ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA SUITE DE CONTROLES DE SERVIDOR ASP.NET ESPECIALIZADOS PARA APLICACIONES DE TIPO E-COMMERCE

PROYECTO PREVIO A LA OBTECIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

MARCELO VINICIO VALLEJO CEVALLOS

marcmvc@hotmail.com

DIRECTOR: ING. ANDRÉS LARCO

andres.larco@epn.edu.ec

Quito, Septiembre de 2011

DECLARACIÓN

Yo Marcelo Vinicio Vallejo Cevallos, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Marcelo Vinicio Vallejo Cevallos

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Marcelo Vinicio Vallejo Cevallos, bajo mi supervisión.

Ing. Andrés Larco
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

A mi familia, por su constante apoyo en el transcurso de mi vida.

CONTENIDO

INTRODUCCIÓN……1

CAPÍTULO 1. MARCO TEÓRICO 3

1.1. PLANTEAMIENTO DEL PROBLEMA 3

1.1.1. ANTECEDENTES 3

1.1.2. DESCRIPCIÓN DEL PROBLEMA 3

1.2. CRITERIOS PARA SELECCIÓN DE CONTROLES DE SERVIDOR ASP.NET 5

1.2.1. PRINCIPALES PROVEEDORES DE CONTROLES DE SERVIDOR ASP.NET .. 5

1.2.2. DESCRIPCIÓN DE CRITERIOS PARA SELECCIÓN DE CONTROLES DE SERVIDOR ASP.NET 13

1.2.3. AYUDA SOBRE LOS CONTROLES 13

1.2.4. SOPORTE TÉCNICO A LOS CONTROLES 14

1.2.5. EJEMPLOS ONLINE QUE DISPONEN CADA UNA DE LAS SUITES 14

1.2.6. LICENCIAMIENTO 14

1.2.7. REPUTACIÓN DEL FABRICANTE 14

1.2.8. FACILIDAD DE USO 14

1.2.9. ESTANDARIZACIÓN 14

1.2.10. MODELO DE PROGRAMACIÓN 14

1.2.11. CASOS DE ÉXITO 14

1.2.12. COBERTURA FUNCIONAL 14

1.3. ANÁLISIS DE LOS CONTROLES DE SERVIDOR SELECCIONADOS 16

1.3.1. LICENCIAMIENTO 16

1.3.2. FUNCIONALIDAD Y FACILIDAD DE USO 17

1.3.3. PERFORMANCE 17

1.4. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA 18

CAPÍTULO 2. DESARROLLO DE LA SUITE DE CONTROLES 21

2.1. FASE DE ANÁLISIS 21

2.1.1. VISIÓN 21

2.1.2. DEFINICIÓN DE REQUERIMIENTOS 24

2.1.3. ESPECIFICACIÓN DE CASOS DE USO 27

2.1.4. DEFINICIÓN DE ARQUITECTURA 44

2.2. FASE DE DISEÑO....51
2.2.1. DISEÑO FUNCIONAL EHEADER....52
2.2.2. DISEÑO FUNCIONAL EMENU....53
2.2.3. DISEÑO FUNCIONAL EPRODUCTCATALOG....55
2.2.4. DISEÑO FUNCIONAL ESHOPPINGCART....57
2.2.5. DISEÑO FUNCIONAL EVIRTUALKEYBOARD....59
2.2.6. DISEÑO FUNCIONAL ECAPTCHA....60
2.2.7. DISEÑO FUNCIONAL EPERSONALDATA....62
2.2.8. DISEÑO FUNCIONAL EPAYMENT....64
2.2.9. DISEÑO FUNCIONAL ESHIPPINGADDRESS....65
2.3. FASE DE IMPLEMENTACIÓN....67
2.4. FASE DE PRUEBAS....79
CAPÍTULO 3. EVALUACIÓN DE LA SUITE DE CONTROLES APLICANDO A UN CASO DE ESTUDIO....80
3.1. DEFINICIÓN DE REQUISITOS DEL PROTOTIPO (E-COMMERCE)....80
3.2. ELABORACIÓN DEL PROTOTIPO USANDO LA SUITE DE CONTROLES....83
3.2.1. ARQUITECTURA DE LA APLICACIÓN....83
3.2.2. DISEÑO FUNCIONAL....84
3.2.3. INTERFACES....85
3.3. PRUEBAS DEL PROTOTIPO....91
3.3.1. ESPECIFICACIÓN DE PRUEBAS....92
3.3.2. EJECUCIÓN DE LAS PRUEBAS....92
3.4. ANÁLISIS DE RESULTADOS....98
CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES....101
4.1. CONCLUSIONES....101
4.2. RECOMENDACIONES....102
BIBLIOGRAFÍA....103
ANEXOS....104

LISTA DE TABLAS

Tabla 1-1 Comparativo de las suites de controles existentes 15
Tabla 2-1 Plataforma Tecnológica para el desarrollo de la aplicación 47
Tabla 2-2 Hardware y Software para el desarrollo de la suite de controles 48
Tabla 2-3 Definición de librerías para la suite de controles..... 50
Tabla 3-1 Plataforma tecnológica para el desarrollo del prototipo..... 83
Tabla 3-2 Requerimientos Hardware y Software para el caso de estudio..... 84

LISTA DE FIGURAS

Figura 1-1 Listado de Controles Telerik 7
Figura 1-2 Listado de Controles COMPONENTONE 8
Figura 1-3 Listado de Controles INFRAGISTICS 10
Figura 1-4 Listado de Controles DEVEXPRESS 11
Figura 1-5 Listado de controles EXT.NET 13
Figura 1-6 Estructura aplicación eCommerce 18
Figura 2-1 Modelo Vista Presentador (MVP)..... 46
Figura 2-2 Prototipo EHeader..... 52
Figura 2-3 Diagrama de Clases EHeader 53
Figura 2-4 Prototipo EMenu 54
Figura 2-5 Diagrama de Clases EMenu 55
Figura 2-6 Prototipo EProductCatalog..... 56
Figura 2-7 Diagrama de clases EProductCatalog 56
Figura 2-8 Prototipo EShoppingCart 57
Figura 2-9 Diagrama de Clases EShoppingCart 58
Figura 2-10 Prototipo EVirtualKeyboard..... 59
Figura 2-11 Diagrama de Clases EVirtualKeyboard..... 60
Figura 2-12 Prototipo ECaptcha..... 61
Figura 2-13 Diagrama de Clases ECaptcha..... 61
Figura 2-14 Prototipo EPersonalData..... 62
Figura 2-15 Diagrama de Clases EPersonalData..... 63
Figura 2-16 Prototipo EPayment 64
Figura 2-17 Diagrama de Clases EPayment 65

Figura 2-18 Prototipo EShippingAddress 66
Figura 2-19 Diagrama de Clases EShippingAddress 67
Figura 2-20 Prototipo EGrid..... **¡Error! Marcador no definido.**
Figura 2-21 Diagrama de Clases EGrid ..... **¡Error! Marcador no definido.**
Figura 2-22 Asignación valor, propiedad FullName en control EShippingAddress75
Figura 2-23 Agregar librería de controles al Toolbox de Visual Studio..... 77
Figura 2-24 Seleccionar librería de los nuevos controles..... 78
Figura 2-25 Suite de Controles agregada al Toolbox de Visual Studio 78
Figura 2-26 Control EShippingAddress implementado en una página de ejemplo79
Figura 3-1 Prototipo ECommerce..... 85
Figura 3-2 Implementación control EHeader..... 85
Figura 3-3 Implementación control EMenu..... 86
Figura 3-4 Control EMenu desplegando categorías..... 86
Figura 3-5 Implementación EShoppingCart 87
Figura 3-6 Implementación control ECaptcha 88
Figura 3-7 Implementación control EShippingAddress..... 89
Figura 3-8 Implementación control EVirtualKeyboard 89
Figura 3-9 Implementación control EShoppingCart..... 90
Figura 3-10 Implementación control EPayment..... 91
Figura 3-11 Implementación control EPersonalData..... 91
Figura 3-12Vista Tiempo de Diseño control EMenu..... 92
Figura 3-13 Ventana de propiedades control EMenu..... 93
Figura 3-14 Vista Tiempo de Diseño control EPersonalData 93
Figura 3-15 Ventana de propiedades control EPersonalData 94
Figura 3-16 Vista Tiempo de Diseño control EShippingAddress..... 94
Figura 3-17 Ventana de propiedades control EShippingAddress..... 95
Figura 3-18 Vista Tiempo de Diseño control EProductCatalog 95
Figura 3-19 Ventana de Propiedades control EProductCatalog..... 96
Figura 3-20 Vista Tiempo de Diseño control EHeader 96
Figura 3-21 Ventana de propiedades control EHeader 97
Figura 3-22 Vista Tiempo de Diseño control EShoppingCart..... 97
Figura 3-23 Ventana de propiedades control EShoppingCart..... 98
Figura 3-24 Incorporación de colores en el control EMenu..... 99

Figura 3-25 Cambio de valores mediante ventana de propiedades 99

INTRODUCCIÓN

El proyecto “Desarrollo de una Suite de Controles de Servidor ASP.NET para aplicaciones de tipo e-commerce”, es un proyecto que utiliza los principios básicos del desarrollo de software, por tanto su herramienta fundamental será la aplicación de una metodología relacionada al desarrollo de software para el cumplimiento de las expectativas creadas sobre el presente proyecto.

Las aplicaciones de comercio electrónico se están difundiendo y desarrollando constantemente, por tanto es necesario desarrollar este tipo de aplicaciones de una manera rápida y usando la menor cantidad de recursos, pues eso determina la diferencia entre ingresar al mercado electrónico o quedarse fuera del mismo.

En el presente proyecto de Titulación se desarrollará una suite de controles de servidor ASP.NET para aplicaciones de tipo e-commerce, permitiendo que desarrolladores de ese tipo de aplicaciones disminuyan el tiempo de desarrollo al tener una serie de controles robustos que puedan incorporarse a soluciones e-commerce de una manera simple y consistente.

En el primer capítulo se detalla la problemática existente acerca del desarrollo de aplicaciones de comercio electrónico y cómo influyen los controles de servidor sobre las mismas. Adicionalmente se establece una comparativa entre las suites de controles más importantes que existen actualmente en el mercado, a través de una serie de criterios que permiten definir claramente si una suite de controles es mejor a otra.

En el capítulo dos se establece el proceso de desarrollo de la suite de controles siguiendo y adaptando la metodología de desarrollo de software Open UP, desarrollando los artefactos de esa metodología que realmente brinden valor agregado en el desarrollo de la aplicación.

En el tercer capítulo se aplica la suite de controles desarrollada a un caso de estudio que para este caso es una aplicación de comercio electrónico para validar el correcto funcionamiento de los controles creados y que además cumplan con el estándar de programación de controles de servidor que define la tecnología ASP.NET.

Por último, el capítulo cuatro muestra las conclusiones tomadas a partir del desarrollo del presente proyecto y se menciona una serie de recomendaciones que se pueden adaptar en proyectos de este mismo tipo.

CAPÍTULO 1. MARCO TEÓRICO

1.1. PLANTEAMIENTO DEL PROBLEMA

1.1.1. ANTECEDENTES

De manera general, los desarrolladores de aplicaciones de tipo e-commerce consumen y usan un prolongado tiempo en el desarrollo de los mismos controles de servidor que se requieren para una aplicación de comercio electrónico, los cuales pueden ser suministrados por terceros a través de una suite de controles y disminuir el tiempo de desarrollo en esa tarea repetitiva.

Esta problemática ocurre permanentemente en cada nuevo desarrollo de aplicaciones de tipo e-commerce y afecta en primer lugar el costo del proyecto, en segundo lugar el tiempo en el cronograma de trabajo previamente estipulado, en tercer lugar en la calidad misma del producto y en la imagen de la empresa que genera el producto dado que su imagen corporativa queda empañada si se produce una aplicación poco competitiva, no obstante, la mayoría de empresas ponen en producción esos sistemas pero no aseguran los márgenes de calidad mínimos.

1.1.2. DESCRIPCIÓN DEL PROBLEMA

Los controles de servidor, al ser el componente principal para la construcción de aplicaciones ASP.NET, se transforman o determinan la calidad y la duración del desarrollo de cualquier tipo de aplicación Web, basada en esa tecnología.

Precisamente, el problema del excesivo tiempo de duración en el desarrollo de aplicaciones de tipo e-commerce al momento de realizar una tarea repetitiva como la composición de controles para formar la estructura o la apariencia visual de una aplicación de comercio electrónico, hace notar la falta de una suite de controles específica para ese tipo de aplicaciones que evite realizar la tarea descrita anteriormente.

No se ha señalado falta de conocimiento por parte de los desarrolladores de aplicaciones de e-commerce en tecnología ASP.NET, pero si se ha notado el tiempo que éstos últimos consumen en realizar una tarea. En otras palabras, los desarrolladores son capaces técnicamente, pero consumen tiempo valioso el cual se podría enfocar en otras actividades más complejas que aparecen durante el desarrollo, como por ejemplo los mecanismos de autenticación, autorización, seguridad, etc. Lo que realmente brinda valor agregado a la aplicación que se está construyendo.

Si se considera que el tiempo es dinero en el desarrollo de cualquier tipo de aplicación y que emplearlo en actividades más complejas que brindan valor agregado a la aplicación, implica al final del desarrollo un producto de calidad, se podría decir que se emplea demasiado tiempo en una tarea relacionada a la construcción y composición de controles de servidor, que si bien no se puede obviar, se puede agilizar re-usando determinados componentes comunes a través de una suite de controles que contenga los mismos, por ejemplo: un componente de menú, un banner de publicidad, un visor de productos, formularios para el ingreso de datos personales y formas de pago para los diferentes productos, un componente de autenticación, un teclado virtual, un componente de Captcha[1], un componente para un Shopping Cart[2].

Desde el punto de vista del desarrollo de software y sus principios, se identifica un inconveniente como es la no reutilización de componentes o código en una actividad repetitiva dada, lo que ocasiona pérdida de tiempo, incluso duplicación de código. La falta de una suite de controles de servidor para aplicaciones de tipo e-commerce y por ende la no aplicación de los principios de desarrollo de software, conllevan a un mal uso del tiempo, y mala calidad del producto final. Los desarrolladores de este tipo de aplicaciones en ocasiones se equivocan en la construcción y composición de los controles de servidor previamente descritos, lo que provoca una elevación en los costos del desarrollo por pérdida de tiempo y mala calidad.

[1] Un captcha es una prueba de desafío empleada en informática para comprobar si un usuario es humano.
[2] Permite visualizar las compras realizadas por un cliente en una tienda virtual.

Como se puede colegir de lo descrito anteriormente, la problemática (porque se trata de un conjunto de problemas interrelacionados) gira en torno al tiempo de desarrollo de componentes comunes y repetitivos en aplicaciones de comercio electrónico, más específicamente, ese desarrollo eleva los costos de esas aplicaciones e incrementa el margen de incertidumbre porque de antemano no se puede saber cómo quedarían construidos esos componentes y cuál sería el resultado de emplearlos en un ambiente de producción donde se puede tener una gran demanda de usuarios, por tanto se reconoce la existencia de los problemas señalados y se dice que se ocasiona porque actualmente no existe una suite de controles que se encargue de solucionar la problemática identificada, a través del suministro de controles comunes que se requieren para una aplicación de tipo e-commerce, que permita disminuir el tiempo de desarrollo y mejorar la calidad en ese tipo de aplicaciones, porque el mercado se enfoca en proveer controles de uso general que por razones de negocio o cobertura funcional no se enfocan para determinadas aplicaciones y que por tanto el desarrollador debe realizar un trabajo adicional para especializar esos controles para el desarrollo de su aplicación cualquiera que esta sea, incluyendo en ellas las aplicaciones de comercio electrónico.

Como se puede observar, existen problemas ligados al incremento en el tiempo de desarrollo de este tipo de aplicaciones y que se expresan como la falta de una suite de controles específica, y a partir de esta última se podría disminuir el tiempo al proveer controles de servidor que cubran las necesidades de una aplicación de comercio electrónico y que sean más simples de usar.

1.2. CRITERIOS PARA SELECCIÓN DE CONTROLES DE SERVIDOR ASP.NET

1.2.1. PRINCIPALES PROVEEDORES DE CONTROLES DE SERVIDOR ASP.NET

1.2.1.1. TELERIK

La empresa Telerik fue fundada en el año 2002, se tiene datos de que su planta de empleados bordea alrededor de 500 empleados y su primera versión de controles para ASP.NET apareció en el tercer trimestre del año 2007.

Su producto o suite de controles para ASP.NET son los RadControls que están construidos sobre la tecnología ASP.NET AJAX y no introducen una implementación propietaria de AJAX. Como resultado se tienen beneficios al mantenerse usando una tecnología estándar. Por ende al usar los controles de Telerik las aplicaciones serán más rápidas y seguras como la tecnología Microsoft ASP.NET AJAX la cual tiene beneficios de rendimiento gracias al caché de recursos que posee.

Adicionalmente usando los RadControls no es necesario aprender otra API de programación, dado que el único conocimiento que se requiere es la tecnología ASP.NET AJAX. El API de los RadControls empata o es muy similar a la de los controles ASP.NET básicos, estos controles usan los mismos eventos, nombres y métodos base como el Framework de Microsoft.

Ofrece soporte para Microsoft Visual Studio y .NET Framework 4.0, incrementan la experiencia de desarrollo en Visual Studio a través del soporte de plantillas de usuario, Toolbox, integración con el motor de IntelliSense de Java Script, soporte inteligente en tiempo de diseño.

Los controles de Telerik también poseen un soporte completo para el Framework 3.5 de .NET y por ende soportan LINQ, LinqDataSource, EntityDataSource, ADO.NET Data Services, MVC, y mientras los RadControls permiten adoptar las últimas tecnologías, también tienen soporte para Framework 2.0

1.2.1.1.1. LISTADO DE CONTROLES TELERIK

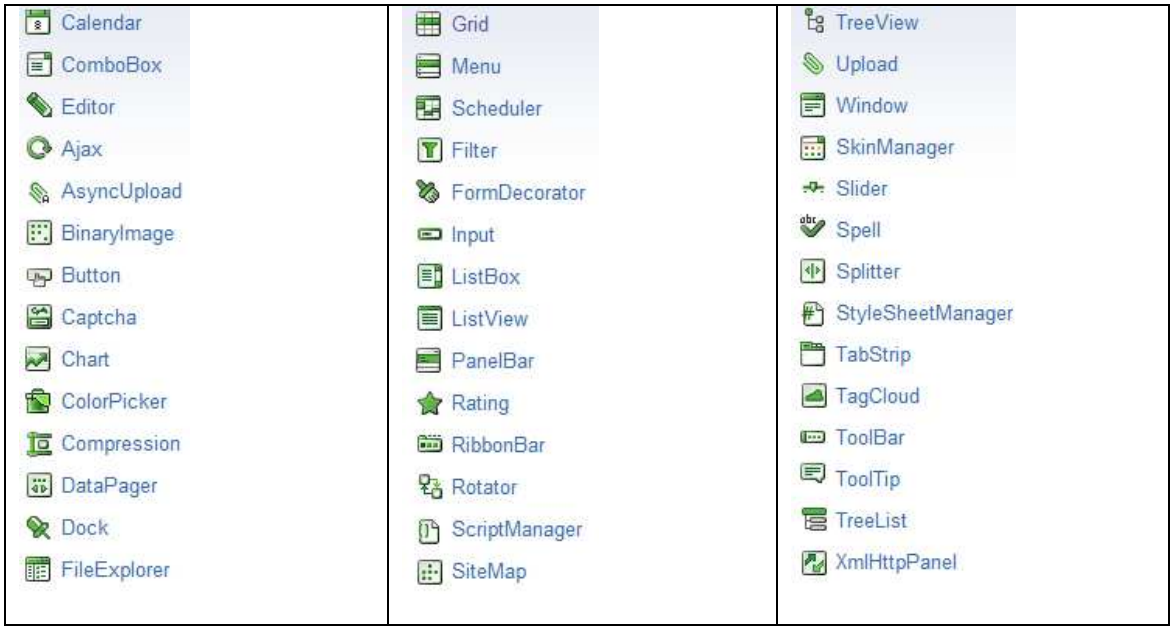

Figura 1-1 Listado de Controles Telerik
Elaborado por: Marcelo Vallejo C.

1.2.1.2. COMPONENTONE

La empresa fue fundada en año 1990 y ofrece un sólido conjunto de controles relacionados con la empresa de hoy y las necesidades de la mayoría de los desarrolladores de ASP.NET, en un mundo que depende cada vez más del código del desarrollador, así todos los controles de ComponentOne tienen un modelo flexible para el programador.

Al revisar las suites de controles, se debe tener algunas métricas puntuales para comparar, sabiendo que el número de controles es difícil de cuantificar. El conjunto de ComponentOne añade unas tres docenas de controles a la caja de herramientas de Visual Studio, una media docena menos que Telerik ASP.NET. Un punto a favor de estos controles son las herramientas de reportes que hacen de esta una suite única.

Al final, una de las ventajas más grandes de esta suite es la facilidad de uso, implementación y excelente funcionalidad de estos controles y lo que más destaca es que están diseñados para que los desarrolladores sean productivos.

La Configuración de la capa de datos en la parte web es rápida e intuitiva con el diseñador.

ComponentOne mejora la visualización de datos, acceso a datos, y la interactividad de los sitios sin necesidad de una línea de código.

Todos los elementos de ComponentOne poseen una interfaz de configuración que los usuarios de cualquier nivel técnico pueden utilizar, integrando características de seguridad y control de versiones.

ComponentOne se instala en el servidor y está listo para funcionar, no son herramientas de desarrollo, están empaquetadas y listas para que el programador pueda agregar arrastrar y agregar a la aplicación.

Posee un asistente de instalación que instala cualquier combinación de elementos Web a cualquier número de servidores. Los instaladores están disponibles para máquinas de 32 bits y 64 bits y los tiempos de carga son bastante rápidos.

1.2.1.2.1. LISTADO DE CONTROLES COMPONENTONE

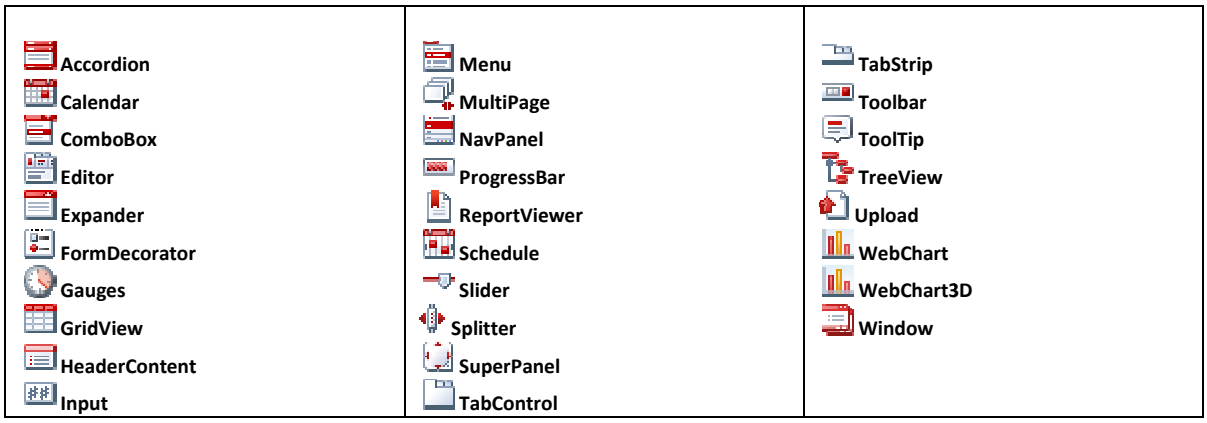

Figura 1-2 Listado de Controles COMPONENTONE
Elaborado por: Marcelo Vallejo C.

1.2.1.3. INFRAGISTICS

La nueva versión de NetAdvantage para el conjunto de controles de ASP.NET permite dar un estilo de Microsoft Office 2010. La exportación a Excel, XPS y PDF, además de un rendimiento mejorado con el estado del cliente optimizada, minified (y comprimido) CSS, y otras características, código más flexible y controles avanzados ASP.NET AJAX que se necesitan para crear rápidamente un estilo de alta calidad y diseño, características ASP.NET AJAX para aplicaciones empresariales.

Este conjunto de herramientas incluye dos redes de ASP.NET AJAX incluida la WebHierarchicalDataGrid, tablas, calendarios, menús, árboles, pestañas, edición de HTML, corrector ortográfico integrado a los controles, CAPTCHA, barras de explorador, la programación, la entrada de los editores de datos y otras características más que permiten crear aplicaciones Web como si fueran aplicaciones de escritorio que todos los usuarios están acostumbrados a utilizar.

Cada control incluye un alto rendimiento y funcionalidad extraordinariamente sofisticada, como menús desplegables, capacidades de arrastrar y soltar, soporte CSS y un ajuste de diseño global para todos los controles ASP.NET. Con todo esto, es fácil crear aplicaciones con funcionalidad extrema, facilidad de uso completo.

Ahora con soporte para Mono, una plataforma de código abierto diseñada para permitir a los desarrolladores construir aplicaciones que se ejecuten sobre múltiples sistemas operativos.

InfraShop es un escaparate de la muestra de comercio impulsado por NetAdvantage para ASP.NET AJAX, controles que se ejecutan en un servidor Linux.

En un futuro NetAdvantage for Web Developers, tendrá productos centrados en el desarrollo independiente del servidor, las soluciones basadas en estándares en el navegador moderno basado en HTML 5, CSS 3, los controles de jQuery, ASP. NET MVC y JavaScript.

NetAdvantage for ASP.NET incluye todos los controles ASP.NET para la línea de negocio de interfaces Web de usuario de aplicaciones.

1.2.1.3.1. LISTADO DE CONTROLES INFRAGISTCS

WebCalcManager	**WebDocumentExporter ***	**WebSlider**
WebCaptcha	**WebEditors** WebCurrencyEditor WebDateTimeEditor WebMaskEditor WebNumericEditor WebPercentEditor WebTextEditor	**WebSpellChecker**
WebChart		**WebSplitter**
WebDataGrid		**WebTab**
WebDataMenu		**Infragistics.Documents** PDF Export XPS Export
WebDataTree	**WebHierarchicalDataSource**	**Infragistics.Excel**

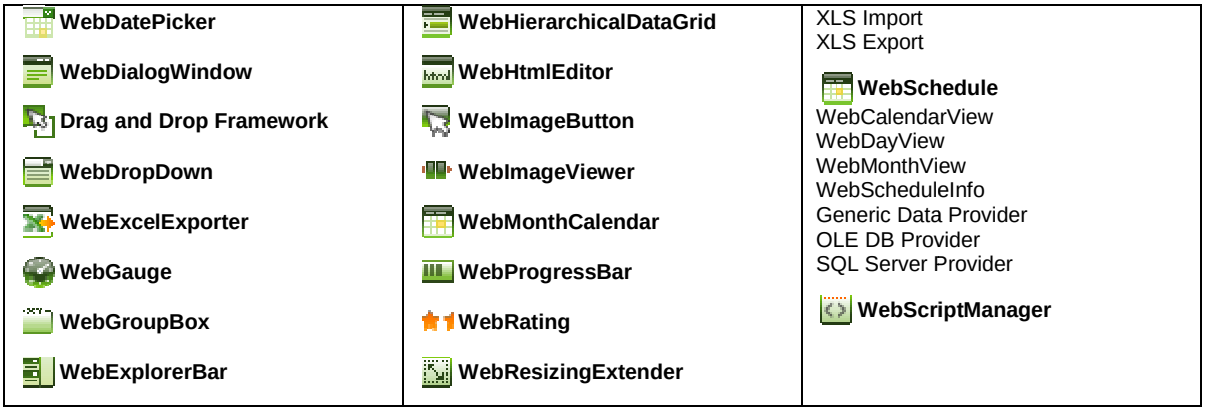

Figura 1-3 Listado de Controles INFRAGISTICS
Elaborado por: Marcelo Vallejo C.

1.2.1.4. DEVEXPRESS

La empresa fue fundada en 1998, su planta de empleados se encuentran en alrededor de 200 personas su primer producto lanzado sobre tecnología ASP.NET se lo hizo en el primer trimestre del año 2008.

Los controles DevExpress ASP.NET son totalmente compatibles con las propiedades de estilo para que se pueda personalizar la apariencia de los elementos individuales y gestionar la apariencia general de un control determinado. Además, todos los controles vienen con una serie de formatos predefinidos automática / presets. Estas configuraciones permiten especificar la apariencia del control y el comportamiento como las necesidades del negocio.

Existe un API permitiendo a la funcionalidad juntarse con la capacidad de controlar los eventos en el cliente lo que permite mantener un control total sobre el comportamiento del control en el cliente.

La velocidad es tal vez uno de los aspectos más importantes de una aplicación. Los Controles DevExpress son altamente optimizados para un rendimiento excepcional y el uso eficiente de la memoria sin la pérdida de funcionalidad o características. Entre las muchas innovaciones a través de la línea de productos de ASP.NET es la disponibilidad de dos fuentes de datos que apoyan en la demanda de carga de datos. DevExpress controles ASP.NET también admite optimizaciones como la compresión de HTML, las devoluciones de llamada, los

sprites [3]de la imagen, el almacenamiento en caché de los recursos y la distribución.

1.2.1.4.1. LISTADO DE CONTROLES DEVEXPRESS

Figura 1-4 Listado de Controles DEVEXPRESS
Elaborado por: Marcelo Vallejo C.

1.2.1.5. EXT.NET

EXT.NET es una suite profesional de controles Web ASP.NET AJAX, los cuales están basados en el Framework de JavaScript ExtJS.

[3] Es una colección de imágenes ubicadas dentro de una sola imagen.

La suite de controles Web está construida y se enfoca en brindar el soporte en tiempo de diseño que posee Visual Studio, obviamente basándose en el Framework de javascript ExtJS. Una combinación entre Frameworks del lado del servidor y del lado del cliente.

La suite de controles incluye lo siguiente:

- Integración del Framework de JavaScript Ext.
- Soporte en tiempo de diseño para Microsoft Visual Studio y Visual Web Developer.
- Doble Licenciamiento (Open Source AGPL3 y Licencia Profesional)

1.2.1.5.1. LISTADO DE CONTROLES EXT.NET

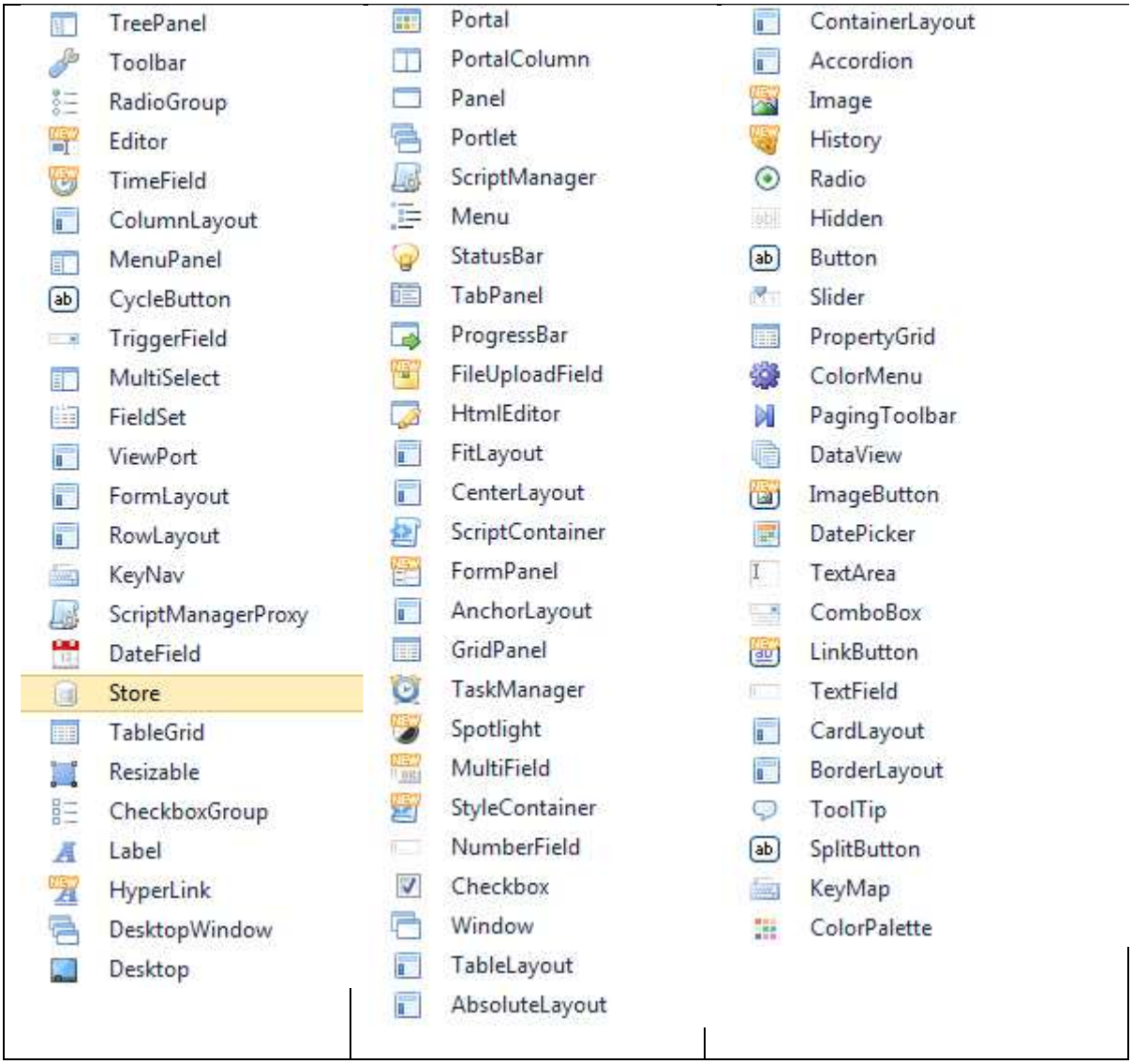

Figura 1-5 Listado de controles EXT.NET
Elaborado por: Marcelo Vallejo C.

1.2.2. DESCRIPCIÓN DE CRITERIOS PARA SELECCIÓN DE CONTROLES DE SERVIDOR ASP.NET

Entre los distintos criterios para selección de controles de servidor ASP.NET se debe considerar lo siguiente:

1.2.3. AYUDA SOBRE LOS CONTROLES

Permite identificar la cantidad de información que existe en foros con respecto a la suite de controles.

1.2.4. SOPORTE TÉCNICO A LOS CONTROLES

Muestra la calidad del soporte técnico, es decir la rapidez y calidad en la respuesta a una solicitud por parte de un usuario.

1.2.5. EJEMPLOS ONLINE QUE DISPONEN CADA UNA DE LAS SUITES

Calidad y cantidad de ejemplos en línea que posee cada una de las suites para mostrar el uso de sus controles.

1.2.6. LICENCIAMIENTO

Permite calificar los esquemas de licenciamiento que posee cada una de las suites, otorgándose un punto favorable para aquellas que presenten un esquema de licenciamiento Dual, lo que a futuro permitirá tener una visión más clara de la suite de controles, porque un esquema con solo 30 días para revisar la suite no es tiempo suficiente para evaluar la misma.

1.2.7. REPUTACIÓN DEL FABRICANTE

Está dada por los comentarios favorables por parte de terceros.

1.2.8. FACILIDAD DE USO

Permite identificar si es fácil o no usar el API de programación propuesta por cada una las suites de controles.

1.2.9. ESTANDARIZACIÓN

Permite identificar si la suite de controles cumple o se asemeja a los estándares de programación de ASP.NET

1.2.10. MODELO DE PROGRAMACIÓN

Permite identificar si la suite de controles empata su modelo de programación con los controles estándar de Microsoft, es decir si es factible combinar controles de servidor estándar de Microsoft con los controles de una suite cualquiera que esta sea.

1.2.11. CASOS DE ÉXITO

Número de empresas que usan una determinada suite de controles.

1.2.12. COBERTURA FUNCIONAL

Tipos de aplicaciones que una determinada suite de controles es capaz de cubrir.

Para los criterios anteriores se construyó la siguiente tabla que califica en una escala de 1 a 10 las suites de controles que se han descrito a lo largo del presente capítulo.

Criterios	Telerik	Devexpress	ComponentOne	Infragistics	Ext.NET
Ayuda sobre los controles	9	9	8	8	8
Calidad Soporte Técnico	10	10	10	10	10
Demos en línea	10	10	8	9	9
Reputación del Fabricante	10	10	8	10	8
Modelo de Programación	10	9	8	9	8
Licenciamiento	9	9	9	9	10
Funcionalidad	9	10	8	8	9
Facilidad de uso	10	9	8	8	8
Performance	9	9	8	6	8
Casos de Éxito	10	9	9	9	6
Estandarización	10	8	8	8	8
Cobertura Funcional	10	10	9	9	9
Total	9,66	9,33	8,42	8,58	8,42

Tabla 1-1 Comparativo de las suites de controles existentes
Elaborado por: Marcelo Vallejo

Nota:

- En el caso de EXT.NET y dado que la suite es relativamente nueva, obtuvo un puntaje inferior con respecto a las otras suites en los casos de éxito.

RESULTADO

- Las dos empresas Telerik y DevExpress obtuvieron una buena calificación por tanto la compra de cualquiera de las dos suites garantiza la inversión que se vaya a realizar.

- El caso de EXT.NET es una apuesta interesante para el uso de una nueva tecnología, obviamente no se lo puede comparar con empresas que están varios años en el mercado.
- Basándose en la calificación anterior, tanto Telerik como DevExpress tienen una ventaja sobre el resto de suites, pero su principal inconveniente es el tema de licenciamiento puesto que no disponen de un esquema de licenciamiento que permita desarrollar el presente proyecto, por tanto se va usar la suite de controles EXT.NET que tiene un esquema de licenciamiento AGPL3, que permite crear controles específicos usando la suite con propósitos de demostración, lo que lamentablemente las otras suites de controles no poseen.

1.3. ANÁLISIS DE LOS CONTROLES DE SERVIDOR SELECCIONADOS

Para el presente proyecto se utilizarán las suites de controles de EXT.NET, la suite estándar de Microsoft y la suite de controles de AJAX, las cuales poseen controles que se pueden combinar para completar el desarrollo de la nueva suite de controles. Para el análisis de la suite seleccionada se describirán los siguientes aspectos:

1.3.1. LICENCIAMIENTO

Ext.NET ofrece un esquema de licenciamiento flexible, el cual permite realizar una composición o mejoramiento de los componentes de esa suite para crear los nuevos controles de servidor ASP.NET sin tener que pagar por el uso de esa suite, dado que el propósito del presente proyecto es puramente educativo o con propósitos de demostración por tanto no es necesario realizar un pago previo por el uso de la suite, adicionalmente el uso no se restringe solamente a 30 días como es el caso de las otras suites de controles descritas anteriormente, así mismo la suite de controles estándar de Microsoft no requiere el pago de una licencia adicional para su uso, sino que simplemente si se tiene un tipo de licencia para el IDE Visual Studio en cualquiera de sus versiones se puede usar sus controles de servidor, lo mismo sucede para la suite de AJAX.

1.3.2. FUNCIONALIDAD Y FACILIDAD DE USO

La suite de controles de servidor Ext.NET es una abstracción o porting del Framework de *“EXTJS”* a .NET, lo que permite adaptar el estilo de programación del Framework antes mencionado al esquema ASP.NET donde existe la separación de código de cliente del código de servidor mediante un archivo con extensión *“.aspx”* y un archivo con extensión *“.cs”* normalmente denominado *“code behind”*, entonces la suite de controles Ext.NET adapta el Framework *“EXTJS”* al esquema ASP.NET obviamente no del todo dado que es una suite nueva todavía existen determinados aspectos que se deben incorporar para adoptar completamente el esquema de programación ASP.NET porque todavía se requiere un uso hasta cierto punto excesivo de *“javascript”*, lo que implica una curva de aprendizaje mayor por parte de los desarrolladores al usar esa suite, entonces lo que busca el presente proyecto es disminuir esa curva de aprendizaje y proveer controles de servidor simples de usar y que se adapten a aplicaciones de tipo e-commerce, porque es de entender que mejorar todos los controles de la suite EXT.NET es un trabajo complejo y está fuera del alcance del presente proyecto, más bien lo que se busca es mejorar y realizar una composición de ciertos controles los cuales se ajusten o se adapten mejor para aplicaciones de comercio electrónico.

No obstante tampoco se puede dejar de lado a la suite de controles estándar de Microsoft, puesto que ésta última posee determinada funcionalidad o controles que pueden ser útiles para realizar la construcción de la nueva suite de controles porque se puede combinar los controles Ext.NET con los controles estándar de Microsoft.

1.3.3. PERFORMANCE

Tanto la suite de Microsoft como Ext.NET poseen un rendimiento aceptable, similar y que comparado con las otras suites de controles tomadas en cuenta para la presente comparativa no existe mayor diferencia.

Cuando hablamos de rendimiento en controles de servidor un aspecto importante es el manejo del view state en cada uno ellos, puesto que de este último depende el tamaño de una página por ende cuán rápido ésta se puede visualizar en un browser y cuántos usuarios pueden acceder a ella simultáneamente, entonces en

ese contexto los controles tanto de Ext.NET como Microsoft se manejan en el rango estándar para el uso del view state y por tanto no representan ningún problema de rendimiento, incluso el HTML que generan los controles de Ext.NET como Microsoft está dentro del rango aceptable, entonces con las suites seleccionadas no se tendrá ningún problema de rendimiento.

1.4. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

La mayor parte de los sitios de comercio electrónico, se enfocan cada vez más en atraer clientes para su tienda virtual y realizan inversiones en áreas como: marketing web, relaciones públicas, optimización para motores de búsqueda, análisis web, investigaciones sobre usabilidad, soporte al cliente, etc. Muchos de los aspectos descritos se los puede resolver usando controles de servidor.

A continuación se muestra la estructura general que poseen todas las aplicaciones de tipo e-commerce, lo único que realmente cambia entre los sitios es el look & feel[4].

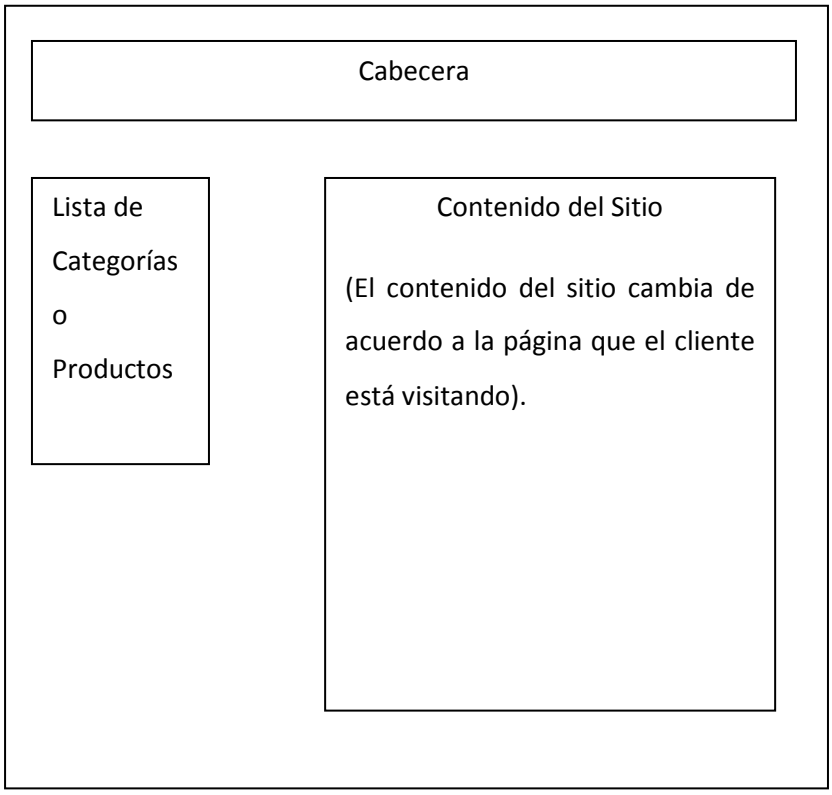

Figura 1-6 Estructura aplicación eCommerce
Elaborado por: Marcelo Vallejo C.

[4] Define la apariencia de un sitio web

Observando la figura anterior se puede desarrollar controles comunes para aplicaciones de comercio electrónico: un control de menú que permita desplegar la lista de categorías o productos, un control para desplegar la cabecera del sitio que generalmente involucra el logotipo de la empresa o de la tienda virtual.

Con respecto al contenido del sitio este puede variar, pero todas las tiendas virtuales en ese contenido muestran un control para visualizar el catálogo de productos que ofrece la tienda con la descripción, fotografía y costo de cada producto. Una vez que el cliente selecciona un determinado producto este ingresa a un carrito de compras el cual se desarrolla usando un control de servidor. Adicionalmente se puede incluir un control de Captcha para validar la información que ingresa el cliente, también están presentes los formularios para el ingreso tanto de datos personales como la información de pago, esta última es información sensible por tanto para el caso de ingreso de información relacionada a la tarjeta de crédito se debe usar un teclado virtual.

Basándose en el análisis anterior la nueva suite de controles para el presente proyecto incluye los siguientes componentes:

- Control de menú (EMenu).
- Control para presentar logotipos (EHeader).
- Teclado Virtual (EVirtualKeyBoard).
- Visor de Catálogo de productos (EProductCatalog).
- Carrito de Compras (EShoppingCart).
- Formulario de ingreso para datos personales (EPersonalData).
- Formulario de ingreso para información de pago (EPayment).
- Captcha (ECaptcha).
- Formulario de ingreso para información de envío de productos (EShippingAddress).

Cada uno de los controles descritos permitirán al desarrollador de la aplicación de tipo e-commerce programar su aplicación directamente en código o usando el modo de diseño, para poder configurar los controles de una manera más simple dado que cada uno de los controles incorpora o permite manejar sus propiedades a través del visor de propiedades de Visual Studio, la mayoría de los controles

incluyen un manejo completo de su presentación y visualización en tiempo de diseño, permitiendo al desarrollador de la aplicación visualizar previamente el estado del control sin necesidad de ejecutar la aplicación, ahorrando tiempo en el desarrollo del e-commerce, los controles también incluyen nombres de propiedades y métodos normalmente usados dentro del comercio electrónico por tanto para el desarrollador de la aplicación le será muy simple adaptarse a la programación de los mismos, dado que se usará el esquema de programación de controles estándar de Microsoft.

En relación al tema de performance, dado que los controles se basan tanto en la suite de Microsoft como en EXT.NET, se podrán construir aplicaciones para un alto número de usuarios concurrentes.

Con respecto al licenciamiento, los controles se basan en el esquema de licenciamiento de Ext.NET y Microsoft, por tanto los controles se pueden usar solamente con fines de demostración más no con fines comerciales.

CAPÍTULO 2. DESARROLLO DE LA SUITE DE CONTROLES

En el presente capítulo se adaptará la metodología de desarrollo de software Open UP es decir, se usarán solamente los artefactos que brinden valor agregado a la solución que se está construyendo.

2.1. FASE DE ANÁLISIS

2.1.1. VISIÓN

Permite establecer o definir el punto de vista de los usuarios finales con respecto a la solución técnica que se va a desarrollar. Esta definición se especifica en base a las necesidades y características claves que los usuarios esperan de la aplicación que se va a construir. Este documento posee los requerimientos importantes que debe tener el sistema.

A continuación se define el documento de Visión del presente proyecto.

2.1.1.1. Introducción

El presente documento tiene por objetivo mostrar a un alto nivel las necesidades o la motivación para crear la suite de controles y servirá para posteriormente definir y detallar los requerimientos para el presente proyecto.

2.1.1.2. Posicionamiento

2.1.1.2.1. Planteamiento del problema

El Problema	Incremento del tiempo desarrollo en aplicaciones de comercio electrónico por falta de una suite de controles de servidor con componentes comunes específicos para ese tipo de aplicaciones y que actualmente no existe en el mercado, porque este último se enfoca en proveer controles de uso general que no se enfocan en determinadas aplicaciones y que por ende el desarrollador debe realizar un trabajo adicional para especializar esos

	controles para el desarrollo de su aplicación cualquiera que esta sea, incluyendo en ellas las aplicaciones de comercio electrónico.
A quiénes afecta	A los desarrolladores de aplicaciones de tipo e-commerce.
Impacto del problema	Incremento en el tiempo de desarrollo de aplicaciones de tipo e-commerce y por tanto también se incrementa el costo para desarrollar ese tipo de aplicaciones.
Una solución exitosa sería	Proveer una suite de controles que permita disminuir el tiempo de desarrollo para aplicaciones de comercio electrónico.

2.1.1.2.2. Product Position Statement

Para	Desarrolladores de aplicaciones de tipo e-commerce.
Mercado	En el mercado no existe una suite de controles específica para aplicaciones de comercio electrónico.
La suite (eCommerceToolkit)	Es una herramienta RAD (Rapid Application Development).
Razón para realizar la compra	Permite disminuir el tiempo de desarrollo y por ende dinero al momento de crear una aplicación de comercio electrónico.

2.1.1.3. Descripciones de los Usuarios

2.1.1.3.1. Resumen del Usuario

Nombre	Descripción	Resposabilidades
Desarrollador	Se encarga de crear las aplicaciones de comercio electrónico	Asegurar que la aplicación a desarrollar cumpla con los estándares de calidad.

2.1.1.3.2. Ambiente de Usuario

Para construir una aplicación de comercio electrónico generalmente se requirieren de 2 a 3 desarrolladores, la duración de un proyecto de ese tipo es alrededor de 8 meses y gran parte del tiempo (alrededor del 40%) se emplea en crear controles de servidor usando tecnología ASP.NET

2.1.1.4. Descripción del Producto

2.1.1.4.1. Necesidades y Características

Necesidad	Características
Creación de una suite de controles de servidor ASP.NET para aplicaciones de tipo e-commerce.	La suite de controles tendrá los siguientes componentes: • Control de menú (EMenu). • Control para presentar logotipos (EHeader). • Teclado Virtual (EVirtualKeyBoard). • Visor de Catálogo de productos (EProductCatalog). • Carrito de Compras (EShoppingCart). • Formulario de ingreso para datos personales

	(EPersonalData). • Formulario de ingreso para información de pago (EPayment). • Captcha (ECaptcha). • Formulario de ingreso para información de envío de productos (EShippingAddress).

2.1.1.5. Otros Requerimientos del Producto

Requerimientos
Descripción del uso de cada uno de los controles de servidor desarrollados.
Creación manual de instalación.

2.1.2. DEFINICIÓN DE REQUERIMIENTOS

Esta actividad permite recolectar y definir los requerimientos de la aplicación antes de proceder a la implementación de la misma. Se debe tomar como base el documento de visión para proceder al desarrollo de este artefacto.

El presente artefacto debe reflejar las necesidades de todos los involucrados en el desarrollo de la aplicación. A continuación se muestra el documento de especificación de requerimientos.

Especificación de Requerimientos

1. Introducción

El presente documento tiene por objetivo mostrar de manera detallada los

requerimientos de la suite de controles de servidor.

2. Requerimientos Funcionales

La suite debe proveer los siguientes controles:

- Control de menú que permita mostrar el listado de categorías o productos que posee la tienda virtual.
- Control de cabecera que permita mostrar el logotipo de la empresa o de la tienda virtual.
- Control que muestra en detalle la información de cada producto que ofrece *la tienda virtual, esto es la* descripción, precio y fotografía para cada uno los productos.
- Control que permita mostrar los ítems que el cliente ha seleccionado para comprar antes de realizar el proceso de checkout[5].
- Teclado virtual para las transacciones relacionadas a tarjetas de crédito e ingreso de información sensible.
- Captcha para verificar la identidad de un usuario.
- Control de Formulario para el ingreso de datos personales, este control debe tener validaciones para cada uno de los campos que conforman el formulario.
- Control de Formulario para el ingreso de información de pagos, este control debe tener validaciones para cada uno de los campos que conforman el formulario.
- Control de formulario para el ingreso de datos de envío de un paquete.

3. Cualidades del Sistema

3.1 Usabilidad

Cada uno de los controles debe permitir el manejo de sus propiedades y eventos usando el visor de propiedades de Visual Studio para simplificar su uso.

3.2 Performance

[5] Checkout: Proceso previo al pago de un producto.

Bajo una alta demanda de usuarios concurrentes se usará un compresor de View State para no afectar los tiempos de respuesta de los controles de servidor.

3.3 Compatibilidad

Los controles de servidor deben funcionar perfectamente usando tanto Framework 3.5 como 4.0 de .NET.

4. Interfaces del Sistema

4.1 User Interfaces

Cada uno de los controles presentará una interfaz en tiempo de diseño, para manipular sus propiedades y eventos.

4.1.1 Look & Feel

Los controles deben adaptarse fácilmente a estilos o skins previamente creados por el desarrollador de la aplicación.

5. Limitaciones del Sistema

La suite de controles se desarrolla usando solamente la suite de controles EXT.NET y la suite estándar de Microsoft.

6. Condiciones del Sistema

6.1 Requerimientos de Licenciamiento

El esquema licenciamiento de la suite de controles es solamente con propósitos de demostración y no con fines comerciales.

6.2 Estándares

El modelo de programación para los controles debe basarse en el estándar de programación de ASP.NET, para el manejo de propiedades y eventos.

7. Documentación del Sistema

Cada uno de los controles debe tener un manual de usuario que describa como

usar cada control.

2.1.3. ESPECIFICACIÓN DE CASOS DE USO

A partir de los casos de uso se puede visualizar y explicar de manera detallada el funcionamiento del sistema y su interacción con los usuarios finales del aplicativo.

A continuación se describen los casos de uso para cada uno de los controles que conforman la suite.

2.1.3.1. Caso de uso Configurar Control EHeader

Caso de Uso: Configurar Control de Cabecera (EHeader)

1. Breve Descripción

Este caso de uso describe cómo el desarrollador de una aplicación de comercio electrónico usa el control EHeader para desplegar el logotipo de la empresa para la cual está desarrollando la aplicación.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control EHeader a la página previamente creada.

2. El desarrollador configura el path, donde se encuentra la imagen con el logotipo.

3. El desarrollador configura el url, hacia donde se redirige la aplicación en cada clic sobre el logotipo de la empresa.

4. El desarrollador ejecuta o carga la página para mostrar el logotipo, y se produce un re direccionamiento en cada clic sobre el logotipo.

5. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

5.1 Path Incorrecto

Si en el paso 2 del flujo básico, el path es configurado de manera incorrecta, entonces:

1. El caso de uso termina con una condición de falla.

5.2 Url Incorrecto

Si en el paso 3 del flujo básico, el desarrollador configura de manera incorrecta el url para realizar el re direccionamiento, entonces:

1. El caso de uso termina con una condición de falla.

6. Post-Condiciones

6.1 Finalización con Éxito

El control despliega el logotipo de la empresa en la página.

En cada clic sobre el logotipo este redirecciona la aplicación al url configurado previamente.

6.2 Condición de Falla

No se muestra el logotipo de la empresa y por ende no se produce el re-direccionamiento al url configurado previamente

2.1.3.2. Caso de Uso Configuración control de Menú (EMenu)

Caso de Uso: Configurar Control de Menú (EMenu)

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de EMenu para desplegar las diferentes categorías de productos que existen en la tienda virtual.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control de Menú a la página.

El desarrollador usando el visor de propiedades de Visual Studio y para cada categoría:

2. Ingresa el nombre de la categoría.

3. Configura el path de una imagen que identifica cada categoría.

4. Crea sub-items para cada categoría.

5. El desarrollador para cada sub-item y usando el visor de propiedades de Visual Studio

6. Ingresa el nombre de cada sub-item.

7. Enlaza un url a cada sub-item.

8. El desarrollador ejecuta o carga la página para mostrar el menú con las

categorías y sus respectivos sub-items.

9. El caso de uso termina satisfactoriamente.

10. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

5.1 Path Incorrecto

Si en el paso 3 del flujo básico, el desarrollador configura el path de manera incorrecta, entonces:

1. No se muestra la imagen que identifica a la categoría.

2. El caso de uso se reanuda en el paso 7.

5.2 Url Incorrecto

Si en el paso 6 del flujo básico, el desarrollador ingresa de manera incorrecta el url para un sub-item, entonces:

1. El caso de uso termina con una condición de falla.

6. Post-Condiciones

6.1 Finalización con Éxito

Se despliega el menú con sus respectivas categorías y sus respectivos sub-items.

6.2 Condición de Falla

No se muestra el url enlazado a un sub-item de una categoría.

2.1.3.3. Caso de Uso Control para Catálogo de Productos (EProductCatalog)

Caso de Uso: Configurar Control EProductCatalog

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de EProductCatalog para desplegar varios productos referentes a una categoría.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso comienza cuando el desarrollador de la aplicación arrastra el control para mostrar los productos de una categoría en la página creada previamente.

El desarrollador para cada producto:

2. Ingresa el nombre del producto.

3. Una descripción de cada producto.

4. Una descripción corta de cada producto.

5. El precio del producto

6. Configura el Path donde se encuentra la imagen del producto.

7. El desarrollador carga la página para mostrar los productos.

8. El caso de uso termina exitosamente.

5. Flujos Alternos

5.1 Path Incorrecto

Si en el paso 6 del flujo básico, el path de la imagen del producto a mostrar es configurado de manera incorrecta, entonces:

1. No se muestra la imagen del producto.

2. El caso de uso se reanuda en el paso 6.

6. Post-Condiciones

6.1 Finalización con Éxito

Se muestra el catálogo de productos para una determinada categoría mostrando el nombre, descripción, costo e imagen del producto.

2.1.3.4. Caso de uso configurar el control para desplegar productos adquiridos por parte de un cliente (EShoppingCart)

Caso de Uso: Configurar Control EShoppingCart

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de EShoppingCart para desplegar los productos seleccionados para comprar por parte de un cliente.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control EShoopingCart a la página previamente creada.

2. El desarrollador crea dinámicamente un objeto ShoppingCart con las propiedades Date, Name, Price, Quantity y lo enlaza al control EShoopingCart para mostrar el producto agregado.

3. El desarrollador ejecuta la aplicación y carga la página donde se encuentra el control EShoppingCart, para mostrar los productos agregados al carrito de compras.

4. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

No Aplica

6. Post-Condiciones

No Aplica

6.1 Finalización con Éxito

Se muestran los productos agregados al carrito de compras.

6.2 Condición de Falla

No se muestran los productos agregados al carrito de compras.

2.1.3.5. Caso de Uso Teclado Virtual (EVirtualKeyBoard)

Caso de Uso: Configurar Control EVirtualKeyboard

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de Teclado Virtual para ingresar datos sensibles de la aplicación.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control EVirtualKeyboard a la página previamente creada.

2. El desarrollador configura el control para mostrarlo visible la primera vez que se carga la página o cuando se da clic sobre el ícono en forma de teclado ubicado en la parte derecha del TextBox.

3. El desarrollador ejecuta o carga la página para mostrar el control del Teclado Virtual.

4. El desarrollador ingresa la información en la caja de texto del control a través del teclado virtual.

5. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

No Aplica

6. Post-Condiciones

No Aplica

6.1 Finalización con Éxito

El control de Teclado Virtual se muestra en la página.

6.2 Condición de Falla

El control no se muestra en la página.

2.1.3.6. Caso de Uso Configurar Control de Captcha (ECaptcha)

Caso de Uso: Configurar Control EVirtualKeyboard

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de Captcha para validar la autenticidad de un cliente.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control ECaptcha a la página previamente creada.

2. El desarrollador configura el control para mostrarlo visible la primera vez que se carga la página o bajo ciertas circunstancias que define el programador de la aplicación.

3. El desarrollador ejecuta o carga la página para mostrar el control de Captcha.

4. El desarrollador ingresa en la caja de texto del control las letras que se muestran en la imagen que también presenta en el Captcha.

5. El texto ingresado es validado por el control para comparar con las letras que muestra la imagen del Captcha. El control responde con un valor booleano de verdadero si la comparación es exitosa.

6. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

5.1 Texto Ingresado es Incorrecto

Si en el paso 4 del flujo básico la comparación del texto ingresado con el texto mostrado en el Captcha no coincide, entonces:

1. El control de Captcha devolverá un valor booleano de falso.

2. El caso de uso se reanuda en el paso 4.

6. Post-Condiciones

No Aplica

6.1 Finalización con Éxito

El control devuelve un valor booleano de verdadero una vez realizada la comparación entre el texto ingresado y el que se muestra en el control de Captcha.

6.2 Condición de Falla

El control devuelve un valor booleano de falso una vez realizada la comparación entre el texto ingresado y el texto que muestra el control de Captcha.

2.1.3.7. Caso de Uso Formulario Ingreso de Datos Personales (EPersonalData)

Caso de Uso: Configurar Control EPersonalData

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de EPersonalData para el ingreso de los datos personales del cliente.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control EPersonalData a la página previamente creada.

2. El desarrollador ejecuta o carga la página para mostrar el control de Datos Personales.

3. El cliente ingresa en la primera caja de texto del control el nombre de usuario.

4. El cliente ingresa en la segunda caja de texto del control la dirección de correo electrónico.

5. El cliente ingresa en la tercera caja de texto del control nuevamente la confirmación de correo electrónico.

6. El cliente ingresa en la cuarta caja de texto del control de manera opcional el número de teléfono fijo o móvil.

7. El cliente ingresa en la quinta caja de texto del control la contraseña.

8. El cliente ingresa en la última caja de texto la confirmación de la contraseña.

9. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

5.1 Correo Electrónico Inválido

Si en el paso 4 del flujo básico, el cliente ingresa una dirección de correo electrónico inválida, entonces:

1. Se desplegará un aviso indicando que la dirección de correo electrónico es inválida.

2. El caso de uso se reanuda en el paso 4.

5.2 Longitud de contraseña incorrecta

Si en el paso 7 del flujo básico, el cliente ingresa para la contraseña un determinado número de caracteres que no se encuentran dentro del rango

establecido (Ver requerimientos especiales WC-1 para longitud de contraseña), entonces:

1. El control desplegará un aviso indicando que no se cumple con la longitud de la contraseña.

2. El caso de uso se reanuda en el paso 7.

6. Post-Condiciones

No Aplica

6.1 Finalización con Éxito

La información ingresada por el usuario es correcta y no existe ningún aviso de error.

6.2 Condición de Falla

El control despliega avisos de error, indicando que existen campos que no se han completado correctamente.

7. Requerimientos Especiales

[SpReq:WC-1] La longitud mínima de la contraseña son 8 caracteres y la longitud máxima son 16 caracteres.

2.1.3.8. Caso de Uso Configurar Formulario para el Ingreso de Información de Pagos (EPayment)

Caso de Uso: Configurar Control EPayment

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de EPayment para el ingreso de una tarjeta de crédito.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control EPayment a la página previamente creada.

2. El desarrollador ejecuta o carga la página para mostrar el control de información de pago.

3. El desarrollador ingresa el número de su tarjeta de crédito.

4. El desarrollador ingresa el nombre de la tarjeta de crédito.

5. El desarrollador selecciona una fecha de vencimiento de la tarjeta.

6. El desarrollador suscribe un método al evento clic para capturar la información de la tarjeta de crédito ingresada, para su posterior validación.

7. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

5.1 Campo número tarjeta de crédito vacío

Si en el paso 3 del flujo básico, el desarrollador no ingresa el número de su tarjeta de crédito, entonces:

1. Se desplegará un aviso indicando que no ha ingresado el número de la tarjeta de crédito

2. El caso de uso se reanuda en el paso 1.

5.2 Campo nombre de la tarjeta de crédito vacío

Si en el paso 4 del flujo básico, el desarrollador no ingresa el nombre de la tarjeta de crédito, entonces:

1. Se desplegará un aviso indicando que no ha ingresado el nombre de la tarjeta de crédito.

2. El caso de uso se reanuda en el paso 4.

5.3 Fecha de vencimiento inválida

Si en el paso 5 del flujo básico, el desarrollador ingresa una fecha de vencimiento incorrecta, entonces:

1. Se desplegará un aviso indicando que la fecha de vencimiento es inválida.

2. El caso de uso se reanuda en el paso 5.

6. Post-Condiciones

No Aplica

6.1 Finalización con Éxito

La información ingresada por el usuario está completa y no existe ningún aviso de error.

6.2 Condición de Falla

El control despliega avisos de error, indicando que existen campos que no se han completado correctamente.

2.1.3.9. Caso de Uso Configurar Formulario para ingreso de información de envío de Productos (EShippingAddress)

Caso de Uso: Configurar Control EShippingAddress

1. Breve Descripción

Este caso de uso describe como el desarrollador de una aplicación de comercio electrónico usa el control de EShippingAddress para el ingreso de una dirección de envío de un cliente.

2. Descripción Breve del Actor

2.1 Desarrollador de la aplicación de comercio electrónico.

3. Precondiciones

- Crear una página con la extensión “aspx”.

- La librería de controles se encuentra en el Toolbox de Visual Studio.

4. Flujo Básico

1. El caso de uso empieza cuando el desarrollador de la aplicación arrastra el control EShippingAddress a la página previamente creada.

2. El desarrollador ejecuta o carga la página para mostrar el control de información de envío.

3. El cliente ingresa su nombre completo.

4. El cliente ingresa la dirección principal de envío.

5. El cliente ingresa una dirección alternativa.

6. El cliente ingresa la ciudad.

7. El cliente ingresa el estado, provincia o región.

8. El cliente ingresa el código postal.

9. El cliente ingresa el país de residencia.

10. El cliente ingresa un número de teléfono.

11. El sistema captura de manera exitosa la información ingresada por el cliente.

12. El caso de uso termina satisfactoriamente.

5. Flujos Alternos

5.1 Nombre Completo vacío

Si en el paso 3 del flujo básico, el cliente no ingresa su nombre completo, entonces:

1. Se desplegará un aviso indicando que falta el nombre completo

2. El caso de uso se reanuda en el paso 3.

5.2 Dirección de envío vacía

Si en el paso 4 del flujo básico, el cliente no ingresa una dirección principal de envío, entonces:

1. Se desplegará un aviso indicando que falta la dirección principal.

2. El caso de uso se reanuda en el paso 2.

5.3 Ciudad de envío vacía

Si en el paso 6 del flujo básico, el cliente no ingresa una ciudad para el envío, entonces:

1. Se desplegará un aviso indicando que falta ingresar la ciudad hacia donde se dirige el envío

2. El caso de uso se reanuda en el paso 6.

5.4 Estado, provincia, región vacía

Si en el paso 7 del flujo básico, el cliente no ingresa un Estado, provincia,

para el envío, entonces:

1. Se desplegará un aviso indicando que falta ingresar un Estado, provincia, hacia donde se dirige el envío

2. El caso de uso se reanuda en el paso 7.

5.5 Código Postal Vacío

1. Si en el paso 8 del flujo básico, el cliente no ingresa un código postal para el envío, entonces:

2. Se desplegará un aviso indicando que falta ingresar un código postal, hacia donde se dirige el envío

El caso de uso se reanuda en el paso 8

6. Post-Condiciones

No Aplica

6.1 Finalización con Éxito

La información ingresada por el usuario es correcta y no existe ningún aviso de error.

6.2 Condición de Falla

El control despliega avisos de error, indicando que existen campos que no se han completado correctamente.

2.1.4. DEFINICIÓN DE ARQUITECTURA

En esta tarea se debe establecer una especificación técnica del sistema, definiendo las restricciones definidas para el proyecto, tomando como parte importante los requerimientos del usuario final de la aplicación.

Durante esta actividad se crea un borrador inicial con respecto a las decisiones técnicas que se tomarán antes del desarrollo del producto, obteniendo un acercamiento técnico de alto nivel.

2.1.4.1. Cuaderno de Arquitectura

Documento de Arquitectura

1. Propósito

El presente documento tiene como propósito, describir la arquitectura aplicada al proyecto "**Desarrollo de una suite de controles ASP.NET para aplicaciones de tipo e-commerce**¨ utilizando ASP.NET como tecnología para su desarrollo. La intención de este documento es capturar y transferir las decisiones de arquitectura más significativas y brindar una vista integral del proyecto conjuntamente con los principales mecanismos arquitectónicos a ser utilizados.

Adicionalmente, se describen un conjunto de buenas prácticas, las cuales se recomienda que sean usadas cuando se implanten soluciones ASP.Net, así como la descripción de los patrones involucrados en la arquitectura planteada.

2. Definiciones

Arquitectura de Software.- "La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución".

Calidad.- La totalidad de características de un producto o servicio soportadas en su habilidad de satisfacer las necesidades indicadas o involucradas.

Componente.- Una parte del sistema no trivial, casi independiente y reemplazable, que completa por sí sola una función clara en el contexto de una

arquitectura bien definida. Conforma y proporciona una realización de un conjunto de interfaces. Una parte de un sistema modular, desplegable y reemplazable que encapsula la implementación y expone un conjunto de interfaces. Un componente es típicamente especificado por uno o más clasificadores (por ejemplo:clases de implementación) que residen en él, y pueden ser implementados por uno o más artefactos (por ejemplo: binarios, ejecutables o archivos de script).

Mecanismo.- Un mecanismo es una instancia de un patrón. Puede requerir algún refinamiento adicional para convertirse en una colaboración en un modelo particular.

Un mecanismo en este contexto es una solución específica (a un problema recurrente) en un contexto simple.

Un mecanismo puede ser empleado para encajar o conformar un patrón. Cualquier colaboración puede ser denominada mecanismo, pero el término es usualmente reservado para colaboraciones que brindan una solución a un problema con frecuencia recurrente en aplicaciones de software, por ejemplo, para manejar persistencia, en donde un patrón es aplicable. En el análisis y diseño, la noción de mecanismo puede ser utilizado como un placeholder teniendo identificado que la persistencia es requerida, por ejemplo, el analista y el diseñador establecen que un mecanismo de persistencia será utilizado, lo cual será direccionado sistemática y consistentemente.

Mecanismo de Arquitectura.- Representa una solución común y concreta a problemas frecuentemente encontrados. Pueden ser patrones de estructura, patrones de comportamiento o ambos. En el RUP (Rational Unified Process), los mecanismos de arquitectura son usados para agrupar los mecanismos de análisis, diseño e implementación.

Namespace.- Un mecanismo de propósito general para organización de elementos dentro de grupos. También se define como una colección de nombres, identificados por un URI (Uniform Resource Identifier), que se utiliza en los documentos XML (Extensible Markup Language) para identificar los nombres de los elementos y atributos.

Patrón de Diseño (Model View Presenter).- El patrón de MVP es un patrón de arquitectura utilizada para construir aplicaciones ASP.net. Se refiere a la división de las responsabilidades para reunir, mostrar y almacenar datos de una página Web en objetos diferentes: un modelo de objetos, un objeto de vista, y un objeto de presentador. La vista es a menudo una interfaz que luego es aplicada por una página Web ASP.net, el modelo es un objeto de negocio. El presentador negocia la transferencia de datos entre los otros dos objetos. El patrón MVP desciende del patrón MVC (Model View Controller).

Una ventaja de tener la vista de una interfaz que puede ser aplicado por muchos diferentes modelos de presentación que no sea una página Web o una Windows. El hecho de que el código en el nivel de presentador permita la reutilización de la lógica del código tecnologías de presentación totalmente diferentes.

MVP está permitiendo que la definición de una vista y que la interfaz puede ser la base inmediata para que los desarrolladores puedan empezar a programar, sin esperar los requisitos.

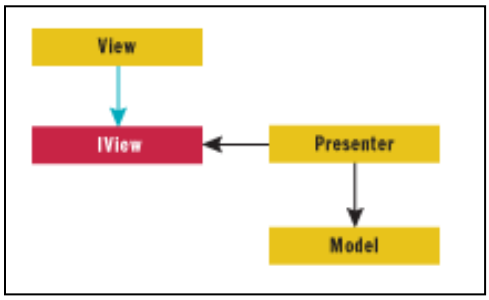

Figura 2-1 Modelo Vista Presentador (MVP)
Fuente: http://www.microsoft.com

IDE (Integrated Development Environment).- Es un conjunto de herramientas que facilitan la programación.

Sitio Web.- Constituye el Front-End de la aplicación, contiene todas las páginas Web, así como también la lógica propia de la interfaz de usuario.

3. Filosofía y Metas de la Arquitectura

La suite de controles para aplicaciones de comercio electrónico pretende ser una herramienta de desarrollo, basada en tecnología vigente que permita optimizar los tiempos de desarrollo e implantación de aplicaciones e-commerce, usando

estándares de programación, para asegurar el cumplimiento de los requerimientos del cliente.

3.1 Plataforma Tecnológica

La aplicación será desarrollada bajo la plataforma Microsoft Visual Studio 2010 y Microsoft .Net Framework 3.5 sobre sistema operativo Windows 7.

La plataforma de software para el desarrollo será:

Herramienta de Desarrollo	Uso
Microsoft Visual Studio 2010	Desarrollo, Modelamiento Visual, Desarrollo de Interfaces
Metodología OpenUp	Proceso de Desarrollo
Microsoft .net Framework 3.5	Desarrollo
EXT.NET	Desarrollo de Interfaces
AjaxControlToolkit	Desarrollo de Interfaces

Tabla 2-1 Plataforma Tecnológica para el desarrollo de la aplicación
Elaborado por: Marcelo Vallejo C.

3.2 Desempeño

Cada control de la suite debe soportar dentro de los estándares un enlace a un alto número de registros (1000 registros) sin realizar uso excesivo del ViewState de cada control y en caso de ser necesario el desarrollador de la aplicación deberá usar un compresor de ViewState.

4. Dependencias y Supuestos

Los usuarios de la suite de controles deben conocer del modelo de programación de controles de servidor de ASP.NET.

4.1 Tamaño y Rendimiento

Las características mínimas de hardware y software que se necesitan para poder implementar la arquitectura para la nueva suite de controles son:

PRODUCTO	HARDWARE	SOFTWARE

Suite de Controles	- Procesador 2 x 3 GHz - 4GB memoria RAM - 50 GB libres en disco duro - Tarjeta de red	- Windows 2003 Server R2 (SP 2) o Windows 7 - IIS 6.0 o IIS 7. - Microsoft .NET 3.5

Tabla 2-2 Hardware y Software para el desarrollo de la suite de controles
Elaborado: Marcelo Vallejo C.

5. Mecanismos de Arquitectura

5.1 Mecanismo para Manejo de Excepciones

Permite un manejo coherente de los errores que se puedan generar al usar los controles de la nueva suite.

5.2 Mecanismo para Manejo de Recursos

Permite un manejo de errores e interfaz de usuario en dos idiomas inglés y español, para una comprensión fácil del desarrollador de la aplicación y del usuario final de esa aplicación.

5.3 Mecanismo para el Logueo de Errores

Permite escribir un log, con los errores que aparecen donde la configuración y uso de los controles de la nueva suite.

6. Capas

6.1 Capa de Presentación

Para usar los controles de la nueva suitees recomendable usar una arquitectura utilizando el Patrón de Diseño MVP (Model - View - Presenter).

6.1.1 Capa Model – Modelo

Es la capa que representa las entidades de negocio que van a ser usadas en la interface gráfica.

6.1.2 Capa View – Vista

Es la capa encargada de manejar la presentación gráfica. Justamente en esta capa es donde actúan los nuevos controles de servidor.

6.1.3 Capa Presenter – Presentador

Es la capa en la que se define la lógica de presentación y la invocación al modelo de negocios, y debe ser implementada por cada página de la aplicación de comercio electrónico, es totalmente aislada de la tecnología de presentación (Windows, Web, etc.) e interactúa con el usuario a través de llamadas a la Capa View, utilizando para esto la interfaz definida.

7. Vistas de Arquitectura

7.1 Vista Lógica

7.1.1 Paquetes de Diseño Significativos para la Arquitectura

7.1.1.1 Paquetes

La estructura principal de los paquetes en el proyecto: “Desarrollo de una suite de controles ASP.NET para aplicaciones de tipo e-commerce¨, será:

7.1.1.2 ECommerceToolkit

En este proyecto estarán las clases base que serán necesarias para construir la funcionalidad de la nueva suite de controles. En este proyecto se pueden encontrar clases pertenecientes a configuración, interfaz de usuario, seguridades, manejo de Excepciones. Las clases de este proyecto estarán organizadas en paquetes dentro de *ECommerceToolkit.,* su contenido se muestra a continuación:

Paquete	Definición
ECommerceToolkit.UI	Ámbito en el cual se encuentran las clases que manejan lógica de presentación y

	clases base de las cuales heredan las páginas Web.
ECommerceToolkit.UI.Controls	Ámbito en el cual se encuentran las clases que contienen los controles web que se utilizarán en el proyecto.
ECommerceToolkit.Exceptions	Ámbito en el cual se encuentran las clases que permiten el manejo de excepciones de los controles de servidor
ECommerceToolkit.Commons	Ámbito en el que estarán las clases comunes para los proyectos .NET, que servirán tanto para la suite de controles como para otro tipo de aplicaciones.

Tabla 2-3 Definición de librerías para la suite de controles
Elaborado por: Marcelo Vallejo

7.2 Vista de Casos de Uso

La vista de casos de uso contempla la selección de un conjunto de escenarios y casos de uso que representen la funcionalidad central o significativa para el sistema y a la vez permitan determinar la arquitectura más adecuada para la misma.

A continuación se listan los casos de uso de la suite de controles:

- Configurar control EHeader.
- Configurar control EMenu.
- Configurar control EShoppingCart.
- Configurar control EPersonalData.
- Configurar control EPayment.
- Configurar control EProductCatalog.
- Configurar control EShippingAddress.
- Configurar control EVirtualKeyBoard.
- Configurar control ECaptcha.
-

8. Criterios de Calidad

El disponer de una infraestructura fundamental basada en .NET hace que el producto resultante sea fácil de manejar para el usuario final, puesto que ofrece una interfaz de manejo intuitivo.

La nueva suite de controles y los componentes de la interfaz que presente cada control, se encuentran desarrollados bajo los estándares de codificación definidos por Microsoft Corporation, lo que hace que el producto sea mantenible, y fácil de usar.

Durante el desarrollo de este producto se aplicará la metodología de desarrollo OpenUp, lo que garantiza, que los requerimientos definidos por el usuario final sean plasmados en el proceso de desarrollo, la correcta aplicación de la arquitectura y patrones definidos; dotando a la nueva suite de controles la característica de ser un producto robusto en su arquitectura y funcionalidad.

Mediante el mecanismo de manejo de excepciones, recursos; la nueva suite de controles asegura que los módulos que lo conforman, presenten al usuario final y técnico la información adecuada en caso de fallos.

2.2. FASE DE DISEÑO

La orientación de esta fase es una vez obtenidos los requerimientos a un buen nivel de detalle, proceder a implementar el producto de software de una manera más técnica a través del uso de diagramas de clase que muestren de manera independiente cada uno de los componentes que se van a desarrollar.

Para continuar con esta actividad es necesario tomar como base los requerimientos del aplicativo y el libro de arquitectura, para establecer el diseño funcional de cada uno de los controles de servidor.

A continuación se muestra el diseño funcional de cada uno de los controles que conforman la nueva suite que se va a desarrollar.

2.2.1. DISEÑO FUNCIONAL EHEADER

Diseño Funcional: Control de EHeader

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita mostrar el logotipo de la empresa que está comercializando los productos y que a su vez sea simple de usar y se adapte al esquema de programación de los controles de servidor de Microsoft ASP.NET.

2. Definición del Problema

Actualmente para mostrar el logotipo de una empresa en una aplicación de tipo e-commerce, es necesario realizar composición de varios controles, entre ellos un control de imagen y un hipervínculo, entre otros. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles y que el desarrollador de la aplicación simplemente lo use sin tener que realizar ningún trabajo adicional.

3. Necesidades

Crear un control de cabecera que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario.

Figura 2-2 Prototipo EHeader
Elaborado por: Marcelo Vallejo C.

5. Diagrama de Clases

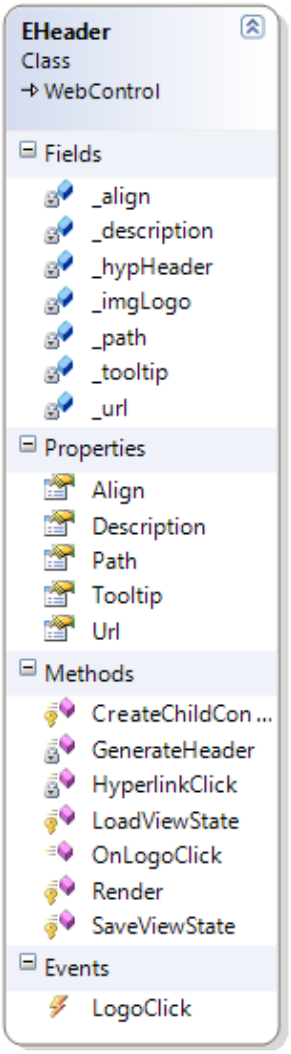

Figura 2-3 Diagrama de Clases EHeader
Elaborado por: Marcelo Vallejo

2.2.2. DISEÑO FUNCIONAL EMENU

Diseño Funcional: Control de EMenu

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita desplegar las categorías y sub-categorías para los productos que una empresa está comercializando.

2. Definición del Problema

Actualmente para mostrar un menú con categorías y sub-categorías en un e-commerce, es necesario realizar composición de varios controles, lo que implica pérdida de tiempo para el desarrollador e incluso se genera incertidumbre al no saber de antemano como quedará construido el control. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles y que el desarrollador de la aplicación simplemente lo use sin tener que realizar ningún trabajo adicional, a través del diseñador de Visual Studio.

3. Necesidades

Construir un control de menú que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario

Figura 2-4 Prototipo EMenu
Elaborado por: Marcelo Vallejo C.

5. Diagrama de Clases

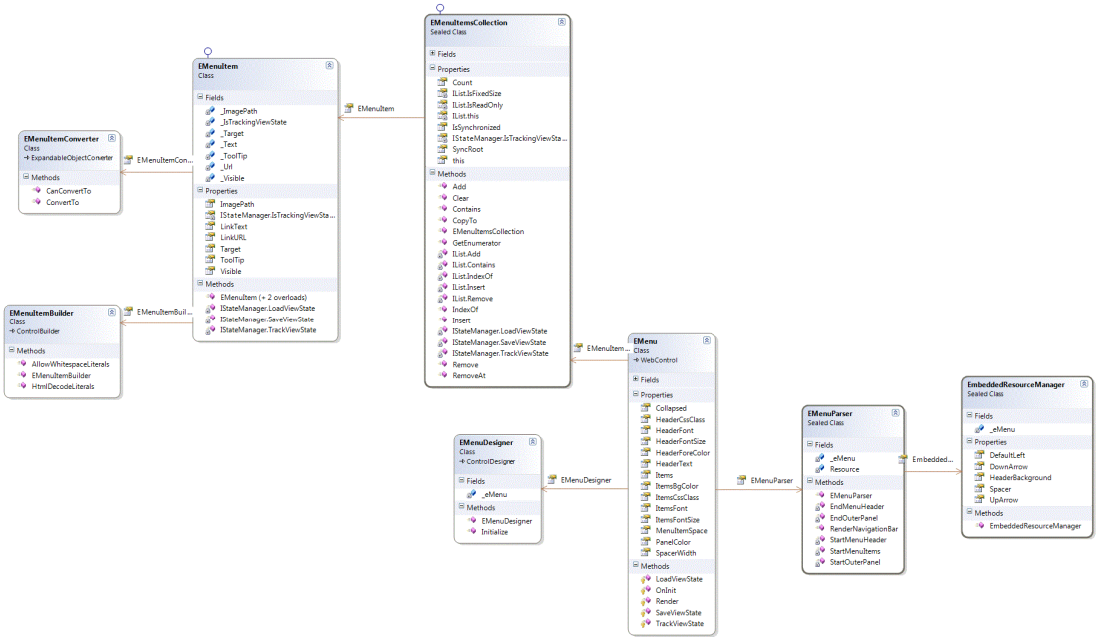

Figura 2-5 Diagrama de Clases EMenu
Elaborado por: Marcelo Vallejo

2.2.3. DISEÑO FUNCIONAL EPRODUCTCATALOG

Diseño Funcional: Control de EProductCatalog

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita mostrar los productos que la tienda virtual ofrece, esto incluye una descripción corta, el precio y una imagen por cada producto.

2. Definición del Problema

Actualmente para tener un control que permita mostrar todos los productos en un e-commerce, es necesario realizar composición de varios controles desde un control de imagen hasta un botón para agregar un producto al carrito de compras, lo que implica pérdida de tiempo para el desarrollador e incluso se genera incertidumbre al no saber de antemano como quedará construido el control.

Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles, que se mencionan.

3. Necesidades

Crear un control para mostrar los productos de la tienda virtual que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario

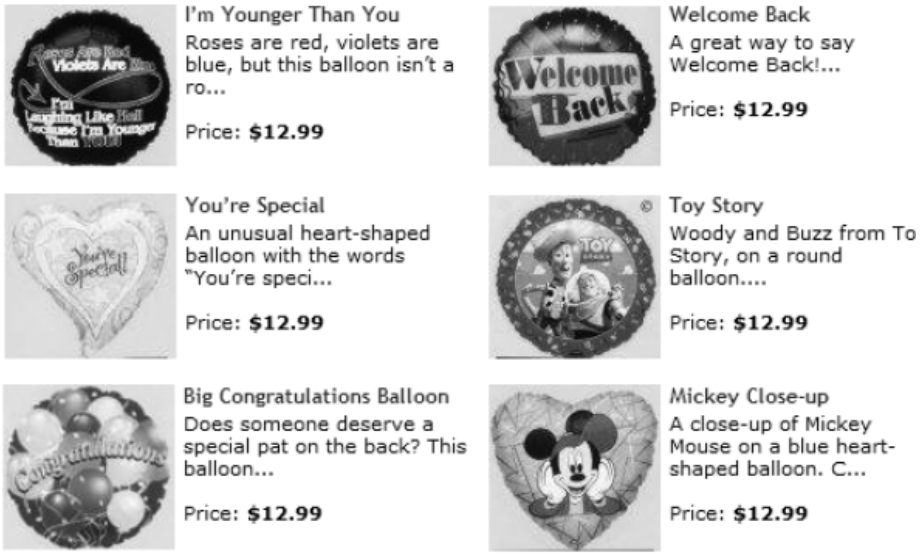

Figura 2-6 Prototipo EProductCatalog
Referencia: http://www.apress.com

5. Diagrama de Clases

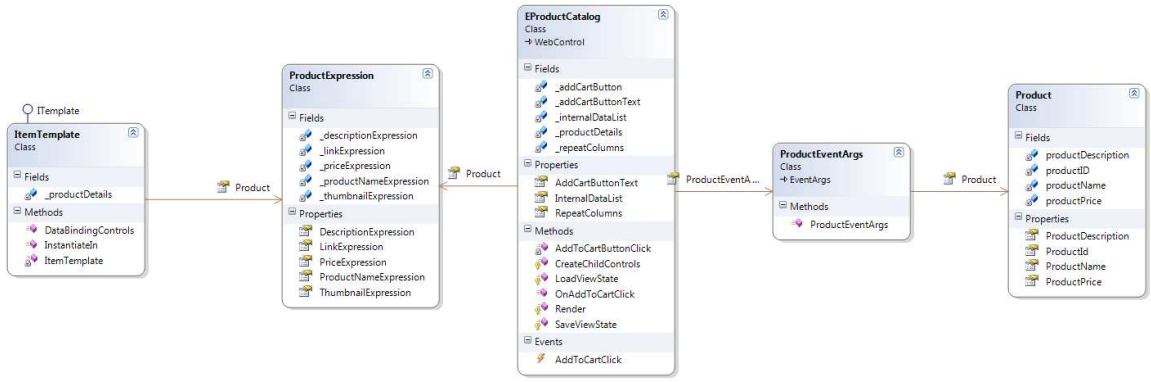

Figura 2-7 Diagrama de clases EProductCatalog
Elaborado por: Marcelo Vallejo C.

2.2.4. DISEÑO FUNCIONAL ESHOPPINGCART

Diseño Funcional: Control de EShoppingCart

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita mostrar los productos que el cliente adquirió durante la visita a la tienda virtual, esto incluye el nombre del producto, precio con y sin impuestos, además se debe permitir borrar el producto que se haya seleccionado.

2. Definición del Problema

Actualmente para mostrar un shoppingcart en un e-commerce, es necesario realizar composición de varios controles para mostrar la descripción detallada de un producto que se ha agregado al carrito de compras. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles.

3. Necesidades

Crear un ShoppingCart que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario.

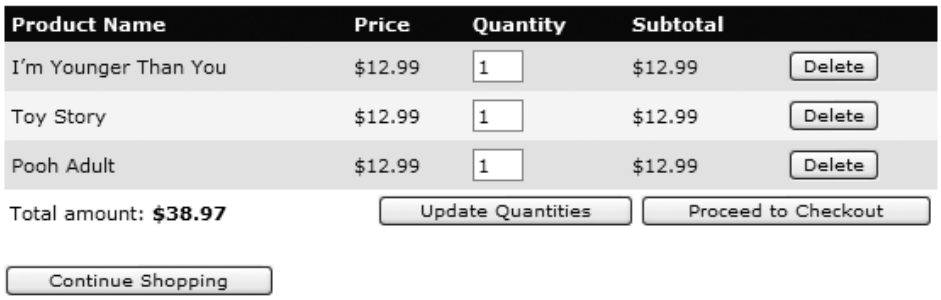

Figura 2-8 Prototipo EShoppingCart
Referencia: http://www.apress.com

5. Diagrama de Clases

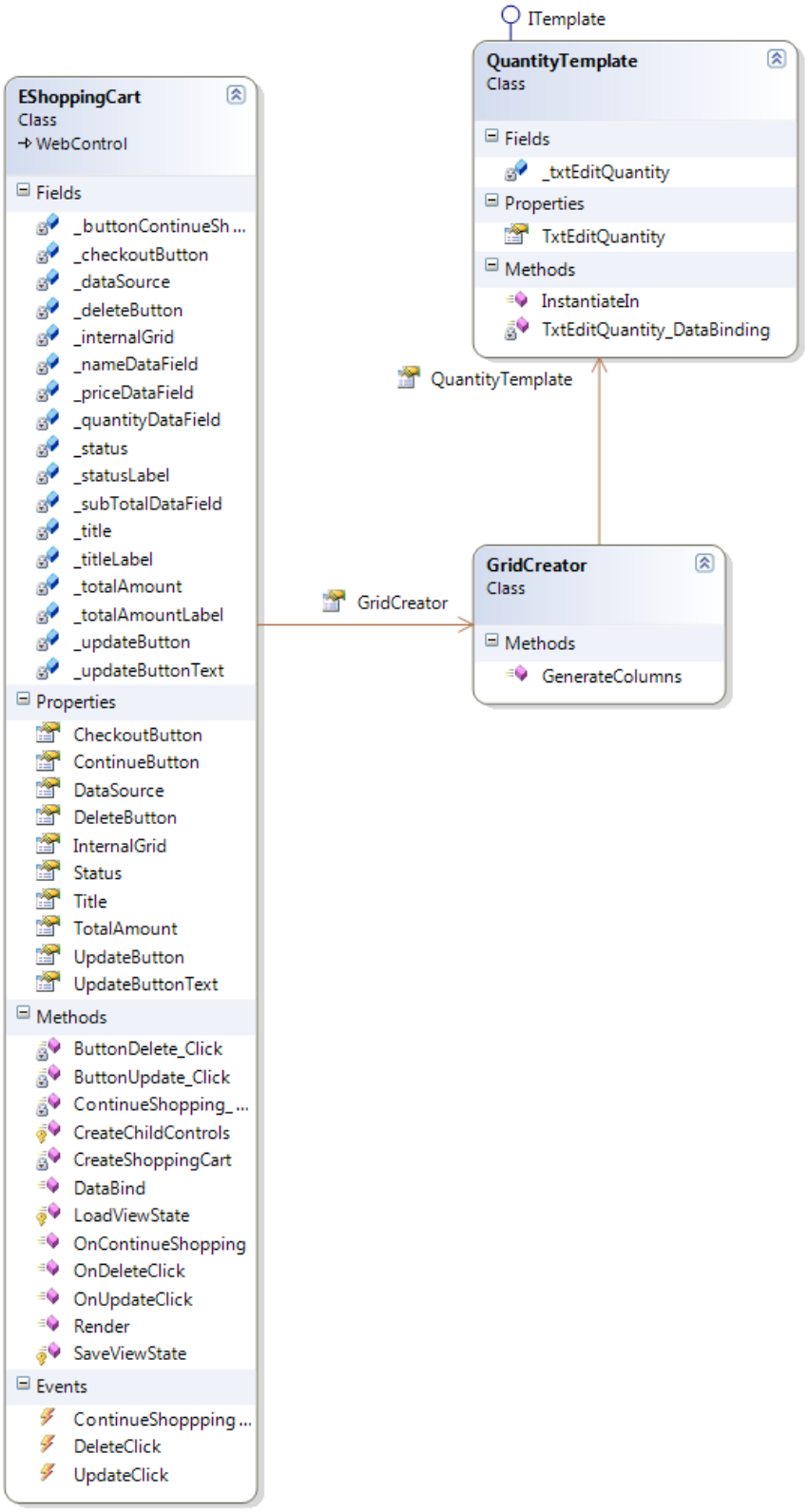

Figura 2-9 Diagrama de Clases EShoppingCart
Elaborado por: Marcelo Vallejo C.

2.2.5. DISEÑO FUNCIONAL EVIRTUALKEYBOARD

Diseño Funcional: Control de EVirtualKeyboard

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita mostrar un teclado virtual para el ingreso de datos sensibles como por ejemplo información sobre tarjetas de crédito.

2. Definición del Problema

Actualmente para tener un control que permita mostrar un teclado virtual en un e-commerce, es necesario realizar composición de varios controles para un teclado virtual es necesario realizar la composición de varios LinkButtons y manejar eventos por cada uno ellos, lo que implica pérdida de tiempo para el desarrollador e incluso se genera incertidumbre al no saber de antemano como quedará construido el control. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles.

3. Necesidades

Tener un control para mostrar un teclado virtual que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario.

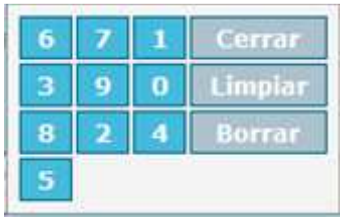

Figura 2-10 Prototipo EVirtualKeyboard
Elaborado por: Marcelo Vallejo C.

5. Diagrama de Clases

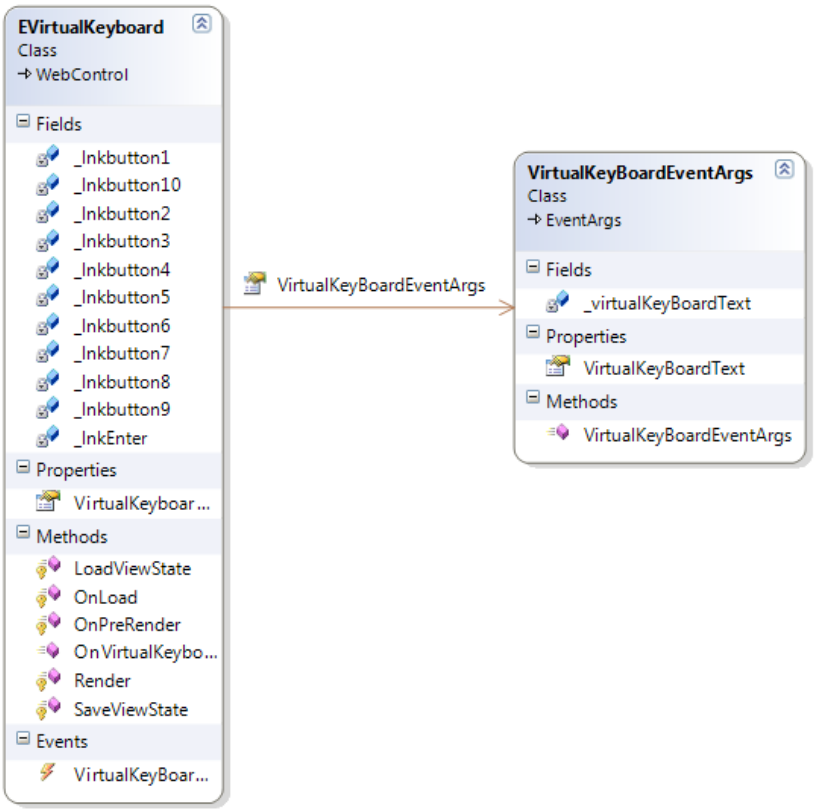

Figura 2-11 Diagrama de Clases EVirtualKeyboard
Elaborado por: Marcelo Vallejo C.

2.2.6. DISEÑO FUNCIONAL ECAPTCHA

Diseño Funcional: Control de ECaptcha

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita validar la identidad de un usuario y que no sea un robot el que está ingresando al aplicativo, esto se logra a través de un control que despliegue una figura en este caso un conjunto de letras las cuales deben ser identificadas por el usuario del aplicativo para validar su propia identidad.

2. Definición del Problema

Actualmente para mostrar un control de captcha en un e-commerce, es necesario realizar composición de varios controles e incluso usar librerías adicionales para

el manejo de imágenes, lo que implica pérdida de tiempo para el desarrollador e incluso se genera incertidumbre al no saber de antemano como quedará construido el control. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles y manejo de imágenes y que el desarrollador de la aplicación simplemente lo use sin tener que realizar ningún trabajo adicional.

3. Necesidades

Construir un control de captcha que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario

Figura 2-12 Prototipo ECaptcha
Realizado por: Marcelo Vallejo C.

5. Diagrama de Clases

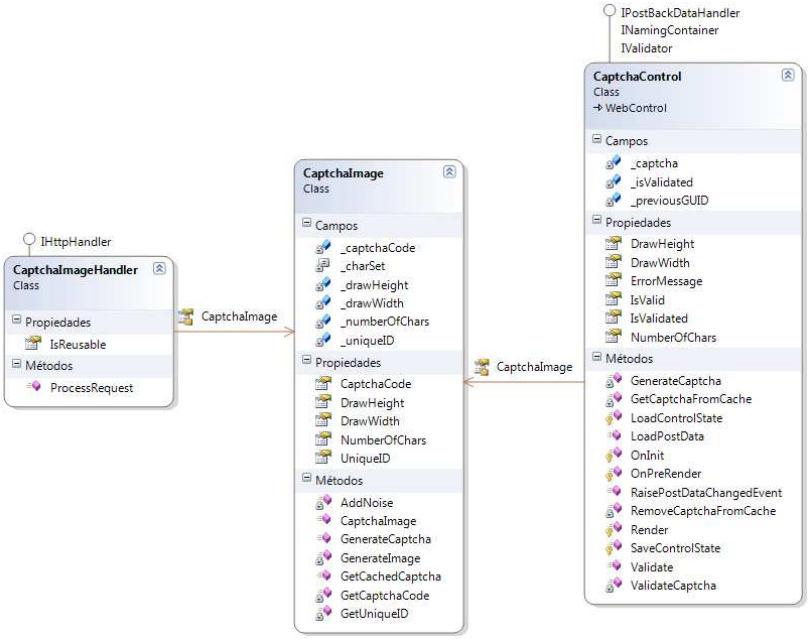

Figura 2-13 Diagrama de Clases ECaptcha
Realizado por: Marcelo Vallejo C.

2.2.7. DISEÑO FUNCIONAL EPERSONALDATA

Diseño Funcional: Control de EPersonalData

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que le permita al usuario de la aplicación ingresar sus datos personales, esto es: nombre de usuario, contraseña y correo electrónico.

2. Definición del Problema

Actualmente para mostrar un control para el ingreso de datos personales en un e-commerce, es necesario realizar composición de varios controles sobre todo TextBox con validaciones sobre cada uno de ellos, lo que implica pérdida de tiempo para el desarrollador e incluso se genera incertidumbre al no saber de antemano como quedará construido el control. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles.

3. Necesidades

Tener un control para el ingreso de datos personales que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

5. Prototipo Interfaz de usuario

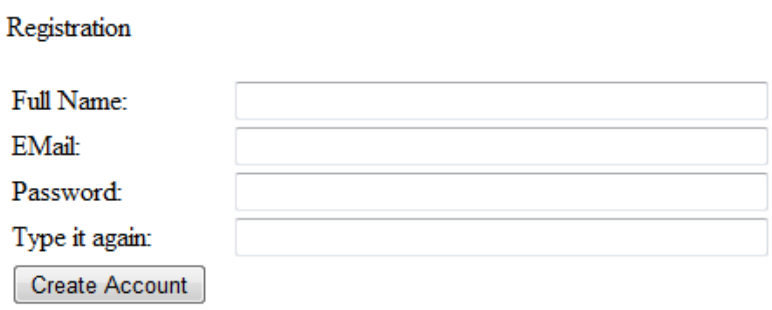

Figura 2-14 Prototipo EPersonalData
Elaborado por: Marcelo Vallejo C.

5. Diagrama de Clases

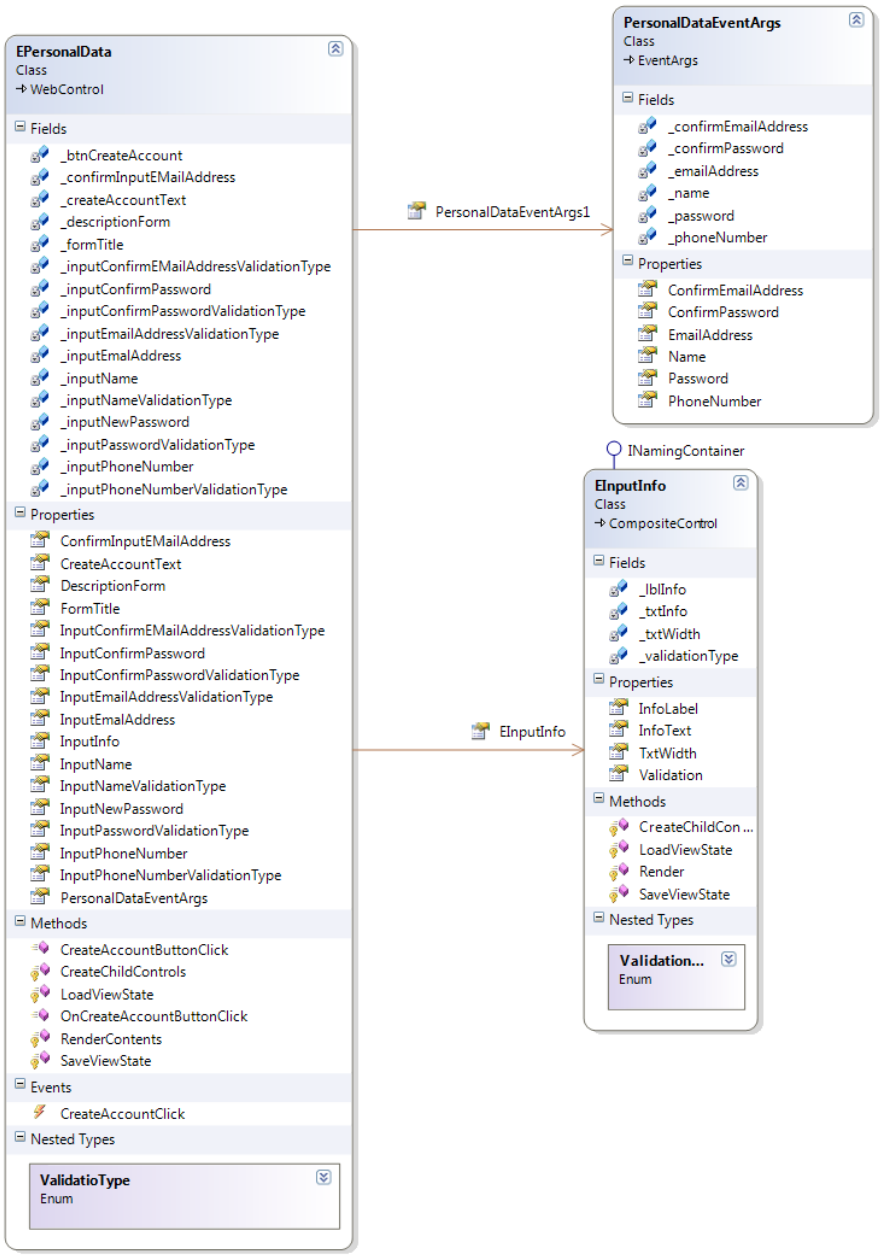

Figura 2-15 Diagrama de Clases EPersonalData
Elaborado por: Marcelo Vallejo C.

2.2.8. DISEÑO FUNCIONAL EPAYMENT

Diseño Funcional: Control de EPayment

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que permita ingresar información relacionada al pago de una orden de uno o varios productos esta información generalmente son datos de tarjetas de crédito.

2. Definición del Problema

Actualmente para tener un control que permita ingresar la forma de pago para una orden, es necesario realizar composición de varios controles los cuales deben tener reglas de validación para las tarjetas de crédito, y toda la información relacionada a pagos. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, sobre el cual se haya realizado previamente la composición de controles.

3. Necesidades

Construir un control para ingreso de tarjetas de crédito que sea simple de usar y que se base en el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario.

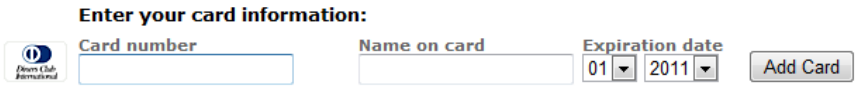

Figura 2-16 Prototipo EPayment
Elaborado por: Marcelo Vallejo C.

5. Diagrama de Clases

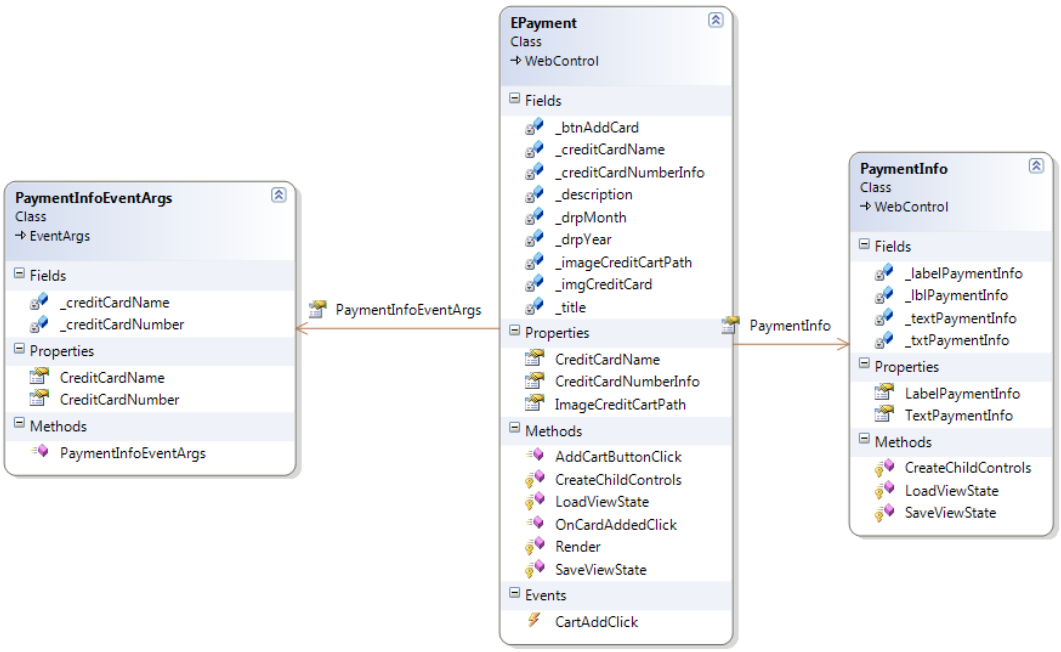

Figura 2-17 Diagrama de Clases EPayment
Elaborado por: Marcelo Vallejo C.

2.2.9. DISEÑO FUNCIONAL ESHIPPINGADDRESS

Diseño Funcional: Control de EShippingAddress

1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario usar un control que le permita al usuario de la aplicación ingresar los datos de envío para un determinado paquete.

2. Definición del Problema

Actualmente para mostrar un control para el ingreso de datos de envío en un e-commerce, es necesario realizar la composición de varios controles sobre todo los TextBox además de validaciones sobre cada uno de ellos, lo que implica pérdida de tiempo para el desarrollador e incluso se genera incertidumbre al no saber de antemano como quedará construido el control. Entonces es necesario crear un control de servidor a partir de la suite de controles de Microsoft, que incluya la composición de controles, para agilizar el proceso.

3. Necesidades

Construir un control para ingresar los datos de envío para un determinado paquete, el cual sea simple de usar y mantenga el estándar de programación de los controles de servidor de Microsoft.

4. Prototipo Interfaz de usuario.

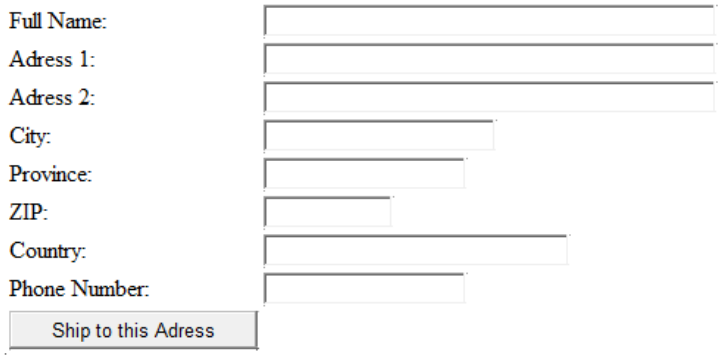

Figura 2-18 Prototipo EShippingAddress
Elaborado por: Marcelo Vallejo C.

5. Diagrama de Clases

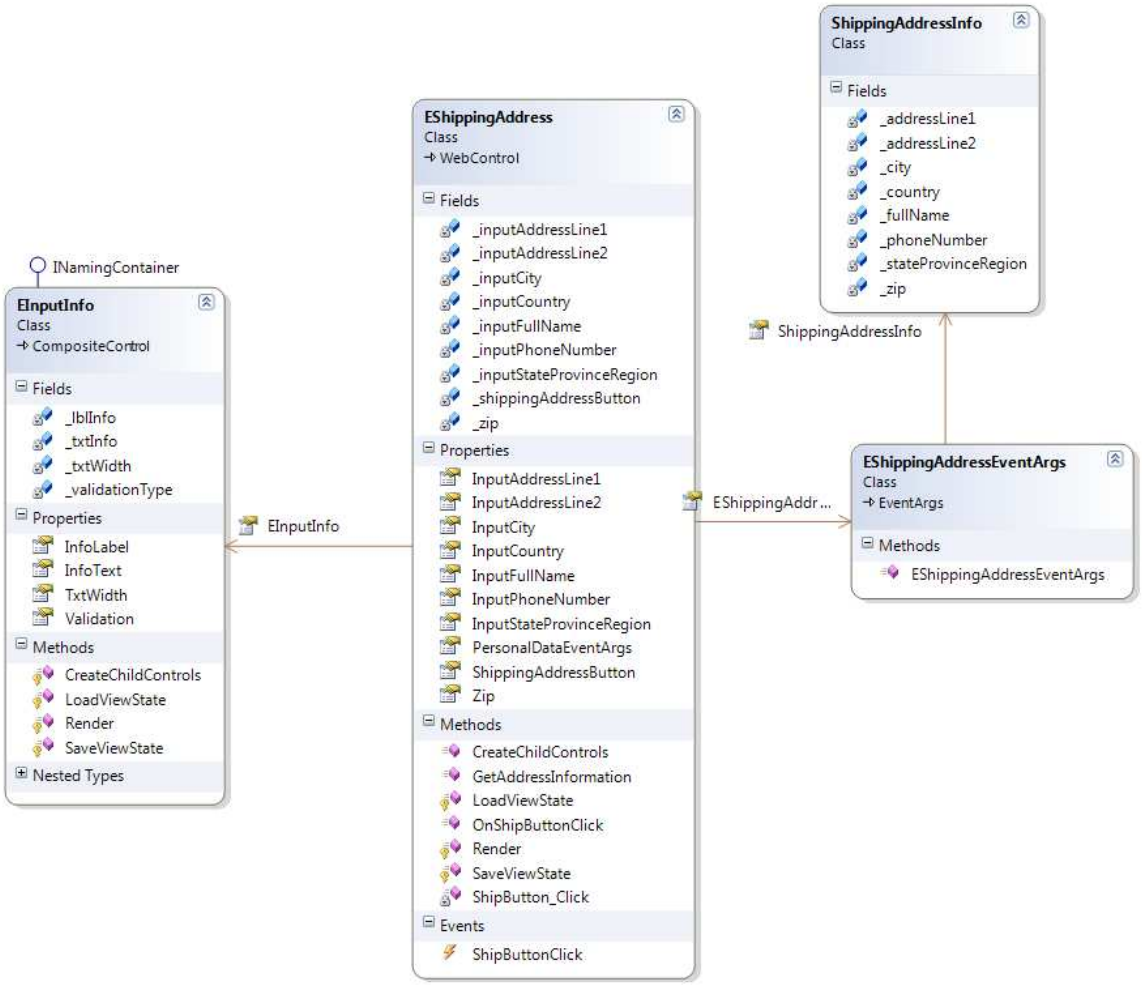

Figura 2-19 Diagrama de Clases EShippingAddress
Elaborado por: Marcelo Vallejo C.

2.3. FASE DE IMPLEMENTACIÓN

En la fase anterior una vez claros los requerimientos, la arquitectura de la solución y el diseño funcional de cada uno de los componentes el siguiente paso es proceder a la construcción de los mismos generando posteriormente una versión estable de la solución que pueda ser usada por los usuarios finales de la aplicación.

Con el diseño concluido el siguiente paso es escribir el código fuente de los componentes según se haya establecido en el diseño funcional.

Una vez culminada la etapa de codificación se integran los componentes en una sola librería de enlace dinámico la cual contiene los controles de servidor que formarán parte de la suite.

Antes de continuar con la codificación es necesario nombrar nuevamente los controles que se van a crear y son los siguientes:

- Control de menú (EMenu).
- Control para presentar logotipos (EHeader).
- Teclado Virtual (EVirtualKeyBoard).
- Visor de Catálogo de productos (EProductCatalog).
- Carrito de Compras (EShoppingCart).
- Formulario de ingreso para datos personales (EPersonalData).
- Formulario de ingreso para información de pago (EPayment).
- Captcha (ECaptcha).
- Formulario de ingreso para información de envío de productos (EShippingAddress).

A manera de ejemplo se va a mostrar cómo construir el control:

EShippingAddress

Para la creación de cada uno de los controles de servidor para el presente proyecto es necesario heredar de la clase WebControl o CompositeControl del Framework de .NET, también es posible heredar de un control existente para extender su funcionalidad, las dos clases mencionadas al inicio permiten manejar el estado de los controles de servidor que se van a construir.

Entre los métodos más importantes que proveen las clases WebControl y CompositeControl para el manejo del estado de los controles de servidor, están los siguientes:

- **CreateChildControls**
    - Es llamado por el Framework de ASP.NET para notificar al control de servidor que usa una implementación basada en la composición

de controles para crear cualquier de los controles hijo que éste contiene.

- **OnPreRender**
    - Provoca el evento PreRender. este evento generalmente se usa para realizar las actualizaciones antes de que el control de servidor se represente en la página. Durante este evento, pueden guardarse los cambios realizados en el estado de vista del control de servidor. Los cambios realizados en la fase de representación no se guardarán.
- **OnLoad**
    - Provoca el evento Load, este evento se produce cuando el control de servidor se carga en el objeto Page, o la página que visualiza el cliente en este método generalmente se establece la suscripción de los eventos de los controles internos de un control de servidor compuesto.
- **SaveViewState**
    - Guarda los cambios realizados en el estado de vista del control de servidor desde la devolución de la página al servidor.
- **LoadViewState**
    - Restaura la información sobre el estado de vista de una solicitud de página anterior guardada por el método SaveViewState.
- **Render**
    - Envía el contenido del control de servidor a un objeto HtmlTextWriter[6], que escribe el contenido que se va a representar en el cliente.

[6] Escribe caracteres de marcado y texto en un flujo de salida de un control de servidor ASP.NET. Esta clase proporciona funciones de aplicación de formato que los controles de servidor de ASP.NET utilizan al representar marcado en los clientes.

- **EnsureChildControls**
    - El método EnsureChildControls se usa normalmente en controles compuestos, que son controles que usan controles secundarios para parte de su funcionalidad o para toda su funcionalidad. Se llama al método EnsureChildControls para garantizar que se han creado los controles secundarios y que están preparados para procesar la entrada, realizar el enlace de datos o realizar otras tareas.

Ahora se va a revisar la implementación del control ETextBoxValidator y del control EShippingAddress para mostrar el uso de los métodos que se mencionaron al inicio.

Para el Control ETextValidator primero se debe heredar de la clase WebControl para tener acceso a los métodos que permiten la creación del control de servidor compuesto tal y como se muestra en el siguiente fragmento de código:

```
[DefaultProperty("Text")]
[ToolboxData("<{0}:EValidatorTextBox runat=server></{0}:EValidatorTextBox>")]
public class EValidatorTextBox : WebControl
{

}
```

Una vez creada la clase que hereda de WebControl, se deben declarar los controles internos que formaran parte de la composición para este caso el control interno será un TextBox proveniente de la suite estándar de Microsoft:

```
private TextBox _txtValidator;
```

A continuación se va a crear una instancia del control interno y ello se lo realiza en el método CreateChildControls(), y una vez creada la instancia del control a este último se lo agrega a la jerarquía de controles de la página donde se está usando el control `Controls.Add(_txtValidator)` con un identificador único `(ID = this.ClientID + "txtValidator")`:

```
protected override void CreateChildControls()
        {
            base.CreateChildControls();
            _txtValidator = new TextBox { ID = this.ClientID + "txtValidator" };
            Controls.Add(_txtValidator);
        }
```

El presente control tiene por objetivo validar los datos de entrada del usuario es decir:

- Si el campo es obligatorio.
- Si el dato de ingreso es de tipo correo electrónico.
- Si el dato de ingreso es de tipo numérico.
- Si el campo no necesita ningún tipo de validación.

Para ello se va a crear una enumeración que le permita al usuario del control seleccionar el tipo de validación:

```
public enum ValidationType
        {
            RequiredFieldValidation,
            NumericValidation,
            EmailValidation,
            None
        }
```

De acuerdo al tipo de validación seleccionado por el usuario se va a crear el control de validación y esto se lo va a realizar el método OnPreRender descrito anteriormente:

```
protected override void OnPreRender(EventArgs e)
        {
            switch (Validation)
            {
                case ValidationType.RequiredFieldValidation:
                    CreateRequiredFieldValidator();
                    break;

                case ValidationType.NumericValidation:
                    CreateRequiredFieldValidator();
                    CreateNumericFieldValidator();
                    break;

                case ValidationType.EmailValidation:
                    CreateRequiredFieldValidator();
                    CreateEmailFieldValidator();
                    break;
                default:
                    break;
            }
        }
```

Por último se deben exponer las propiedades que serán visibles al usuario del control, en este ejemplo las propiedades serán:

- *Text:* Permite acceder al texto ingresado en el TextBox por parte del usuario del control.
- *Validation:* Permite seleccionar el tipo de validación para el TextBox.

- *ValidationErrorMessage:* Permite ingresar el texto que se mostrará cuando el texto ingresado no cumple con el tipo de validación que se seleccionó en el instante que se creó el control.
- *Width:* Permite ingresar el ancho del TextBox.

A continuación se detalla la creación de las propiedades:

```
[Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]
        public ValidationType Validation
        {
            get
            {
                try
                {
                    var validationType = (ValidationType)ViewState["Validation"];
                    return ((validationType == null) ? ValidationType.None :
validationType);
                }
                catch (Exception)
                {

                    return ValidationType.None;
                }

            }

            set
            {
                ViewState["Validation"] = value;
                //_validation = value;
            }
        }

        [Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]
        public string Text
        {
            get
            {
                EnsureChildControls();
                return _txtValidator.Text;
            }

            set
            {
                EnsureChildControls();
                _txtValidator.Text = value;
            }
        }

        [Bindable(true)]
        [Category("Appearance")]
```

```
        [DefaultValue("")]
        [Localizable(true)]
        public string ValidationErrorMessage
        {
            get
            {
                var s = (String)ViewState["ValidationErrorMessage"];
                return (s ?? "*");
            }

            set
            {
                ViewState["ValidationErrorMessage"] = value;
            }
        }


        [Bindable(true)]
        [Category("Appearance")]
        [Localizable(true)]
        public Unit Witdh
        {
            get
            {
                EnsureChildControls();
                return _txtValidator.Width;
            }

            set
            {
                EnsureChildControls();
                _txtValidator.Width = value;
            }
        }
```

Sobre cada una de las propiedades se incluyen atributos(Bindable, Category, etc.) los cuales permiten determinar cómo se mostrarán las propiedades en el diseñador de Visual Studio, a continuación se mencionan los atributos más importantes:

BindableAttribute: especificado como true o false, indica a los diseñadores visuales si tiene sentido enlazar la propiedad a datos. Por ejemplo, en Visual Studio 2010, si una propiedad se marca con Bindable(true), ésta se mostrará en el cuadro de diálogo DataBindings. Si la propiedad no se marca con este atributo, el examinador de propiedades deduce que el valor es Bindable(false).

CategoryAttribute: especifica en qué categoría se debe situar la propiedad en el examinador de propiedades del diseñador visual. Por ejemplo,

Category("Appearance") indica al examinador de propiedades que muestre la propiedad en la categoría Apariencia cuando el desarrollador de páginas utilice la vista por categorías del examinador de propiedades. Puede especificar un argumento de cadena correspondiente a una categoría existente en el examinador de propiedades o crear una categoría propia.

DescriptionAttribute: especifica una breve descripción de la propiedad. En Visual Studio 2010, el examinador de propiedades muestra la descripción de la propiedad seleccionada en la parte inferior de la ventana Propiedades.

DefaultValueAttribute: especifica un valor predeterminado para la propiedad. Este valor debe ser idéntico al valor predeterminado que se devuelve desde el descriptor de acceso de la propiedad (captador). En Visual Studio 2010, **DefaultValueAttribute**: permite al desarrollador de páginas restablecer el valor de una propiedad a su valor predeterminado si se abre el menú contextual en la ventana Propiedades y se hace clic en el botón Restablecer.

LocalizableAttribute: especificado como true o false, indica a los diseñadores visuales si tiene sentido traducir la propiedad. Cuando una propiedad se marca con Localizable(true), el diseñador visual la incluye cuando se serializan los recursos traducibles. El diseñador conservará el valor de la propiedad en el archivo de recursos de referencia cultural neutra u otro origen de localización cuando el control se sondee para buscar propiedades localizables.

Los atributos en tiempo de diseño aplicados a un control y sus miembros no afectan al funcionamiento del control en tiempo de ejecución, pero mejoran la experiencia del desarrollador cuando el control se utiliza en un diseñador visual.

Se puede observar sobre cada una de los propiedades del control servidor el método EnsureChildControls (), el cual permite asegurar que los controles hijo del control padre se han creado para poder asignarlos a un determinado valor.

```
                EnsureChildControls();
                _txtValidator.Width = value;
```

Ahora se va mostrar la implementación de un control más complejo el cual incluye el uso del control descrito anteriormente, adicionalmente se incluye etiquetas para

identificar a cada ETextBoxValidator, los que se van a usar en el control EShippingAddress que se detalla a continuación:

En primer lugar se declaran las variables o los controles que van a formar parte del control EShippingAddress:

```
private EValidatorTextBox _inputFullName = new EValidatorTextBox();
private EValidatorTextBox _inputAddressLine1 = new EValidatorTextBox();
private EValidatorTextBox _inputAddressLine2 = new EValidatorTextBox();
private EValidatorTextBox _inputCity = new EValidatorTextBox();
private EValidatorTextBox _inputStateProvinceRegion = new EValidatorTextBox();
private EValidatorTextBox _zip = new EValidatorTextBox();
private EValidatorTextBox _inputCountry = new EValidatorTextBox();
private EValidatorTextBox _inputPhoneNumber = new EValidatorTextBox();

private Label _lblFullName;
private Label _lblAddressLine1;
private Label _lblAddressLine2;
private Label _lblCity;
private Label _lblStateProvinceRegion;
private Label _lblzip;
private Label _lblCountry;
private Label _lblPhoneNumber;
```

A continuación se establecen las propiedades visibles para el desarrollador que usa el control, éstas últimas permiten establecer por ejemplo la etiqueta para el TextBox, como se muestra en el siguiente fragmento de código:

```
[Bindable(true)]
[Category("Appearance")]
[DefaultValue("Full Name:")]
[Localizable(true)]
public string LblFullName
{
    get
     {
       EnsureChildControls();
       return _lblFullName.Text;
      }
     set
      {
        EnsureChildControls();
        _lblFullName.Text = value;
        }
}
```

Y en la siguiente figura:

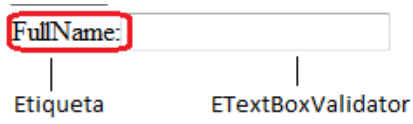

Figura 2-20 Asignación valor, propiedad FullName en control EShippingAddress

En el método CreateChildControls se crean las etiquetas y los ETextBoxValidator, todos ellos se incluyen en un control Table, como se muestra en el siguiente fragmento de código:

```
protected override void CreateChildControls()
        {

            _inputFullName.Text = InputFullName;
            _inputAddressLine1.Text = InputAddressLine1;
            _inputAddressLine2.Text = InputAddressLine2;
            _inputCity.Text = InputCity;
            _inputStateProvinceRegion.Text = InputStateProvinceRegion;
            _zip.Text = Zip;
            _inputCountry.Text = InputCountry;
            _inputPhoneNumber.Text = InputPhoneNumber;
            _lblFullName = new Label();
            _lblAddressLine1=new Label();
            _lblAddressLine2=new Label();
            _lblCity=new Label();
            _lblStateProvinceRegion=new Label();
            _lblzip=new Label();
            _lblCountry=new Label();
            _lblPhoneNumber=new Label();
            _shippingAddressButton = new Button();

            var contentTable = new Table();

            var trFullName = new TableRow();
            var trAddressLine1 = new TableRow();
            var trAddressLine2 = new TableRow();
            var trCity = new TableRow();
            var trStateProvinceRegion = new TableRow();
            var trZip = new TableRow();
            var trCountry = new TableRow();
            var trPhoneNumber = new TableRow();
            var trShippingAdressButton = new TableRow();



            var tcFullName = new TableCell();
            var tcAddressLine1 = new TableCell();
            var tcAddressLine2 = new TableCell();
            var tcCity = new TableCell();
            var tcStateProvinceRegion = new TableCell();
            var tcZip = new TableCell();
            var tcCountry = new TableCell();
            var tcPhoneNumber = new TableCell();
            var tcShippingAdressButton = new TableCell();


            AddControlToTableRow(tcFullName, trFullName, _lblFullName,
_inputFullName);
            AddControlToTableRow(tcAddressLine1, trAddressLine1, _lblAddressLine1,
_inputAddressLine1);

AddControlToTableRow(tcAddressLine2,trAddressLine2,_lblAddressLine2,_inputAddressLin
e2);
```

```
            AddControlToTableRow(tcCity, trCity, _lblCity, _inputCity);
            AddControlToTableRow(tcStateProvinceRegion, trStateProvinceRegion,
_lblStateProvinceRegion, _inputStateProvinceRegion);
            AddControlToTableRow(tcZip, trZip, _lblzip, _zip);
            AddControlToTableRow(tcCountry, trCountry, _lblCountry, _inputCountry);
            AddControlToTableRow(tcPhoneNumber, trPhoneNumber, _lblPhoneNumber,
_inputPhoneNumber);

            tcShippingAdressButton.Controls.Add(_shippingAddressButton);


            contentTable.Controls.Add(trFullName);
            contentTable.Controls.Add(trAddressLine1);
            contentTable.Controls.Add(trAddressLine2);
            contentTable.Controls.Add(trCity);
            contentTable.Controls.Add(trStateProvinceRegion);
            contentTable.Controls.Add(trZip);
            contentTable.Controls.Add(trCountry);
            contentTable.Controls.Add(trPhoneNumber);
            trShippingAdressButton.Controls.Add(tcShippingAdressButton);

            Controls.Add(contentTable);
        }
```

El siguiente paso es generar la librería de enlace dinámico que contiene en este caso el control EShippingAddress, una vez generada la librería se la agrega al Toolbox de Visual Studio, realizando el siguiente procedimiento:

- Clic derecho sobre el Toolbox de Visual Studio y se escoge la opción “Seleccionar ítems”, tal y como se muestra en la siguiente figura:

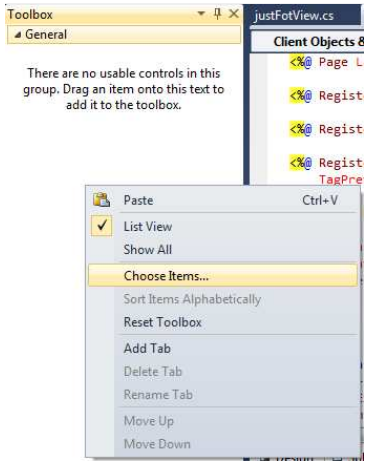

Figura 2-21 Agregar librería de controles al Toolbox de Visual Studio
Elaborado por: Marcelo Vallejo C.

- Se selecciona la respectiva librería de enlace dinámico.

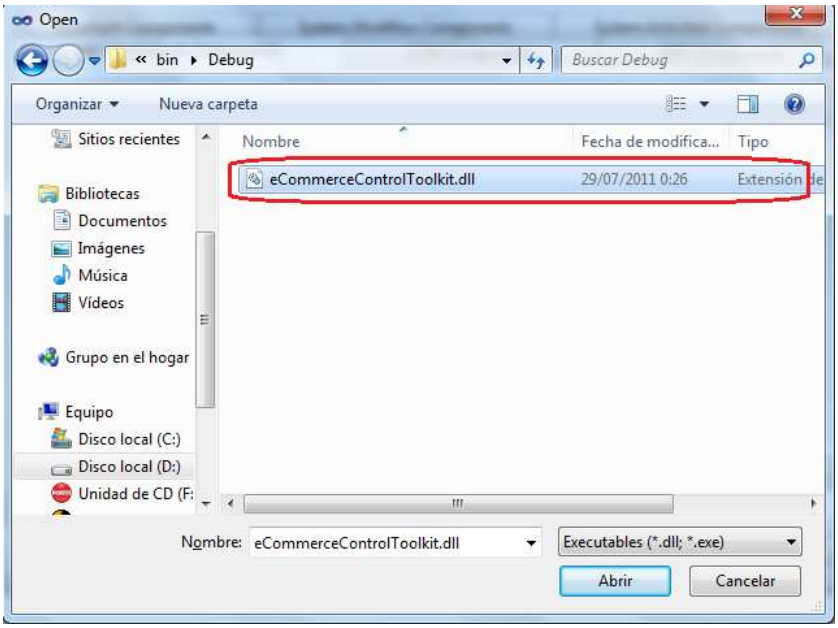

Figura 2-22 Seleccionar librería de los nuevos controles
Elaborado por: Marcelo Vallejo C.

Una vez agregada la librería, en el Toolbox de Visual Studio se pueden observar los controles que se han creado, en este caso se puede ver el control EShippingAddress.

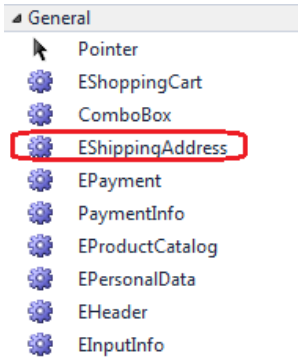

Figura 2-23 Suite de Controles agregada al Toolbox de Visual Studio
Elaborado por: Marcelo Vallejo C.

El siguiente paso es arrastrar el control EShippingAddress desde el Toolbox hacia la página donde se va a usar el control y configurarlo con sus respectivas propiedades.

```
<cc2:EShippingAddress ID="EShippingAddress1" runat="server" LblFullName="FullName:"
LblAddressLine1="Direccion 1:" LblAddressLine2="Direccion 2:" LblCity="Ciudad:"
LblCountry="Pais:" LblPhoneNumber="Numero Telefono:"
LblStateProvinceRegion="Estado:" Lblzip="Zip:" />
```

Por último se debe ejecutar la página donde se encuentra el control para observar la implementación del mismo.

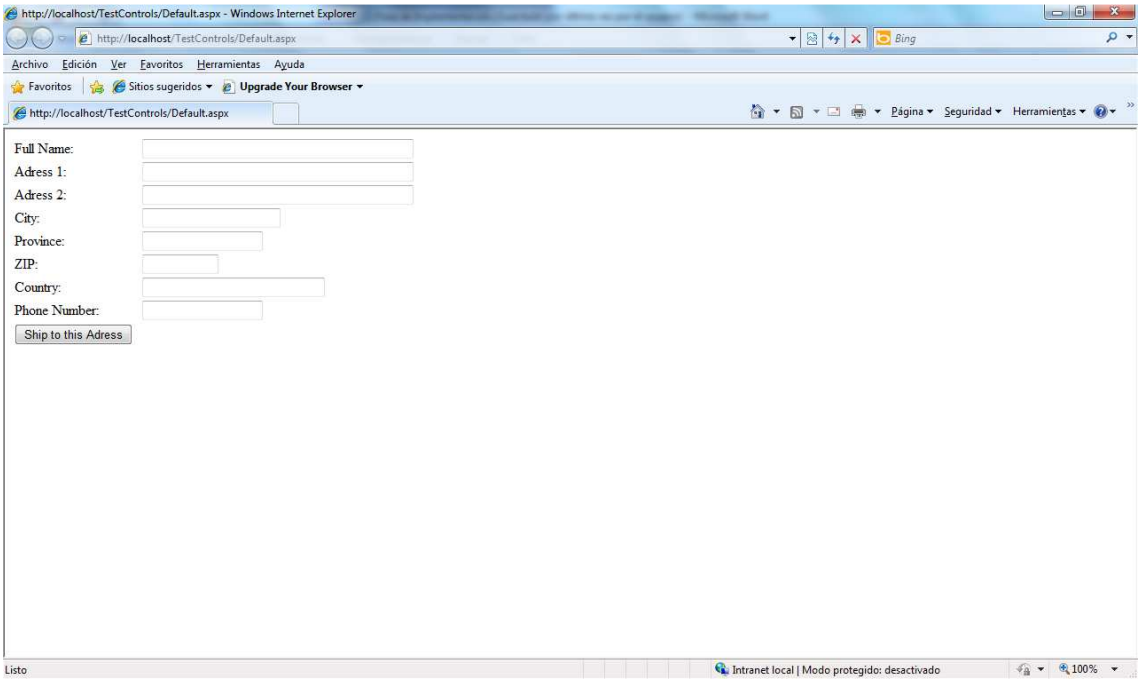

Figura 2-24 Control EShippingAddress implementado en una página de ejemplo
Elaborado por: Marcelo Vallejo C.

Para el resto de controles de la Suite la implementación es similar a la mostrada durante este ejemplo.

2.4. FASE DE PRUEBAS

La fase de pruebas será ejecutada en el siguiente capítulo puesto que para validar el funcionamiento de los controles es necesario usarlos o implementarlos en un caso de estudio, para posteriormente realizar un análisis de resultados.

CAPÍTULO 3. EVALUACIÓN DE LA SUITE DE CONTROLES APLICANDO A UN CASO DE ESTUDIO

3.1. DEFINICIÓN DE REQUISITOS DEL PROTOTIPO (E-COMMERCE)

Para validar la suite de controles es necesario implementarla en un sitio de comercio electrónico, para ello se ha definido la siguiente especificación de requerimientos:

Especificación de Requerimientos

1. Introducción

El presente documento tiene por objetivo mostrar de manera detallada los requerimientos necesarios para la construcción del e-commerce que servirá como caso de estudio en la aplicación de la suite de controles que se ha construido. El presente caso de estudio permitirá mostrar los controles de servidor en una tienda virtual de prendas de vestir.

2. Requerimientos Funcionales

El caso de estudio debe poseer las siguientes características o requerimientos:

- Despliegue de cada una de las categorías y subcategorías de los productos que ofrece la tienda virtual.
- Muestrario de los productos que ofrece la tienda virtual, esto incluye:
    - Fotografía del producto.
    - Breve descripción del producto.
    - Precio del Producto.
- Registro de información de los usuarios del e-commerce, esto incluye:
    - Información personal.
    - Direcciones de envío.
    - Información de pago para las transacciones.
    - Control de servidor específico (teclado virtual) para el ingreso de información sensible.
    - Mecanismo de validación de usuarios mediante un control Captcha.
    - Mostrar el detalle de la compra realizada por los usuarios.

- Mostrar el logotipo de la tienda virtual.

3. Cualidades del Sistema

3.1 Usabilidad

La navegación para los usuarios de la tienda virtual debe permitir mostrar en todo momento los productos que se ofrecen, para incentivar la compra por parte del cliente de la tienda.

3.2 Performance

Bajo una alta demanda de usuarios concurrentes se usará por ejemplo un compresor de View State y otros mecanismos en caso de ser necesario para no afectar los tiempos de respuesta en el aplicativo.

3.3 Compatibilidad

El aplicativo de comercio electrónico debe ser compatible con cualquier otra suite de controles que se desee incorporar posterior a su desarrollo.

3.4 Soporte

El aplicativo será instalado sobre un sistema operativo Windows XP, Windows 2003 server o Windows 7, bajo un Internet Information Services 6.0 o superior.

4. Interfaces del Sistema

4.1 Interfaces de Usuario

La tienda virtual debe presentar a sus usuarios las siguientes páginas durante la navegación de la misma:

- Una página de Login para el ingreso de los usuarios del aplicativo.
- Una página para el ingreso de datos personales.
- Una página para el ingreso de datos de envío, cuando se haya realizado una compra.
- Una página para el ingreso de información de pagos, cuando se haya realizado una compra.
- Una página principal donde se muestren las categorías y sus respectivos productos.
- Una página donde se detalle la información relacionada a una compra.
- Una página para mostrar un producto en mayor detalle.
- Una página de checkout para confirmar una comprar con los datos ingresados en las respectivas páginas de ingreso de información.

4.1.1 Look & Feel

Cada una de las páginas descritas deben adaptarse a estilos o skins creados por el desarrollador de la aplicación.

4.1.2 Requerimientos de Navegación

La página principal del aplicativo debe incluir el listado de las categorías, subcategorías y sus respectivos productos. Con respecto a la navegación, tanto usuarios registrados como no registrados podrán navegar por el aplicativo para observar los productos que ofrece la tienda virtual.

5. Restricciones del Sistema

- El aplicativo no incluye la validación de pagos en línea.

6. Requerimientos adicionales del Sistema

6. 1 Licenciamiento

El esquema licenciamiento para el caso de estudio es solamente con propósitos de demostración y no con fines comerciales.

6. 2 Estándares

El modelo de programación del aplicativo debe basarse en el estándar de programación de ASP.NET, para este tipo de aplicaciones.

7. Documentación del Sistema

Se incluirá un manual de instalación del sitio de ejemplo como ayuda en línea una vez que se instale el aplicativo.

3.2. ELABORACIÓN DEL PROTOTIPO USANDO LA SUITE DE CONTROLES

3.2.1. ARQUITECTURA DE LA APLICACIÓN

3.2.1.1. Plataforma Tecnológica

El aplicativo de ejemplo será desarrollado bajo la plataforma Microsoft Visual Studio 2010 y Microsoft .Net Framework 3.5 sobre sistema operativo Windows 7.

La plataforma de software para el desarrollo será:

Herramienta de Desarrollo	Uso
Microsoft Visual Studio 2010	Desarrollo, Modelamiento Visual, Desarrollo de Interfaces
Microsoft .net Framework 3.5	Desarrollo
ECommerceToolkit	Desarrollo de Interfaces
AjaxControlToolkit	Desarrollo de Interfaces

Tabla 3-1 Plataforma tecnológica para el desarrollo del prototipo
Elaborado por: Marcelo Vallejo C.

Las características mínimas de Hardware y software que se requieren para poder implementar la nueva suite de controles en el caso de estudio son las siguientes:

PRODUCTO	HARDWARE	SOFTWARE
Suite de Controles	• Procesador 2 x 3 GHz • 4GB memoria RAM • 50 GB libres en disco duro • Tarjeta de red	• Windows 2003 Server R2 (SP 2) o Windows 7. • IIS 6.0 o IIS 7. • Microsoft .NET 3.5.

Tabla 3-2 Requerimientos Hardware y Software para el caso de estudio
Elaborado por: Marcelo Vallejo C.

3.2.2. DISEÑO FUNCIONAL

3.2.2.1. Descripción del Contexto

En las aplicaciones de comercio electrónico es necesario incluir una suite de controles de servidor que sea simple de usar y que permita al desarrollador de la aplicación construirla con bases sólidas y en el menor tiempo posible.

3.2.2.2. Definición del Problema

Actualmente en cada de una las páginas de una aplicación de comercio electrónico es necesario incluir controles de servidor o realizar composición de los mismos para conseguir la funcionalidad deseada aunque esto tiene un inconveniente denominado “tiempo”, el cual se transforma en un atributo crítico en la mayoría de proyectos y se hace necesario la inclusión de una suite de controles para agilizar el proceso de desarrollo.

3.2.2.3. Interfaz de Usuario

Para definir la interfaz de usuario se toma como base la figura 1-6. Donde se muestra la estructura común que poseen las aplicaciones de comercio electrónico, partiendo de lo descrito anteriormente la interfaz principal del caso de estudio quedaría de la siguiente manera:

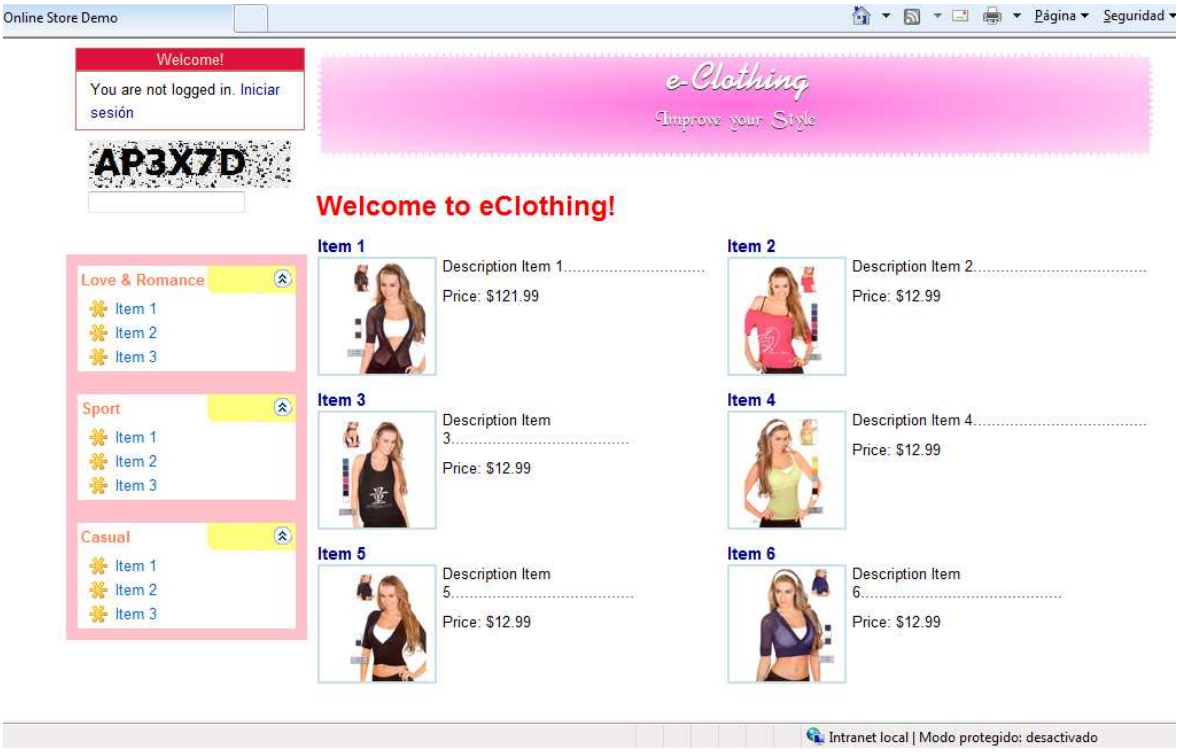

Figura 3-1 Prototipo ECommerce
Elaborado por: Marcelo Vallejo

3.2.3. INTERFACES

A continuación se describe cada una de las páginas del prototipo donde se usan los controles de servidor de la suite desarrollada:

3.2.3.1. Control EHeader

El control EHeader se muestra en la página Default.aspx a través del uso de una Master Page y permite incluir el logotipo de la tienda virtual mediante la propiedad “Path” y redireccionar el clic sobre el logotipo mediante la propiedad “Url”.

La siguiente figura muestra el uso del control EHeader sobre la página de ejemplo:

Figura 3-2 Implementación control EHeader

3.2.3.2. Control EMenu

El control EMenu al igual que el control EHeader se muestra en la página Default.aspx, a través de una Master Page y permite agregar categorías y subcategorías mediante la colección EMenuItem así mismo cada subcategoría permite mediante la propiedad Url, redireccionar a la página donde se encuentran listados sus respectivos productos tal y como se muestra en la siguiente figura:

Figura 3-3 Implementación control EMenu
Elaborado por: Marcelo Vallejo

Figura 3-4 Control EMenu desplegando categorías
Elaborado por: Marcelo Vallejo

3.2.3.3. Control EShoppingCart

El control EShoopingCart permite mostrar un resumen de los productos que se han adquirido por parte de un usuario durante la visita a la tienda virtual y el desarrollador de la aplicación enlaza los ítems mediante la propiedad DataSource y el método DataBind expuestos por el control, adicionalmente el programador de la página puede manipular eventos como:

- Eliminar un Ítem o producto.
- Actualizar el número de productos del mismo tipo.
- Proceder al Pago de los productos
- Observar más productos de la tienda virtual.

En la página ShoppingCart.aspx se puede observar el control EShoppingCart, como se muestra en la siguiente figura:

Figura 3-5 Implementación EShoppingCart
Elaborado por: Marcelo Vallejo

3.2.3.4. Control ECaptcha

El Control ECaptcha permite ayuda a validar la identidad de un usuario, se lo usa en distintas partes del aplicativo, por ejemplo en la página de inicio de sesión del aplicativo, como se muestra en la siguiente figura:

Figura 3-6 Implementación control ECaptcha
Elaborado por: Marcelo Vallejo

3.2.3.5. Control EShippingAddress

El control EShippingAddress permite el ingreso de información de destino para el envío de productos adquiridos por parte de un usuario, este control actualmente se lo usa en la página ShippingAddress.aspx y permite manipular la información ingresada por un usuario a través del evento:

- ShippingButtonClick

También permite manipular la información de ingresada por el usuario mediante las propiedades:

- InputAddressLine1
- InputAddressLine2
- InputCity
- InputCountry
- InputFullName
- InputPhoneNumber
- InputStateProvinceRegion

En la siguiente figura se puede observar el control EShippingAddress en la página ShippingAddress.aspx del aplicativo:

Figura 3-7 Implementación control EShippingAddress
Elaborado por: Marcelo Vallejo

3.2.3.6. Control EVirtualKeyboard

El control EVirtualKeyboard permite el ingreso de información sensible como números de tarjetas de crédito de un usuario, la información ingresada se la puede manipular mediante el evento:

- OnKeyBoardClick

En la página Checkout.aspx se puede observar el control EVirtualKeyBoard en funcionamiento, como se muestra en la siguiente figura:

Figura 3-8 Implementación control EVirtualKeyboard
Elaborado por: Marcelo Vallejo C.

3.2.3.7. Control EProductCatalog

El control EProductCatalog permite mostrar información relacionada a los productos que ofrece la tienda virtual, los productos se enlazan mediante la propiedad InternalDataList, actualmente el control se usa en la página ProductCatalog.aspx y se muestra en la siguiente figura:

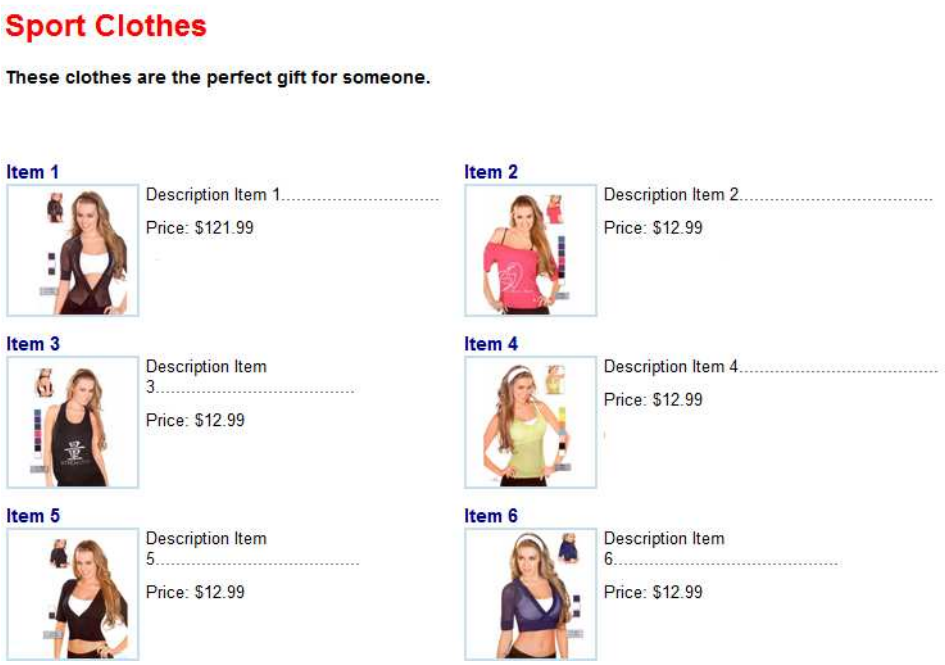

Figura 3-9 Implementación control EShoppingCart
Elaborado por: Marcelo Vallejo C.

3.2.3.8. Control EPayment

El control EPayment permite el ingreso de información de pagos, cuando se ha realizado una compra de uno o varios productos, para manipular la información se puede usar el evento:

- CardAddClick

Adicionalmente para manipular la información ingresada se puede utilizar las propiedades:

- CreditCardName
- CreditCardNumberInfo

El control se lo puede observar en la página “Checkout.aspx”, como se muestra en la siguiente figura:

Figura 3-10 Implementación control EPayment
Elaborado por: Marcelo Vallejo C.

3.2.3.9. Control EPersonalData

El control EPersonalData permite el ingreso de información de registro para un nuevo usuario, para manipular la información se puede usar el evento:

- CreateAccountClick.

Y a través de la clase "PersonalDataEventArgs" se puede manipular los datos ingresados por el cliente.

En la página Sign.aspx se incluye el control EPersonalData tal y como se muestra en la siguiente figura:

Figura 3-11 Implementación control EPersonalData
Elaborado por: Marcelo Vallejo C.

3.3. PRUEBAS DEL PROTOTIPO

Las pruebas que se realicen al aplicativo de e-commerce como caso de estudio permitirán definir si efectivamente se cumplen con los objetivos por los cuales fue construida la suite de controles.

3.3.1. ESPECIFICACIÓN DE PRUEBAS

Para determinar si construir una aplicación de comercio electrónico toma o consume menos tiempo a partir de la suite de tener una suite de controles se realizarán las siguientes comprobaciones:

1. Soporte a nivel de ventana de propiedades de Visual Studio para los controles de la suite.
2. Soporte en tiempo de diseño de los controles de la suite.

Basándose en los dos criterios anteriores se puede determinar si realmente se reduce el tiempo de desarrollo de un aplicativo de comercio electrónico, puesto que el soporte en tiempo de diseño y su facilidad de uso permiten al desarrollador de una aplicación reducir el tiempo de desarrollo de la misma.

3.3.2. EJECUCIÓN DE LAS PRUEBAS

Para ejecutar las comprobaciones previamente descritas, principalmente relacionadas a la ejecución de los controles en tiempo de diseño se incluyen las Figuras 3-12 hasta la Figura 3-23 tomando de ejemplo los controles más complejos de la suite, donde se puede observar que los controles poseen soporte en tiempo de diseño.

Figura 3-12Vista Tiempo de Diseño control EMenu
Elaborado por: Marcelo Vallejo

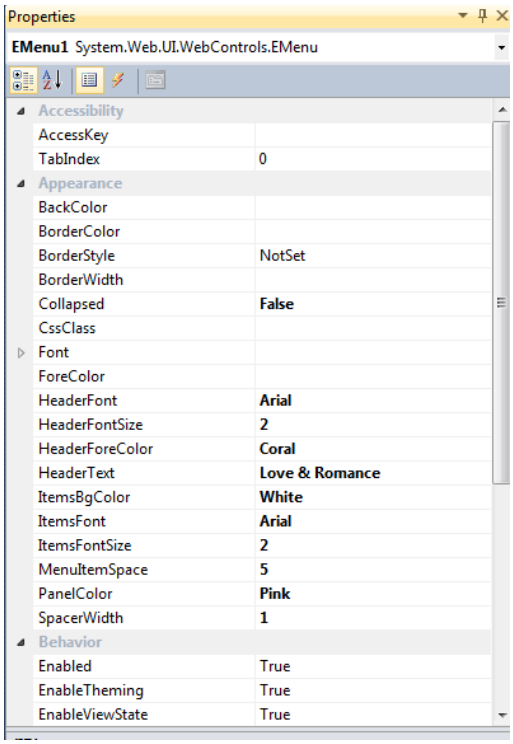

Figura 3-13 Ventana de propiedades control EMenu
Elaborado por: Marcelo Vallejo C.

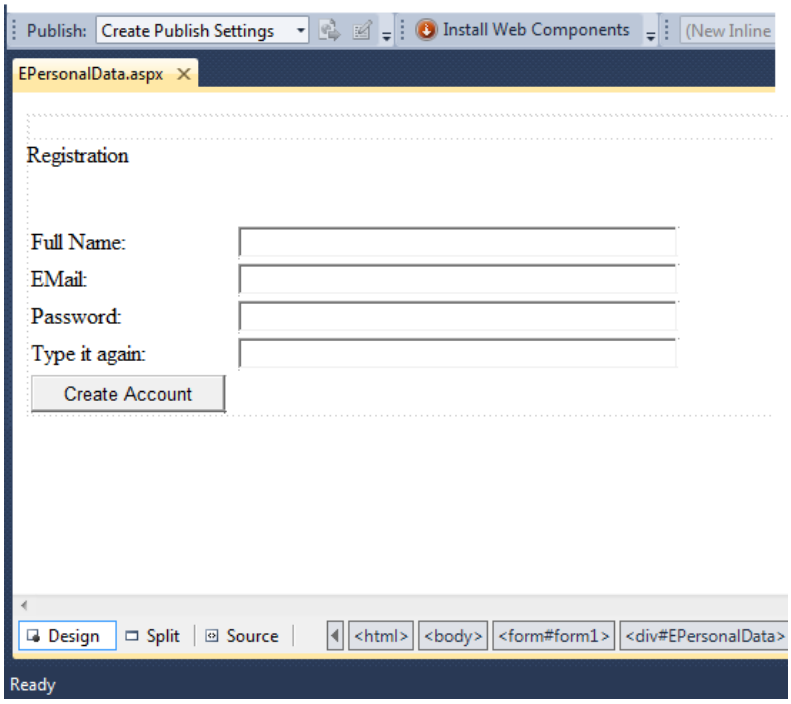

Figura 3-14 Vista Tiempo de Diseño control EPersonalData
Elaborado por: Marcelo Vallejo C.

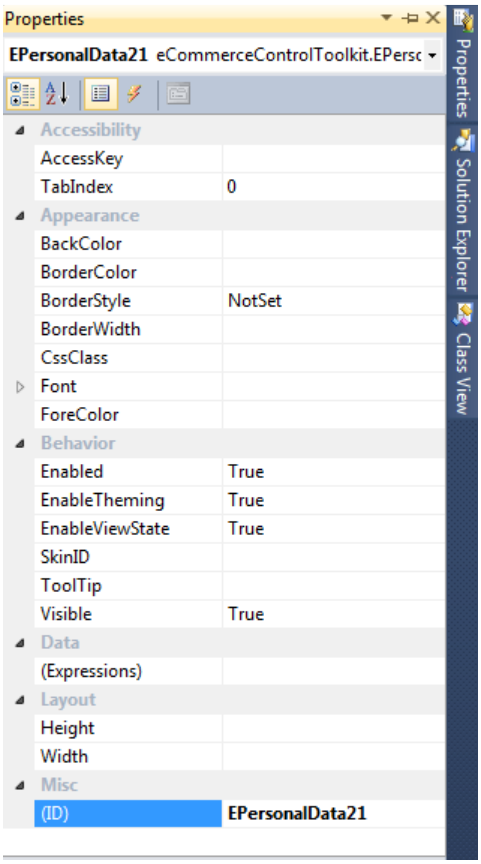

Figura 3-15 Ventana de propiedades control EPersonalData
Elaborado por: Marcelo Vallejo C.

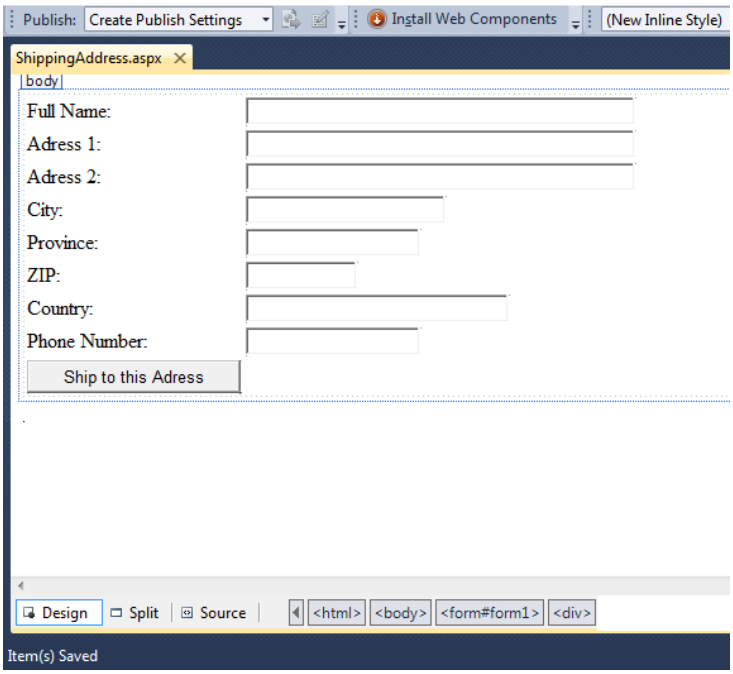

Figura 3-16 Vista Tiempo de Diseño control EShippingAddress
Elaborado por: Marcelo Vallejo

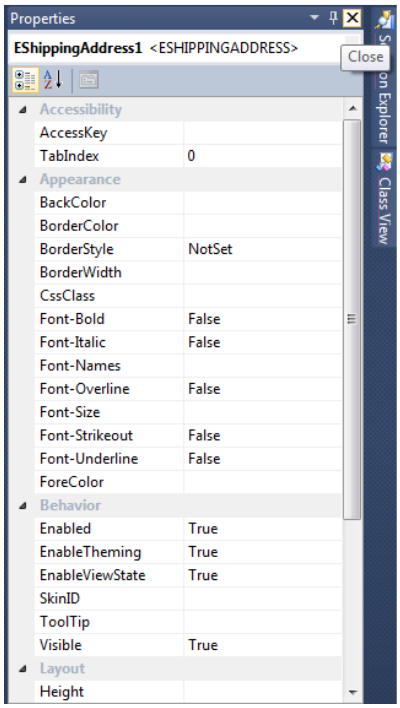

Figura 3-17 Ventana de propiedades control EShippingAddress
Elaborado por: Marcelo Vallejo

Figura 3-18 Vista Tiempo de Diseño control EProductCatalog
Elaborado por: Marcelo Vallejo

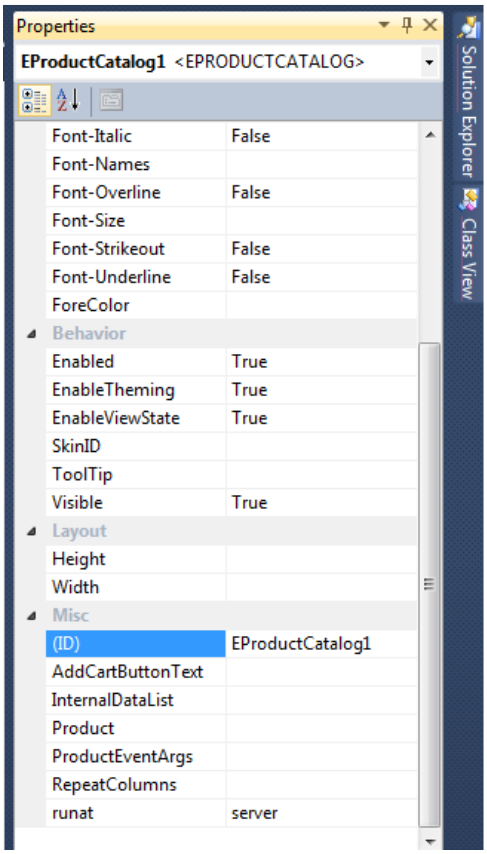

Figura 3-19 Ventana de Propiedades control EProductCatalog
Elaborado por: Marcelo Vallejo C.

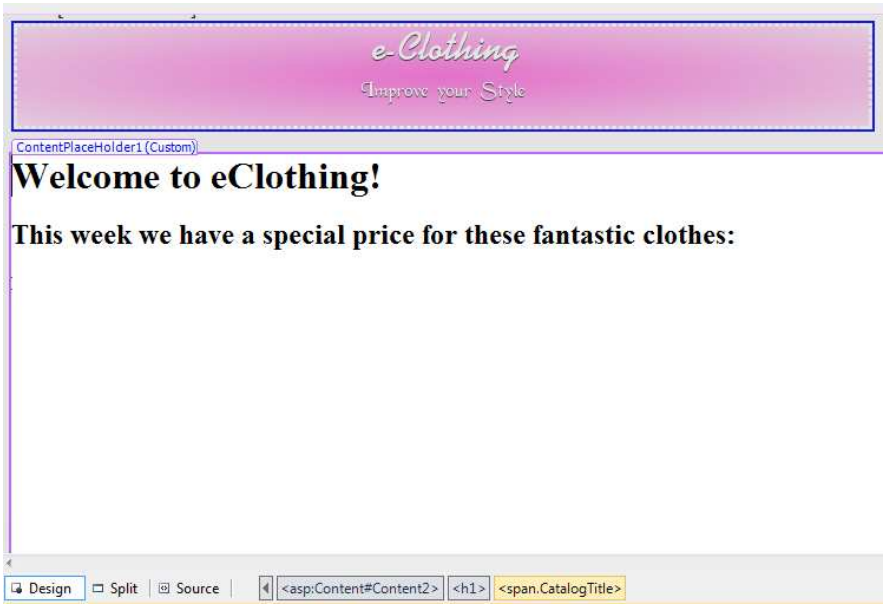

Figura 3-20 Vista Tiempo de Diseño control EHeader
Elaborado por: Marcelo Vallejo C.

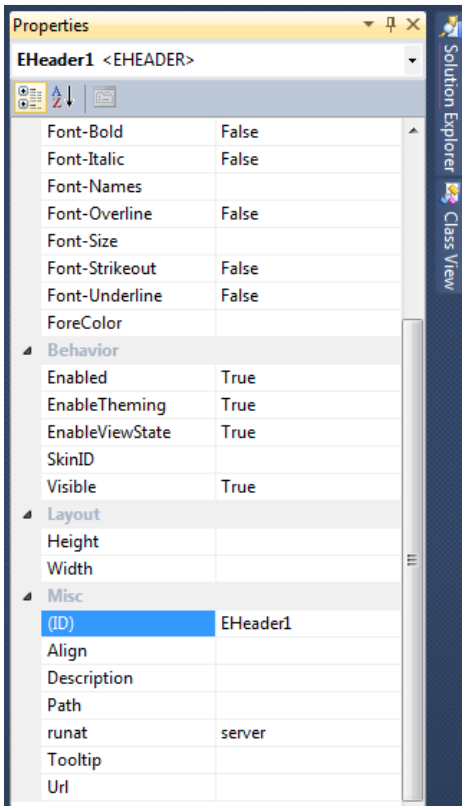

Figura 3-21 Ventana de propiedades control EHeader
Elaborado por: Marcelo Vallejo C.

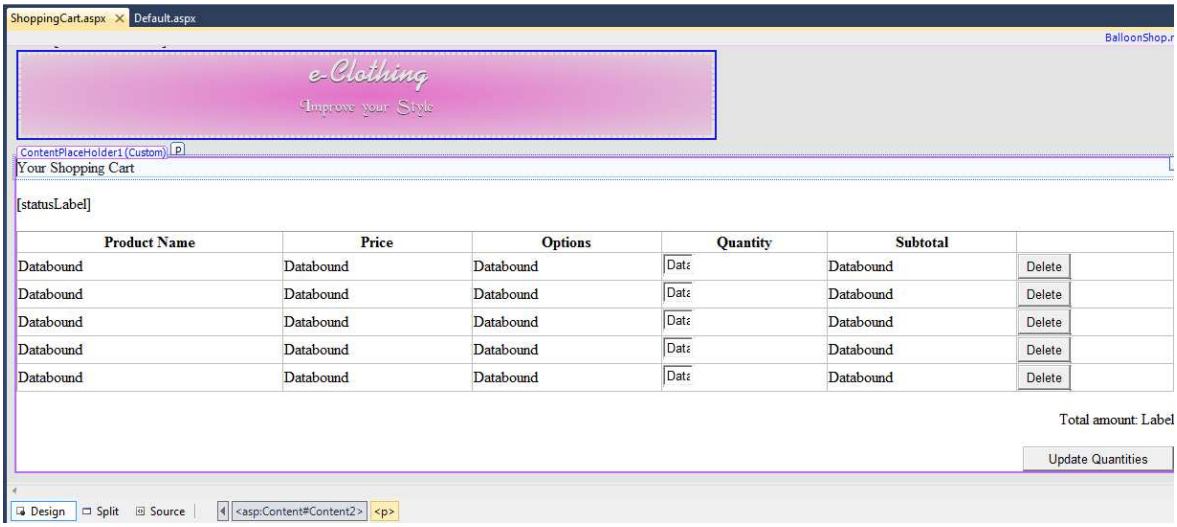

Figura 3-22 Vista Tiempo de Diseño control EShoppingCart
Elaborado por: Marcelo Vallejo C.

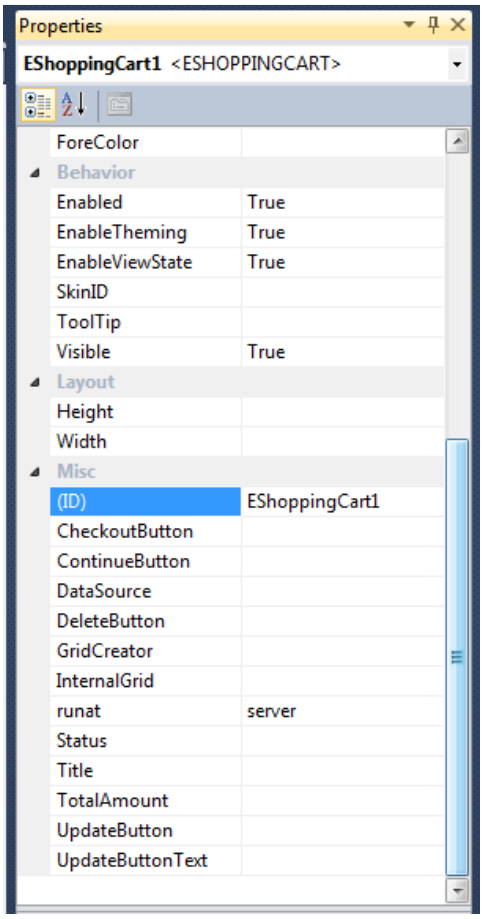

Figura 3-23 Ventana de propiedades control EShoppingCart
Elaborado por: Marcelo Vallejo C.

3.4. ANÁLISIS DE RESULTADOS

De las figuras que se muestran en la sección 3.3.3 y tomando en cuenta que figuras forman parte de criterios de comprobación que se definieron en la sección 3.3.2, se puede decir lo siguiente:

- Las ventanas de propiedades y el soporte en tiempo de diseño de cada uno de los controles permiten configurarlos de una manera simple sin necesidad de ejecutar el aplicativo para observar la apariencia visual del control.

- Los controles de la suite creada se adaptan fácilmente a cualquier tipo de elemento o control ASP.NET, en el caso de estudio se creó una Master Page dentro de la cual se pudo incluir los controles EHeader y EMenu,

mostrando su compatibilidad con el estándar de programación Microsoft para controles de servidor.

- La vista en tiempo de diseño incluso permite manipular los colores con los que se va a mostrar el control en la ejecución del aplicativo, como se muestra en la Figura comparativa 3-24.

Figura 3-24 Incorporación de colores en el control EMenu
Elaborado por: Marcelo Vallejo C.

- Mediante la ventana de propiedades es posible cambiar el ancho, alto entre otras propiedades para un determinado control.

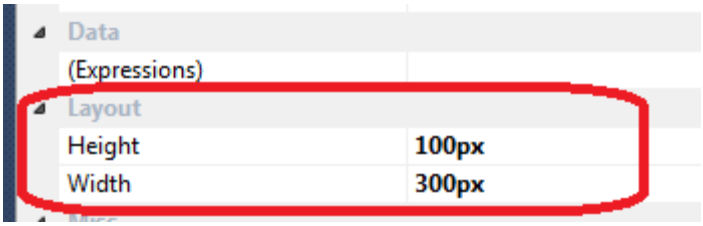

Figura 3-25 Cambio de valores mediante ventana de propiedades
Elaborado por: Marcelo Vallejo C.

- Como se puede observar en la Figura 3-1 también es factible combinar los controles de la suite con controles de otros proveedores dentro de una

misma página ASP, lo que permite mostrar la interoperabilidad de la suite desarrollada.

- Para finalizar se puede observar en las capturas de pantalla anteriores que para todos los controles de la suite existe un manejo completo en tiempo de diseño lo que facilita la configuración de los controles y disminuye el tiempo de desarrollo de aplicaciones porque no es necesario estar ejecutando el aplicativo cada instante que se haya realizado un cambio visual a un control.

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

Para finalizar el proyecto “Desarrollo de una suite de controles de servidor ASP.NET para aplicaciones de tipo e-commerce” es necesario establecer un conjunto de conclusiones y recomendaciones las cuales se detallan a continuación:

4.1. CONCLUSIONES

- Para desarrollar controles de servidor ASP.NET deben estar claros los requerimientos funcionales, puesto que éstos determinan la calidad misma del control que se va a desarrollar, también es importante mencionar que se deben adaptar las metodologías de desarrollo de software para este tipo de componentes, dado que muchos de los artefactos que se deben generar con esas metodologías realmente no brindan valor agregado al desarrollo de controles de servidor.

- Actualmente el mercado con relación a las suites de controles de servidor ofrece una amplia gama de controles, que se enfocan en cubrir funcionalidad para cualquier tipo de aplicación y por tanto se debe realizar un trabajo adicional para incorporar una determinada suite a la aplicación que se esté desarrollando.

- La suite de controles para aplicaciones de tipo e-commerce aporta en la disminución en el tiempo de desarrollo, dado que el proceso de composición de controles consume alrededor del 25% del tiempo total de implementación de un sistema de comercio electrónico, entonces al tener una suite de controles específica se estaría disminuyendo el tiempo que toma implementar una solución de ese tipo en el valor antes mencionado.

- A través de la implementación del prototipo de e-commerce se ha demostrado que construir aplicaciones de ese tipo toma menos tiempo y adicionalmente se tiene una solución estable que se basa en los requerimientos comunes de un aplicativo de comercio electrónico.

4.2. RECOMENDACIONES

- Para el desarrollo de suites de controles de servidor, es conveniente partir el desarrollo de la suite de manera paralela al desarrollo de la aplicación, logrando así abstraer claramente los requerimientos funcionales que deben tener los controles.

- Es una buena práctica incluir los controles de servidor en una librería de enlace dinámico para distribuir y re-utilizar fácilmente los controles en cualquier otra aplicación que se vaya a desarrollar.

- Si se desea adquirir componentes de terceros, en este caso suites de controles, se debe evaluar a cada una de las suites ofertantes y revisar la funcionalidad que cada una de ellas ofrece adicionalmente a los criterios que se mencionan en este proyecto.

- La suite de controles puede evolucionar hasta convertirse en un producto que no solamente sea orientado para desarrolladores, sino también para personas que no posean conocimiento técnico en el desarrollo de aplicaciones de comercio electrónico y que mediante la implementación o desarrollo de wizards puedan crear su aplicación de e-commerce usando los controles.

BIBLIOGRAFÍA

- Cristian Darie, Karli Watson, Beginning ASP.NET E-Commerce in C#: From Novice to Professional (Expert's Voice in .NET), Segunda Edición. Apress. Estados Unidos. Año 2009.
- Nikhil Kothari, Vandana Datye, Developing Microsoft ASP.NET Server Controls and Components (Pro-Developer). Microsoft Press. Estados Unidos. Año 2003.
- Adam Calderon, Joel Rumerman, Advanced ASP.NET AJAX Server Controls For .NET Framework 3.5. Addison Wesley. Estados Unidos. Año 2009.
- David Sussman , Beginning Dynamic Websites with ASP.NET Web Matrix, Wrox Press. Estados Unidos. Año 2003.
- Michael A. Kittel, Geoffrey T. LeBlond, ASP.NET Cookbook, Segunda Edición. O'Reilly. Año 2005
- Rob Cameron and Dale Michalk, Building ASP.NET Server Controls. Apress. Estados Unidos. Año 2004.
- Soporte Microsoft
    - http://support.microsoft.com

ANEXOS

El anexo que se detalla a continuación, se encuentra en el disco compacto que acompaña a este documento

Anexo 1. Manual de uso de los controles de servidor creados.