation
step of checkout.</value>
        protected System.Web.UI.WebControls.Panel SummaryPanel;
        /// <value>The DataGrid used to dispaly the items for the current
order.</value>
        protected System.Web.UI.WebControls.DataGrid ShoppingCartDataGrid;
        /// <value>The Label for the displayed order's sub total.</value>
        protected System.Web.UI.WebControls.Label SubTotalLabel;
        /// <value>The Label for the displayed order's tax.</value>
        protected System.Web.UI.WebControls.Label TaxLabel;
        /// <value>The Label for the displayed order's shipping and handling
charges.</value>
        protected System.Web.UI.WebControls.Label ShippingHandlingLabel;
        /// <value>The Label for the displayed order's total.</value>
        protected System.Web.UI.WebControls.Label TotalLabel;
        //
        // END SUMMARY CONTROLS
        //

        //
```

```
        // Controls present for all panels.
        //
        /// <value>The ImageButton used to navigate to the next step of
checkout.</value>
        protected System.Web.UI.WebControls.ImageButton NextImageButton;
        /// <value>The ImageButton used to navigate to the previous step of
checkout.</value>
        protected System.Web.UI.WebControls.ImageButton PreviousImageButton;
        /// <value>The CheckoutModule user control used to display the current
step of the checkout process.</value>
       //protected PryEcommerce.Web.ModuloPago  ModuleCheckout = new
PryEcommerce.Web.ModuloPago();
                protected PryEcommerce.Web.ModuloPago ModuloPago;

        //
        // Validator Controls.
        //
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator1;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator2;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator3;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator4;
        protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator1;
        protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator2;
        protected System.Web.UI.WebControls.ValidationSummary
ValidationSummary1;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator5;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator6;
        protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator3;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator7;
        protected System.Web.UI.WebControls.ValidationSummary
ValidationSummary2;

        // transaction amount key value
        private const String KEY_TRANSAMOUNT = "transamount";
        // the current stage of the checkout process
        private int stage;

        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <summary>
        ///     Constructor for Checkout.
        /// </summary>
        public Pago()
       {
                Page.Init += new System.EventHandler(Page_Init);
        }

        /// <summary>
```

```
        ///     Display the checkout information to step through the checkout
process.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            stage = IsPostBack ? Int32.Parse(ViewState[KEY_STAGE].ToString()) :
0;

            //
            // Get the cart, but don't populate if it isn't already filled
            //
            Cart checkoutShoppingCart = base.CarritoCompra(false);

            //
            // Check to make sure the correct state exists to display this page
            //
            //
            // Have we got a valid shoppingcart?
            //
            ApplicationAssert.Check(!checkoutShoppingCart.IsEmpty, "Empty
Shopping Cart at Checkout", ApplicationAssert.LineNumber - 4);
            if (checkoutShoppingCart == null || checkoutShoppingCart.IsEmpty)
            {
                //Go to a page that can add items if we don't have any
                Response.Redirect(PageBase.UrlBase + @"/CarritoCompra.aspx",
false);
                return;
            }

            //
            // Have we got a customer ?
            //
            if (Customer == null)
            {
                //session has probably timed out force a new logon
                checkoutShoppingCart.Customer = null;
                FormsAuthentication.SignOut();
                Response.Redirect(@"Pago.aspx", false);
                return;
            }


            //
            // Open a DataSource against the cart items and attach it to the
grid
            //
            ShoppingCartDataGrid.DataSource =
(ICollection)checkoutShoppingCart.OrderItems.DefaultView;
            //
            // Fill in the order summary fields. OrderSummary needs to be
recalculated
            //    whenever the items in the cart of the shipping address
changes.
            //
            checkoutShoppingCart.CalculateOrderSummary();
```

```
            DataRow row = checkoutShoppingCart.OrderSummary.Rows[0];
            SubTotalLabel.Text = System.String.Format("{0:C}",
row[OrderData.SUB_TOTAL_FIELD]);
            TaxLabel.Text = System.String.Format("{0:C}",
row[OrderData.TAX_FIELD]);
            ShippingHandlingLabel.Text = System.String.Format("{0:C}",
row[OrderData.SHIPPING_HANDLING_FIELD]);
            TotalLabel.Text = System.String.Format("{0:C}",
row[OrderData.TOTAL_FIELD]);

            ShoppingCartDataGrid.DataBind();

            //
            // START PAYMENT CONTROLS
            //
            //allow multiline editing for billing Info
            BillingInfoTextBox.TextMode = TextBoxMode.MultiLine;
            AmountLabel.Text =
System.String.Format("{0:C}",row[OrderData.TOTAL_FIELD]);
            //
            // END PAYMENT CONTROLS
            //

            if (!IsPostBack)
            {
                row = checkoutShoppingCart.ShippingAddress.Rows[0];

                ShipToNameTextBox.Text =
row[OrderData.SHIP_TO_NAME_FIELD].ToString();
                AddressTextBox.Text = row[OrderData.ADDRESS_FIELD].ToString();
                CountryTextBox.Text = row[OrderData.COUNTRY_FIELD].ToString();
                PhoneNumberTextBox.Text =
row[OrderData.PHONE_NUMBER_FIELD].ToString();
                FaxTextBox.Text = row[OrderData.FAX_FIELD].ToString();

                SetPanelDisplay();

                //
                // Set the expiration year based off the current year
                //
                int startYear = DateTime.Now.Year;
                    for (int i=0; i < 10; i++)
                    {

       ExpYearDropDownListBox.Items.Add(startYear++.ToString());
                    }

            }
        }

        //------------------------------------------------------------------
        // This code was automatically generated
        //------------------------------------------------------------------
        /// <summary>
        ///     Used to wireup this page's event handlers.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Init(object sender, EventArgs e)
```

```
        {
            //
            // CODEGEN: This call is required by the ASP+ Windows Form
Designer.
            //
            InitializeComponent();
        }

        //--------------------------------------------------------------
        // This code was automatically generated
        //--------------------------------------------------------------
        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
             {
                    this.PreviousImageButton.Click += new
System.Web.UI.ImageClickEventHandler(this.PreviousImageButton_Click);
                    this.NextImageButton.Click += new
System.Web.UI.ImageClickEventHandler(this.NextImageButton_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

             }


        /// <summary>
        ///     Move to the next step in the checkout process.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
             public void NextImageButton_Click (object sender,
System.Web.UI.ImageClickEventArgs e)
        {
            bool stepSuccess = false;

            //
            // Trim the phone and fax numbers
            //
            ShipToNameTextBox.Text = ShipToNameTextBox.Text.Trim();
            AddressTextBox.Text = AddressTextBox.Text.Trim();
            CountryTextBox.Text = CountryTextBox.Text.Trim();
            PhoneNumberTextBox.Text = PhoneNumberTextBox.Text.Trim();
            FaxTextBox.Text = FaxTextBox.Text.Trim();
            NameOnCardTextBox.Text = NameOnCardTextBox.Text.Trim();
            CardNumberTextBox.Text = CardNumberTextBox.Text.Trim();
            BillingInfoTextBox.Text = BillingInfoTextBox.Text.Trim();
            foreach (IValidator val in Page.Validators)
            {
                val.Validate();
            }

            switch (stage)
            {
                case 0:
                    stepSuccess = ValidateShipping();
                    break;

                case 1:
```

```
                    stepSuccess = ValidatePayment();
                    break;

                case 2:
                    stepSuccess = SubmitOrder();
                    break;
            }

            if (stepSuccess)
            {
                ++stage;
            }

            SetPanelDisplay();
        }

        /// <summary>
        ///     Move to the previous step in the checkout process.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
             public void PreviousImageButton_Click (object sender,
System.Web.UI.ImageClickEventArgs e)
        {
            --stage;

            SetPanelDisplay();
        }

        //----------------------------------------------------------------
        // Sub SetPanelDisplay:
        //    Determine which panels to display and control the button states.
        //----------------------------------------------------------------
        private void SetPanelDisplay()
        {
            switch (stage)
            {
                case 0:
                    ShowPanel(ShippingPanel, true);
                    ShowPanel(PaymentPanel, false);
                    ShowPanel(SummaryPanel, false);

                    NextImageButton.Enabled = true;
                    NextImageButton.ImageUrl = "../images/next.gif";
                    PreviousImageButton.Enabled = false;
                    PreviousImageButton.ImageUrl =
"../images/previousdisabled.gif";
                    break;

                case 1:
                    ShowPanel(ShippingPanel, false);
                    ShowPanel(PaymentPanel, true);
                    ShowPanel(SummaryPanel, false);

                    NextImageButton.Enabled = true;
                    NextImageButton.ImageUrl = "../images/next.gif";
                    PreviousImageButton.Enabled = true;
                    PreviousImageButton.ImageUrl = "../images/previous.gif";
                    break;
```

```
                case 2:
                    ShowPanel(ShippingPanel, false);
                    ShowPanel(PaymentPanel, false);
                    ShowPanel(SummaryPanel, true);

                    NextImageButton.Enabled = true;
                    NextImageButton.ImageUrl = "../images/confirm.gif";
                    PreviousImageButton.Enabled = true;
                    PreviousImageButton.ImageUrl = "../images/previous.gif";
                    break;
            }

            ViewState[KEY_STAGE] = stage.ToString();
            ModuloPago.Stage  = stage;

        }

        //-----------------------------------------------------------------
        // Function ValidateShipping:
        //   Helper function used to validate that all required web form
        //   ui fields have not been left empty. In this case, validates
        //   shipping information.
        // Returns:
        //   Boolean, true if input is valid, false if input is not valid.
        //-----------------------------------------------------------------
        private bool ValidateShipping()
        {
            //
            // Validation is done via validator controls.
            //
            bool errorsFound = !Page.IsValid;

            //
            //  FUTURE: Call to external web services for address verification
goes here.
            //


            if (errorsFound)
            {
                //
                // Show that validation failed
                //
                return false;
            }
            else
            {
                //
                // Update the shopping cart
                //
                CarritoCompra(false).SetShippingAddress(ShipToNameTextBox.Text,
                                                        AddressTextBox.Text,
                                                        CountryTextBox.Text,
                                                        PhoneNumberTextBox.Text,
                                                        FaxTextBox.Text);
                //
                // Show that validation succeeded
                //
                return true;
```

```
            }
        }

        //-------------------------------------------------------------
        // Function ValidatePayment:
        //   Helper function used to validate that all required web form
        //   ui fields have not been left empty. In this case, validates
        //   payment information.
        // Returns:
        //   Boolean, true if input is valid, false if input is not valid.
        //-------------------------------------------------------------
        private bool ValidatePayment()
        {
            bool errorsFound = false;


            //
            // Most validation is done via validator controls.
            // MultiLine TextBoxes do not use MaxLength so use
            // a custom validator to check for length.
            //
            if (BillingInfoTextBox.Text.Length >= 255)
            {
                BillingInfoCustomValidator.IsValid = false;
                errorsFound = true;
            }
            else
            {
                BillingInfoCustomValidator.IsValid = true;
            }


            errorsFound = !Page.IsValid;

            //
            //  FUTURE: Call to external web services for payment verification
goes here.
            //
            if (errorsFound)
            {
                //
                // Show that validation failed
                //
                return false;
            }
            else
            {
                String expDate = ExpMonthDropDownList.SelectedItem.Value + "/"
+ ExpYearDropDownListBox.SelectedItem.Value;
                //
                // Update the shopping cart
                //

CarritoCompra(false).SetPayment(CartTypeDropDownList.Items[CartTypeDropDownList
.SelectedIndex].Text,
                                                NameOnCardTextBox.Text,
                                                CardNumberTextBox.Text,
                                                expDate,
                                                BillingInfoTextBox.Text,
                                                "AuthorizationCode");
```

```
                //
                // Show that validation succeeded
                //
                return true;
            }
        }

        //--------------------------------------------------------------------
        // Function SubmitOrder:
        //   Submits an order to the business facade layer.
        // Returns:
        //   Boolean, true if successful, false if not.
        //--------------------------------------------------------------------
        private bool SubmitOrder()
        {
            //Write trace log.
            ApplicationLog.WriteTrace("PryEcommerce.Web.Pago.SubmitOrder:");

            CarritoCompra(false).AddOrder();

            //
            // Show the order summary page
            //
            Response.Redirect("Orden.aspx", false);

            return false;
        }


        //--------------------------------------------------------------------
        // Sub ShowPanel:
        //   Helper sub used to make certain that the validators do not
        //   fire when their parent panel is not visible.
        //--------------------------------------------------------------------

        private void ShowPanel(Panel panel, bool visible)
        {
            IValidator validator;
            foreach (Control ctrl in panel.Controls)
            {
                //check to see if its a validator
                if ( ctrl is IValidator)
                {
                    validator = (IValidator)ctrl;
                    ctrl.Visible = visible;
                    if (!visible)
                    {
                        validator.Validate();
                    }
                }
            }
            panel.Visible = visible;
        }

    } // class Pago
} // namespace PryEcommerce.Web
```

ModuloBanner.cs - ModuloBanner.ascx

```
namespace PryEcommerce.Web
{

	using System;
	using System.Web;
	using System.Web.UI;
	using System.Web.UI.WebControls;
	using System.Collections;
	using System.ComponentModel;
	using System.Data;


	using PryEcommerce.CapaNegocio;
	using PryEcommerce.Comun;
	using PryEcommerce.Comun.Data;
	using PryEcommerce.SystemFramework;



	/// <summary>
	///		Descripción breve de WebUserControl1.
	/// </summary>
	public class ModuloBanner : ModuleBase
	{
		protected System.Web.UI.WebControls.ImageButton ImageButton1;

		private void Page_Load(object sender, System.EventArgs e)
		{


//		HomeImage.ImageUrl = PathPrefix + "/images/banner/home.gif";
//		CartImage.ImageUrl = PathPrefix +
"/images/banner/bannercart.gif";
//		AccountImage.ImageUrl = PathPrefix + "/images/banner/key.gif";

//			Image20.ImageUrl = PathPrefix +
"/images/banner/reminder.gif";
//			Image21.ImageUrl = PathPrefix +
"/images/banner/asistente.gif";
//			Image22.ImageUrl = PathPrefix + "/images/banner/grupo.gif";

			//Image1.ImageUrl = PathPrefix +
"/images/banner/bannerregalos.gif";

			//Image4.ImageUrl = PathPrefix + "/images/b1.gif";
			//Image5.ImageUrl = PathPrefix + "/images/b2.gif";
		//	Image6.ImageUrl = PathPrefix + "/images/b3.gif";
			//Image7.ImageUrl = PathPrefix + "/images/b4.gif";
			//Image8.ImageUrl = PathPrefix + "/images/b5.gif";
		//	Image9.ImageUrl = PathPrefix + "/images/b6.gif";
		//	Image10.ImageUrl = PathPrefix + "/images/b7.gif";



			//banner1.ImageUrl = PathPrefix + "/images/banner/4.jpg";
```

```
                        //Image11.ImageUrl = PathPrefix + "/images/banner/4.jpg";


                }

                #region Web Form Designer generated code
                override protected void OnInit(EventArgs e)
                {
                        //
                        // CODEGEN: llamada requerida por el Diseñador de Web Forms
ASP.NET.
                        //
                        InitializeComponent();
                        base.OnInit(e);
                }

                ///             Método necesario para admitir el Diseñador, no se
puede modificar
                ///             el contenido del método con el editor de código.
                /// </summary>
                private void InitializeComponent()
                {
                        this.Load += new System.EventHandler(this.Page_Load);

                }
                #endregion

        }
}
```

ModuloBusqueda.cs - ModuloBusqueda.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Text;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Collections;
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.Comun;
    using PryEcommerce.SystemFramework;

    /// <summary>
    ///     This user control module collecta the  search criteria and
    ///     redirects to the SearchResults page to process the request.
    ///     <remarks>This class is derived from ModuleBase.</remarks>
    /// </summary>
    public class ModuloBusqueda : ModuleBase
    {
```

```
        /// <value>The Image for the separating line for the search module user
control.</value>
        protected System.Web.UI.WebControls.Image LineImage;
        /// <value>The Button used to start a search.</value>
        protected System.Web.UI.WebControls.Button SearchButton;
        /// <value>The TextBox used to specify the text to search for.</value>
        protected System.Web.UI.WebControls.TextBox SearchTextBox;
        /// <value>The DropDownList used to specify the types of searches
available.</value>
        protected System.Web.UI.WebControls.DropDownList SearchDropDownList;

        /// <summary>
        ///     Initialize the user control.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            LineImage.ImageUrl = PathPrefix + "/images/banner/line.gif";
                    //LineImage2.ImageUrl = PathPrefix +
"/images/banner/line.gif";
        }

             private void InitializeComponent()
             {
                    this.SearchTextBox.TextChanged += new
System.EventHandler(this.SearchTextBox_TextChanged);
                    this.SearchButton.Click += new
System.EventHandler(this.SearchButton_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

             }

        /// <summary>
        ///     Collects user input and redirects information to the
SearchResults page.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void SearchButton_Click(Object sender, EventArgs e)
        {
            //
            // Retrieve the search text
            //
                    //PageBase XYZ = new PageBase();
            String searchText = SearchTextBox.Text.Trim();
            //
            // Redirect to search results page...
            //
            int index = SearchDropDownList.SelectedIndex;
            Response.Redirect((new StringBuilder(PageBase.UrlBase))
                                .Append("/ResultadoBusqueda.aspx?type=")
                                .Append(index)
                                .Append("&fullType=")

.Append(Server.HtmlEncode(SearchDropDownList.Items[index].Text))
                                .Append("&text=")
```

```
                              .Append(Server.UrlEncode(searchText)).ToString(),
false);
        }

             private void SearchTextBox_TextChanged(object sender,
System.EventArgs e)
             {

             }

    } // class SearchModule

} // namespace PryEcommerce.Web
```

ModuloBusquedaAsistida.cs – ModuloBusquedaAsistida.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Text;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Collections;
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.Comun;
    using PryEcommerce.SystemFramework;

    /// <summary>
    ///     This user control module collecta the  search criteria and
    ///     redirects to the SearchResults page to process the request.
    ///     <remarks>This class is derived from ModuleBase.</remarks>
    /// </summary>
    public class ModuloBusquedaAsistida : ModuleBase
    {
             protected System.Web.UI.WebControls.Button SearchButton;
             protected System.Web.UI.WebControls.Label Label1;
             protected System.Web.UI.WebControls.Label Label3;
             protected System.Web.UI.WebControls.DropDownList DdlMotivo;
             protected System.Web.UI.WebControls.DropDownList DdlEdad;
             protected System.Web.UI.WebControls.DropDownList DdlSexo;
             protected System.Web.UI.WebControls.DropDownList DdlEstadoCivil;
             protected System.Web.UI.WebControls.DropDownList DdlMonto;
             protected System.Web.UI.WebControls.Label Label4;
             protected System.Web.UI.WebControls.Label Label5;
             protected System.Web.UI.WebControls.Label Label6;
             protected System.Web.UI.WebControls.Image ImageAs;
             protected System.Web.UI.WebControls.Image LineImage;
             protected System.Web.UI.WebControls.TextBox SearchTextBox;
             protected System.Web.UI.WebControls.Label Label2;
```

```
        /// <summary>
        ///     Initialize the user control.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            ImageAs.ImageUrl = PathPrefix + "/images/banner/asistente.gif";
                    LineImage.ImageUrl = PathPrefix +
"/images/banner/line.gif";

        }

             private void InitializeComponent()
             {
                    this.SearchButton.Click += new
System.EventHandler(this.SearchButton_Click);
                    this.Load += new System.EventHandler(this.Page_Load);

             }

        /// <summary>
        ///     Collects user input and redirects information to the
SearchResults page.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void SearchButton_Click(Object sender, EventArgs e)
        {
            //
            // Retrieve the search text
            //

            String searchText = SearchTextBox.Text.Trim();

            //
            // Redirect to search results page...
            //
            int index = DdlMotivo.SelectedIndex;


////                Response.Redirect((new StringBuilder(PageBase.UrlBase))
////                       .Append("/ResultadoBusqueda.aspx?type=")
////                       .Append(index)
////                       .Append("&fullType=")
////
       .Append(Server.HtmlEncode(SearchDropDownList.Items[index].Text))
////                       .Append("&text=")
////                       .Append(Server.UrlEncode(searchText)).ToString(),
false);


                    //
.Append(Server.HtmlEncode(DdlMotivo.SelectedItem.Text))

            Response.Redirect((new StringBuilder(PageBase.UrlBase))
```

```
.Append("/ResultadoBusquedaAsistida.aspx?Category=")

        .Append(Server.HtmlEncode(DdlMotivo.SelectedValue))
                                                .Append("&Edad=")
        .Append(Server.HtmlEncode(DdlEdad.SelectedValue))
                                                .Append("&Sexo=")
        .Append(Server.HtmlEncode(DdlSexo.SelectedValue ))
                                                .Append("&Estado=")
        .Append(Server.HtmlEncode(DdlEstadoCivil.SelectedValue ))
                                                .Append("&Monto=")
        .Append(Server.HtmlEncode(DdlMonto.SelectedItem.Text))
                                .Append("&Textobusqueda=")
                                .Append(Server.UrlEncode(searchText)).ToString(),
false);
         }

    } // class SearchModule

} // namespace PryEcommerce.Web
```

ModuloCategorias.cs – ModuloCategorias.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Text;
    using System.Collections;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.CapaNegocio;
    using PryEcommerce.Comun;
    using PryEcommerce.Comun.Data;
    using PryEcommerce.SystemFramework;

    /// <summary>
    ///     This user control module displays the list of categories on many
    ///     of the Duwamish web pages.
    ///     <remarks>This class is derived from ModuleBase.</remarks>
```

```
    /// </summary>
    public class ModuloCategorias : ModuleBase
    {

        /// <summary>
        ///     This inner class is used to access a single item for the
categories list.
        /// </summary>
        public class DataItem
        {
            private String moduleDescription; //Description for the category
            private String moduleUrl;         //URL for obtaining category
listing

            /// <summary>
            ///     Constructor for DataItem:  saves the target and
description.
            ///     <param name="description">The description for the category
item.</param>
            ///     <param name="target">The URL to link to for this category
item.</param>
            /// </summary>
            public DataItem(String description, String target)
            {
                moduleDescription = description;
                moduleUrl = target;
            }

            /// <value>
            ///     Property Description is used to get the Description string
for a category item.
            /// </value>
            public String Description
            {
                get
                {
                    return moduleDescription;
                }
            }

            /// <value>
            ///     Property URL is used to get the URL string for a category
item.
            /// </value>
            public String Url
            {
                get
                {
                    return moduleUrl;
                }
            }
        }

        private System.Collections.ArrayList moduleDataList;
             protected System.Web.UI.WebControls.Image LineImage;
             protected System.Web.UI.WebControls.DataList CategoriesList;

        private const String KEY_CATEGORYSET = "Cache:CategoryData";

        private void InitializeComponent()
```

```
        {
                    this.Load += new System.EventHandler(this.Page_Load);

             }

        /// <value>
        ///     Property SubCategoryList is used to get the ArrayList control
that contains the list of
        ///     the top level categories to display.
        /// </value>
        public System.Collections.ArrayList SubCategoryList
        {
            get
            {
                return moduleDataList;
            }
        }

        /// <summary>
        ///     Populate the pagelet with the list of categories and store
        ///     the data in the ASP+ Cache.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {

                    //PageBase XYZ = new PageBase();

                    LineImage.ImageUrl = PathPrefix +
"/images/banner/line.gif";


            //
            // Always do a root category query
            //
            int categoryId = 1;
            String categoryIdString = categoryId.ToString();

            //
            // Try to get the data from the cache first
            //
            CategoryData categorySet = (CategoryData)Cache[KEY_CATEGORYSET];

            if (null == categorySet)
            {
                //
                // There's no data in the cache, so load it and cache it
                //
                // Create the Facade Object and request the data
                //
                categorySet = (new ProductSystem()).GetCategories(categoryId);
                //
                // Now cache the data for an hour
                //
                Cache.Insert(KEY_CATEGORYSET, categorySet, null,
DateTime.Now.AddHours(1), System.Web.Caching.Cache.NoSlidingExpiration);
            }
```

```
            //
            // Ensure that we have the category data
            //
            ApplicationAssert.CheckCondition(null != categorySet, "No Category
data available", ApplicationAssert.LineNumber);
            //
            // get the default view for the data
            //
            DataView categoryView =
categorySet.Tables[CategoryData.CATEGORIES_TABLE].DefaultView;
            //
            // Create the array of items for the list display
            //
            moduleDataList = new System.Collections.ArrayList();
            categoryView.RowFilter = (new
StringBuilder(CategoryData.PARENT_ID_FIELD))
                                     .Append(" = ")
                                     .Append(categoryIdString)
                                     .Append(" AND ")
                                     .Append(CategoryData.PKID_FIELD)
                                     .Append(" <> ")
                                     .Append(categoryIdString).ToString();

            //
            // Store each of the top level categories
            //



            String catUrlBase =  PageBase.UrlBase + "/Categorias.aspx?id=";
            foreach (DataRowView categoryRow in categoryView)
            {
                moduleDataList.Add(new
DataItem(Server.HtmlEncode(categoryRow[CategoryData.DESCRIPTION_FIELD].ToString
()), catUrlBase + categoryRow[CategoryData.PKID_FIELD].ToString()));
            }
            //
            // Finally, do the data binding
            //
            DataBind();
        }

    } // class CategoriesModule
} // namespace PryEcommerce.Web
```

ModuloCuenta.cs – ModuloCuenta.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Collections;
```

```
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.Comun;
    using PryEcommerce.Comun.Data;
    using PryEcommerce.SystemFramework;
    using PryEcommerce.CapaNegocio;

    /// <summary>
    ///     This user control module supports creation and editing of account
information.
    ///     <remarks>This class is derived from ModuleBase.</remarks>
    /// </summary>
    ///

    public class ModuloCuenta : ModuleBase
    {
        //----------------------------------------------------------------
        // This code was automatically generated
        //----------------------------------------------------------------
        /// <value>The TextBox for the user's email address.</value>
        protected System.Web.UI.WebControls.TextBox EmailTextBox;
        /// <value>The TextBox for the user's password.</value>
        protected System.Web.UI.WebControls.TextBox PasswordTextBox;
        /// <value>The TextBox for the user's password confirmation.</value>
        protected System.Web.UI.WebControls.TextBox ConfirmPasswordTextBox;
        /// <value>The TextBox for the user's name.</value>
        protected System.Web.UI.WebControls.TextBox AcctNameTextBox;
        /// <value>The TextBox for the user's address.</value>
        protected System.Web.UI.WebControls.TextBox AddressTextBox;
        /// <value>The TextBox for the user's country.</value>
        protected System.Web.UI.WebControls.TextBox CountryTextBox;
        /// <value>The TextBox for the user's phone number.</value>
        protected System.Web.UI.WebControls.TextBox PhoneTextBox;
        /// <value>The TextBox for the user's fax number.</value>
        protected System.Web.UI.WebControls.TextBox FaxTextBox;
        /// <value>The label for the displayed when a user is being
created.</value>
        protected System.Web.UI.WebControls.Label CreateLabel;
        /// <value>The label for the displayed when a user is being
edited.</value>
        protected System.Web.UI.WebControls.Label EditLabel;
        /// <value>The label for the displayed when a user has been
updated.</value>
        protected System.Web.UI.WebControls.Label UpdatedLabel;

        //
        // Error handling
        //
        /// <value>The CustomValidator used when the email address given is
already in use.</value>
        protected System.Web.UI.WebControls.CustomValidator
EmailUniqueCustomValidator;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator1;
        protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator1;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator2;
        protected System.Web.UI.WebControls.CompareValidator CompareValidator1;
```

```
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator3;
        protected System.Web.UI.WebControls.CompareValidator CompareValidator2;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator4;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator5;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator6;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator7;
        protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator2;
        protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator3;
        protected System.Web.UI.WebControls.ValidationSummary
ValidationSummary1;

              //
        // private member variables
        //

        private CustomerData moduleCustomerInfo;
        private bool         moduleEditMode;         //edit or create mode -
default is create

        /// <value>
        ///     Property EditMode is used to get or set the edit mode for this
user control.
        ///     <remarks>
        ///         Sets the value EditMode.
        ///         Gets the value EditMode.
        ///     </remarks>
        /// </value>
        ///

              public bool ProcessChanges()
              {
                     //
                     // Update the Database
                     //
                     if ( SaveCustomer() )
                     {
                            EditLabel.Visible = false;
                            CreateLabel.Visible = false;
                            UpdatedLabel.Visible = true;

                            return true;
                     }
                     else
                     {
                            //
                            // Show any errors
                            //
                            DisplayErrors();

                            UpdatedLabel.Visible = false;

                            return false;
                     }
```

```
            }

        public bool EditMode
        {
            get
            {
                return moduleEditMode;
            }
            set
            {
                moduleEditMode = value;
            }
        }

        /// <summary>
        ///     Sets up the user control.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            if ( moduleEditMode )
            {
                EditLabel.Visible = true;
                CreateLabel.Visible = false;
            }
            else
            {
                CreateLabel.Visible = true;
                EditLabel.Visible = false;
            }
            //
            // If we are here as a result of a postback (form submit) then we
know
            // what action was in progress. Do not repopulate the form.
            //
            if ( IsPostBack )
            {
                //
                // Intentionally blank
                //
            }
            else
            {
                if ( moduleEditMode )
                {
                    //
                    // Ensure that the customer data is in the session for edit
mode
                    //
                    if ( Customer != null )
                    {
                        //
                        // Populate the form from the customer data
                        //
                        DataRow row =
Customer.Tables[CustomerData.CUSTOMERS_TABLE].Rows[0];
```

```
                        EmailTextBox.Text           =
(string)row[CustomerData.EMAIL_FIELD];
                        AcctNameTextBox.Text        =
(string)row[CustomerData.NAME_FIELD];
                        AddressTextBox.Text         =
(string)row[CustomerData.ADDRESS_FIELD];
                        CountryTextBox.Text         =
(string)row[CustomerData.COUNTRY_FIELD];
                        PasswordTextBox.Text        =
(string)row[CustomerData.PASSWORD_FIELD];
                        ConfirmPasswordTextBox.Text =
(string)row[CustomerData.PASSWORD_FIELD];
                        PhoneTextBox.Text           =
(string)row[CustomerData.PHONE_FIELD];
                        FaxTextBox.Text             =
(string)row[CustomerData.FAX_FIELD];
                    }
                    else
                    {
                        //
                        // Missing customer data is handled by the account and
logon pages
                        //
                    }
                }
                else
                {
                    //
                    // Leave the form blank
                    //
                }
            }
        }

        /// <summary>
        ///     Save the current state to the session and database.
        ///     <remarks>If this is a new customer, then create the item, or
edit for an existing customer.</remarks>
        ///     <remarks>This is called from the page where this user control
exists.</remarks>
        ///     <retvalue>Returns true on success, false otherwise.</retvalue>
        /// </summary>
        private bool SaveCustomer()
        {
            DataRow      customerRow;
            bool         retVal        = false;


            AcctNameTextBox.Text = AcctNameTextBox.Text.Trim();
            PhoneTextBox.Text = PhoneTextBox.Text.Trim();
            FaxTextBox.Text = FaxTextBox.Text.Trim();
            AddressTextBox.Text = AddressTextBox.Text.Trim();
            CountryTextBox.Text = CountryTextBox.Text.Trim();

            String tmpPassword = PasswordTextBox.Text;

            if (tmpPassword != ConfirmPasswordTextBox.Text)
            {
                tmpPassword = "";  // Force an error on password
```

```
            }
            foreach(IValidator val in Page.Validators)
            {
                val.Validate();
            }

            if ( Page.IsValid )
            {
                if ( moduleEditMode )
                {
                    //
                    // WORKAROUND: Create and Update the dataset. Ideally we
would do Customer.Clone()
                    //              here and update the row, however
DataSet.Clone() is broken. Instead,
                    //              we create a new dataset, add a row, accept
the changes then update
                    //              the row before passing it to the facade
layer. This ensures that
                    //              the row is marked as dirty when the data
access layer does the
                    //              update call.
                    //
                    moduleCustomerInfo = new CustomerData();
                    customerRow    =
moduleCustomerInfo.Tables[CustomerData.CUSTOMERS_TABLE].NewRow();

                    customerRow[CustomerData.EMAIL_FIELD]    =
EmailTextBox.Text;
                    customerRow[CustomerData.PASSWORD_FIELD] = tmpPassword;
                    customerRow[CustomerData.NAME_FIELD]     =
AcctNameTextBox.Text;
                    customerRow[CustomerData.ADDRESS_FIELD]  =
AddressTextBox.Text;
                    customerRow[CustomerData.COUNTRY_FIELD]  =
CountryTextBox.Text;
                    customerRow[CustomerData.PHONE_FIELD]    =
PhoneTextBox.Text;
                    customerRow[CustomerData.FAX_FIELD]      = FaxTextBox.Text;
                    //
                    // Add the row to the dataset
                    //

moduleCustomerInfo.Tables[CustomerData.CUSTOMERS_TABLE].Rows.Add(customerRow);

                    moduleCustomerInfo.AcceptChanges();

                    customerRow[CustomerData.PKID_FIELD]     =
Customer.Tables[CustomerData.CUSTOMERS_TABLE].Rows[0][CustomerData.PKID_FIELD];
                    //
                    // WORKAROUND: End of workaround.
                    //
                    // Save changes to the database
                    //
                    retVal = (new
CustomerSystem()).UpdateCustomer(moduleCustomerInfo);
                }
                else
                {
```

```
                    retVal = (new
CustomerSystem()).CreateCustomer(EmailTextBox.Text,
                                                                   tmpPassword,

AcctNameTextBox.Text,

AddressTextBox.Text,

CountryTextBox.Text,

PhoneTextBox.Text,

FaxTextBox.Text,
                                                                   out
moduleCustomerInfo);
                }
            }

            //
            // Store customer in the Session
            //
            if ( retVal && ( moduleCustomerInfo != null ) )
            {
                //
                // 1. update customer in session
                // 2. update customer in cart
                //
                Customer                = moduleCustomerInfo;
                CarritoCompra().Customer = moduleCustomerInfo;
            }

            return retVal;
        }

        /// <summary>
        ///     Validates then saves off the customer account information.
        ///     <remarks>If this is a new customer, then create the item, or
edit for an existing customer.</remarks>
        ///     <remarks>This is called from the page where this user control
exists.</remarks>
        ///     <retvalue>Returns true on success, false otherwise.</retvalue>
        /// </summary>
        ///


        private void InitializeComponent()
        {
                    this.EmailTextBox.TextChanged += new
System.EventHandler(this.EmailTextBox_TextChanged);
                    this.Load += new System.EventHandler(this.Page_Load);

              }

        //-----------------------------------------------------------------
        // void DisplayErrors:
        //   Displays the error text and marks the fields in error.  The
        //   fields in error are marked in m_CustomerData by the Business
        //   Rules.
        //-----------------------------------------------------------------
        private void DisplayErrors()
```

```
        {
            if ( moduleCustomerInfo != null )
            {
                //
                // Check errors for all of the customer fields.
                //
                DataRow customerRow =
moduleCustomerInfo.Tables[CustomerData.CUSTOMERS_TABLE].Rows[0];
                //
                // Check the email special cases
                //
                String  fieldError  =
customerRow.GetColumnError(CustomerData.EMAIL_FIELD);

                if (0 == String.Compare(fieldError,
CustomerData.EMAIL_FIELD_NOT_UNIQUE))
                    EmailUniqueCustomValidator.IsValid = false;
            }
        }

             private void EmailTextBox_TextChanged(object sender,
System.EventArgs e)
             {

             }

    } // class AccountModule

} // namespace PryEcommerce.Web
```

ModuloPago.cs - Moduloaspx.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Collections;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;

    /// <summary>
    ///     This user control module displays the steps through the checkout
process.
    ///     <remarks>This class is derived from ModuleBase.</remarks>
    /// </summary>
    public class ModuloPago : ModuleBase
    {

        /// <value>The Image for the arrow used to show shipping info is the
current checkout step.</value>
        protected System.Web.UI.WebControls.Image ShipArrowImage;
        /// <value>The Image for the arrow used to show payment info is the
current checkout step.</value>
        protected System.Web.UI.WebControls.Image PaymentArrowImage;
```

```
        /// <value>The Image for the arrow used to show order confirmation is
the current checkout step.</value>
        protected System.Web.UI.WebControls.Image ConfirmArrowImage;
        /// <value>The Label for the text associated with shipping
step.</value>
        protected System.Web.UI.WebControls.Label ShipTextLabel;
        /// <value>The Label for the text associated with payment step.</value>
        protected System.Web.UI.WebControls.Label PaymentTextLabel;
        /// <value>The Label for the text associated with order confirmation
step.</value>
        protected System.Web.UI.WebControls.Label ConfirmTextLabel;

        private int moduleStage;

        /// <value>
        ///     Property Stage is used to set the current stage of checkout for
the user control.
        /// </value>
        public int Stage
        {
            set
            {
                moduleStage = ((value >= 0) && (value < 3)) ? value : 0;

                //
                // set the image and text display according to the current
checkout stage
                //
                ToggleDisplay();
            }
        }

        /// <summary>
        ///     Sets the display for the current stage of checkout.
        ///     <remarks>Called whenever the page is loaded.</remarks>
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            //
            // set the image and text display according to the current checkout
stage
            //
            ToggleDisplay();
        }

             private void InitializeComponent()
             {
                    this.Load += new System.EventHandler(this.Page_Load);

             }

        //------------------------------------------------------------------
        // void ToggleDisplay:
        //   Sets up the images, arrows, etc for the display.
        //------------------------------------------------------------------
        private void ToggleDisplay()
        {
```

```
            switch (moduleStage)
            {
                case 0:
                    ShipArrowImage.Visible = true;
                    PaymentArrowImage.Visible = false;
                    ConfirmArrowImage.Visible = false;
                    ShipTextLabel.ForeColor = System.Drawing.Color.White;
                    ShipTextLabel.Font.Bold = true;
                    PaymentTextLabel.ForeColor = System.Drawing.Color.Silver;
                    PaymentTextLabel.Font.Bold = false;
                    ConfirmTextLabel.ForeColor = System.Drawing.Color.Silver;
                    ConfirmTextLabel.Font.Bold = false;
                    break;

                case 1:
                    ShipArrowImage.Visible = false;
                    PaymentArrowImage.Visible = true;
                    ConfirmArrowImage.Visible = false;
                    ShipTextLabel.ForeColor = System.Drawing.Color.Silver;
                    ShipTextLabel.Font.Bold = false;
                    PaymentTextLabel.ForeColor = System.Drawing.Color.White;
                    PaymentTextLabel.Font.Bold = true;
                    ConfirmTextLabel.ForeColor = System.Drawing.Color.Silver;
                    ConfirmTextLabel.Font.Bold = false;
                    break;

                case 2:
                    ShipArrowImage.Visible = false;
                    PaymentArrowImage.Visible = false;
                    ConfirmArrowImage.Visible = true;
                    ShipTextLabel.ForeColor = System.Drawing.Color.Silver;
                    ShipTextLabel.Font.Bold = false;
                    PaymentTextLabel.ForeColor = System.Drawing.Color.Silver;
                    PaymentTextLabel.Font.Bold = false;
                    ConfirmTextLabel.ForeColor = System.Drawing.Color.White;
                    ConfirmTextLabel.Font.Bold = true;
                    break;
            }
        }

    } // class CheckoutModule
} // namespace PryEcommerce.Web
```

ModuloSeleccionDia.cs - ModuloSeleccion.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Collections;
    using System.ComponentModel;
    using System.Data;

    using PryEcommerce.CapaNegocio;
    using PryEcommerce.Comun;
```

```
using PryEcommerce.Comun.Data;
using PryEcommerce.SystemFramework;

/// <summary>
///     This user control module displays the DailyPicks for a category.
///     <remarks>This class is derived from ModuleBase.</remarks>
/// </summary>
public class ModuloSeleccionDia : ModuleBase
{
    //
    // DataView for the DailyPick items in the selected category
    //
    private DataView moduleItemView;
         protected System.Web.UI.WebControls.DataList DailyPickList;
         protected System.Web.UI.WebControls.Panel DailyPickPanel;
    //
    // CategoryId for the DailyPick items
    //
    private int moduleCategoryID = 830;  //Default for DailyPick


    /// <value>
    ///     Property ItemList is used to get the DataView that contains the
list of Daily Pick
    ///     products for the currently selected category.
    /// </value>
    public DataView ItemList
    {
        get
        {
            return moduleItemView;
        }
    }


    /// <value>
    ///     Property CategoryID is used to get or set the category id to
retrieve daily pick products for.
    ///     <remarks>
    ///         Sets the value CategoryID.
    ///         Gets the value CategoryID.
    ///     </remarks>
    /// </value>
    public int CategoryID
    {
        get
        {
            return moduleCategoryID;
        }
        set
        {
            moduleCategoryID = value;
        }
    }


    /// <summary>
    ///     Populate the user control with the daily pick items.
    ///     <remarks>Called whenever the page is loaded.</remarks>
    ///     <param name="sender">The source of the event.</param>
```

```
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        protected void Page_Load(Object sender, EventArgs e)
        {
            if ( !IsPostBack )
            {
                loadDailyPicks();
            }
        }

        //-----------------------------------------------------------------
        // Function loadDailyPicks:
        //   Retrieves the DailyPick information for the currently
        //      selected category and binds it to the UI.  If no data is
        //      found for the category the control is not shown.
        // Parameters:
        // Returns:
        //   bool - true for success, false for failure
        //-----------------------------------------------------------------
        private bool loadDailyPicks()
        {
            //
            // Read the products in the category
            //
            ProductData ItemSet = (new
ProductSystem()).GetDailyPickItems(moduleCategoryID);
            //
            // Save the view for data binding
            //
            moduleItemView =
ItemSet.Tables[ProductData.PRODUCTS_TABLE].DefaultView;
            //
            // If everything succeeded, then enable page caching as indicated
            // by the current application configuration.
            //
            if (PryEcommerceConfiguration.EnablePageCache)
            {
                //Enable  Page Caching...
                Response.Cache.SetExpires (
DateTime.Now.AddSeconds(PryEcommerceConfiguration.PageCacheExpiresInSeconds));
                Response.Cache.SetCacheability(HttpCacheability.Public);
            }

            if (moduleItemView.Count > 0)
            {
                //
                // Remind the page that it needs bind its controls to the data.
                //
                DataBind();

                DailyPickPanel.Visible = true;
                return true;
            }
            else
            {
                DailyPickPanel.Visible = false;
                return false;
            }
        }
```

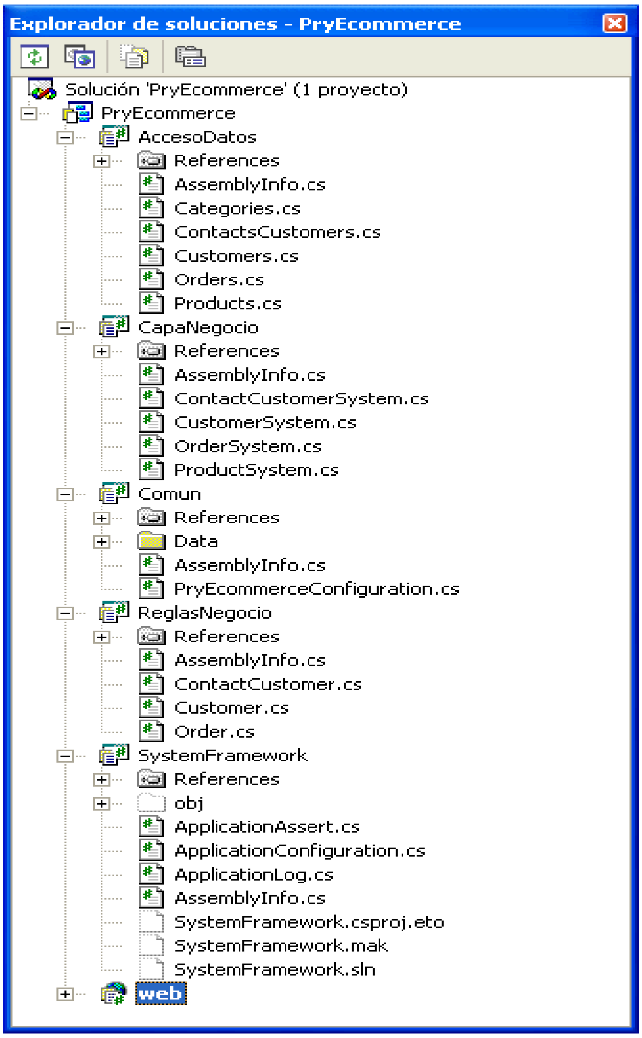
```
        private void InitializeComponent()
        {
                    this.Load += new System.EventHandler(this.Page_Load);

              }


        /// <summary>
        ///     Adds a predefined product to the shopping cart.
        ///     <param name="sender">The source of the event.</param>
        ///     <param name="e">An EventArgs that contains the event
data.</param>
        /// </summary>
        public void AddToCartButton_Click(Object sender, EventArgs e)
        {
            Button itemButton = (Button)sender;
            string arguments = itemButton.CommandArgument;
            char [] parsChar = {'|'};
            string[] addArgs = arguments.Split(parsChar);
            CarritoCompra().AddItem(Int32.Parse(addArgs[0]), addArgs[1] ,
Decimal.Parse(addArgs[2]));
            Response.Redirect("CarritoCompra.aspx", false);

        }

             private void DailyPickList_SelectedIndexChanged(object sender,
System.EventArgs e)
             {

             }

    } // class SeleccionDia
} // namespace PryEcommerce.Web
```

ModuloServicios.cs - ModuloServicios.ascx

```
namespace PryEcommerce.Web
{
    using System;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Collections;
    using System.ComponentModel;
    using System.Data;

    /// <summary>
    ///     This user control module used to link to the behind the scenes
viewsource.
    ///     <remarks>This class is derived from ModuleBase.</remarks>
    /// </summary>
    public class ModuloServicios : ModuleBase
    {
             protected System.Web.UI.WebControls.Image Line2Image;
             protected System.Web.UI.WebControls.HyperLink HyperLink1;
             protected System.Web.UI.WebControls.HyperLink HyperLink2;
```

```
		protected System.Web.UI.WebControls.HyperLink HyperLink3;
		protected System.Web.UI.WebControls.Image Line1Image;

		private void InitializeComponent()
		{
			this.Load += new System.EventHandler(this.Page_Load);

		}

	/// <summary>
	///     Initialize the user control.
	///     <remarks>Called whenever the page is loaded.</remarks>
	///     <param name="sender">The source of the event.</param>
	///     <param name="e">An EventArgs that contains the event
data.</param>
	/// </summary>
	protected void Page_Load(Object sender, EventArgs e)
	{
			string prefix = PathPrefix;

	//    Image1.ImageUrl = prefix + "/images/banner/reminder.gif";
	    Line2Image.ImageUrl = prefix + "/images/banner/line.gif";
	//    Image2.ImageUrl = prefix + "/images/banner/asistente.gif";
	//    Image3.ImageUrl = prefix + "/images/banner/grupo.gif";
	 //   Image4.ImageUrl = prefix + "/images/banner/line.gif";

			//'      Line1Image.ImageUrl + PathPrefix &
"/images/banner/line.gif"
	}


	/// <value>
	///     Property SourceUrl is used to set the URL of the hyperlink for
source viewing.
	/// </value>
//        public String SourceUrl
//        {
//            set
//            {
//                //SourceHyperLink.NavigateUrl = PageBase.UrlBase + "/" +
value;
//
//            }
//        }

    }
      // class About
} // namespace PryEcommerce.Web
```

Resto de Capas



Capa Comun:

Referencias a Librerías:

Librerías Componentes de Proyecto:
- SystemFramework.dll

Librerías .Net :
- MsCorLib.dll
- System.dll
- System.Data.dll
- System.Xml

Archivos de Clases:

AssemblyInfo.cs
CategoryData.cs
ContactCustomerData.cs
CustomerData.cs
OrderData.cs
ProductData.cs
PryEcommerceConfiguration.cs

Capa de Negocio

Referencias a Librerías:

Librerías Componentes de Proyecto:
- AccesoDatos.dll
- Comun.dll
- ReglasNegocio.dll
- SystemFramework.dll

Librerías .Net :
- MsCorLib.dll
- System.dll
- System.Data.dll
- System.Xml

Archivos de Clases:

AssemblyInfo.cs

ContactCustomerSystem.cs
CustomerSystem.cs
OrderSystem.cs
ProductSystem.cs

Capa de ReglasNegocio

Referencias a Librerías:

Librerías Componentes de Proyecto:
- AccesoDatos.dll
- Comun.dll
- SystemFramework.dll

Librerías .Net :
- MsCorLib.dll
- System.dll
- System.Data.dll
- System.Xml

Archivos de Clases:

AssemblyInfo.cs
ContactCustomer.cs
Customer.cs
Order.cs

Capa de Acceso a Datos

Referencias a Librerías:

Librerías Componentes de Proyecto:
- Comun.dll
- SystemFramework.dll

Librerías .Net :
- MsCorLib.dll
- System.dll
- System.Data.dll

System.Xml

Archivos de Clases:

AssemblyInfo.cs
Categories.cs
ContactsCustomers.cs
Customers.cs
Orders.cs
Products.cs

Capa de SystemFramework

Referencias a Librerías:

Librería .Net :
MsCorLib.dll
System.dll
System.Data.dll
System.Xml

Archivos de Clases:

AssemblyInfo.cs
ApplicationAssert.cs
ApplicationConfiguration.cs
ApplicationLog.cs

BIBLIOGRAFÍA

1. BORGHELLO CRISTIAN F., Seguridad Informática: Sus Implicaciones e Implementación (Tesis Universidad Tecnológica Nacional de Argentina, 2001)

2. FIGUEROA PABLO, Etapas y actividades en el desarrollo OO basado en UML, Versión 1.1

3. ICSA Inc., Firewall Buyer's Guide, 1998

4. NOVA PEDRO CONCEPCION, Análisis y Diseño de Sistemas, Azua – Republica Dominicana, 2002

5. ROGER S. PRESSMAN, Ingeniería del Software, Un enfoque practico, Tercera Edición, Editorial McGraw Hill

6. SOUDERS CINDY, Fifteen tips for properly securing IIS Web servers, Agosto 2003

7. WARWICK FORD and MICHAEL S. BAUM, Secure Electronic Commerce, Second Edition, Editorial Prentice Hall PTR

8. WILLIAM STALLINGS, Netowrk Security Essentials, Applications and Standards, Second Edition, Editorial Prentice Hall

9. MICROSOFT CORPORATION, Guía del Desarrollador de .NET Framework (http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpguide/html/cpovrIntroductionToNETFrameworkSDK.asp)

10. MICROSOFT CORPORATION, Microsoft Internet Security and Acceleration (ISA) Server 2000 Caching Overview

11. MICROSOFT CORPORATION, Microsoft Internet Security and Acceleration (ISA) Server 2000 Firewall Security Services Overview, 2001

12. MICROSOFT CORPORATION, Microsoft Internet Security and Acceleration (ISA) Server 2000 Installation and Deployment Guide

13. MICROSOFT CORPORATION, MSA Enterprise Data Services (http://www.microsoft.com/resources/documentation/msa/edc/all/solution/en-us/pak/pag/edcpag05.mspx)

14. MICROSOFT CORPORATION, MSDN Online, Visual Studio .NET (http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/vbcon/html/vboriManagedDevelopmentStartPage.asp)

15. MICROSOFT CORPORATION, Solutions for Security, 2003

16. MICROSOFT CORPORATION, SQL Server 2000 Resource Kit (http://www.microsoft.com/resources/documentation/sql/2000/all/reskit/en-us/part1/c0161.mspx)

17. MICROSOFT CORPORATION, SQL Server 2000 SP3 Security Features and Best Practices: Security Best Practices Checklist, Mayo 2003 (http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/sp3sec04.mspx)

18. MICROSOFT CORPORATION, Vistual Studio Samples Duwamish 7.0 (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dwamish7/html/vtoriDuwamish70Overview.asp)

19. MICROSOFT CORPORATION, Windows 2000 Security Hardening Guide, 2003