UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

“Mejoramiento de ICallbackEventHandler mediante una herramienta basada en Reflection y JavaScript”

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO INFORMÁTICO

PRESENTADO POR: CHÁVEZ GALLEGOS, CHRISTIAN ; PANTOJA ASCA, MICHAEL MOAMMAR ALÍ

LIMA – PERÚ

NOVIEMBRE – 2014

Dedicamos esta tesis a Dios por darnos la vida y siempre estar a nuestro lado dándonos fuerzas para seguir adelante. Agradecemos a nuestros padres por estar siempre con nosotros cuando más los necesitamos y por todo su amor que nos brindan cada día y por el esfuerzo que han realizado para poder culminar nuestros estudios, dedicado también a todas las personas que les gusta la programación.

AGRADECIMIENTOS

Agradecemos a nuestro asesor Ing. Humberto Linares, quien estuvo apoyándonos en la realización de nuestro proyecto y a los miembros de nuestro jurado al Dr. Hugo Vega, Ing. Francisco Aguilar, Ing. Juana Segura e Ing. Silvia Campos. Gracias por su apoyo y contribución a nuestra tesis. A nuestra alma mater la Universidad Ricardo Palma, la cual nos ha formado y guiado en este tiempo para ser personas de bien y profesionales destacados.

EPÍGRAFE

Sueña cuando acaba el día, Sueña y tus sueños podrían hacerse realidad, Las cosas nunca son tan malas como parecen, Así que sueña, sueña, sueña.

Philip Roth

RESUMEN

Titulo	:	Mejoramiento de ICallbackEventHandler mediante una herramienta basada en Reflection y JavaScript
Autores	:	Chávez Gallegos Christian y Pantoja Asca Michael Moammar Alí
Asesor de Tesis	:	Ing. Humberto Víctor Linares Coloma
Jurado Evaluador	:	Presidente: Dr. Hugo Vega (Director de la Escuela Académico de Ingeniería Informática)
Miembros	:	Ing. Francisco Aguilar, Ing. Juana Segura e Ing. Silvia Campos
Fecha	:	14 de Noviembre del 2014

Este trabajo alcanza el objetivo de encontrar una manera fácil y rápida de utilizar la interfaz iCallbackEventHandler de Microsoft. El proyecto potencia la forma de utilizar iCallbackEventHandler, elimina la utilización directa de sus métodos GetCallbackResult y RaiseCallbackEvent permitiendo a los usuarios crear sus propios métodos los cuales serán invocados desde el lado del cliente, esto permite enviar varios parámetros al método del servidor a diferencia de la manera actual que solo permite enviar uno, para lograr este cometido se utiliza “events” y “reflection”. En la usabilidad se tomó como referencia la

forma en que la librería jQuery realiza las llamadas asíncronas utilizando su función AJAX, es decir se utiliza "object literal notation" de JavaScript para invocar a los métodos del servidor. El producto final es una DLL que contiene todas las clases necesarias para que puedan ser referenciadas simplemente agregándola en cualquier proyecto.

PALABRAS CLAVES

ICallbackEventHandler, Llamada asíncrona, C#

ABSTRACT

Title	:	Improvement ICallbackEventHandler using a tool based on JavaScript and Reflection
Authors	:	Chávez Gallegos Christian and Pantoja Asca Michael Moammar Alí
Thesis Advisor	:	Eng. Humberto Víctor Linares Coloma
Jury Reviewers	:	President: Dr. Hugo Vega (Head Teacher)
Members	:	Eng. Francisco Aguilar, Eng. Juana Segura and Eng. Silvia Campos
Date	:	November 14th, 2014

This work achieves the goal of finding a quick and easy way to use interface ICallbackEventHandler Microsoft. The project how to use power ICallbackEventHandler eliminates the direct use of their methods and RaiseCallbackEvent GetCallbackResult allowing users to create their own methods which will be called from the client side, this allows to send multiple parameters to the server method unlike current so that only lets you send one to achieve this goal "events" is used and "reflection". Usability was taken as reference how the jQuery

library does asynchronous AJAX calls using your function, that is to say "object literal notation" JavaScript is used to invoke the server methods. The final product is a DLL that contains all the necessary classes so they can be referenced simply adding it in any project.

KEYWORDS

ICallbackEventHandler, Asynchronous Call, C#

ÍNDICE

INTRODUCCIÓN

Las buenas prácticas en programación ayudan a que el código generado sea más entendible, además hace que el sistema tenga un mejor funcionamiento y que su mantenimiento sea más fácil.

Hay dos tipos de lenguaje de programación, lenguajes de bajo nivel y lenguajes de alto nivel. Los lenguajes de bajo nivel son instrucciones directas que se hacen al microprocesador donde se esté desarrollando el programa. El problema que existe al desarrollar programas con este tipo de lenguaje, es la portabilidad. No se puede esperar que un programa desarrollado en una maquina X, funcione de la misma manera (o simplemente funcione) en una maquina Y.

Los lenguajes de alto nivel solucionan la desventaja de los lenguajes de bajo nivel pero pierden su ventaja, dar instrucciones directas al procesador, sin embargo desarrollar un programa con un lenguaje de alto nivel es mucho más fácil.

Existen varios lenguajes de programación de alto nivel, entre los que se encuentran: C#, Java, C++, Visual Basic, etc. Todos estos tienen una sintaxis parecida, lo cual hace que aprender uno, ya sabiendo otro, sea fácil o por lo menos que la curva de aprendizaje sea corta.

En el desarrollo de páginas web no es necesario u obligatorio el uso de los lenguajes mencionados anteriormente, sin embargo su utilización hace que las webs sean más potentes. Lo importante en una página web es el uso de HTML.

HTML, siglas de HyperText Markup Language (*Lenguaje de Marcado de Hipertexto*), es el lenguaje de marcado predominante para la elaboración de páginas web. Se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>).

Actualmente al desarrollar páginas webs se prima que el comportamiento de la misma se parezca al de una aplicación de escritorio, para esto se utilizan llamadas asíncronas. Las

llamadas asíncronas permiten eliminar el postback que se encuentra por defecto en todas las páginas web. Una forma de realizar llamadas asíncronas es utilizando Ajax.

Ajax, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

En el capítulo uno se describe la visión del proyecto, en el segundo capítulo se trata el tema del marco teórico, en el tercer capítulo se muestra el estado del arte, en el cuarto capítulo se muestra el modelado del negocio, en el quinto capítulo se realizan los requerimientos del proyecto, en el sexto capítulo se desarrolla la arquitectura, en el séptimo capítulo se muestra lo necesario para el desarrollo y las pruebas, en el octavo capítulo se muestra la gestión del proyecto. Se finaliza con las conclusiones y recomendaciones.

CAPÍTULO I: VISIÓN DEL PROYECTO

1.1 Antecedentes del Problema

A continuación se muestra a rasgos grandes el negocio en la cual está centrada la tesis y posteriormente se indica el problema que se desea solucionar.

1.1.1 El Negocio

Esta tesis está enfocada en empresas que desarrollan páginas webs. Estas empresas tienen una serie de procesos que pueden variar según la organización. En la Figura 1 se muestran los procesos identificados y que son los más significativos:

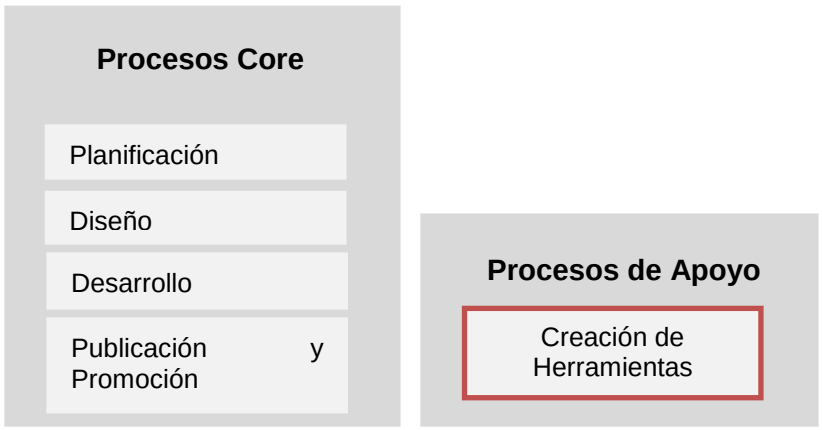

Figura 1: Procesos del Negocio

Fuente: Elaboración Propia, 2014

Esta tesis colabora con los procesos de apoyo de este tipo de organizaciones, es una herramienta que facilita la creación de páginas Webs.

1.2 Fundamentación del problema

Hay mucha dificultad en implementar la interfaz ICallbackEventHandler cuando se desea agregar varias llamadas asíncronas en una misma página web. Existen una serie de pasos que se deben realizar cada vez que se quiera ejecutar un evento que tenga las características

asíncronas utilizando ICallbackEventHandler, esto demanda un exceso de tiempo y esfuerzo que podría ser reducido.

A continuación se describen los pasos:

Implementar la interface

Implementar los métodos de la interface

Colocar el script del servidor

Colocar el script del lado del cliente

Modificar los métodos de la interface según el funcionamiento que se desea obtener.

Realizar llamada asíncrona

Este es justamente el problema que se ha encontrado y se atacará, para reducir los pasos y dar facilidad para implementar más eventos en una misma interfaz de una manera más fácil y rápida.

1.3 Objetivos del proyecto

A continuación se muestra el marco lógico, donde se expone tanto el árbol de problemas y objetivos, luego se muestra el objetivo general y los objetivos específicos.

1.3.1 Marco Lógico

1.3.1.1 Árbol de problemas

El problema principal es la dificultad en desarrollar páginas utilizando la interfaz ICallbackEventHandler, esto se debe a la cantidad de pasos que se han de realizar, otra causa es que solo hay un método que recibe la información del cliente y éste se hace más confuso según se vayan agregando más eventos asíncronos. Esto trae dificultades a la hora de implementar varios eventos en una solo página, debido a que el método RaiseCallbackEvent es el único que procesa la información del cliente. En la Figura 2 se muestra el árbol de problemas.

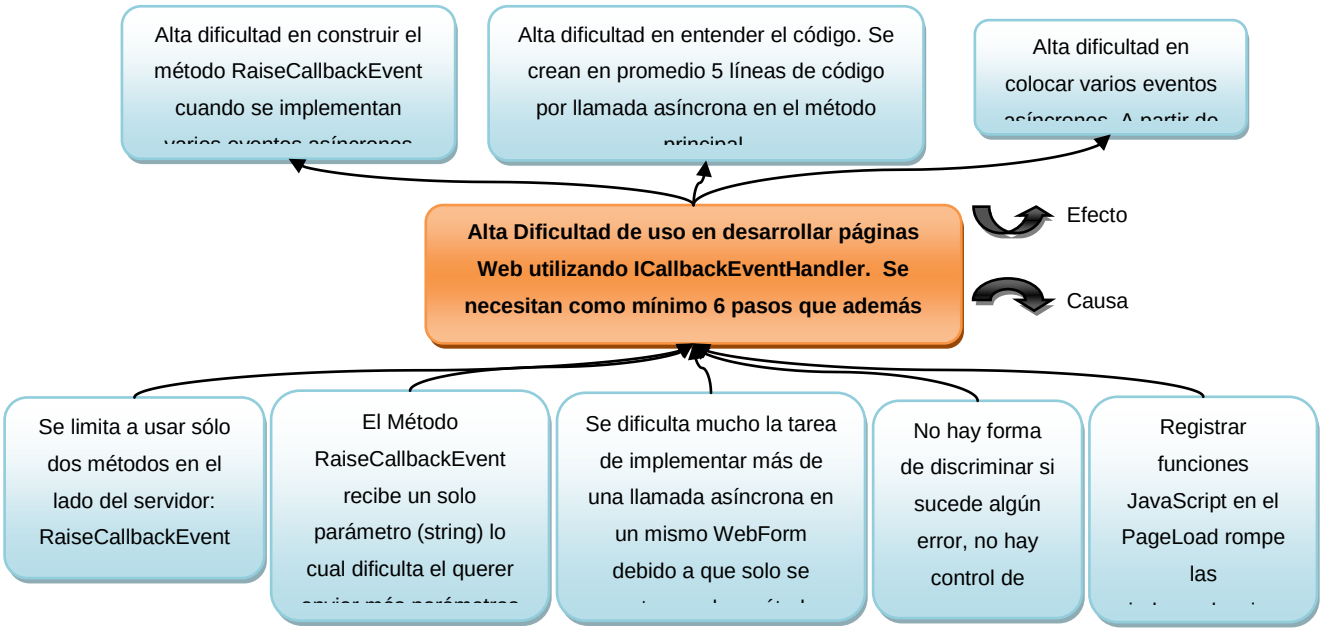

Figura 2: Árbol de Problemas

Fuente: Elaboración Propia, 2013

1.3.1.2 Árbol de objetivos

El árbol de objetivos muestra lo inverso del árbol de problemas, se descubre que el objetivo central es bajar la dificultad reduciendo la cantidad de pasos y optimizando cada uno de ellos. En la Figura 3 se muestra el árbol de objetivos.

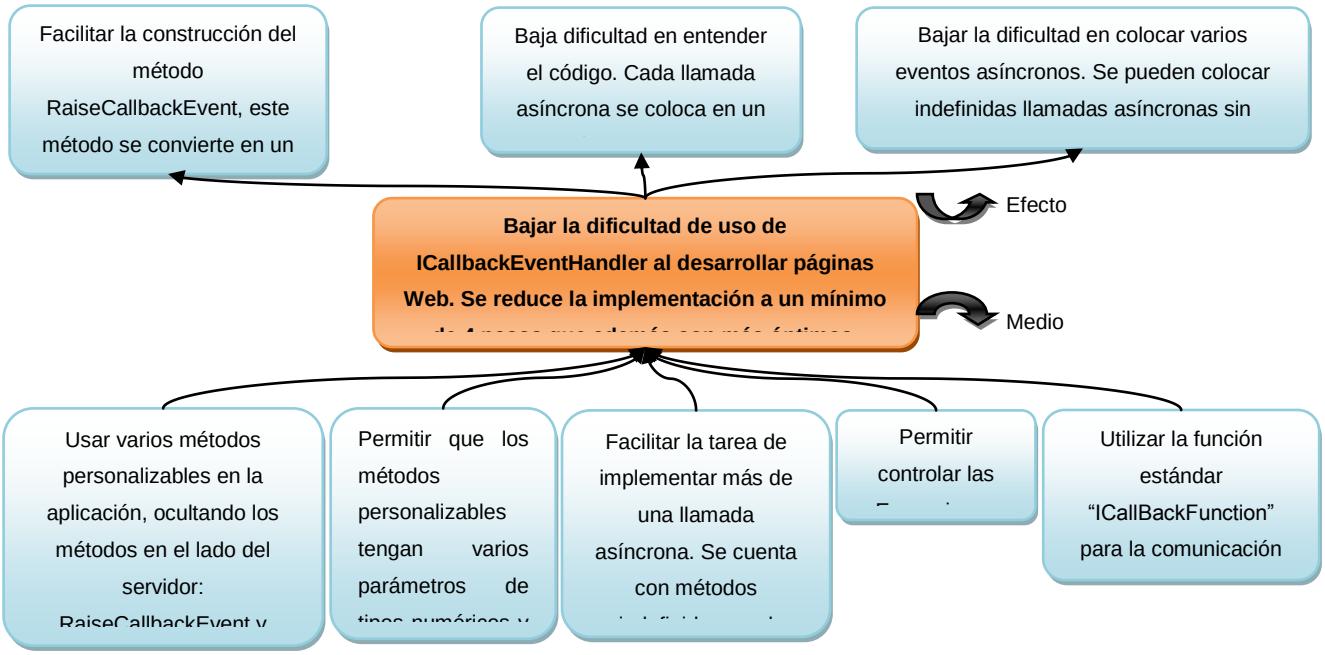

Figura 3: Árbol de Objetivos

Fuente: Elaboración Propia, 2013

1.3.2 Objetivo General

Bajar la dificultad de uso de ICallbackEventHandler al desarrollar páginas Web. Se reduce la implementación a un mínimo de 4 pasos que además son más óptimos.

1.3.3 Objetivos Específicos

Usar varios métodos personalizables en la aplicación, ocultando los métodos en el lado del servidor: RaiseCallbackEvent y GetCallbackResult.

Permitir que los métodos personalizables tengan varios parámetros de tipos numéricos y tipo String.

Facilitar la tarea de implementar más de una llamada asíncrona. Se cuenta con métodos indefinidos en el servidor

Permitir controlar las Excepciones

Utilizar la función estándar "ICallBackFunction" para la comunicación con el servidor

1.4 Importancia (Justificación)

A continuación se muestra los beneficios tangibles e intangibles que se obtendrán.

1.4.1 Beneficios Tangibles

Tratándose de un tema que busca facilitar la implementación de ICallbackEventHandler, se han encontrado los siguientes beneficios tangibles:

Aumento de la eficiencia en el código

Reducción de tiempo para desarrollar páginas con ICallbackEventHandler

Mayor facilidad para implementar ICallbackEventHandler.

1.4.2 Beneficios Intangibles

Los beneficios intangibles que se han encontrado son los siguientes:

Rapidez en aprendizaje para implementar ICallbackEventHandler.

1.5 Alcance del proyecto

El alcance del proyecto señala lo que la tesis se propone a desarrollar y que cosas no tiene considerado hacer.

Que se va a hacer

Se desarrolla una DLL que permita implementar las mejoras que se mencionan en los objetivos específicos.

Que no se va a hacer

No se creó una aplicación que permita comparar la velocidad de la DLL creada, con todas las tecnologías que se toman como referencias en el proyecto, que son: PageMethods, Ajax con JQuery y el mismo ICallbackEventHandler.

CAPÍTULO II: MARCO TEÓRICO

2.1 Optimización de Procesos

Dice [Garimella, lees, & Williams, 2009] que los procesos operacionales transforman los recursos y materiales en productos o servicios para los clientes y consumidores finales. Esta "transformación" es el funcionamiento de un negocio; que es el elixir mágico de la empresa. Esta transformación es la más efectiva, el mayor éxito que te crea valor.

La ciencia aplicada de los procesos y la transformación abarca la historia de la gestión industrial moderna de los gurús de la calidad como Deming, Juran, Shingo, Crosby y Peters y recientemente las prácticas de Lean y Six Sigma. BPM incorpora plenamente estas metodologías, y los acelera con una mejora dramáticamente de los sistemas de definición, medición, análisis y control.

La eficacia del proceso

Los procesos efectivos son más coherentes, generan menos residuos, y crean mayor valor neto para los clientes y las partes interesadas.

BPM promueve directamente el aumento de la efectividad del proceso a través de la automatización de adaptación y coordinación de personas, información y sistemas.

A diferencia de los métodos y herramientas del pasado, BPM no impone la eficacia a través de los sistemas rígidos y de inflexibles controles centrado en los dominios funcionales. En su lugar, BPM permite la respuesta continua y la adaptación a los eventos y condiciones del mundo real y en tiempo real. En la Figura 4 se puede apreciar el ciclo de la excelencia del proceso del negocio.

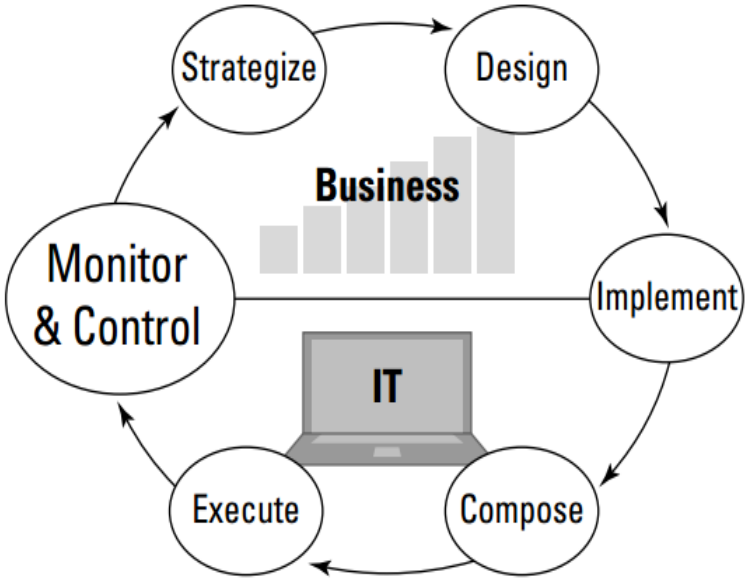

Figura 4: El ciclo de la excelencia del proceso del negocio

Elaboración: Obtenido de [Garimella, lees, & Williams, 2009]

Dice [Arsenovski, 2009] que el objetivo de la refactorización es mejorar el diseño del código, lo que se logra generalmente mediante la aplicación de las modificaciones a la misma. El proceso de refactorización es bastante simple y consiste en tres pasos básicos:

Identificar la hediondez del código: Este primer paso tiene que ver con la identificación de los posibles escollos en el código, y la hediondez del código es muy útil en la identificación de esas trampas.

Aplicar la refactorización apropiada: Este paso se dedica a cambiar la estructura del código por medio de transformaciones de refactorización. Estas transformaciones a menudo pueden ser automatizadas y realizadas por una herramienta de refactorización.

Ejecutar pruebas unitarias: Este paso ayuda a rectificar el estado del código después de las transformaciones. Refactorizar no pretende cambiar cualquier comportamiento del código observable desde la afuera. Este paso consiste generalmente en la ejecución de las pruebas unitarias adecuadas para garantizar que el comportamiento del código no ha cambiado después de realizar la refactorización.

A la luz de todas las reglas y las técnicas relacionadas con la refactorización, es pertinente preguntarse qué beneficios trae la refactorización. Después de todo, no añade nuevas características ni resuelve errores, y al final se termina con el código que básicamente hace lo que solía, así que ¿por qué se debería invertir tiempo y dinero para llevar a cabo esta actividad? ¿Cuáles son los beneficios de mantener óptimo el diseño en todo momento?

Mantiene el codigo simple: Debido a que el desarrollo de software es un proceso continuo, en evolución, la refactorización puede simplificar los numerosos aspectos que hay que hacer como desarrollador, permiten la racionalización de trabajo y proporcionan cualidades importantes para el código. Mantener el código conciso en todo momento puede ser un reto, especialmente cuando se está bajo presión para entregar los resultados rápidamente.

Mantiene el codigo legible: La programación es intelectualmente una actividad muy intensa. A menudo se tiende a tener una comprensión más profunda y detallada del codigo propio con el fin de mantener el control total sobre él. Se puede tratar de memorizar todos los detalles del código. A menudo uno se siente orgulloso cuando es capaz de corregir de inmediato un error o algún comportamiento. A medida que se más productivo, a desarrollar estrategias y ganar un estilo propio de programación. No hay nada malo en ser experto con el código que uno crea, a menos que la experiencia se convierte en la única arma que se tiene.

Por desgracia, a veces se puede olvidar un hecho importante: En el desarrollo de software, rara vez se trabaja solo y con el fin de ser capaz de trabajar en un equipo, se debe escribir un código que sea fácil de entender para los demás, ya que podrían necesitarlo para modificar, mantener , u optimizar el código. En ese caso, cuando se enfrentan con código críptico o hermético, otros miembros del equipo podrían perder muchas horas tratando de entenderlo. Tarde o temprano se tendrá una computadora para hacer todo, pero hasta entonces, el codigo fuente debe ser tal que sea fácil de entender para los demás. La memoria es otra consideracion a tener en cuenta, ésta tiene sus límites, y después de un tiempo puede que no se sea capaz de recordar todos los detalles del código que uno mismo escribió.

Dice [Deitel & Deitel 2010] Los programadores se concentran en elaborar nuevas clases y reutilizar las existentes. Existen muchas bibliotecas de clases, y otras se están desarrollando en diversas partes del mundo. Así, el software se construye a partir de componentes existentes, bien definidos, cuidadosamente probados, bien documentados, portables, optimizados para el mejor rendimiento, y ampliamente disponibles. Este tipo de reutilización de software agiliza el desarrollo de software poderoso y de alta calidad. El desarrollo rápido de aplicaciones (RAD) es de gran interés hoy en día.

Microsoft proporciona a los programadores de C# miles de clases en la Biblioteca de clases del .NET Framework, para ayudar a implementar aplicaciones en C#. El .NET Framework permite a los desarrolladores en C# trabajar para lograr una verdadera reutilización y un rápido desarrollo de aplicaciones. Los programadores en C# se pueden enfocar en la tarea en cuestión a la hora de desarrollar sus aplicaciones, y dejan los detalles de bajo nivel a las clases de la FCL. Por ejemplo, para escribir una aplicación que dibuje gráficos, un programador de la FCL no requiere conocer los gráficos en cada una de las plataformas computacionales en las que se ejecutará la aplicación. En vez de ello, el programador se puede concentrar en aprender las capacidades de gráficos de .NET (que son bastante sustanciales y aumentan en forma constante) y escribir una aplicación en C# que dibuje los gráficos, usando clases de la FCL como las que se encuentran en el espacio de nombres System.Drawing. Cuando la aplicación se ejecuta en cierta computadora, es responsabilidad de la CLR traducir los comandos de MSIL compilados del código C# en comandos que la computadora local pueda entender.

Las clases de la FCL permiten a los programadores en C# llevar las nuevas aplicaciones al mercado con más rapidez, mediante el uso de componentes preexistentes ya probados. Esto no sólo reduce el tiempo de desarrollo, sino que también mejora la habilidad del programador para depurar y dar mantenimiento a las aplicaciones. Para aprovechar las diversas herramientas de C#, es esencial que los programadores se familiaricen con la variedad de clases en el .NET Framework. El recurso principal para aprender acerca de la FCL es la Referencia de .NET Framework.

Para comprender todo el potencial de la reutilización de software. Se necesita:

1. Mejorar los esquemas de clasificación, los esquemas de licenciamiento, los mecanismos de protección que evitan la corrupción de copias maestras de las clases.

2. Mejorar los esquemas de descripción que utilizan los desarrolladores de sistemas para determinar si las clases existentes cumplen con sus necesidades,

3. Mejorar los mecanismos de exploración que determinan qué clases hay disponibles y qué tan cerca está de cumplir con los requerimientos de los desarrolladores de software.

4. Se tiene que evitar reinventar la rueda. Se debe estudiar las herramientas de la FCL. Si la FCL contiene una clase que cumpla con los requerimientos de su aplicación, se debe utilizar en lugar de crear una clase propia.

Muchos problemas interesantes de investigación y desarrollo se han resuelto, y hay muchos más por resolver. Es probable que estos problemas se resuelvan, debido a que el valor potencial del aumento en la reutilización de software es enorme

2.2 ICallbackEventHandler

Dice [MacDonald, Freeman, & Szpuszta 2010] Para crear una devolución de llamada del cliente en ASP.NET, primero se tiene que planificar cómo va a funcionar la comunicación. He aquí el modelo básico:

1. En algún momento, un JavaScript desencadena el evento, lo que provocó la devolución de llamada del servidor.

2. En este punto, se produce el ciclo de vida de la página normal, lo que significa que todos los eventos normales del lado del servidor se disparan, tales como Page.Load.

3. Cuando se complete este proceso (y la página inicializó correctamente), ASP.NET ejecuta el método de devolución de llamada del lado del servidor. Este método debe tener una firma fija que acepta un solo parámetro de cadena y devuelve una única cadena.

4. Una vez que la página recibe la respuesta desde el método del lado del servidor, este utiliza código JavaScript en consecuencia para modificar la página web. La arquitectura de ASP.NET está diseñado para abstraer el proceso de comunicación, por lo que puede construir una página que utiliza devoluciones de llamada sin preocuparse acerca de este nivel, de la misma manera que usted puede aprovechar el estado de vista y el ciclo de vida de la página.

Para recibir una devolución de llamada, se necesita una clase que implemente la interfaz ICallbackEventHandler. Si se sabe que la devolución de la llamada se utiliza en varias páginas, tiene sentido crear una clase especial. Sin embargo, si desea definir la funcionalidad que está destinado a una sola página, se puede implementar ICallbackEventHandler en la misma página web.

La interfaz ICallbackEventHandler define dos métodos. RaiseCallbackEvent () que recibe datos de eventos desde el navegador como un parámetro de cadena, este se activa primero. GetCallbackResult () se activa después, y devuelve el resultado de nuevo a la página.

Dice [Deitel & Deitel 2010] que ASP.NET también incluye un mecanismo conveniente para permitir el comportamiento básico de Ajax, o el Lado de Cliente de devoluciones de llamada, en un control de servidor. Las devoluciones de las llamadas del cliente permiten que se tome ventaja de los componentes XmlHttp que se encuentran en la mayoría de los navegadores modernos para comunicarse con el servidor sin tener que realizar una devolución de datos completa.

Para habilitar las devoluciones de llamada en un control del servidor, se implementa la interfaz System.Web.UI.ICallBackEventHander. Esta interfaz se requiere para implementar los dos métodos: el RaiseCallbackEvent y la GetCallbackResult. Estos eventos del lado del servidor se activan cuando el cliente ejecuta la devolución de llamada.

Después de implementar esta interfaz, se debe de amarrar sus eventos del cliente al servidor. Esto se hace mediante el uso de Page, ClientScript, y el método GetCallbackEventReference. También se debe especificar las dos funciones del lado del

cliente: uno para servir como el manejador de devolución de llamadas y otro para servir como un controlador de errores

La Figura 5 muestra cómo las devoluciones de llamada de cliente funcionan en el marco ASP.NET

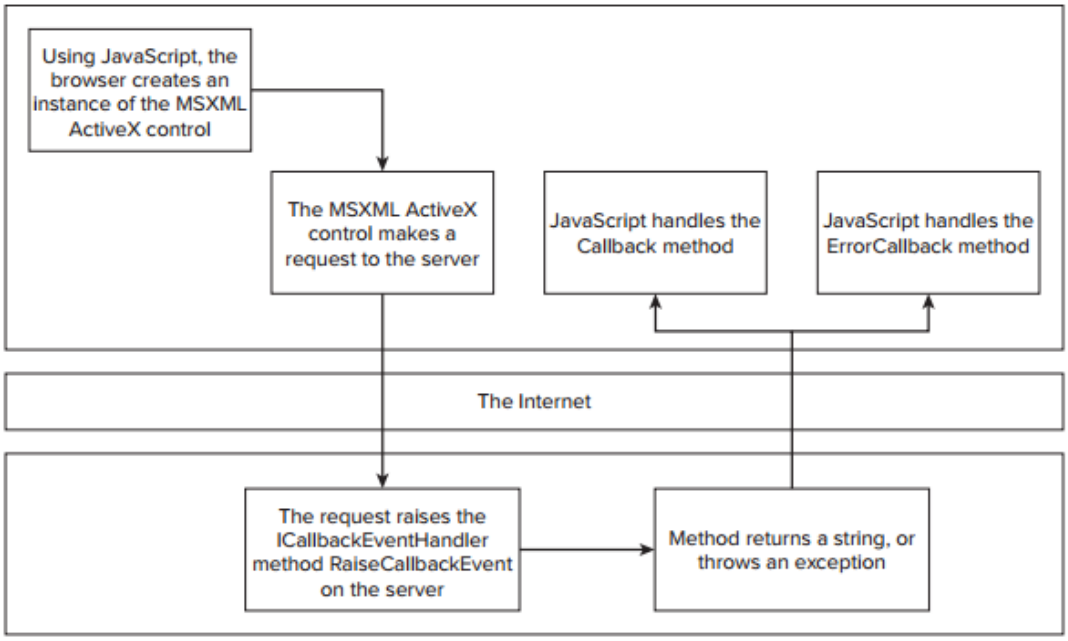

Figura 5: Devoluciones de llamadas en ASP.NET

Elaboración: Obtenido de [Deitel & Deitel 2010]

2.3 Reflection

Dice [Albahari & Albahari 2010] que un programa de C# se compila en un ensamblado que incluye metadatos, código compilado y recursos. La inspección de los metadatos y el código compilado en tiempo de ejecución se denomina Reflection.

El código compilado en un ensamblado contiene casi todo el contenido original del código fuente. En Reflection, parte de la información se pierde, tales como nombres de variables locales, los comentarios, y las sentencias de preprocesador. Sin embargo, Reflection puede acceder a casi todo, incluso por lo que es posible describirlo como un descompilador.

Muchos de los servicios disponibles en .NET y expuestos a través de C # (como unión dinámica, la serialización, el enlace de datos, y Remoting) dependerá de la presencia de metadatos.

Los programas también pueden hacer uso de estos metadatos, e incluso pueden ampliarlo con nueva información usando atributos personalizados. El namespace System.Reflection alberga la API de Reflection. También es posible en tiempo de ejecución crear dinámicamente nuevas instrucciones de metadatos y ejecutables en el IL (Intermediate Language) a través de las clases en el System.Reflection.Emit namespace. En la Figura 6 se muestra la estructura de tipos de miembros a los que Reflection puede acceder.

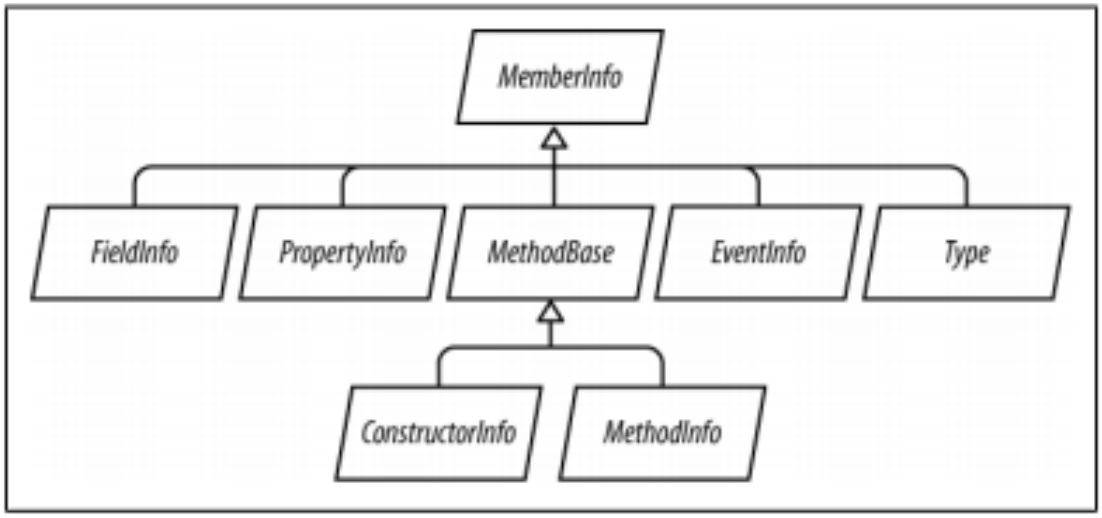

Figura 6: Tipos de Miembros

Elaboración: Obtenido de [Albahari & Albahari 2010]

Dice [Solis, 2012] que la mayoría de los programas se escriben para trabajar en los datos. Los programas leen, escriben, manipulan y visualizan datos. (Los gráficos son una forma de datos).

Los datos sobre los programas y sus clases se llaman metadatos y se almacena en el ensamblador de programas.

Un programa puede mirar los metadatos de otros ensamblados o de sí mismo, mientras que se está ejecutando. Se llama Reflection cuando un programa en ejecución ve sus propios metadatos, o los de otros programas.

Un navegador de objetos es un ejemplo de un programa que muestra los metadatos. Se puede leer y mostrar los conjuntos de los tipos que contienen, junto con todas las características y miembros.

Estos incluyen los tipos predefinidos (int, long, string, etc.), los tipos de la Biblioteca de Clases (consola, IEnumerable, y así sucesivamente), y los tipos definidos por el usuario (MyClass, MyDEL, y así sucesivamente). Cada tipo tiene sus propios miembros y características.

La Biblioteca de Clases declara una clase abstracta llamada Type, que está diseñada para contener las características de un tipo. El uso de objetos de esta clase le permite obtener información sobre los tipos de un programa que se está utilizando.

Como Type es una clase abstracta, no se puede tener instancias reales. En cambio, en tiempo de ejecución, el CLR crea instancias de un tipo de clase derivado de (RuntimeType) que contiene la información de Type. Al acceder a uno de estas instancias.

En la siguiente Figura 7 se muestran las propiedades y métodos de la clase System.Type.

Member	Member Type	Description
`Name`	Property	Returns the name of the type.
`Namespace`	Property	Returns the namespace containing the type declaration.
`Assembly`	Property	Returns the assembly in which the type is declared. If the type is generic, it returns the assembly in which the type is defined.
`GetFields`	Method	Returns a list of the type's fields.
`GetProperties`	Method	Returns a list of the type's properties.
`GetMethods`	Method	Returns a list of the type's methods.

Figura 7: Miembros de la clase System.Type

Elaboración: Obtenido de [Solis, 2012]

2.4 Javascript

Dice [Suehring, 2013] que JavaScript no es Java. Con esa aclaración fuera del camino, se puede pasar a lo siguiente, JavaScript es una implementación de una especificación conocida como ECMAScript.

La historia de JavaScript es importante para entender cómo se implementa el lenguaje en diferentes entornos de hoy.

JavaScript fue desarrollado originalmente por Brendan Eich en Netscape en algún momento de 1995 a 1996. En aquel entonces, el lenguaje se llamaba LiveScript. Ese era un gran nombre para un nuevo lenguaje y la historia podría haber terminado ahí. Sin embargo, en una decisión desafortunada, la gente en la mercantilización se salieron con la suya renombrándolo a JavaScript. En ese momento Java era el nuevo lenguaje emocionante en el momento, y alguien decidió tratar de sacar provecho de la popularidad de Java mediante el uso de su nombre. Como resultado, se encontró JavaScript asociado con el lenguaje Java. Esta era una desventaja para JavaScript, Java, porque aunque era popular en el sentido de que se utiliza con frecuencia, también fue impopular porque se había ganado una reputación bastante mala, utilizaban Java en sitios web para presentar los datos o añadir mejoras inútiles (como el molesto desplazamiento de texto). La experiencia de navegacion del usuario sufrió porque Java requiere un plug-in que se carga en el navegador web, lo que frena el proceso de navegación y causa dolor de cabeza en los visitantes. Sólo en los últimos años JavaScript ha comenzado a separarse de esta asociación negativa con Java. Los programadores nuevos de JavaScript pronto se dieron cuenta de sus fortalezas y utilidad tanto para la simulación y creación de interactividad en la World Wide Web. Los programadores desarrollaron muchos sitios web utilizando sólo simples Hypertext Markup Language (HTML) y con gráficos que a menudo carecían de atractivo visual y la capacidad de interactuar con el contenido del sitio. Con Microsoft Windows 8, JavaScript tiene ahora una vía para la creación de aplicaciones en toda regla.

Los inicios de JavaScript estaban centrados en la validación de formularios del lado del cliente y trabajaba con imágenes en las páginas Web para proporcionar páginas rudimentarias, aunque útil en la interactividad y la retroalimentación para el visitante. Cuando un visitante de una página web llena un formulario, JavaScript valida instantáneamente el contenido en lugar de hacer una ida y vuelta al servidor. Esto fue una gran manera de ayudar a las aplicaciones haciéndolas un poco más rápidas y más sensibles.

Múltiples estándares: Si uno piensa que los estándares de programación de JavaScript se definen vagamente, tiene razón. Cada navegador es compatible con JavaScript de una forma ligeramente diferente, haciendo su trabajo y el trabajo de los demás mucho más difíciles.

Tratar de escribir sobre todos estos matices es más difícil que escribir sobre un lenguaje que se implementa por una sola entidad específica, como una cierta versión de Microsoft Visual Basic o Perl. El trabajo es hacer un seguimiento de estas diferencias y darse cuenta de ellos cuando sea necesario, y tratar de encontrar un terreno común entre ellos tanto como sea posible.

El DOM: Otro estándar en evolución pertinente para el programador JavaScript es el Document Object Model (DOM) estándar desarrollado por el World Wide Web Consortium (W3C). El W3C define el DOM como "una interfaz de la plataforma y de lenguaje neutro que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, estructura y estilo de los documentos. ¿Qué significa esto? es que se puede trabajar con una especificación de navegadores web que se adhieren a desarrollar una página web de una manera dinámica. El DOM crea una estructura de árbol de objetos para documentos HTML y Extensible Markup Language (XML) y permite a los scripts de los objetos JavaScript interactuar fuertemente con el DOM para muchas funciones importantes.

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo y dinámico.

Dice [Freeman & Robson 2014] que JavaScript es bastante único en el mundo de la programación. Con un lenguaje de programación típico se tiene que codificar, compilar,

enlazar y desplegar. JavaScript es mucho más fluido y flexible. Con JavaScript todo lo que se necesita hacer es escribir código JavaScript, y luego cargarlo en un navegador. A partir de ahí, el navegador estará feliz de comenzar a ejecutar el código. A continuación, se muestra más de cerca cómo funciona esto:

Codificar: Al igual que HTML y CSS, se puede poner todo junto en un solo archivo, o se puede colocar JavaScript en su propio archivo, que se incluirá como referencia en la página.

Carga: Cuando el navegador ve el código, comienza analizarlo de inmediato, hasta tenerlo a punto para ejecutarlo. Tener en cuenta que al igual que HTML y CSS, si el navegador ve errores en el código, hará todo lo posible para mantener el movimiento y la lectura de JavaScript, HTML y CSS. La última cosa que el navegador desea es que el usuario no sea capaz de ver la página.

Ejecución: El navegador comienza a ejecutar el código tan pronto como se lo encuentra en la página, y continúa ejecutándolo hasta el cierre de la misma. A diferencia de las primeras versiones de JavaScript, el JavaScript de hoy es más potente, utiliza técnicas de compilación avanzadas para ejecutar el código casi a la misma velocidad que muchos lenguajes de programación nativos.

2.5 JQuery

Dice [Sawyer McFarland 2012] que una biblioteca de JavaScript es una colección de código JavaScript que ofrece soluciones sencillas a muchos de los detalles del día a día de JavaScript. Se piensa en ello como una colección de funciones JavaScript pre-escritas que se agregan a las páginas webs. Estas funciones hacen que sea fácil completar tareas comunes. En muchos casos, se puede sustituir muchas líneas de su propia programación JavaScript (y las horas requeridas para probarlos) con una sola función de una biblioteca de JavaScript. Hay un montón de librerías JavaScript por ahí, y muchos de ellas ayudan a crear grandes sitios web como Yahoo, Amazon, CNN, Apple, y Twitter. JQuery tiene muchas ventajas:

Es relativamente pequeño en tamaño. Una versión comprimida de la biblioteca pesa alrededor de 90 kb.

Se aceptan a los diseñadores web. jQuery no asume que eres un científico de la computación. Aprovecha los conocimientos de CSS que la mayoría de los diseñadores web ya tienen.

jQuery se utiliza en millones de sitios, entre ellos muchos populares, sitios web como Digg, Dell, y NBC. Incluso Google lo utiliza en algunos lugares, y está construido en el número uno de software de blogs, WordPress. El hecho de que jQuery es tan popular es un testimonio de lo bueno que es.

Ampliación de la comunidad de desarrolladores. Al leer esto, decenas de personas están trabajando en el proyecto de escritura de código de jQuery, corrigiendo errores, añadiendo nuevas características, y la actualización de la página web con documentación y tutoriales. Una biblioteca JavaScript creado por un solo programador (o suministrada por un solo autor) puede desaparecer fácilmente si el programador (o autor) se cansa del proyecto.

Según [Chaffer & Swedberg 2013] Con el resurgimiento del interés en HTML dinámico viene una proliferación de JavaScript. JQuery emplea varias estrategias:

Apalancamiento conocimiento de CSS: Al basar el mecanismo para la localización elementos de la página de selectores CSS, jQuery hereda una forma todavía legible concisa de expresar la estructura de un documento. La biblioteca jQuery se convierte en una entrada punto para los diseñadores que quieren añadir comportamientos a sus páginas, porque un requisito previo para hacer el desarrollo web profesional es el conocimiento de Sintaxis CSS.

Extensiones de apoyo: El método para crear nuevos plugins es simple y bien documentado, esto ha estimulado el desarrollo de una amplia variedad de módulos útiles. Incluso la mayoría de las características de la base de jQuery se realiza internamente a través de la arquitectura de plugin, y se puede quitar si se desea, produciendo una biblioteca aún más pequeña.

Peculiaridades del navegador: Una triste realidad de desarrollo web es que cada navegador tiene su propio conjunto de desviaciones de los estándares publicados. A parte importante de cualquier aplicación web puede ser relegado a funciones de gestión de manera diferente en cada plataforma. Mientras que el paisaje siempre cambiante navegador hace una base de código perfectamente navegador neutral imposible para algunas avanzadas características, jQuery añade una capa de abstracción que normaliza las tareas comunes.

Permitir varias acciones en una línea: para evitar el uso excesivo de variables temporales, jQuery emplea un modelo de programación Esto significa que el resultado de la mayoría de las operaciones en un objeto es el objeto en sí mismo, listo para la siguiente acción para ser aplicado a la misma.

2.6 XML

Dice [Fawcett, Ayers, & Quin 2012] XML significa Extensible Markup Language (presumiblemente los autores originales pensaron que sonaba más emocionante que EML) y su desarrollo y la utilización de haber seguido un camino común en el software y el mundo de las TI. Comenzó hace más de diez años y fue utilizado originalmente por muy pocos; más tarde comenzó a invadir el mundo de intercambio de datos. Posteriormente, las herramientas disponibles para procesar y gestionar XML se hicieron más sofisticadas, hasta el punto de que muchas personas comenzaron a usarla sin ser realmente conscientes de su existencia. Últimamente ha habido un poco de un retroceso en algunos sectores sobre sus fallas percibidas y los puntos débiles, lo que ha dado lugar a diversas alternativas y propuestas de mejora.

Sin embargo, XML ahora tiene un lugar permanente en los sistemas de TI y es difícil imaginar cualquier aplicación no trivial que no utiliza XML, ya sea para su configuración o los datos en algún grado. Por esta razón es esencial que los desarrolladores de software modernos tengan un conocimiento profundo de sus principios, lo que es capaz de hacer, y cómo utilizarlo para su mejor ventaja.

Hay dos usos principales para XML: Una de ellas es una forma de representar los datos de bajo nivel, para los archivos de configuración. La segunda es una forma de añadir metadatos a los documentos; por ejemplo, es posible que se desee hacer hincapié en una frase particular, en un informe al ponerlo en cursiva o negrita.

El primer uso de XML está pensado como un reemplazo de las formas más tradicionales de esta se ha hecho antes, por lo general por medio de listas de pares nombre / valor, como se ve en los archivos INI. La segunda aplicación de XML es similar a cómo funcionan los archivos HTML. El texto está contenido en un recipiente general. Para ambos escenarios se ha producido una multiplicidad de técnicas elaboradas en los últimos años. El problema con estos enfoques dispares ha sido más evidente que nunca, ya que el aumento del uso de Internet y una amplia existencia de aplicaciones distribuidas, en particular los que se basan en componentes diseñados y gestionados por diferentes partes.

Es ciertamente posible diseñar un sistema distribuido que tiene dos componentes, uno como salida de datos utilizando un archivo INI de Windows y el otro, que lo convierte en un formato de Propiedades Java. Desafortunadamente, esto significa un montón de desarrollo en ambos lados, que no deben ser necesarias y realmente detrae recursos del objetivo principal, el desarrollo de nuevas funcionalidades que proporciona valor de negocio.

XML fue concebido como una solución a este tipo de problemas; está destinado a hacer que los datos pasen entre los diferentes componentes mucho más fácil y aliviar la necesidad de preocuparse continuamente sobre los diferentes formatos de entrada y salida, lo que lleva a los desarrolladores a concentrarse en los aspectos más importantes de la codificación, como la lógica de negocio. XML también es visto como una solución a la cuestión de si los archivos deben ser fácilmente legibles por software o por los seres humanos; El objetivo de XML consiste en ser ambos a la vez. En la Figura 8 se muestra un ejemplo del código XML.

Figura 8: Ejemplo de XML

Elaboración: Obtenido de [Fawcett, Ayers, & Quin 2012]

[Deitel & Deitel 2010] dice que XML permite a los desarrolladores crear totalmente nuevos lenguajes de marcado para describir cualquier tipo de datos, como fórmulas matemáticas, instrucciones de software de configuración, estructuras moleculares químicas, música, noticias, recetas e informes financieros.

La próxima generación de la web se está construyendo sobre una base XML, lo que le permite desarrollar aplicaciones más sofisticadas basadas en web.

XML permite a los desarrolladores crear marcas (es decir, una anotación de texto para la descripción de datos) para prácticamente cualquier tipo de información.

XML describe los datos de una manera que tanto los seres humanos y las computadoras pueden entender.

XML permite asignar significado a los datos

Es importante darse cuenta de que XML no es realmente un "lenguaje" en absoluto, sino un estándar para la creación de lenguas que cumplan con los criterios, en otras palabras, XML describe una sintaxis que se utiliza para crear sus propios idiomas. Por ejemplo, supongamos disponer de datos acerca de un

nombre, y desea ser capaz de compartir esa información con los demás, así como el uso de la información en un programa de ordenador

Cada etiqueta de comienzo debe tener una etiqueta final que coincida, o sea una etiqueta de cierre automático.

Las etiquetas no pueden solaparse; los elementos deben estar anidados correctamente.

Los documentos XML pueden tener sólo un elemento raíz.

Los nombres de elementos deben obedecer las convenciones de nombres XML.

XML distingue entre mayúsculas y minúsculas.

2.7 JSON

Según [Deitel & Deitel 2010] JSON (JavaScript Object Notation): es un modo sencillo para representar objetos JavaScript como marcos y es una alternativa a XML como una técnica de intercambio de datos.

JSON se ha ganado la aclamación debido a su formato sencillo, haciendo objetos fáciles de leer, crear y analizar.

JSON ofrece una forma sencilla de manipular objetos en JavaScript, y muchos otros lenguajes de programación ahora soportan este formato. Además de simplificar la creación de datos.

JSON permite que los programas puedan manipular los datos con facilidad y transmiten datos de manera más eficiente a través de Internet.

Cada objeto JSON se representa con una lista de nombres y valores de propiedad que figuran entre llaves, en el siguiente formato:

{propertyName1: value1, propertyName2: value2}

Las matrices se representan en JSON con corchetes en el siguiente formato:

[value1, value2, value3]

Cada valor puede ser una cadena, un número, un objeto JSON puede ser verdadero, falso o nulo. Para apreciar la simplicidad de los datos JSON, usted puede examinar esta representación de una serie de entradas de una libreta de direcciones:

[{ first: 'Cheryl', last: 'Black'},

{ first: 'James', last: 'Blue'},

{ first: 'Mike', last: 'Brown'},

{ first: 'Meg', last: 'Gold'}]

Dice [Sawyer McFarland D. 2012] Otro formato popular para enviar datos desde el servidor se llama JSON, que significa JavaScript Object Notation. JSON es un formato de datos que está escrito en JavaScript, y que es algo así como un formato XML, en el que se trata de un método para el intercambio de datos.

Se puede crear colecciones aún más complejas de información mediante el uso de objetos literales como los valores dentro de un objeto JSON, en otras palabras, los literales de objetos anidados dentro objetos literales.

JSON es generalmente mejor que XML; Sin embargo, desde que JSON es JavaScript esto funciona rápido y fácilmente en el programa JavaScript.

XML necesita ser tomado aparte por JavaScript, que es generalmente más lento y requiere de mucha programación.

JSON es más prescriptivo acerca de sus requisitos de sintaxis y más restrictiva acerca de los valores que permite.

JSON especifica que todas las claves de objeto, así como todos los valores de cadena deben ser entre comillas dobles.

Dice [Ferguson & Heilmann, 2013] en lugar de leer un archivo XML como XML y analizarlo a través del DOM o leer como texto y el uso de expresiones regulares, sería mucho más fácil y tendría menos esfuerzo que el sistema tenga los datos en un formato que JavaScript pueda utilizar directamente. Este formato se llama JSON. Esto permite que un conjunto de datos que debe expresarse en una notación literal de objeto.

El formato JSON se introdujo como un reemplazo para el formato XML debido a que no se requiere la extensibilidad y la verbosidad de XML y por lo tanto para levantar el consumo complejo de recursos de procesamiento de XML para permitir que los dispositivos más pequeños consuman flujos de datos o paquetes de datos producidos por diferentes servicios que necesitan para interactuar.

Los desarrolladores pueden manipular documentos JSON en una manera estándar similar al procesamiento de XML.

La API de JSON ofrece dos métodos de análisis para analizar documentos JSON, los mismos dos modelos que están disponibles para analizar documentos XML. En la Figura 9 se muestra una comparación entre XML y JSON.

```
<albums>
  <album>
    <id>1</id>
    <artist>Depeche Mode</artist>
    <title>Playing the Angel</title>
    <comment>They are back and finally up to speed again</commer
 </album>
  <album>
    <id>2</id>
    <artist>Monty Python</artist>
    <title>The final Rip-Off</title>
    <comment>Double CD with all the songs</comment>
  </album>
  <album>
    <id>3</id>
    <artist>Ms Kittin</artist>
    <title>I.com</title>
    <comment>Good electronica</comment>
  </album>
</albums>
```

```
{
  "album":
   [
    {
      "id" : "1",
      "artist" : "Depeche Mode",
      "title" : "Playing the Angel",
      "comment" : "They are back and finally up to speed again"
    },
    {
      "id" : "2",
      "artist" : "Monty Python",
      "title" : "The final Rip-Off",
      "comment" : "Double CD wiid all the songs"
    },
    {
      "id" : "3",
      "artist" : "Ms Kittin",
      "title" : "I.com",
      "comment" : "Good electronica"
    }
   ]
}
```

Figura 9: XML vs JSON

Elaboración: Obtenido de [Ferguson & Heilmann, 2013]

2.8 AJAX

Según [Ferguson & Heilmann, 2013] AJAX, Asynchronous JavaScript y XML, un término que fue acuñado por Jesse James Garrett en Adaptive Path en febrero de 2005. En él se describe una metodología de desarrollo de aplicaciones web que es diferente de la tradicional. Las aplicaciones web tradicionales y sitios funcionan de forma sincrónica cada vez que se sigue un enlace o se envía un formulario, el navegador envía los datos al servidor, el servidor (con suerte) responde, y toda la página se actualiza.

Las aplicaciones AJAX funcionan de forma asíncrona, lo que significa que se envían datos de ida y vuelta entre el navegador del usuario y el servidor sin tener que recargar toda la página. AJAX Reemplaza sólo las partes de la página que cambia. AJAX también puede enviar varias solicitudes y continuar el desplazamiento y el uso de la página mientras se cargan las otras partes en el fondo. Una buena comparación es que Ajax es a páginas web tradicionales lo que la mensajería instantánea es al correo electrónico-mensajes: retroalimentación inmediata, sin largos tiempos de espera y con más opciones para comunicarse. En la Figura 10 se muestra una comparación entre AJAX y las llamadas tradicionales.

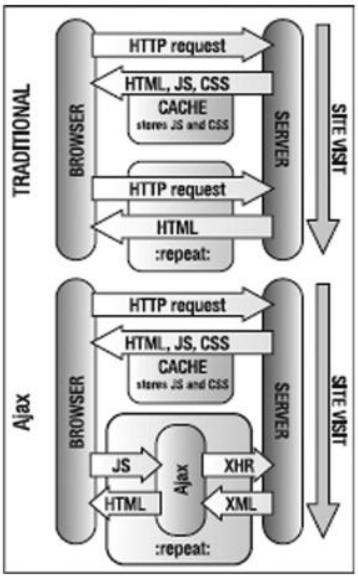

Figura 10: AJAX vs Llamadas Tradicionales

Elaboración: Obtenido de [Ferguson & Heilmann, 2013]

Según [Sawyer McFarland, 2012] JavaScript es grande, pero no puede hacerlo todo. Si se desea mostrar la información de una base de datos, garabatear un correo electrónico con los resultados de un formulario, o simplemente descargar HTML adicional, tiene que comunicarse con un servidor web. Para estas tareas, por lo general, tiene que cargar una nueva página web. Por ejemplo, cuando se busca una base de datos para obtener información, por lo general, sale de la página de búsqueda y se va a otra página de resultados.

Por supuesto, la carga de nuevas páginas requiere tiempo. Y, si se piensa bien, el concepto de desaparecer y luego reaparecer es bastante extraño en general. Es como si se estuviera usando Microsoft Word y cada vez que se abre un nuevo archivo de los menús del programa, los paneles y las ventanas desaparecen de repente y luego reaparecen cuando el nuevo archivo se abre. Sitios como Facebook, Twitter, Google Maps y Gmail están difuminando la línea entre los sitios web y programas de computadora de escritorio. En todo caso, las personas quieren sitios web más rápido y más sensibles, como los programas de escritorio. La tecnología que hace posible esta nueva generación de aplicaciones web, es una tecnología de programación llamada AJAX.

Hay muchas cosas simples que se puede hacer con las tecnologías Ajax, como por ejemplo:

Visualizar nuevo contenido HTML sin necesidad de recargar la página: Por ejemplo, en una página que muestra titulares de noticias y muestra el artículo, cuando un visitante hace clic en un titular, puede salvarle la espera tediosa para que una nueva página se cargue. La noticia podría aparecer justo en la misma página web, sin que esta se recargue.

Presentar un formulario y mostrar los resultados al instante: imaginar un "suscribirse a nuestro boletín de noticias"; cuando alguien rellena y envía el formulario, el formulario desaparece y aparece un mensaje como "has accedido a nuestro boletín" inmediatamente.

Login sin salir de la página: Aquí hay otra forma relacionada uso de JavaScript a una página con un pequeño formulario de "login". El procedimiento seria rellenar el formulario, pulsar el botón "Entrar", y que no sólo muestre el mensaje de estar conectado, sino que la

página se transforme para mostrar el estado de inicio de sesión, nombre de usuario, y tal vez otra información específica.

Obtener calificaciones: En los sitios que listan los libros, películas y otros productos, a menudo se ve una estrella de clasificación-por lo general de 1 a 5 estrellas para indicar cómo los visitantes han valorado la calidad del artículo. Estos sistemas de clasificación por lo general permiten expresar la opinión haciendo clic en un número de estrellas. Usando Ajax, se puede dejar que los visitantes emiten votos sin salir de la web, todo lo que tienen que hacer es clic en las estrellas.

CAPÍTULO III: ESTADO DEL ARTE

3.1 A Study and Toolkit for Asynchronous Programming in C#

Dice [Okur, Hartveld, & Arie van 2014] que la programación asíncrona es demandada hoy en día porque la respuesta es cada vez más importante en todos los dispositivos modernos: escritorio, móvil o aplicaciones web. Por lo tanto, los principales lenguajes de programación tienen APIs que soportan sin bloquear las operaciones asíncronas (por ejemplo, para acceder a la web, o para operaciones de archivo). Si bien estas API hacen posible la programación asíncrona, no hacen que sea fácil.

Las API asíncronas se basan en las devoluciones de llamada. Sin embargo, las devoluciones de llamada invierten el flujo de control. Recientemente, los idiomas principales (F #, C # y Visual Basic) introdujeron construcciones asíncronas que se asemejan al estilo de codificación directa de código síncrono tradicional. Por lo tanto, reconocen la programación asíncrona como un ciudadano de primera clase.

Sabemos poco sobre cómo los desarrolladores utilizan la programación asíncrona y como especifican las nuevas construcciones asíncronas en la práctica. Sin esos conocimientos, otros desarrolladores no pueden educarse a sí mismos.

Actualmente, muchos desarrolladores hacen el mal uso de los métodos async y await, debido a que son herramientas relativamente nuevas, a través de mucha investigación llegan a definir el mal uso como anti-patrones que perjudican el rendimiento y pueden causar problemas graves en los sistemas actuales que se están desarrollando.

Este artículo se centra en las aplicaciones de Windows Phone porque se espera encontrar muchos ejemplos de programación asíncrona, dado que la capacidad de respuesta es crítica. Las aplicaciones móviles pueden fácilmente ser insensibles ya que los dispositivos móviles tienen recursos limitados y tienen una alta latencia (excesivo de la red de accesos). Con la

inmediatez de las interfaces de usuario táctiles, incluso pequeños contratiempos en la capacidad de respuesta son más obvia y chocante que cuando se utiliza un ratón o teclado.

Utilidad para el proyecto de tesis

Gracias a este artículo, se ha tenido sumo cuidado en no implementar código que lejos de ayudar a la optimización, empeoran el rendimiento. Como señala el artículo la implementación de llamadas asíncronas no es fácil por lo tanto se ha tratado de conseguir una notación que ya se viene usando en jQuery por ejemplo.

3.2 Toward a Mobile Rich Web Application – Mobile AJAX and Mobile Web 2.0

Dice [Jeon & Lee 2009] que a medida que la Web de próxima generación se ha vuelto más madura, las aplicaciones que ofrecen respuesta a las interfaces de usuario y capacidades interactivas, se han vuelto cada vez más populares. Las capacidades representan una manera de hacer que los programas sean más fáciles de usar y más funcional, mejorando así la experiencia de usuario.

Ha habido un cambio en la dirección del desarrollo web. Una nueva generación de aplicaciones web, llamado AJAX (Asynchronous JavaScript And XML), está surgiendo en respuesta al limitada grado de interactividad de grande interacciones Web. En el corazón de este nuevo enfoque se encuentra un modelo de interfaz de una sola página que facilita la interactividad. En este modelo, se realizan cambios en los componentes de interfaz de usuarios individuales contenidas en una página web, en lugar de actualizar la página entera.

AJAX es una técnica de desarrollo web que se utiliza para crear aplicaciones web interactivas. La intención es hacer que las páginas web se sientan más sensibles mediante el intercambio de pequeñas cantidades de datos con el servidor, de modo que toda la página web no tiene que ser recargada cada vez que el usuario solicita un cambio. Con ello se pretende aumentar la interactividad de la página web, velocidad, funcionalidad y facilidad de uso.

La mayor parte de los componentes de AJAX fueron desarrollados y estandarizados durante los últimos 10 años. Estas tecnologías han mejorado recientemente, haciéndolos más adecuados para el uso empresarial.

AJAX utiliza una combinación de:

HTML y CSS, para el diseño que acompaña la información.

El DOM se accede con un lenguaje de script del lado del cliente, especialmente con implementaciones de ECMAScript como JavaScript y JScript, para mostrar de forma dinámica e interactuar con la información presentada.

El objeto XMLHttpRequest se utiliza para intercambiar datos de forma asíncrona con el servidor web. En algunos marcos AJAX y en ciertas situaciones, un objeto IFrame se utiliza en lugar del objeto XMLHttpRequest para intercambiar datos con el servidor web, y en otras implementaciones, agregar dinámicamente etiquetas <script>.

XML se utiliza a veces como el formato para transferir data entre el servidor y el cliente, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano y JSON. Estos archivos pueden ser creados dinámicamente por alguna forma de scripting del lado del servidor.

Las aplicaciones Web basadas en navegador convencionales requieren que el usuario envíe una solicitud al servidor, espera a que el servidor procese la solicitud y genere una respuesta, y luego espera a que el navegador actualice la interfaz con los resultados. Este patrón de esperar respuesta es extremadamente perjudicial y reduce la productividad.

Asynchronous JavaScript y XML es una técnica de programación basada en estándares diseñados para hacer que las aplicaciones basadas en Web sean más interactivas y personalizables. AJAX ofrece muchas ventajas sobre los enfoques convencionales para el desarrollo de aplicaciones Web, pero también tiene varias deficiencias que han frenado la adopción generalizada.

Ventajas: Las principales ventajas de Web con estilo AJAX producen menos tiempo de espera y más control para el usuario. AJAX logra esto

La eliminación completa post-backs

Aprovechando la potencia de procesamiento de la máquina cliente y la proximidad temporal al hacer el navegador Web responsable de más aspectos de la ejecución de la aplicación;

La explotación de ricos gráficos con capacidades de transparencia, sombras, animación, composición, añadir más brillo y la interactividad a la presentación de la información.

Separación de los datos, el formato, el estilo y la función, el enfoque AJAX puede tender a alentar a los programadores claramente a separar los métodos y formatos utilizados para los diferentes aspectos de la entrega de información a través de la Web.

Desventajas: Otra cuestión es sobre Interoperabilidad. AJAX se basa en JavaScript, que se implementa a menudo de manera diferente por diferentes navegadores o versiones de un navegador en particular. Debido a esto, los sitios que utilizan JavaScript pueden necesitar ser probado en varios navegadores para comprobar si hay problemas de compatibilidad.

Otro problema de AJAX es cómo se actualiza la interfaz de la aplicación. Cuando se producen cambios a la interfaz, puede que no sea aparentemente visual que se haya producido un cambio. El problema es aún más grave para los usuarios lectores de pantalla. Cuando ocurren cambios en la interfaz, el lector de pantalla del usuario puede que no esté al tanto de los cambios y los nuevos contenidos pueden no leerse.

Utilidad para el proyecto de tesis

Gracias a este artículo, se toma la decisión de realizar el envío de datos al servidor mediante JSON, y se toma en cuenta los problemas de compatibilidad entre los navegadores para la librería creada.

3.3 An Intelligent Web Browser Plug-In for Automatic Translation to Ajax Approach

Dice [Hanakawa 2013] que los sitios web son medios importantes de recopilar información en la investigación, el aprendizaje, e incluso las actividades comerciales. La importancia de los navegadores web crece cada vez más debido a que los sitios web están en crecimiento. Los navegadores Web son herramientas poderosas para referirse a los sitios web.

Sin embargo, debido a la necesidad de la comunicación síncrona con servidores web, La operatividad de las páginas web en el navegador web no es mejor que la operatividad de las aplicaciones de escritorio. Por lo tanto, se propone un nuevo navegador web inteligente plug-in para el acercamiento de Ajax. El navegador plug-in puede mejorar la operatividad como lo mismo que las aplicaciones de escritorio sin revisar códigos de programa de aplicaciones web. Una característica del navegador plug-in es la comunicación asíncrona con los servidores web.

Recientemente, se ha recibido valiosos beneficios de gran cantidad de sitios web en Internet. Mucha información se proporciona a través de sitios web. Por otra parte, los sitios web son útiles en los casos no sólo en la búsqueda de información, sino también a las actividades comerciales, tales como las compras online. Incluso la educación en las universidades y las escuelas no puede ignorar la información de los sitios web. Los sitios web son los recursos de información indispensables en varias escenas en las sociedades actuales. En otras manos, la mayoría de las herramientas habituales para referirse a los sitios web son los navegadores como Internet Explorer (IE) y Firefox. Los Navegadores Web generan una página web visible mientras que los navegadores están analizando los códigos fuente HTML. Este es un sistema cliente-servidor típico llamado "aplicación web". Todo el mundo puede acceder a las aplicaciones web fáciles a través de navegadores web.

Sin embargo, tenemos problemas con las aplicaciones web sobre los navegadores web. La operatividad de las aplicaciones web sobre los navegadores web es menor que la operatividad de las aplicaciones de escritorio. El rendimiento lento e interactividad limitada de operatividad en la aplicación web. Las aplicaciones de escritorio se instalan en un

ordenador de usuario. Las aplicaciones de escritorio son capaces de lograr respuestas rápidas a las solicitudes de los usuarios ya que algunas aplicaciones de escritorio no necesitan la comunicación con los servidores de otros equipos.

Para resolver la debilidad de la operatividad de las aplicaciones web, la tecnología Ajax (Asynchronous JavaScript + XML) se adoptaron para las aplicaciones web. Si una aplicación web se construye en base a la tecnología Ajax, Las recarga total de toda la página se puede evitar porque los programas de cliente (por ejemplo, JavaScript) en código fuente HTML se comunica de forma asíncrona con los servidores web.

Los programas cliente en HTML no necesitan esperar para finalizar la comunicación con los servidores.

Debido a la tecnología Ajax que puede mejorar la operatividad de las aplicaciones web, muchos desarrolladores adoptan la tecnología Ajax para sus proyectos. Una aplicación web más famoso es el Mapa de Google. La aplicación web de Mapa de Google evita la recarga total de toda la página debido a la comunicación asíncrona con los servidores Web.

Aunque el enfoque Ajax es una mejor manera de mejorar la operatividad de las aplicaciones web, no es un problema significativo. Por lo general, los desarrolladores tienen que cambiar los códigos de los programas de software cliente y el software de servidor de las aplicaciones web. Los usuarios finales no pueden recibir beneficios a partir del enfoque Ajax en las aplicaciones web existentes a menos que los desarrolladores modifiquen el código de los programas actuales de las aplicaciones web. Los usuarios finales sólo hacen nada más que esperar a la modificación de los códigos de los programas de las aplicaciones web existentes. Los beneficios de Ajax dependen de los avances de la codificación de los programas, los usuarios tienen que esperar a veces mucho tiempo para recibir los beneficios del enfoque de Ajax en las aplicaciones web actuales.

Por lo tanto, en esta investigación proponen un nuevo navegador web inteligente “plug-in” basado en un modelo de comunicación asíncrona. El navegador “plug-in” es compatible con la comunicación asíncrona con los equipos del servidor web como el enfoque de Ajax. Si los usuarios acceden a las aplicaciones web usando el navegador, los usuarios podrán

recibir los beneficios de Ajax, incluso si las aplicaciones web no adoptan el acercamiento de Ajax. El navegador tiene una función de traducción de código automáticamente de JavaScript que se puede comunicar de forma asíncrona con los servidores web. La función puede mejorar la operatividad de las aplicaciones web que no adoptan el acercamiento de Ajax.

El navegador plug-in se basa en un modelo de comunicación asíncrona. El modelo puede comportarse con diversas operaciones llevadas a cabo por el enfoque de Ajax. Una característica más importante del modelo es que no se produce ningún cambio de los códigos de cliente y programas del servidor de las aplicaciones web. Debido a que el modelo está integrado en un navegador web, los desarrolladores de aplicaciones web no tienen que revisar las aplicaciones web actuales. Adicionalmente, el modelo se puede comunicar de forma asíncrona con los servidores web con el fin de lograr una suave operatividad de aplicaciones web en los navegadores web. Los pasos típicos son los siguientes;

(0) Selección Plug-in: Los usuarios seleccionan un plug-in favorito de los plug-ins soportados de un navegador web.

(1) Procesamiento del Plug-in: El programa del seleccionado Plug-in es procesado en el navegador.

(2) Generación de códigos de JavaScript: Por procesamiento del Plug-in, unos nuevos códigos JavaScript son generados. Los códigos de JavaScript pueden lograr las funciones del plug-in en las páginas web específicas. Los códigos de JavaScript se incrustan automáticamente a los códigos fuente HTML originales de las páginas web específicas.

(3) Solicitud: los códigos de JavaScript incrustados pueden solicitar cualquier función del servidor web. Las solicitudes se producen por las operaciones del usuario, o solicitudes automáticas en el plug-in.

(4) Procesamiento en servidor web: Correspondiente a la solicitud de los códigos de JavaScript, el servidor web está procesando normalmente.

(5) Respuesta: Los resultados del procesamiento en el servidor web se devuelven al navegador web.

(6) Los datos de archivo: Los resultados devueltos desde el servidor Web se almacenan en los datos de la zona. Es decir, la respuesta del servidor no refleja inmediatamente a la página web. A la vez, la respuesta se acumula en el área de valores.

(7) Acceso independiente: El código incrustado JavaScript accede a la zona de datos de valores. Debido a que el acceso es independiente del tiempo de respuesta del servidor web, la página web puede comportarse como el enfoque de Ajax. Es decir, la página web no tiene que esperar para una respuesta desde el servidor web.

(8) Pantalla: Por la ejecución de los códigos incrustados de JavaScript, los usuarios pueden operar sin problemas la página web como las aplicaciones Web que soportan Ajax.

En la Figura 11 se pueden apreciar los pasos típicos mencionados.

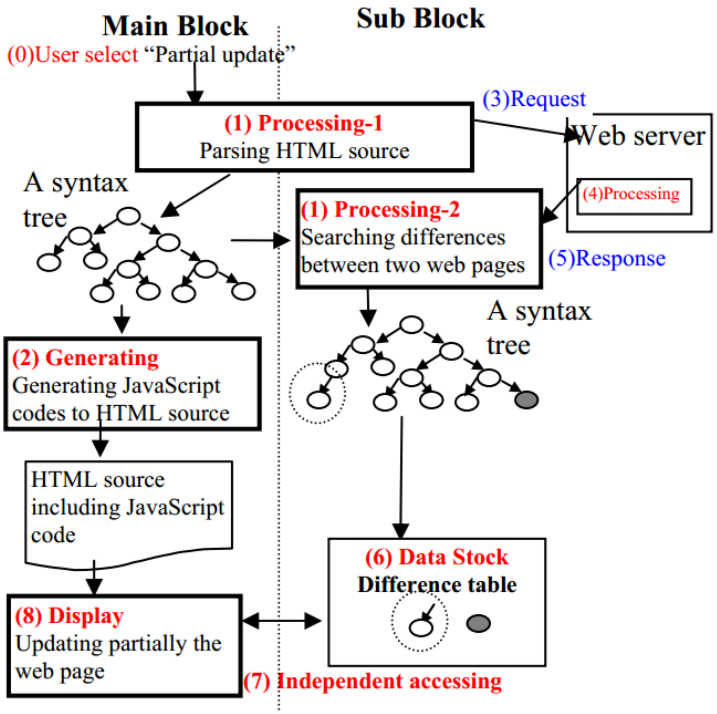

Figura 11: Ejemplo del modelo de actualización parcial

Elaboración: Obtenido de [Hanakawa 2013]

Utilidad para el proyecto de tesis

Gracias a este artículo, usaremos un método similar de comunicación asíncrona. Además este articulo ayuda a identificar los problemas que ocurren en la mayoría de aplicaciones web, y como se pueden superar esos límites creando una nueva librería que haga más eficiente las aplicaciones web actuales.

3.4 Llamadas asíncronas usando ICallbackEventHandler

En el modelo predeterminado para las páginas Web ASP.NET, el usuario interactúa con una página y hace click en un botón o realiza alguna otra acción que da como resultado una devolución de datos. La página y sus controles se vuelven a crear, se ejecuta el código de la página en el servidor y se representa una nueva versión de la página en el explorador. Sin embargo, en algunas situaciones, es útil ejecutar código del servidor desde el cliente sin realizar una devolución de datos. Si el script de cliente de la página incluye información de estado (por ejemplo, valores de variables locales), enviar la página y obtener una nueva copia ocasiona la destrucción de dicho estado. Además, las devoluciones de datos de página producen una sobrecarga de procesamiento que puede disminuir el rendimiento y obligar al usuario a esperar a que la página se procese y se vuelva a crear.

La siguiente aplicación tiene como fin que el servidor realice la conversión a dólares de un valor numérico, que está en soles, utilizando ICallbackEventHandler.

En la Figura 12 se muestra el diagrama de casos de uso del ejemplo que se realiza.

Figura 12: Diagrama del CUS del Ejemplo – Convertir a dólares

Fuente: Elaboración Propia, 2014

Diseñando la interfaz de usuario

Se coloca un campo donde se podrá ingresar el valor numérico, es decir la cantidad de soles que se desea cambiar a dólares. Para esto se utiliza la etiqueta input y un botón el cual será el desencadenante del evento que produzca el cambio de moneda.

En la Tabla 1 se ve el código necesario para crear la interfaz gráfica del ejemplo.

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <span>Moneda:</span>

        <input id="Text1" type="text" />S/.

        <input   id="Button1"   type="button"   value="Cambiar   a
dolares" />

    </form>

</body></html>
```

Tabla 1: Código de la interfaz para implementar ICallbackEventHandler

Fuente: Elaboración Propia, 2013

En la Figura 13 se aprecia la interfaz gráfica del ejemplo.

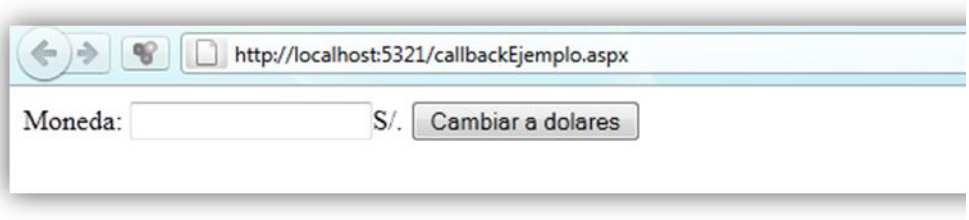

Figura 13: Código de la Interfaz para Implementar ICallbackEventHandler

Fuente: Elaboración Propia, 2013

Preparando el código .cs

Lo primero que se debe hacer es implementar la interface "ICallbackEventHandler" para hacer esto, basta con colocar una coma luego de "System.Web.UI.Page" y colocar el nombre de la interface, como se muestra en la Tabla 2.

```
public partial class callbackEjemplo : System.Web.UI.Page,
ICallbackEventHandler

    {
```

Tabla 2: Implementando la Interface ICallbackEventHandler

Fuente: Elaboración Propia, 2013

Lo siguiente es colocar en el page load el código que permita reconocer los scripts, un encargado de enviar la información al servidor y el otro que tiene la función de recibir la información del servidor. En la Tabla 3 se muestra el código necesario.

```
protected void Page_Load(object sender, EventArgs e)

        {

            String cbReference =

            Page.ClientScript.GetCallbackEventReference(this,

            "arg", "script_ladoCliente", "context");



            String callbackScript;

            callbackScript   =   "function   script_   llamaServidor   (arg,
context)" +

                "{" + cbReference + ";}";

            Page.ClientScript.RegisterClientScriptBlock(this.GetType(),

                "script_llamaServidor", callbackScript, true);        }
```

Tabla 3: Page Load para Implementar ICallbackEventHandler

Fuente: Elaboración Propia, 2013

Lo siguiente es colocar los métodos que la interface nos obliga, en la Tabla 4 se muestra el código necesario:

```
private string _callbackArgument;
        public string GetCallbackResult()
        {
           return _callbackArgument;
      }
        public void RaiseCallbackEvent(string eventArgument)
        {
        }
```

Tabla 4: Métodos de la Interface ICallbackEventHandler

Fuente: Elaboración Propia, 2013

Ahora se coloca el método que permita hacer el cambio de moneda, para esto se utiliza un valor de cambio de 2.7 soles por cada dólar. En la Tabla 5 se muestra el código necesario:

```
public string retornarCantidadDolar(string soles)

        {

            return (Convert.ToDouble(soles) / 2.7).ToString();

        }
```

Tabla 5: Método que Retorna la Conversión de Soles a Dólares

Fuente: Elaboración Propia, 2013

Lo siguiente que se debe hacer es llamar al método "retornarCantidadDolar" y pasarle como argumento la cantidad de soles que se desea convertir a dólares, en la Tabla 6 se muestra el código necesario.

```
public void RaiseCallbackEvent(string eventArgument)

        {

            Label dolares = new Label();

            dolares.Text = retornarCantidadDolar(eventArgument);



            StringWriter sw = new StringWriter();

            HtmlTextWriter htw = new HtmlTextWriter(sw);

            dolares.RenderControl(htw);

            htw.Flush();

            _callbackArgument = sw.ToString();

        }
```

Tabla 6: Asignando el valor a la variable _callbackArgument

Fuente: Elaboración Propia, 2013

Como se ve el método retornarCantidadDolar es llamado desde el método de la interface RaiseCallbackEvent, esto es necesario debido a que este es el que recibe el argumento del script que se encuentra en el lado del cliente.

Crear el script que recibe los datos del servidor.

Esta parte de código se colocó en el HEAD del documento HTML. El código se muestra a continuación, en la Tabla 7.

```
<script                                    src="Scripts/jquery-1.4.1.js"
type="text/javascript"></script>

    <script type="text/javascript">

        function script_ladoCliente(arg) {

            $("#Text1").val($(arg).html());

        }

    </script>
```

Tabla 7: Función que Obtiene los Datos del Servidor

Fuente: Elaboración Propia, 2013

Primero es hacer una referencia a jQuery, posteriormente se crea una función llamada "script:ladoCliente" la cual es encargada de recibir la información que el servidor manda, la función lo único que hace es colocar el valor que envía el servidor en el input que se había creado.

Crear el script que llama al servidor

Es el último paso en este ejemplo, y es el que desencadena todo el proceso que se creó, para poder llamar al script que enviar la información al servidor se hace lo siguiente:

Crear una función como el de la Tabla 8:

```
function Button1_onclick() {

        script_llamaServidor($("#Text1").val(), "");

    }
```

Tabla 8: Función que Llama al Servidor

Fuente: Elaboración Propia, 2013

La función "Button1_onclick()" obtiene el valor del argumento a partir del valor que el usuario escribe en el input cuyo ID es "Text1".

Finalmente se asocia esa función a un evento onclick del botón que se había creado en el primer paso.

En la Tabla 9 se muestra como agregar la función al input:

```
<input id="Button1" type="button" value="Cambiar a dolares"
onclick="return Button1_onclick()" />
```

Tabla 9: Colocando la Función "Button1_onclick" al Input "Text1"

Fuente: Elaboración Propia, 2013

Ejecutando ejemplo

Basta con ejecutar el código en un navegador, para que se muestre lo que hay en la Figura 14:

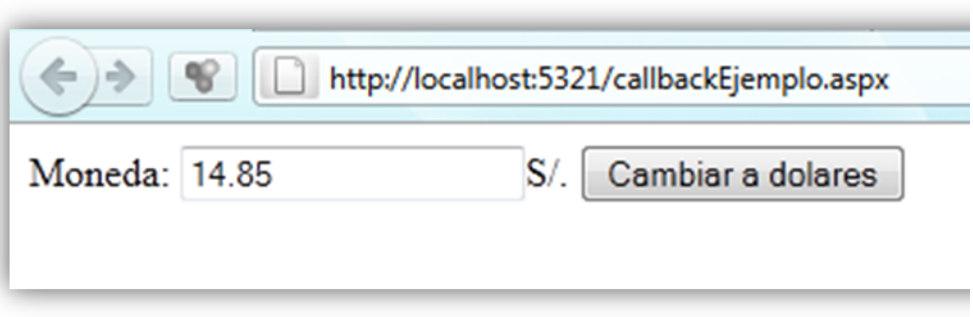

Figura 14: Ejemplo de ICallbackEventHandler en Ejecución

Fuente: Elaboración Propia, 2013

Y en la Figura 15 se muestra el resultado de hacer clic en el botón "Cambiar a dólares", obviamente no se produjo un postback al hacer clic en el botón.

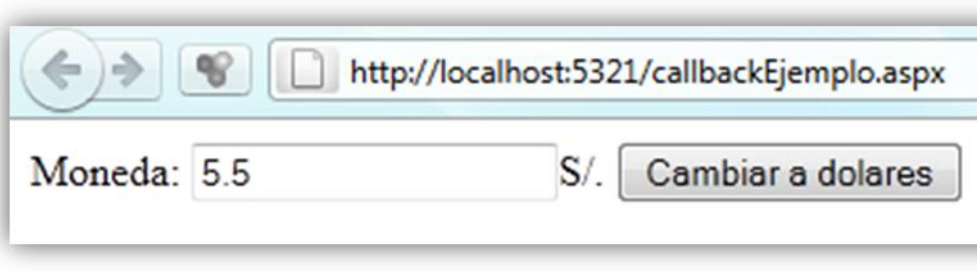

Figura 15: Resultado de Hacer Click en "cambiar a dólares"

Fuente: Elaboración Propia, 2013

El ejemplo también debió cambiar el signo de la moneda, sin embargo el ejemplo ya cumplió su cometido, el mostrar cómo funciona ICallbackEventHandler.

En la Tabla 10 y 11 se muestra el código completo.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="callbackEjemplo.aspx.cs"
Inherits="EjemploJS.callbackEjemplo" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

    <script src="Scripts/jquery-1.4.1.js"
type="text/javascript"></script>

    <script type="text/javascript">

        function script_ladoCliente(arg) {

            $("#Text1").val($(arg).html());

        }

        function Button1_onclick() {

            script_llamaServidor($("#Text1").val(), "");

        }

    </script>

</head>

<body>

    <form id="form1" runat="server">

        <span>Moneda:</span>

        <input id="Text1" type="text" />S/.

        <input id="Button1" type="button" value="Cambiar a dolares"
```

```
onclick="return Button1_onclick()" />

    </form>

</body>

</html>
```

Tabla 10: Código "Source" del Ejemplo de ICallbackEventHandler

Fuente: Elaboración Propia, 2013

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

namespace EjemploJS

{

    public partial class callbackEjemplo : System.Web.UI.Page,
ICallbackEventHandler

    {

        protected void Page_Load(object sender, EventArgs e){

            String cbReference =

            Page.ClientScript.GetCallbackEventReference(this,

            "arg", "script_ladoCliente", "context");
```

```
            String callbackScript;
            callbackScript   =   "function   script_llamaServidor(arg,
context)" +
                "{" + cbReference + ";}";

Page.ClientScript.RegisterClientScriptBlock(this.GetType(),
                "script_llamaServidor", callbackScript, true);
        }
        private string _callbackArgument;
        public string GetCallbackResult()
        {return _callbackArgument;}
        public void RaiseCallbackEvent(string eventArgument){
            Label dolares = new Label();
            dolares.Text = retornarCantidadDolar(eventArgument);

            StringWriter sw = new StringWriter();
            HtmlTextWriter htw = new HtmlTextWriter(sw);
            dolares.RenderControl(htw);
            htw.Flush();
            _callbackArgument = sw.ToString();
        }
        public string retornarCantidadDolar(string soles){
```

```
            return (Convert.ToDouble(soles) / 2.7).ToString();

        }}}
```

Tabla 11: Código “cs” del Ejemplo de ICallbackEventHandler

Fuente: Elaboración Propia, 2013

Utilidad para el proyecto de tesis

ICallbackEventHandler es la esencia del proyecto, es la interface que se trata de mejorar y optimizar en todo sentido.

3.5 Llamadas asíncronas usando JQuery y AJAX

JQuery ofrece un método principal, llamado $.ajax(), que es altamente configurable para adaptarse a cualquier necesidad que se pueda tener. También proporciona un conjunto de métodos abreviados, llamados taquigrafía porque son simplemente contenedores para el método $.ajax () con una configuración preestablecida.

Para comprender un poco como se realiza esto, se desarrolla un pequeño ejemplo.

Requisitos: Solo se necesita descargar jQuery desde su página oficial o desde algún otro lugar que brinde la biblioteca.

¿Es necesario el IDE de desarrollo Visual Studio? No, pero para este caso se utiliza, sin embargo este ejemplo puede ser desarrollado con cualquier IDE e incluso con lenguajes como java (jsp), PHP, etc. La Figura 16 muestra el diagrama de casos de uso del ejemplo “Sumar Numeros”.

Figura 16: Diagrama de CUS del Ejemplo – Sumar Números

Fuente: Elaboración Propia, 2013

Se empezó creando el proyecto, se colocó como nombre del proyecto "Probando_AjaxJquery01" sin embargo puede tener cualquier otro.

Luego se debe agregar un formulario Web cuyo nombre es "ajaxJquery.aspx".

En la Tabla 12 se muestra la interfaz que se utilizará para probar Ajax con jQuery.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="ajaxJquery.aspx.cs"
Inherits="Probando_AjaxJquery01.ajaxJquery" %>



<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

    <h4>Sumar 2 números utilizando Ajax y JQuery</h4>

        <table class="style1">

            <tr>

                <td class="style2">
```

```
                    Número Uno:</td>

                <td>

        <input id="txtNum1" type="text" /></td>

            </tr>

            <tr>

                <td class="style2">

                    Número Dos:</td>

                <td>

        <input id="txtNum2" type="text" /></td>

            </tr>

            <tr>

                <td class="style2">

                     </td>

                <td>

                    <input   id="btnSumar"   type="button"   value="Sumar"
/></td>

            </tr></table>

      </div></form></body></html>
```

Tabla 12: Código de la Interfaz que se Utiliza Para el Ejemplo “Ajax con Jquery”

Fuente: Elaboración Propia, 2013

Ahora se debe colocar el código JavaScript, el cual utiliza jQuery, primero se muestra el código en la Tabla 13, luego se pasa a explicar paso a paso.

```
    <script type="text/javascript" language="javascript">

        jQuery(document).ready(function () {

            $('#btnSumar').click(function () {

                var n1 = $('#txtNum1').val();

                var n2 = $('#txtNum2').val();

                sendDataAjax(n1, n2);

            });

        });

        function sendDataAjax(n1, n2) {

            var actionData = "{'num1': '" + n1 + "','num2': '" + n2 +
"'}";



            $.ajax(

            {

                url: "ajaxJquery.aspx/GetDataAjax",

                data: actionData,

                dataType: "json",

                type: "POST",

                contentType: "application/json; charset=utf-8",

                success: function (msg) { alert(msg.d); },

                error: function (result) {

                    alert("ERROR   "   +   result.status   +   '   '   +
```

```
result.statusText);

                }

            });};

    </script>
```

Tabla 13: Código JavaScript que se utiliza para el ejemplo “Ajax con JQuery”

Fuente: Elaboración Propia, 2013

El código de la Tabla 13 deberá ser incorporado en el "head" del formulario web. Hay dos partes importantes en el código, la primera parte es la que permite obtener el evento click del botón “btnSuma”, se refiere exactamente a esta parte: “$('#btnSuma').click(function ()“. Dentro de este evento se capturan los valores de las cajas de texto y luego se llama a la función sendDataAjax y se le pasa como argumentos los valores de la caja de texto, esta función es la segunda parte a la que se hacía referencia y que se pasará a explicar a continuación.

Dentro de la función sendDataAjax se encuentra lo necesario para poder utilizar Ajax. Con “$.ajax” se llama a un método de jQuery al cual se le deben pasar algunos argumentos de los cuales se explicaran los más importantes:

Url: Es la dirección donde se encuentra el método que se encuentra en el servidor y el que se está intentando invocar. “"ajaxJquery.aspx/GetDataAjax"” la primera palabra hace referencia al nombre de la clase y la segunda palabra es el nombre del método.

Data: son los datos que vamos a enviar al método del servidor, en este caso serían los valores de num1 y num2.

DataType: Es el formato de cómo se van a enviar los datos al servidor, en este caso se elige json.

Success: Esto es lo que se ejecuta si no hubo algún error en el servidor.

Error: De forma contraria que success, este se ejecuta si sucede algún error en el servidor.

El siguiente paso, y ultimo, es escribir el código del servidor, como se mencionó anteriormente, se debe colocar el método GetDataAjax, en realidad podría tener cualquier nombre pero así se especificó en el método "$.ajax" así que deberá tener el mismo nombre.

En la Tabla 14 se muestra el código necesario que debe tener el servidor.

```
using System;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.Services;



namespace Probando_AjaxJquery01

{

    public partial class AjaxJquery : System.Web.UI.Page

    {

        protected void Page_Load(object sender, EventArgs e)

        {}

        [WebMethod]

        public static string GetDataAjax(string num1, string num2)

        {return string.Format("La suma de los numeros es: "+
((Convert.ToInt32(num1)) + (Convert.ToInt32(num2))));

        }}}
```

Tabla 14: Código en el Servidor que se Utiliza para el ejemplo "Ajax con Jquery"

Fuente: Elaboración Propia, 2013

El método que se está colocando debe tener el atributo "[WebMethod]" lo cual indica que ese método puede ser invocado como si fuera un servicio, es decir un web service. Tal vez esa sea una desventaja de este método debido a que vas a tener muchos métodos que ofrecen servicios, dependiendo de la complejidad de lo que se esté programando.

Al ejecutar este ejemplo, se obtiene el resultado de la Figura 17.

Figura 17: Ejecución del Ejemplo "Ajax con Jquery"

Fuente: Elaboración Propia, 2013

Utilidad para el proyecto de tesis

Esta herramienta sirve como modelo para representar las llamadas asíncronas en una Web utilizando ICallbackEventHandler. Gracias a esta herramienta, se decide utiliza la notación literal de JavaScript para realizar la comunicación con el servidor.

3.6 Llamadas asíncronas usando XMLHttpRequest

La mayoría del trabajo con Ajax pasa por el uso de llamadas asíncronas a un servidor, y esto, aunque los distintos Frameworks lo gestionen de una forma u otra, se consigue gracias al objeto XMLHttpRequest.

A continuación se desarrolla un ejemplo del uso de XMLHttpRequest.

Requisitos:

Tener un navegador web.

Se desarrolla el siguiente ejemplo en el IDE Visual Studio 2012 .NET.

Lo primero es desarrollar la interfaz gráfica de la página, para esto se escribe el código que se presenta en la Tabla 15.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="index.aspx.cs"
Inherits="xmlHtppRequest_EjemploTesis.index" %>



<!DOCTYPE html>



<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

    <script type="text/javascript">

     </script>

</head>

<body>

    <form id="form1" runat="server">

    <div>

        <h1>Testing Web Services</h1>

        <b>Nickname:</b>  <input  id="idNombre"  type="text"  />  <input
type="button" value="Aceptar" onclick="testWebService()" />

    </div></form></body></html>
```

Tabla 15: Interfaz de usuario del ejemplo - XMLHttpRequest

Fuente: Elaboración Propia, 2013

Luego, dentro de la etiqueta “script” se coloca el contenido de la Tabla 16.

```
        function testWebService() {argumento = "nombre=" +
document.getElementById("idNombre").value;request = new XMLHttpRequest();

            request.onreadystatechange = responseHandler;

            request.open("POST", "/Services/TestService.asmx/HelloWorld",
true);

            request.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');

            request.send(argumento);

        }function responseHandler() {

            if (request.readyState == 4) {

                if (request.status == 200) {

alert(request.responseXML.getElementsByTagName("string")[0].textContent);

                }else {alert("Ha surgido un problema...");} } }
```

Tabla 16: Código JavaScript del ejemplo XMLHttpRequest

Fuente: Elaboración Propia, 2013

La función “testWebService” se ejecuta cada vez que se hace click en el botón “Aceptar”. Esta función desencadena el código necesario para hacer una llamada al servidor.

“request = new XMLHttpRequest();” es la sentencia que permite la creación del objeto XMLHttpRequest.

“request.onreadystatechange = responseHandler;” es la sentencia que permite asociar una función a la respuesta del servidor, es decir que cada vez que el servidor retorne un valor, se ejecuta la función “responseHandler“.

“request.open (…);” es la sentencia que desencadena la llamada al servidor.

“request.send(argumento);” es la sentencia que permite enviar argumentos al servidor.

La función “responseHandler” es la que se ejecuta cada vez que el servidor retorna un valor. Para ver el tipo de resultado, se consulta “request.readyState” que según el número que tenga se puede saber en qué estado se encuentra la llamda al servidor. El estado número 4 indica finalizado y es cuando el servidor retorna el valor deseado.

Para que el ejemplo funcione, se debe crear un servicio web con la información mostrada en la Tabla 17:

```
using System;

using System.Web;

using System.Web.Services;

namespace xmlHtppRequest_EjemploTesis.Services

{   [WebService(Namespace = "http://tempuri.org/")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

    [System.ComponentModel.ToolboxItem(false)]

    public class TestService : System.Web.Services.WebService

    {   [WebMethod]

        public string HelloWorld(string nombre)

        {return "Hola " + nombre;}}}
```

Tabla 17: Web Service del ejemplo de XMLHttpRequest

Fuente: Elaboración Propia, 2013

El Web Service contiene un único método que devuelve un string. El resultado del ejemplo se muestra en la Figura 18.

Figura 18: Resultado del ejemplo de XMLHttpRequest

Fuente: Elaboración Propia, 2013

Contras:

La desventaja de usar este objeto directamente sin usar un framework, es la compatibilidad con todos los navegadores del mercado. Se necesita de mucho esfuerzo para que funcione en todos los navegadores, por lo que es mejor usar un framework que facilite esa tarea.

Utilidad para el proyecto de tesis

Este objeto es el que permite la realización de llamadas asíncronas en las páginas Web, es de gran utilidad para la tesis conocer de él porque así se trata de corregir las falencias que tiene, como por ejemplo la compatibilidad con todos los navegadores.

3.7 Benchmarking

En la Figura 19 se muestra la comparación de las soluciones encontradas y en la Figura 20 se muestra la leyenda del Benchmarking.

BENCHMARKING				
ANÁLISIS COMPARATIVO				
	Icallback Helper	**IcallbackeventHandler**	**Jquery Ajax**	**XmlHttpRequest**
DETALLES DE PRODUCTO				
Lenguaje(s) que utiliza	C#, Javascript	C#, Javascript	Javascript	Javascript
Precio ($)	0	0	0	0
Sistema Operativo	Windows	Windows	Plataforma Independiente	Plataforma Independiente
Licencia	GPL	GPL	GPL	GPL
Forma de transportar datos	Json	XML,JSON	XML,JSON	XML,JSON
Mejor IDE de desarrollo para utilizarlo	.NET	.NET	.NET	.NET
CARACTERÍSTICAS				
Modo de Implementación (Manual y/o Automática)	5	3	4	3
Eficiencia en transferencia de Datos	5	3	4	4
Facilidad de Uso	4	2	4	3
Variedad de efectos para presentar salida en pantalla	5	5	5	4
Integración con .NET	5	5	3	3
Compatibilidad con la Base de Datos	5	5	5	5
Documentación	5	3	5	5
Variedad de ejemplos de las funcionalidades	5	2	3	4
Control sobre el código	5	5	5	5
Compatibilidad con los navegadores	5	5	5	5
PUNTAJE	49	38	43	41

Figura 19: Benchmarking

Fuente: Elaboración Propia, 2014

Leyenda		
Puntaje	**Porcentaje (%)**	**Descripcion**
0	0	No presenta la caracteristica
1	10	Caracteristica básica deficiente
2	25	Caracteristica básica
3	50	Regular / limitada
4	75	Buena/ extensiones
5	100	Supera la expectativa

Figura 20: Leyenda del Benchmarking

Fuente: Elaboración Propia, 2014

En esta comparación de productos se puede apreciar que la herramienta desarrollada en la tesis es ligeramente superior a las demás en ciertos aspectos.

CAPÍTULO IV: MODELADO DEL NEGOCIO

4.1 Reglas del negocio

Se utiliza .Net para desarrollar páginas Webs.

Se utiliza como leguaje C# o VB.

En el lado del cliente se utiliza JavaScript.

La página Web debe contar con llamadas asíncronas.

4.2 Casos de uso del negocio

Cliente: Persona natural o jurídica que necesita el desarrollo de un sistema informático, específicamente de una página Web con características asíncronas.

Analista de sistemas: Es el responsable del conjunto de requisitos que están modelados en los casos de uso, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específicos. El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente.

Especificador de casos de uso: Asisten al analista de sistemas y asumen las responsabilidades de las descripciones detalladas de uno o más casos de uso, estos trabajadores necesitan trabajar estrechamente con los usuarios reales de sus casos de uso.

Diseñador de interfaz de usuario: Los diseñadores dan forma visual a las interfaces de usuario. Esto puede implicar el desarrollo de prototipos de interfaces de usuario para algunos casos de uso, habitualmente un prototipo para cada actor. Por tanto, es conveniente que cada diseñador de interfaces de usuario de forma a las interfaces de usuario de uno o más actores.

Arquitecto: El arquitecto participa en el flujo de trabajo de los requisitos para describir la vista de la arquitectura del modelo de casos de uso. Es responsable de la integridad del modelo de análisis, garantizando que este sea correcto, consistente y legible como un todo.

Ingeniero de casos de uso: Un ingeniero de casos de uso es responsable de la integridad de una o más realizaciones de caso de uso, garantizando que cumplen los requisitos que recaen sobre ellos.

Ingeniero de componentes: El ingeniero de componentes define y mantiene las responsabilidades, atributos, relaciones, y requisitos especiales de una o varias clases de análisis, asegurándose de que cada clase del análisis cumple los requisitos que se esperan de ella de acuerdo a las realizaciones de caso de uso en las que participa. También, mantiene la integridad de uno o varios paquetes del análisis.

Ingeniero de pruebas: Es responsable de la integridad del modelo de pruebas, asegurando que el modelo cumple con su propósito. También, planean las pruebas, lo que significa que deciden los objetivos de prueba apropiados y la planificación de las pruebas. Además, seleccionan y describen los casos de pruebas y los procedimientos de prueba correspondientes que se necesitan, y son responsables de la evaluación de las pruebas de la integración y de sistema cuando éstas se ejecutan.

Integrador de Sistemas: La integración del sistema está más allá del ámbito de cada ingeniero de componentes individual. Por el contrario, esta responsabilidad recae sobre el integrador del sistema. Entre sus responsabilidades se incluye el planificar la secuencia de construcciones necesarias en cada iteración y la integración de cada construcción cuando sus partes han sido implementadas.

Ingeniero de Pruebas de integración: Los ingenieros de pruebas de integración son los responsables de realizar las pruebas de integración que se necesitan para cada construcción producida en el flujo de trabajo de la implementación. Las pruebas de integración se realizan para verificar que los componentes integrados en una construcción funcionan correctamente juntos. El ingeniero también se encarga de documentar los defectos en los resultados de la prueba de integración.

Ingeniero de Pruebas de Sistemas: Un ingeniero de pruebas de sistema es responsable de realizar las pruebas de sistema necesarias sobre una construcción que muestra el resultado (ejecutable) de una iteración completa. También, se encarga de documentar los defectos en los resultados de las pruebas de sistema.

En la figura 21 se muestra el diagrama de actores y trabajadores del negocio.

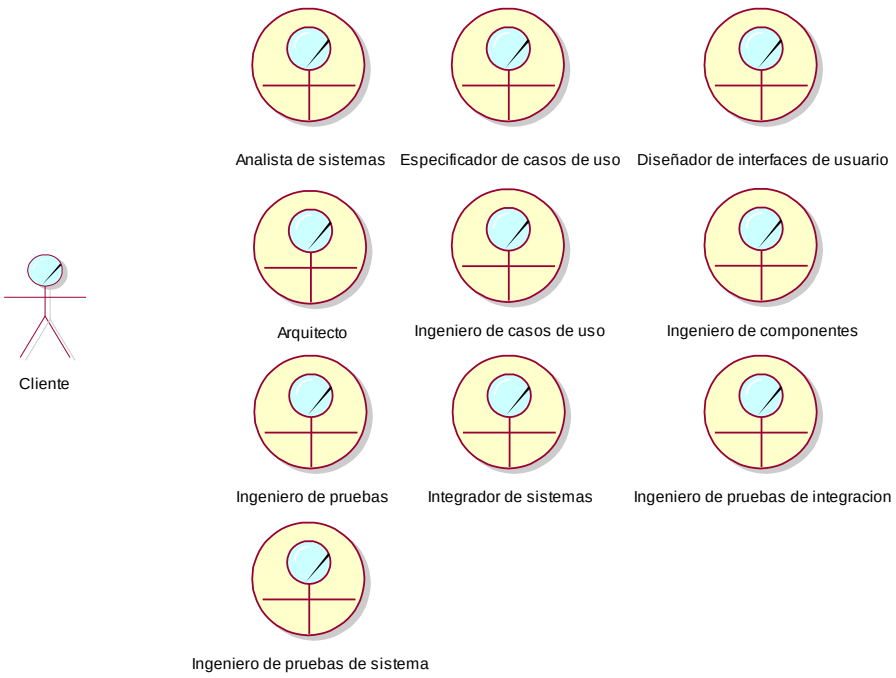

Figura 21: Diagrama de Actores y Trabajadores del Negocio

Fuente: Elaboración Propia, 2014

4.2.1 Diagrama de CUN

Desarrollar Software: Es el caso de uso del negocio principal de la organización, permite la creación de software para sus clientes.

Crear Herramientas Para Desarrollar Software: Es el caso de uso del negocio principal de la tesis, este caso de uso permite la construcción de herramientas para mejorar el desarrollo de software.

En la figura 22 se muestra el diagrama de casos de uso del negocio.

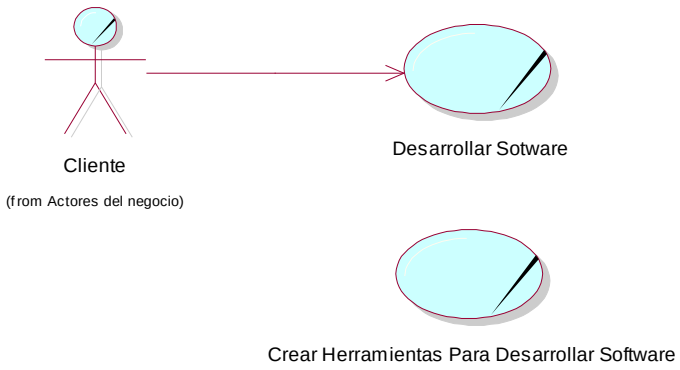

Figura 22: Diagrama de Casos de Uso del Negocio

Fuente: Elaboración Propia, 2014

4.2.2 Diagrama de Actividades

La construcción de un software está dividida en varias etapas. Las etapas que se muestran a continuación, en forma de diagrama de actividades, son las que mínimamente se deben seguir para construir un software de calidad.

Diagrama de Actividad – Requisitos

En los requisitos se desarrolla el modelo del sistema que se va a construir, y la utilización de los casos de uso es una forma adecuada de crear ese modelo. Esto es debido a que los requisitos funcionales se estructuran de forma natural mediante casos de uso, y a que la mayoría de los otros requisitos no funcionales son específicos de un solo caso de uso, y pueden tratarse en el contexto de ese caso de uso. La Figura 23 muestra el diagrama de actividad de Requisitos.

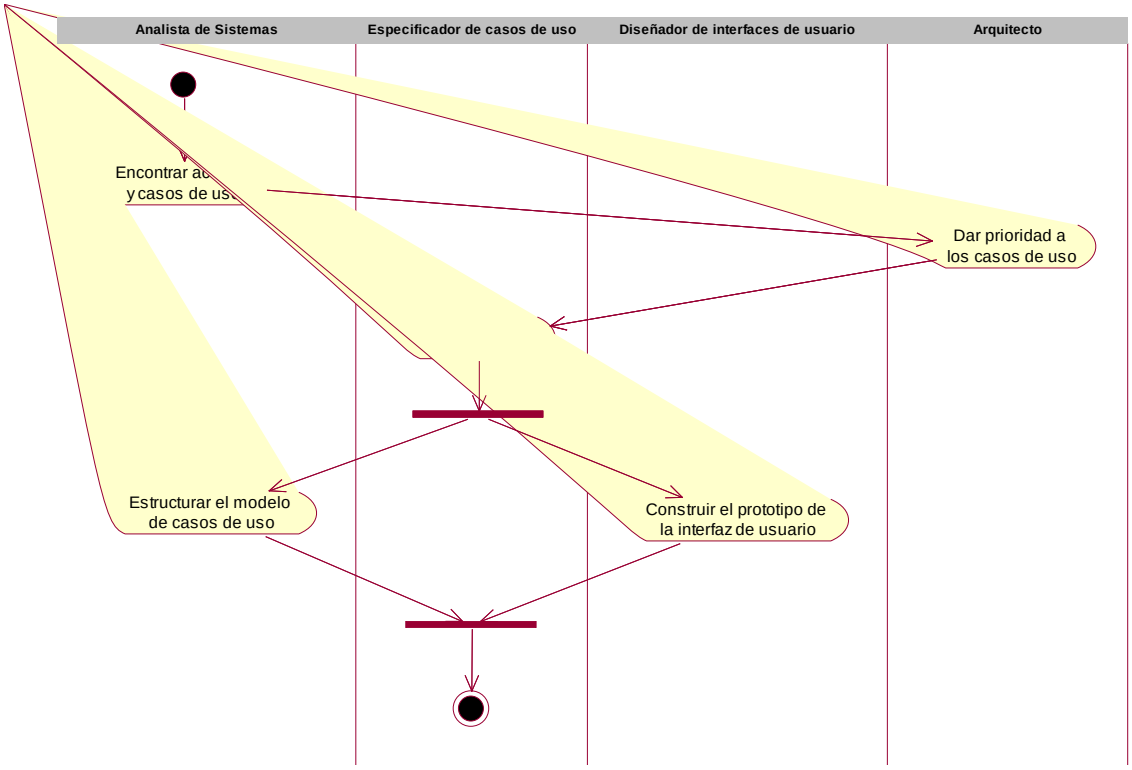

Figura 23: Diagrama de Actividad: Requisitos

Fuente: Elaboración Propia, 2014

Diagrama de Actividad – Análisis

Se analiza los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacer esto es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero incluyendo su arquitectura. La Figura 24 muestra el diagrama de actividad de Análisis.

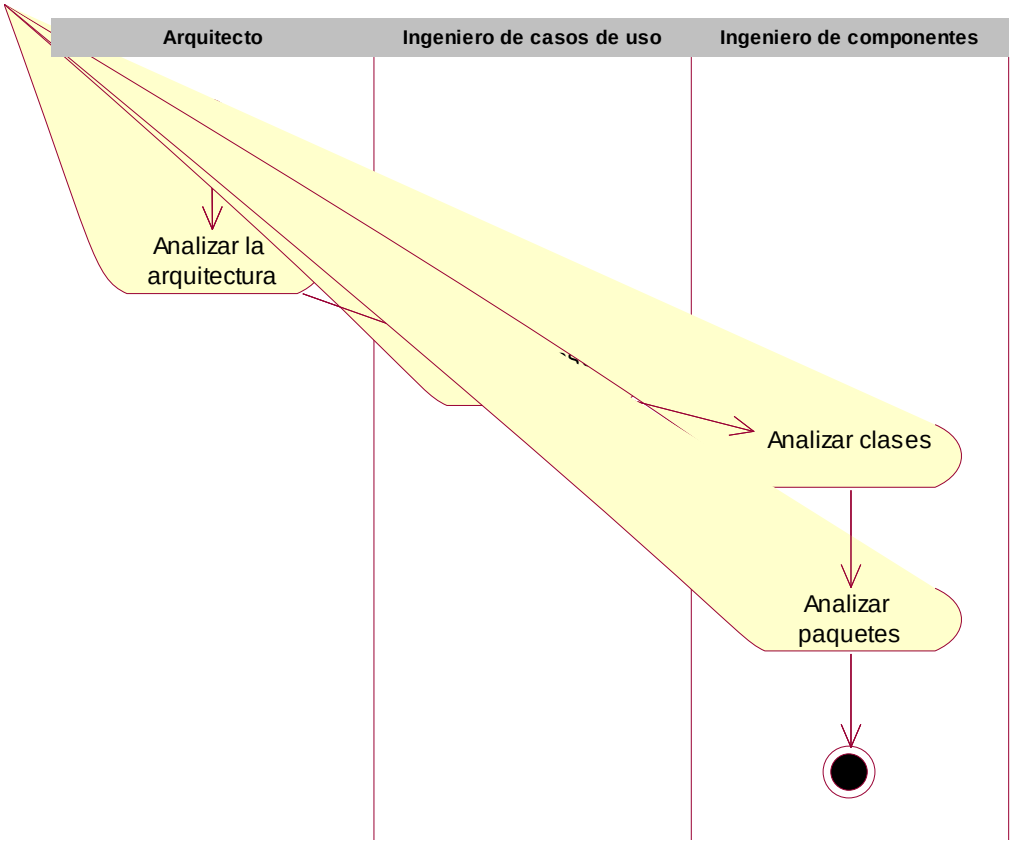

Figura 24: Diagrama de Actividad: Análisis

Fuente: Elaboración Propia, 2014

Diagrama de Actividad – Diseño

El diseño se modela el sistema y se encuentra su forma (incluida la arquitectura), para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones. Se crea una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases. La Figura 25 muestra el diagrama de actividad de Diseño

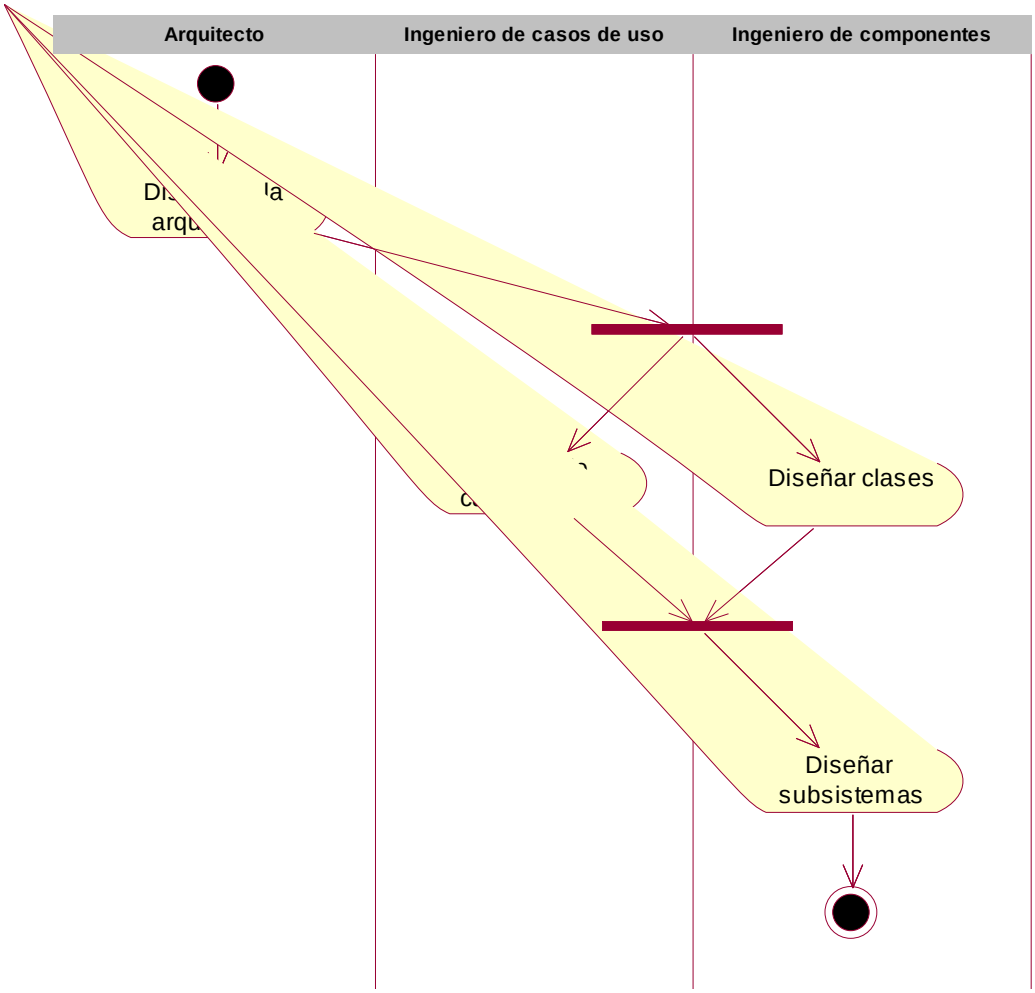

Figura 25: Diagrama de Actividad: Diseño

Fuente: Elaboración Propia, 2014

Diagrama de Actividad – Implementación

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes como ficheros, código fuente, scripts, ficheros de códigos binarios, ejecutables y similares. Además, el propósito principal es desarrollar la arquitectura y el sistema como un todo. La Figura 26 muestra el diagrama de actividad de Implementación.

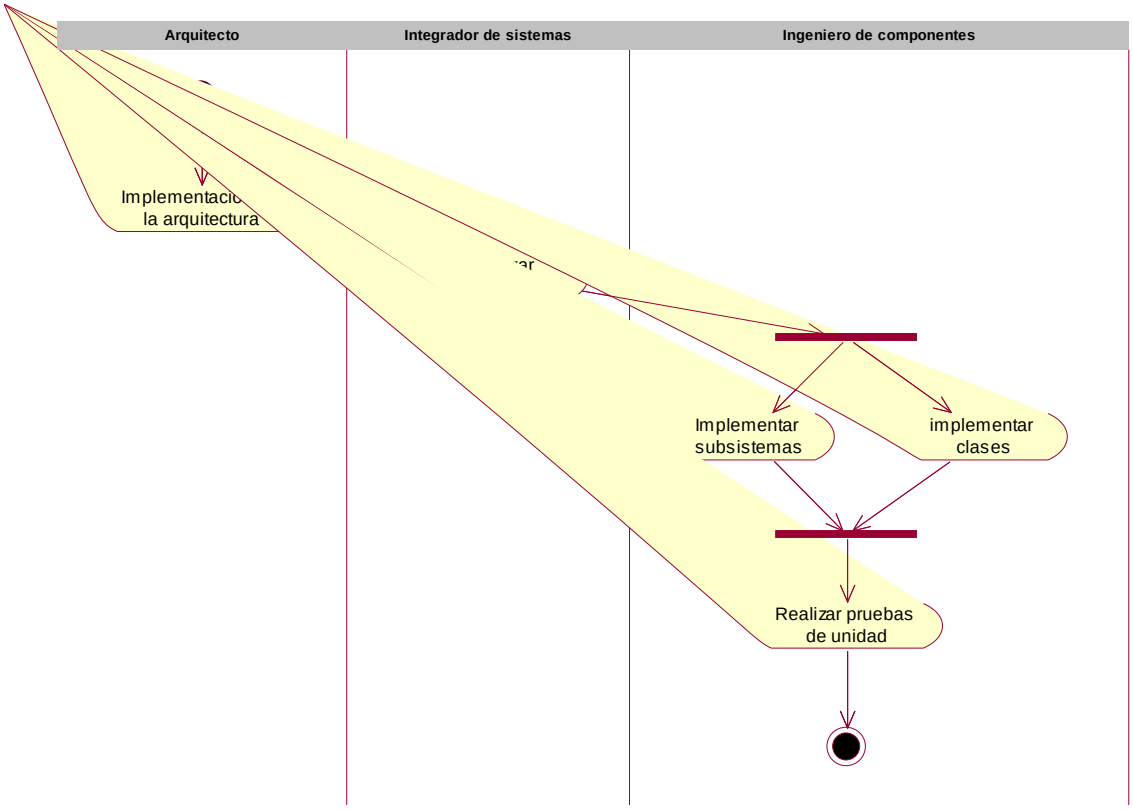

Figura 26: Diagrama de Actividad: Implementación

Fuente: Elaboración Propia, 2014

Diagrama de Actividad – Pruebas

En las pruebas se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros. La figura 27 muestra el diagrama de actividad de Pruebas.

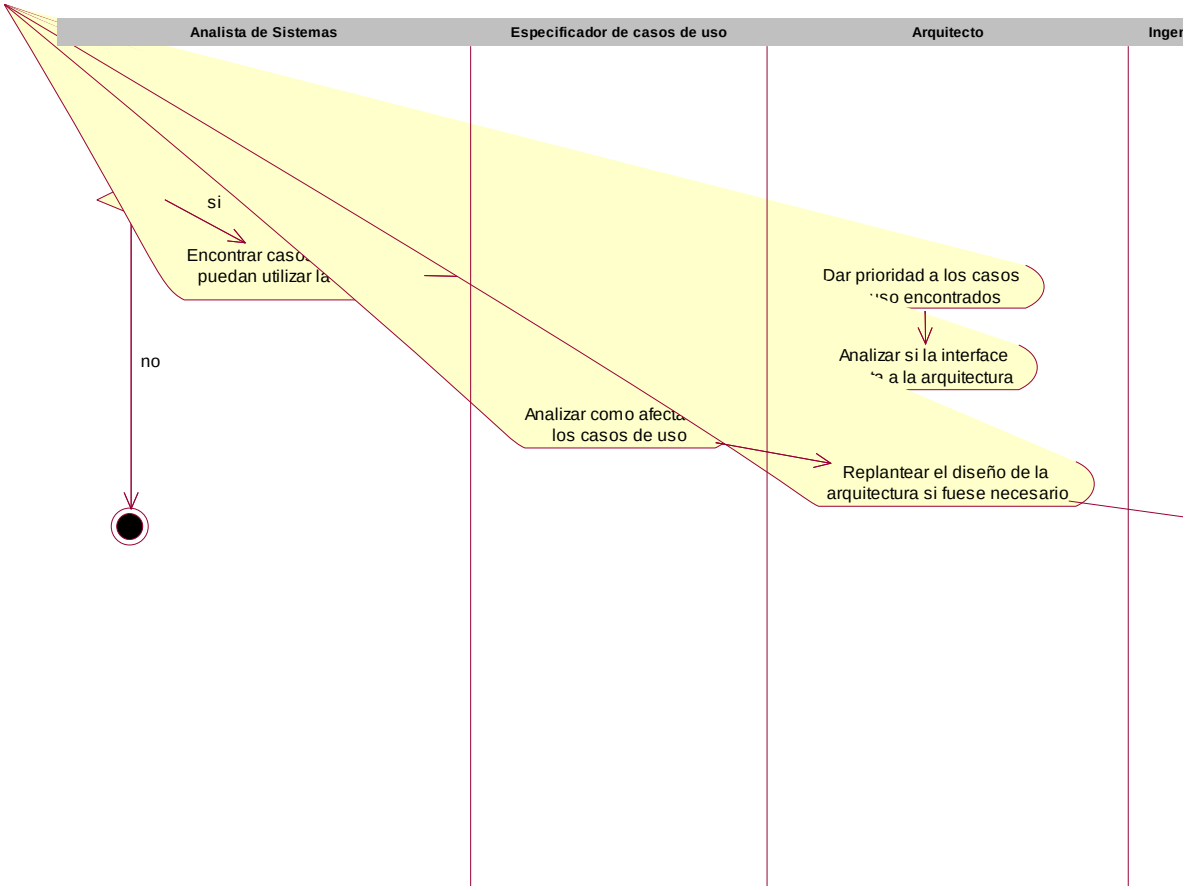

Figura 27: Diagrama de Actividad: Pruebas

Fuente: Elaboración Propia, 2014

Diagrama de Actividad – Crear Herramientas para Desarrollar Software

El siguiente diagrama pertenece al caso de uso del negocio *Crear herramienta para desarrollar software*, este diagrama refleja las actividades principales de la tesis y se verán los problemas que se tratan de solucionar. Las Figuras 28, 29, 30, 31 muestran el diagrama de actividad para el caso de uso Crear Herramientas para Desarrollar Software.

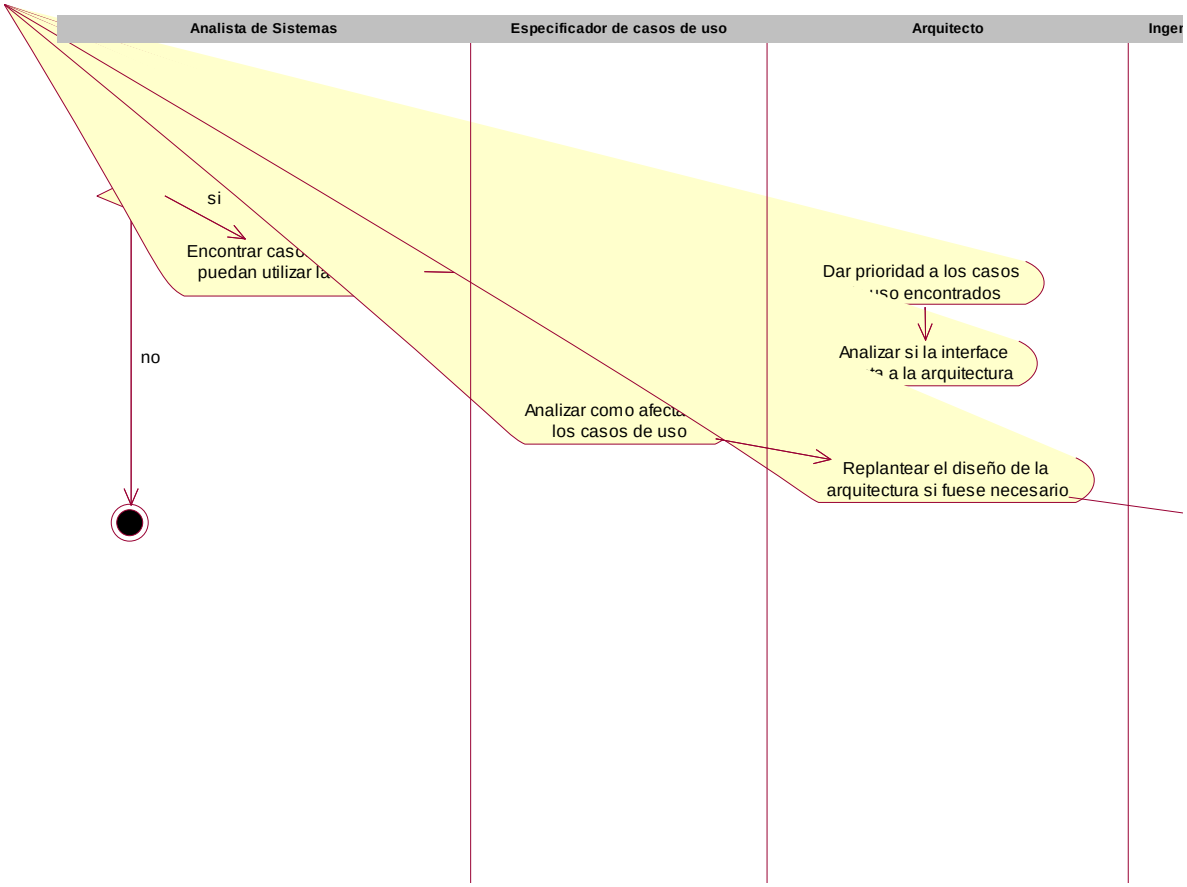

Figura 28: Diagrama de Actividad: Crear Herramienta de Software Parte 1

Fuente: Elaboración Propia, 2014

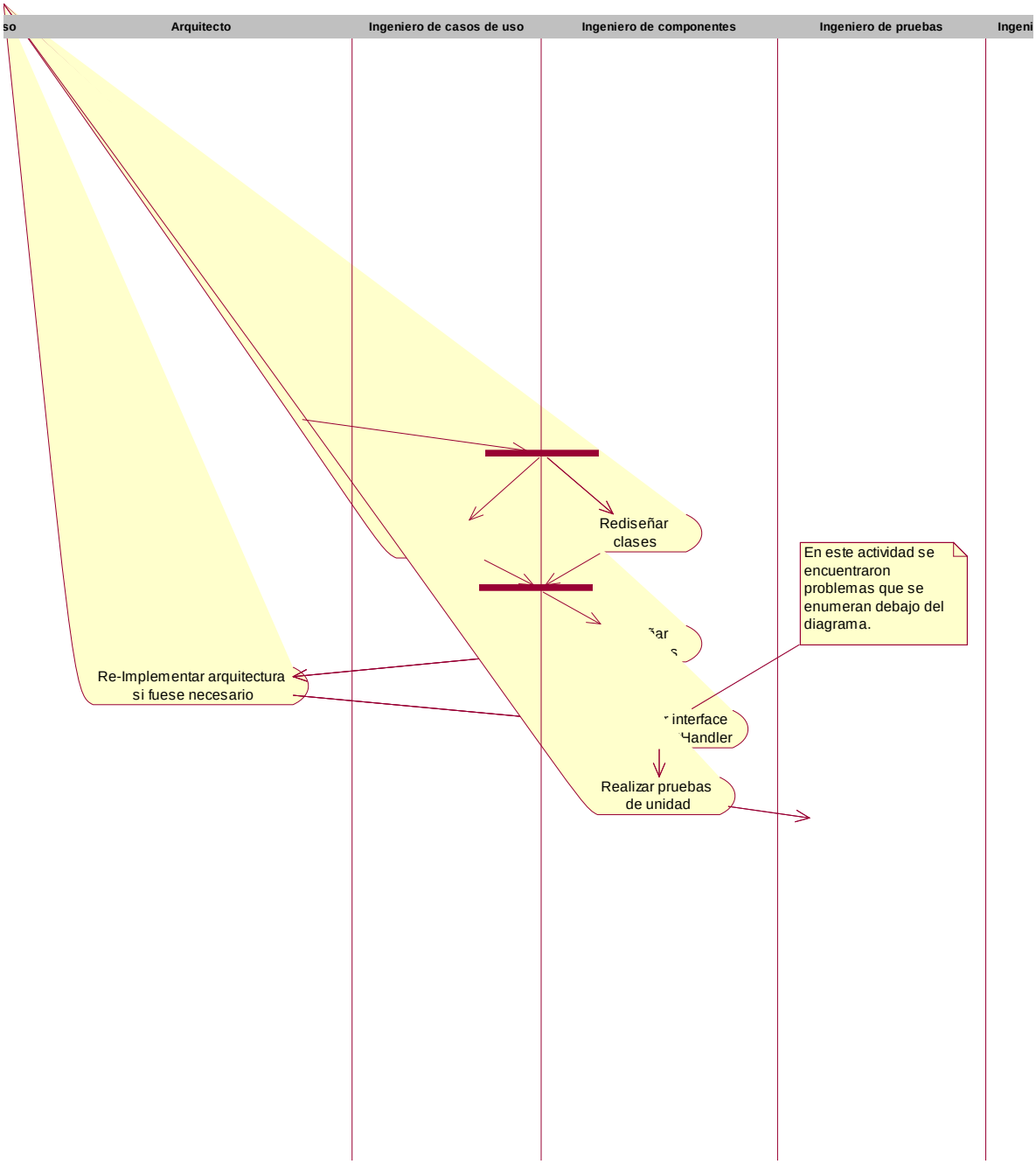

Figura 29: Diagrama de Actividad: Crear Herramienta de Software parte 2

Fuente: Elaboración Propia, 2014

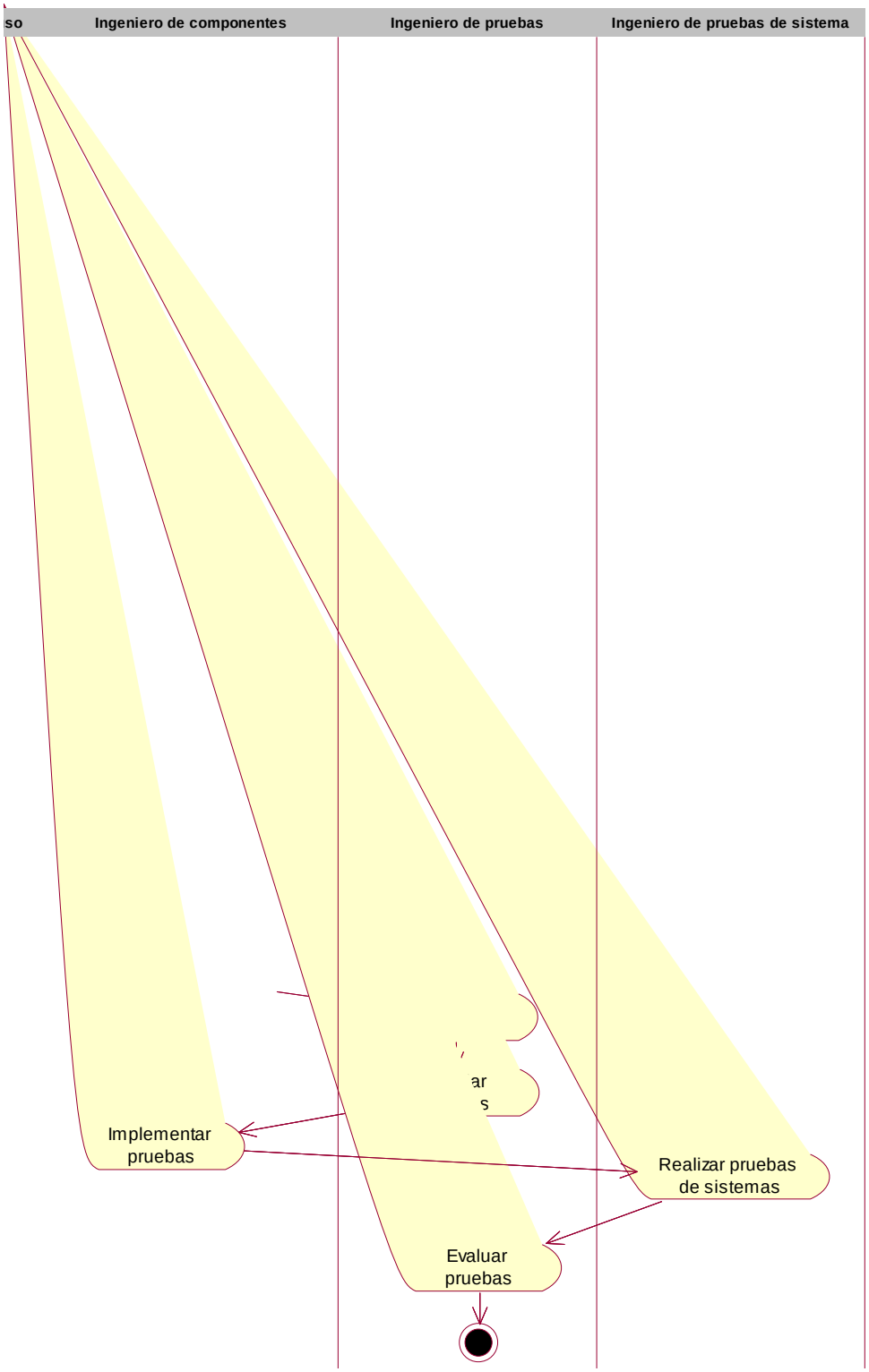

Figura 30: Diagrama de Actividad: Crear Herramienta de Software Parte 3

Fuente: Elaboración Propia, 2014

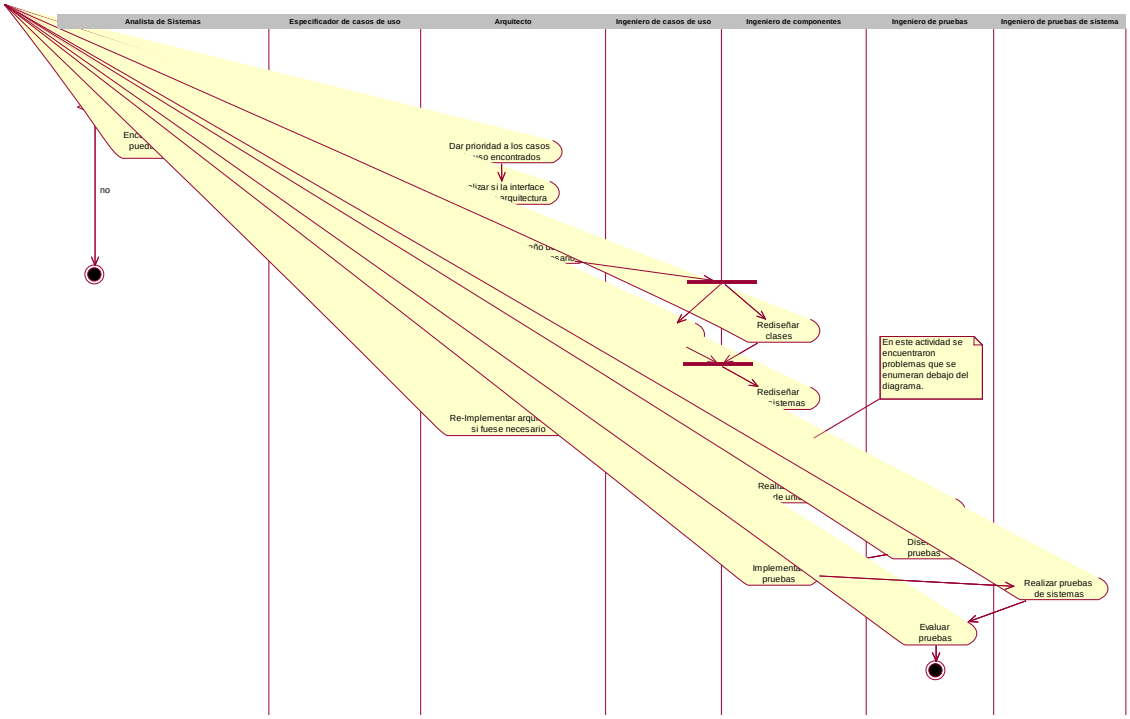

Figura 31: Diagrama de Actividades: Crear Herramienta De Software (Completo)

Fuente: Elaboración Propia, 2014

Problemas Encontrados:

Se limita a usar sólo dos métodos en el lado del servidor: RaiseCallbackEvent y GetCallbackResult.

El Método RaiseCallbackEvent recibe un solo parámetro (string) lo cual dificulta el querer enviar más parámetros al servidor.

Se dificulta mucho la tarea de implementar más de una llamada asíncrona en un mismo WebForm debido a que solo se cuenta con dos métodos en el lado del servidor.

No hay forma de discriminar si sucede algún error, no hay control de Excepciones.

El hecho de registrar funciones javascript en el PageLoad rompe la independencias de cada lado (Cliente/Servidor).

CAPÍTULO V: REQUERIMIENTOS DEL PROYECTO

5.1 Requerimientos del software

En esta sección veremos una lista de los requerimientos funcionales y no funcionales del sistema, también se muestra una descripción de cada uno.

5.1.1 Relación de requerimientos

En las tablas 18 y 19 se pueden ver la lista de requerimientos funcionales y no funcionales respectivamente.

Código de Requerimiento.	Nombre del requerimiento
RF01	Permitir la creación de métodos propios en el lado del servidor.
RF02	Eliminar la utilización de un método que devuelva el resultado del servidor al cliente.
RF03	Permitir la implementación de los métodos del servidor en otras clases.
RF04	Permitir enviar más de un argumento al servidor.
RF05	Permitir enviar argumentos que sean tanto números como cadenas de texto
RF06	Tener una función estándar para comunicarse con el servidor.
RF07	Poder elegir que función va a recibir la información del servidor.
RF08	Controlar el manejo de errores.
RF09	Poder elegir la función que se ejecuta si se encuentra algún error.

Tabla 18: Lista de Requerimientos Funcionales

Fuente: Elaboración Propia, 2014

Código de Requerimiento.	Nombre del requerimiento
RNF01	Destreza en el manejo del sistema
RNF02	Encontrar una función del sistema
RNF03	Lenguaje de programación

Tabla 19: Lista de Requerimientos No Funcionales

Fuente: Elaboración Propia, 2014

5.1.2 Especificación de requerimientos

5.1.2.1 Requerimientos funcionales

RF01: Permitir la creación de métodos propios en el lado del servidor

El sistema debe permitir la comunicación del cliente con el servidor mediante otro método que no sea RaiseCallbackEvent, este otro método puede tener cualquier nombre definido previamente por el programador.

RF02: Eliminar la utilización de un método que devuelva el resultado del servidor al cliente

Actualmente se utiliza el método GetCallbackResult para devolver información al cliente, se debe evitar esto y se debe utilizar el mismo método creado por el programador para tener la comunicación con el cliente.

RF03: Permitir la implementación de los métodos del servidor en otras clases

Actualmente los métodos que se comunican con el cliente desde el servidor, siempre deben ir en el WebForm donde se programa la llamada asíncrona. Ahora se debe permitir colocar estos métodos en cualquier clase, incluso en cualquier otro namespace.

RF04: Permitir enviar más de un argumento al servidor

El método RaiseCallbackEvent solo recibe un parámetro, se desea que ahora se puedan enviar múltiples parámetros al servidor.

RF05: Permitir enviar argumentos que sean tanto números como cadenas de texto

Actualmente solo se puede enviar un parámetro al servidor y este necesariamente tiene que ser de tipo string. Se desea que además del tipo string, se pueda enviar también argumentos de tipo numérico (int, double, decimal, float).

RF06: Tener una función estándar para comunicarse con el servidor

En el PageLoad se define la función que se comunica con el servidor, ahora lo que se desea es que se cree una función estándar que sea la que se comunique con el servidor y que esta no sea declarada en el PageLoad.

RF07: Poder elegir que función va a recibir la información del servidor

Actualmente en el PageLoad se define la función que recibe la información del servidor, ahora se desea que la función sea definida desde el cliente y que se llame de la forma en que el programador desee.

RF08: Controlar el manejo de errores

No se manejan las excepciones que se puedan encontrar en el servidor, se desea cambiar esto y tener un control de errores.

RF09: Poder elegir la función que se ejecuta si se encuentra algún error

Se desea poder elegir que función se debe ejecutar cada vez que se encuentra una excepción en el servidor.

Requerimientos no funcionales

Usabilidad

Los requisitos de capacidad de uso, es decir la usabilidad, están relacionados con la facilidad de comprensión de los usuarios del sistema. Se han contemplado los siguientes requerimientos

RNF01: Destreza en el manejo del sistema

El Usuario es capaz de utilizar cualquier función del sistema sin mayores complicaciones, haciendo uso de los elementos de ayuda del sistema entre 1 a 8 minutos dependiendo del grado de conocimiento del usuario.

RNF02: Encontrar una función del sistema

El usuario es capaz de encontrar una función del sistema en un periodo de tiempo no mayor a 1 minuto, en el 90% de los casos.

Restricciones de diseño

Son las restricciones que tendrá el sistema a la hora de crearlo.

RNF03: Lenguaje de programación

El sistema es programado en el lenguaje C#, utilizando también JavaScript.

5.2 Casos de uso del sistema

Agrupa los requerimientos con los que cuenta el sistema.

5.2.1 Diagrama de actores del sistema

Los dos actores del sistema son los trabajadores del negocio en el capítulo anterior y se muestran en la Figura 32.

Figura 32: Diagrama de Actores del Sistema

Fuente: Elaboración Propia, 2014

5.2.2 Relación de Casos de uso del sistema

Utilizar Función Estándar

Utilizar Método Personalizado

5.2.3 Diagrama de casos de uso de sistema

En la Figura 33 se muestra el diagrama de casos de uso del sistema.

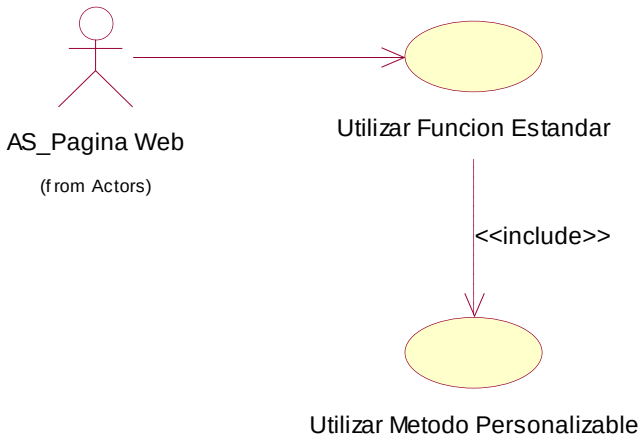

Figura 33: Casos de Uso del Sistema

Fuente: Elaboración Propia, 2014

5.2.4 Especificación de casos de uso

En la Tabla 20 se muestra la especificación de caso de uso de "Utilizar Función Estándar".

Use Case ID:	UC-1.0.0
Use Case Name:	Utilizar Función Estándar
Actors:	AS_Pagina Web
Description:	Este caso de uso permite una comunicación asíncrona entre una página web y el servidor(*) a través de una función javascript. Esta comunicación se da gracias a la interface proporcionada por Microsoft: ICallbackEventHandler. El caso de uso tiene como finalidad mejorar y facilitar la implementación regular de la interface ICallbackEventHandler, proporcionando las siguientes mejoras: Permite enviar más de un argumento al servidor (hasta 100 sin perjudicar rendimiento, pero pueden ser infinitos), a diferencia de la forma regular que solo permite enviar uno. Permite enviar argumentos que sean tanto números como cadenas de texto, a diferencia de la forma regular que solo puede enviar una cadena de texto (string). Tiene una función estándar para comunicarse con el servidor, a diferencia de la forma regular que pide que se implemente esta función en el PageLoad de la página web cada vez que se utiliza la interface. Para cada evento asíncrono en una página web, se puede crear una función javascript personalizada para recibir la información del servidor, a diferencia de la forma regular que obliga al programador a implementar la función en el PageLoad de la página web y recibir los valores devueltos del servidor para todas los eventos asíncronos. Controla el manejo de errores, a diferencia de la forma regular que no lo hace. Poder elegir la función que se ejecuta si se encuentra algún error, especificando la función de error en la función estándar, a diferencia de la forma regular que no proporciona esta facilidad. (*) Llámese servidor a las clases y métodos de código C# que están alojadas en un host central.
Trigger:	El caso de uso se desencadena con cualquier evento html ejecutado por un usuario. Ejemplo: Click en un botón.

Tabla 20: Utilizar Función Estándar (Continúa)

Preconditions:	El objeto html que desencadena el caso de uso debe tener asociado el nombre da la función javascript pertinente. La aplicación web debe contener el archivo javascript que permite mejorar la implementación de ICallbackEventHandler. La aplicación web debe tener referenciado el archivo del punto 2.
Postconditions:	Se ejecuta la función javascript de éxito definida en la función estándar.

Normal Flow:	Se ejecuta un evento html. (Ejemplo: click de un boton).
	El evento desencadena la llamada a una función javascript.
	La función javascript hace una llamada a la función estandar "ICallbackFunction()" y le pasa los siguientes argumentos: namespace, clase, method, funcion de éxito (success), función de error (error) y data.
	Ejemplo:
	`function functionPersonalizada() { //Funcion estandar: iCallbackFunction({ namespace: "nombre del namespace", clase: "nombre de la clase", method: "nombre del metodo", success: function (arg) { }, error: function (error) { }, data: (new Array()) }); }`
	La función ICallback serializa los datos en un string que contenga un formato JSON.
	Ejemplo:
	{'parameters':[{'param':nombre del namespace}, {'param':'nombre de la clase }, {'param':nombre del metodo}, {'param':'data'}]}
	
	La función "ICallbackFunction ()" utiliza "eval()" para ejecutar la función "serverFunction ()" y así realizar la llamada al método RaiseCallbackEvent del Servidor. Revisar caso de uso incluido para ver el funcionamiento en el servidor.
	Ejemplo:
	`eval("serverFunction(serializarJson(options.namespace, options.clase, options.method, options.data));");`
	
	Se ejecuta la función "clientFunction()" que contiene como parámetro el argumento devuelto por el servidor.
	La función "clientfunction()" utiliza "eval()" para ejecutar la función de éxito (proporcionada en el punto 3).
	Ejemplo:
	`eval(successIcallback)(data.data);`
	
	Se ejecuta la función de éxito.

Alternative Flows:	7a. En el paso 7 de normal flow, si el servidor retorna que sucedió un error, entonces: La función "clientFunction()" utiliza "eval()" para ejecutar la función de error (proporcionado en el punto 3). Ejemplo: `eval(errorIcallback)(data.Message);` Se ejecuta la función de error.
Exceptions:	3a. En el paso 3, no se envía uno de estos argumentos necesarios: namespace, class, method, función de éxito o función de error. La función "ICallbackFunction()" no realiza ninguna llamada al servidor y retorna un error.
Notes and Issues:	La función "ICallbackFunction()" puede tener en "data" cero o más de un argumento. Si no se desea enviar argumentos al servidor, no es necesario que se cree "data" en "ICallbackFunction()".

Tabla 20: Utilizar Función Estándar

Fuente: Elaboración Propia, 2014

En la Tabla 21 se muestra la especificación de caso de uso de "Utilizar Método Personalizable"

Use Case ID:	UC-2.0.0
Use Case Name:	Utilizar Método Personalizable
Description:	Este caso de uso permite al sistema ejecutar el método RaiseCallbackEvent(*) en el lado del servidor luego de ser invocado desde el cliente mediante la llamada de la función "serverFunction()". El servidor envía un resultado (cadena string) mediante el método GetCallbackResult (*). El objetivo, es ocultar los métodos de la interface ICallbackEventHandler: RaiseCallbackEvent y GetCallbackResult de tal forma que se tenga una comunicación directa con un método personalizado(**). Así mismo, proporciona las siguientes mejoras: Permite la creación de métodos propios en el lado del servidor, a diferencia de la forma regular que solo permite la utilización de los métodos RaiseCallbackevent y GetCallbackResult. Eliminar la utilización de un método que devuelva el resultado del servidor al cliente, a diferencia de la forma regular que solo devuelve los datos utilizando GetCallbackResult. Permitir la implementación de los métodos del servidor en otras clases, a diferencia de la forma regular que obliga a colocar los métodos en el code behind aspx.cs. (*) Método proporcionado por Microsoft en su interface ICallbackEventHandler. (**) Método creado por el programador, puede tener cualquier nombre e infinita cantidad de parámetros, pero debe ser de tipo string.
Trigger:	El casos de uso lo desencadena la llamada a la función "serverFunction()".
Preconditions:	La aplicación debe tener referenciada la dll del proyecto (iCallback.dll). Se debe crear una instancia de la clase "Callback" en "OnPreInit()" de la página aspx.cs. Ejemplo:

	protected override void OnPreInit(EventArgs e)
	{
	Callback cb = new Callback();
	}

Postconditions:	Se envía un string a la función javascript “clientFunction()” del cliente.
Normal Flow:	Se realiza una llamada al método “RaiseCallbackEvent()”. Dentro del método “RaiseCallbackEvent()” se deserealiza el parámetro string y se convierte al tipo “Params”. Se identifica la ubicación del método mediante “Reflection” utilizando los valores: namespace, clase, method. Ejemplo: Tabla 21: Utilizar Método Personalizable (Continúa) pe = e(parametro.parameters[0].param + "." + parametro.parameters[1].param); MethodInfo claseWebMethod = claseWebType.GetMethod(parametro.parameters[2].param.ToString()); Se guarda en un vector de objetos los valores de “data”. Se invoca al método (del punto 3) utilizando “Invoke()” y se guarda el resultado en una variable. Ejemplo: claseWebMethod.Invoke(Activator.CreateInstance(claseWebType), arguments) Se ejecuta la función “GetCallbackResult()” para devolver un resultado a la función del lado del cliente “clientFunction()”.
Exceptions:	3a. En el paso 3, no se encontró alguna coincidencia. El método “RaiseCallbackEvent()” retorna un error a la función “clientFunction()”. 5a. En el paso 5, la cantidad de parámetros enviada no es la misma que la del método. El método “RaiseCallbackEvent()” retorna un error a la función “clientFunction()”.

Tabla 21: Utilizar Método Personalizable

Fuente: Elaboración Propia, 2014

5.2.5 Matriz CUN vs CUS

La matriz que se muestra en la Tabla 22 permite ver que CUS se desprenden del modelado del negocio.

Nombre de caso de uso de negocio	Nombre de caso de uso de sistema
CUN2: Crear herramienta para desarrollo de SW	- Utilizar Método Personalizable - Utilizar Función Estándar Retorno

Tabla 22: Matriz CUN vs CUS

Fuente: Elaboración Propia, 2014

5.2.6 Matriz Requerimientos Funcionales VS CUS

La matriz que se muestra en la Tabla 23 permite ver que requerimientos funcionales forman el CUS.

CUS/RF	RF 01	RF 02	RF 03	RF 04	RF 05	RF 06	RF 07	RF 08	RF 09
CUS1				↓	↓	↓	↓	↓	↓
CUS2	↓	↓	↓						

Tabla 23: Matriz de CUS vs Requerimientos Funcionales

Fuente: Elaboración Propia, 2014

La Tabla 24 muestra la leyenda de la matriz de casos de uso contra requerimientos funcionales.

Leyenda	
CUS	Caso de uso de sistema
RF	Requerimiento funcional
↓	El requerimiento pertenece al CUS.

Tabla 24: Leyenda de la Matriz de CUS vs Requerimientos Funcionales

Fuente: Elaboración Propia, 2014

5.3 Modelo Conceptual del sistema

En esta sección se muestran las clases con las que contara el sistema. Este proyecto no utiliza una BD relacional hecha en Microsoft SQL Server, Oracle o algún otro sistema de gestión de base de datos debido a que no necesita guardar información, es por eso que las clases que se muestran son las que tendrá la programación.

5.3.1 Clases del sistema

En la Figura 34 se muestran las clases del sistema, junto con sus variables y métodos, tanto privados como públicos.

No se muestran los nombres reales de las variables y métodos, debido a que es muy difícil asegurar como se pueden llamar y cuantos métodos y variables existirán, en la etapa de desarrollo del sistema se irán colocando según se necesite, y posiblemente se agregaran nuevas clases que permitirán mejorar el código.

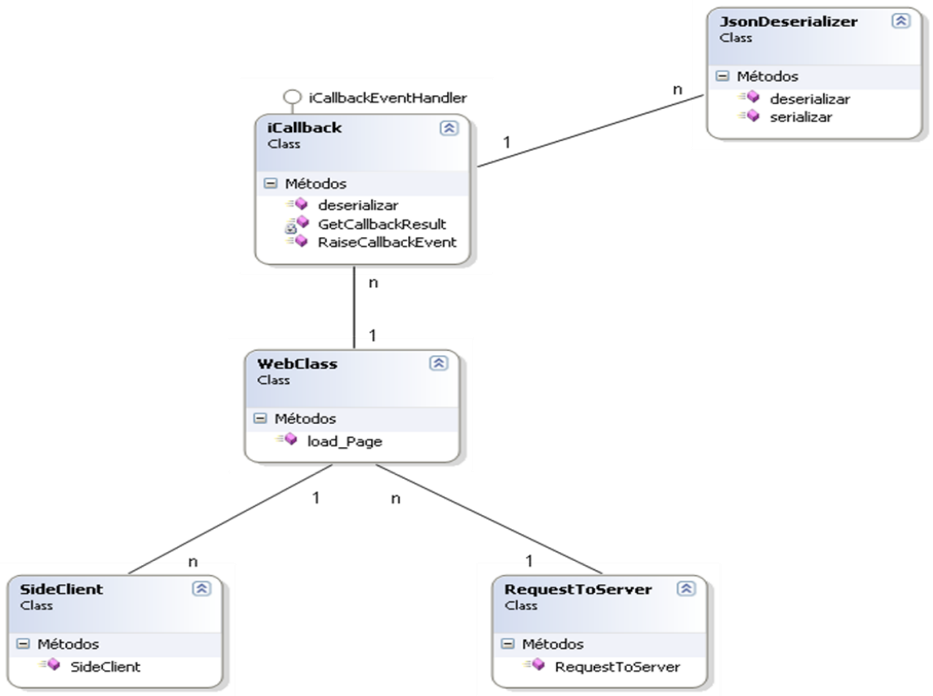

Figura 34: Diagrama de Clases del Sistema

Fuente: Elaboración Propia, 2014

Lo que se debe resaltar de este diagrama es la clase principal llamada iCallback, la cual hereda de la interfaz iCallbackEventHandler que es la interfaz que permite realizar la tecnología de iCallback.

5.3.2 Diccionario de datos

Se muestra a continuación una pequeña descripción de cada clase, se ha agrupado en dos grupos para describir las clases.

Primer grupo lo compone la clase principal iCallback.cs: esta clase es la más importante, es la que permite enlazar las peticiones de los usuarios con la clase correcta, además es la encargada de devolver el resultado al usuario.

El segundo grupo la componen todas las demás clases, cada una de ellas como su nombre lo dice, permiten la correcta implementación de la solicitud del usuario.

5.4 Prototipo de la solución

En la Figura 35 y 36 se muestran los prototipos de lo que podría ser la aplicación. El tipo de dato que intercambia con el servidor está en formato Json, el servidor debe interpretar el formato y obtener las variables. En la url se coloca el nombre del método al cual el script va a enviar los datos, el método debe estar listo para recibir como parámetro, todos los argumentos que se le pasan, en este caso debería existir un método de la siguiente forma: public string nombreMetodo (string dato){ }.

```
function enviarDatos(dato) {
    var datosJson = "{'dato': '" + dato + "'}";
    $.iCallback({
        url: "nombreMetodo",
        data: datosJson,
        dataType: "json",
        success: function (msg) { funcionSucces(msg.d); },
        error: function (result) { funcionError(msg.d); }
    });
}
```

Figura 35: Prototipo de la Función jQuery iCallback

Fuente: Elaboración Propia, 2014

Dependiendo de la respuesta del método, la función ejecuta success o error. En caso de que todo vaya bien y sin errores, se ejecuta la función funcionSucces(msg.d), caso contrario se ejecuta la función funcionError(msg.d). Cada una de estas funciones puede mostrar el resultado en pantalla como guste. En las siguientes imágenes se puede observar esto.

```
function funcionSucces(msg) {
    //muestra el mensaje de la forma que desee, Ejemplo:
    alert(msg);
}
function funcionError(msg) {
    //muestra el error de la forma que desee, Ejemplo:
    $("#error").html(msg); //suponiendo que hay una etiqueta con ID = error
    $("#error").show("slow");
}
```

Figura 36: Prototipo de Función Success y Error

Fuente: Elaboración Propia, 2014

CAPÍTULO VI: ARQUITECTURA

6.1 Vista Lógica

Dirigida a los usuarios finales, es una vista que representa la funcionalidad del software. Para expresar la funcionalidad, se utiliza el diagrama de clases.

En la figura 37 se muestra el diagrama de clases.

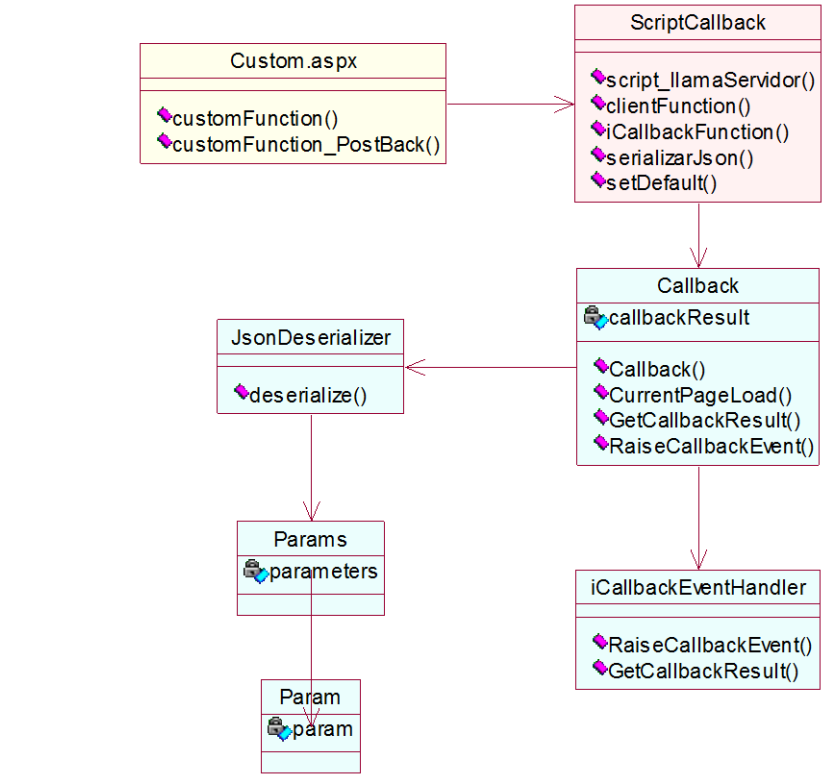

Figura 37: Diagrama de Clases

Fuente: Elaboración Propia, 2014

En la Figura 38 se muestra el diagrama de secuencia que nos permite ver como se carga la página web por primera vez.

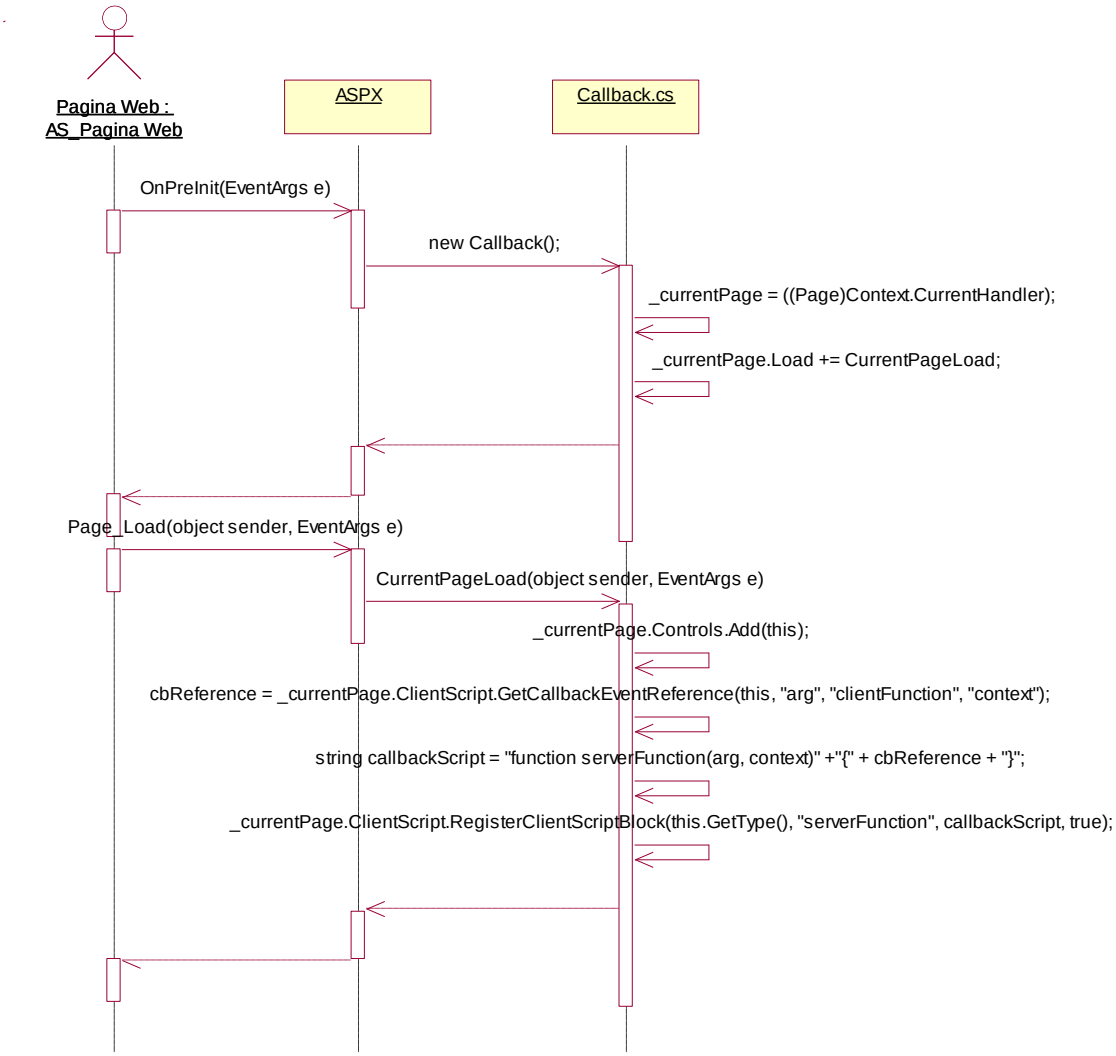

Figura 38: Diagrama de Secuencia – Primera Carga

Fuente: Elaboración Propia, 2014

En la Figura 39 se muestra el diagrama de secuencia que nos permite ver el funcionamiento de la página web en el lado del cliente, cada vez que se realiza un evento asíncrono.

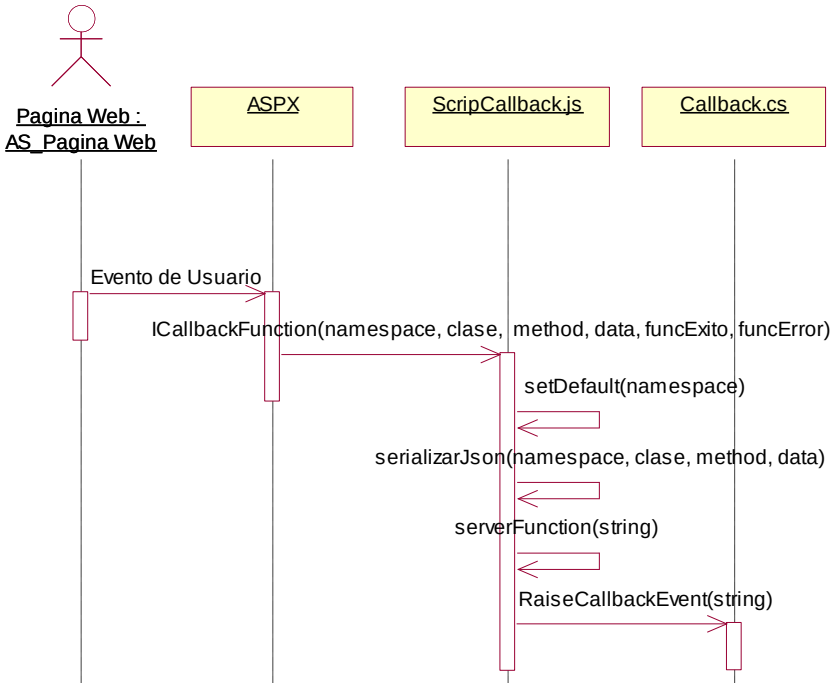

Figura 39: Diagrama de Secuencia – CUS: Utilizar Función Estándar

Fuente: Elaboración Propia, 2014

La Figura 40 muestra el funcionamiento desde el lado del servidor.

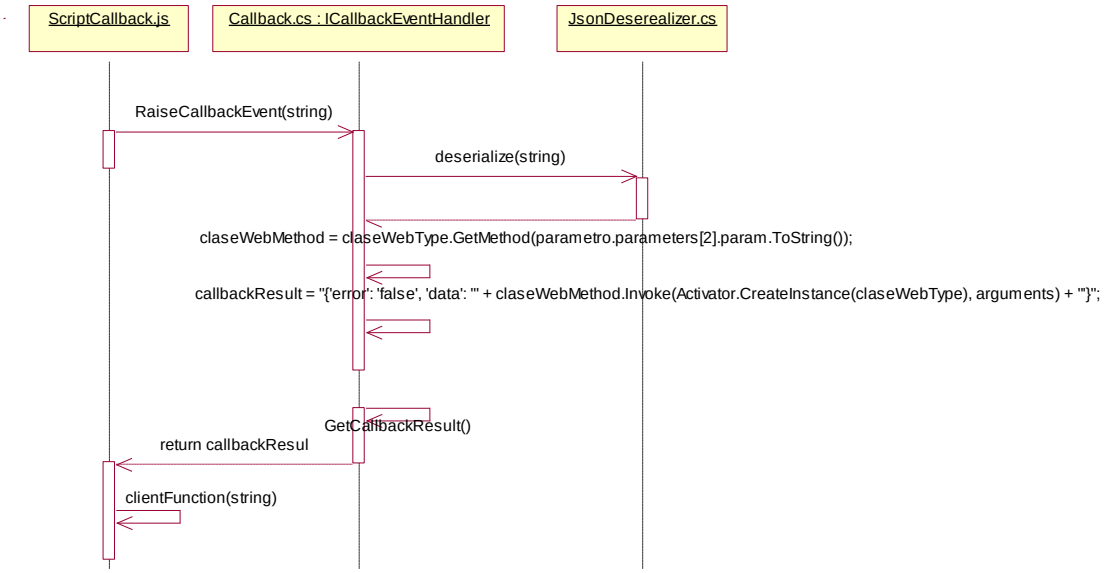

Figura 40: Diagrama de Secuencia – CUS: Utilizar Método Personalizable

Fuente: Elaboración Propia

Se tiene una clase principal llamada Callback, que es invocada desde el cliente mediante una función JavaScript, esta hereda de la interfaz ICallbackEventHandler e implementa sus dos métodos GetCallbackResult y RaiseCallbackEvent. Utiliza la clase JsonDeserealizer para deserializar los valores que envía el cliente (Custom.aspx). Una vez procesada la información, esta es devuelta al cliente (hacia otra función JavaScript).

6.2 Vista de Desarrollo

Esta vista va dirigida a los programadores y se ocupa de la gestión del software. Para representar esta vista, utilizamos el diagrama de componentes que se muestra en la Figura 41 y 42.

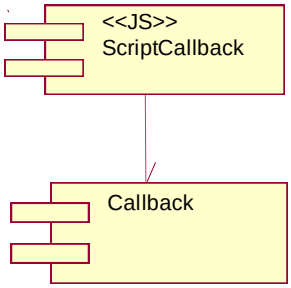

Figura 41: Diagrama de Componentes

Fuente: Elaboracion Propia, 2014

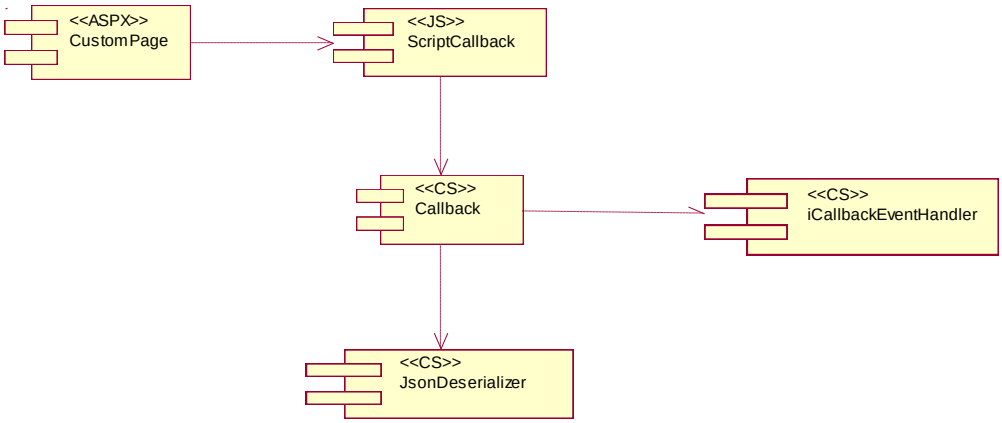

Figura 42: Diagrama de Componentes Fase de Construcción

Fuente: Elaboración Propia, 2014

La Figura 42 representa los componentes del software, se muestran diferentes tipos como:

ASPX: es una página web de Microsoft ASP.Net

JS: es un archivo que contiene código JavaScript

CS: es una clase de .Net con código C#.

El código principal se encuentra en la clase Callback, esta utiliza la interfaz ICallbackEventHandler y la clase JsonDeserealizer. La página CustomPage utiliza las funciones que se encuentran en ScriptCallback para invocar a la clase Callback.

6.3 Vista de Procesos

Esta vista va dirigida a los integradores de sistemas, se utiliza el diagrama de procesos para representar el comportamiento del software.

En la Figura 43 se muestra el diagrama de procesos para la comunicación entre el navegador web y el servidor.

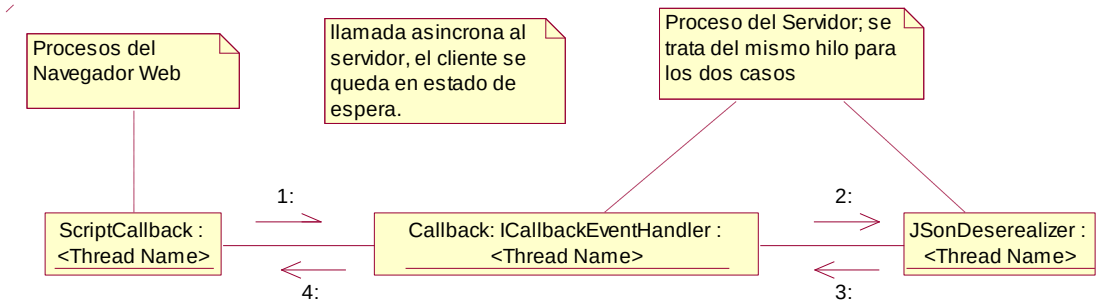

Figura 43: Diagrama de Procesos

Fuente: Elaboración Propia, 2014

6.4 Vista Física

Esta vista va dirigida a los ingenieros de sistemas, normalmente se utilizan los diagramas de despliegue para representar los servidores, pero esta tesis es una herramienta de apoyo para la construcción web, por lo tanto no tiene una arquitectura física.

6.5 Vista Escenarios

Esta vista es la integradora de las demás vistas, utilizaremos el diagrama de casos de uso para representar esta arquitectura de escenarios. En la Figura 44 se muestra el diagrama.

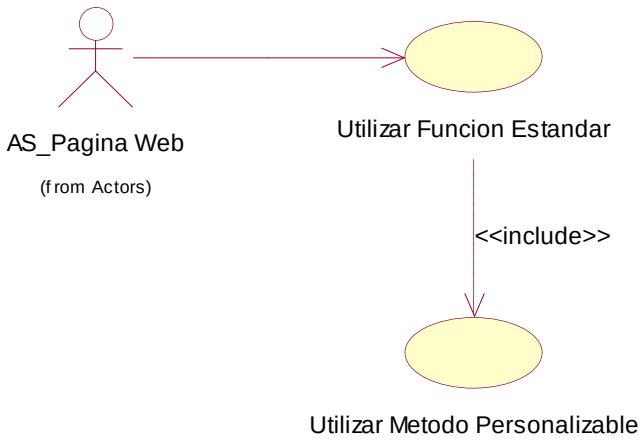

Figura 44: Diagrama de CUS

Fuente: Elaboración Propia, 2014

CAPÍTULO VII: DESARROLLO Y PRUEBAS

7.1 Desarrollo

7.1.1 Construcción del Cliente

En esta sección se muestra la programación realizada en la parte del cliente. El lenguaje que se usa es javascript. Para una mejor visión de lo que se muestra aquí, mirar la siguiente imagen con la estructura javascript que se creará.

El diagrama de clases que se muestra en la Figura 45, permite observar las funciones que intervienen en el lado del cliente.

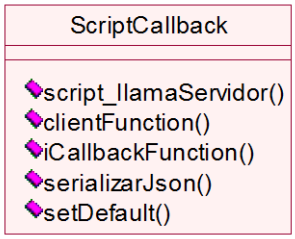

Figura 45: ScriptCallback.js

Fuente: Elaboración Propia, 2014

La Figura 46 muestra las clases y métodos que intervienen.

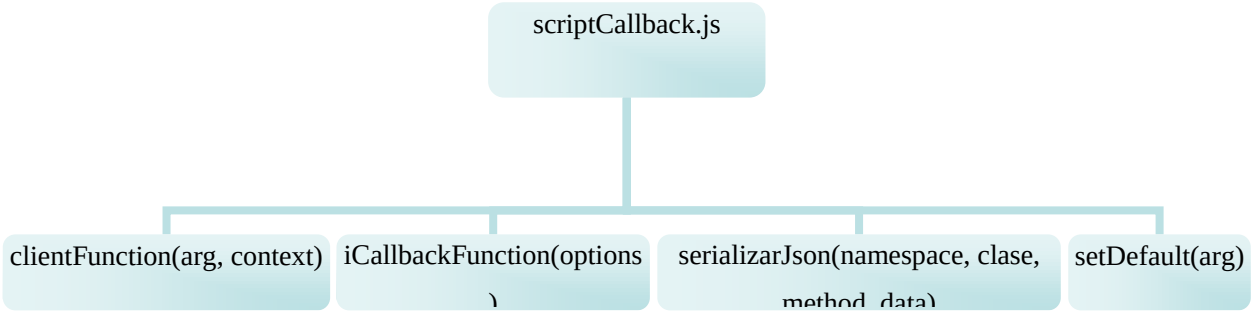

Figura 46: Desarrollo del Cliente

Fuente: Elaboración Propia, 2014

La Tabla 25 muestra el código de ScriptCallback.

```
var successIcallback;
var errorIcallback;

function clientFunction(arg, context) {
    var data = eval("(" + arg + ")");
    if (data.error == "true") {
        eval(errorIcallback)(data.Message);
    }
    else {eval(successIcallback)(data.data);}
}
function iCallbackFunction(options) {
    options.namespace = setDefault(options.namespace);

    successIcallback = options.success;
    errorIcallback = options.error;
    eval("serverFunction(serializarJson(options.namespace,   options.clase,
options.method, options.data));");
}
function serializarJson(namespace, clase, method, data) {

    jsonData = "{'parameters':[{'param':'" + namespace + "'}";
    jsonData += ", {'param':'" + clase + "'}";
    jsonData += ", {'param':'" + method + "'}";

    for (i = 0; i < data.length; i++) {
        if (typeof data[i] == "string") {jsonData += ", {'param':'" +
data[i] + "'}";}
        else {jsonData += "," + JSON.stringify({ param: data[i] });}}
    jsonData += "]}";
    return jsonData;
}
function setDefault(arg) {return arg == null || typeof (arg) ==
'undefined' ? "nulo" : arg;}
```

Tabla 25: ScriptCallback

Fuente: Elaboración Propia, 2014

ClientFunction: Es la función encargada de ejecutar errorIcallback o successIcallback según convenga.

iCallbackFunction: Esta función recibe un parámetro (options) que contiene los valores que serán enviados al método del servidor. El objetivo de esta función es permitir la llamada a la función serverFunction que es la que permite la comunicación con el método del servidor RaiseCallbackEvent.

serializarJson: permite convertir un string en formato Json.

setDefault: Devuelve un string igual a “nulo” siempre que el valor del argumento enviado sea “null” o “undefined”, caso contrario devuelve el mismo valor.

7.1.2 Construcción del Servidor

En esta sección se muestra la programación realizada en la parte del servidor. El lenguaje que se usa es C#. Para una mejor visión de lo que se muestra aquí, mirar la siguiente imagen con los archivos cs que se crearán.

El diagrama de clases que de la Figura 47 permite ver los métodos y variables del cliente.

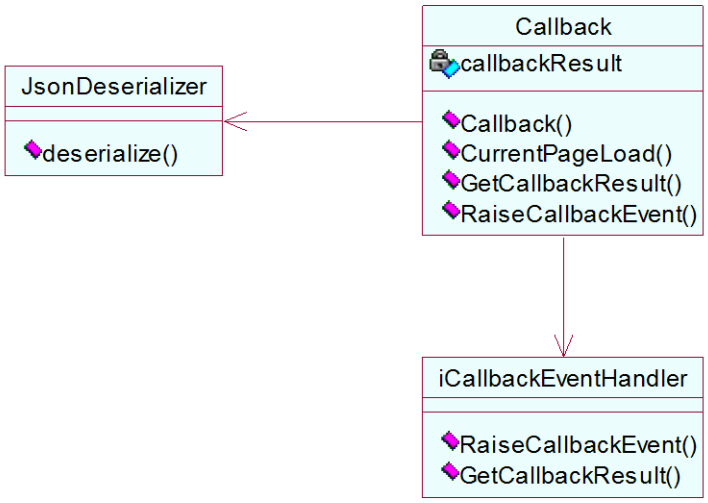

Figura 47: Clases del servidor

Fuente: Elaboración Propia, 2014

Las clases y métodos que intervienen se muestran en las Figuras 48 y 49.

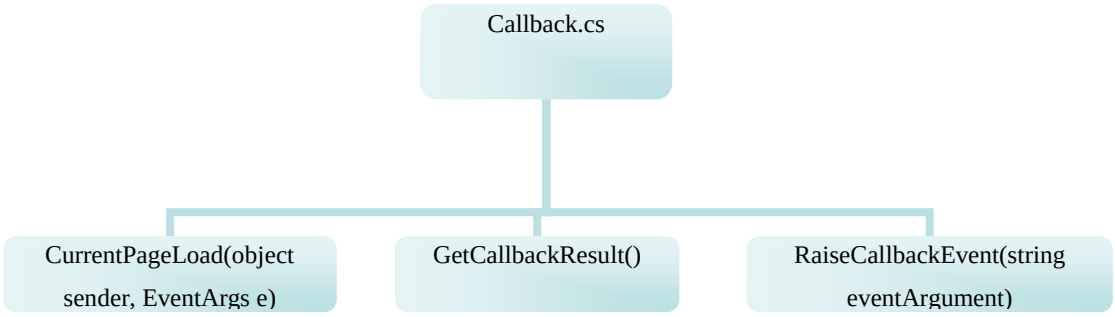

Figura 48: Desarrollo Del Servidor – Callback

Fuente: Elaboración Propia, 2014

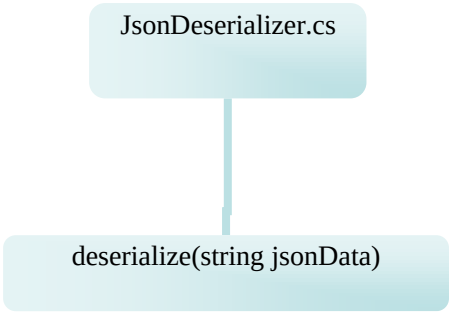

Figura 49: Desarrollo Del Servidor – JsonDeserializer

Fuente: Elaboración Propia, 2014

En la Tabla 26 se muestra el código de Callback.cs.

```
public class Callback : Control, ICallbackEventHandler
{
    private readonly Page _currentPage;
    public Callback()
    {   _currentPage = ((Page)Context.CurrentHandler);
        _currentPage.Load += CurrentPageLoad;}
    void CurrentPageLoad(object sender, EventArgs e)
    { if (_currentPage != null)
        {
            if          (_currentPage.Controls          !=          null)
_currentPage.Controls.Add(this);
            string cbReference =
           _currentPage.ClientScript.GetCallbackEventReference(this,
"arg",
           "clientFunction", "context");
            string callbackScript =
                "function serverFunction(arg, context)" +
                "{" + cbReference + "}";

_currentPage.ClientScript.RegisterClientScriptBlock(this.GetType(),
                "serverFunction", callbackScript, true);
        }
    }private string callbackResult;
    public string GetCallbackResult()
    {return callbackResult;}
    public void RaiseCallbackEvent(string eventArgument)
    { try
        {
            Params parametro = new Params();
            parametro = JsonDeserializer.deserialize(eventArgument);
            Type                        claseWebType                        =
Type.GetType(parametro.parameters[0].param        +        "."        +
parametro.parameters[1].param);
            MethodInfo                    claseWebMethod                    =
claseWebType.GetMethod(parametro.parameters[2].param.ToString());
object[] arguments = new object[parametro.parameters.Count-3];
            int i = 0;
            foreach (var x in parametro.parameters)
            {
                if (i > 2){arguments[i - 3] = x.param;}
                i++;}
          callbackResult    =    "{'error':    'false',    'data':    '"    +
claseWebMethod.Invoke(Activator.CreateInstance(claseWebType),
arguments) + "'}";
        }
        catch (Exception ex)
        {callbackResult    =    "{'error':    'true',    'Message':    '"    +
ex.Message.Replace("'", "\"") + "'}";
        }}}
```

Tabla 26: Callback.cs

Fuente: Elaboración Propia, 2014

Callback: Es el constructor de la clase y permite registrar un evento, el cual permite que se ejecute la llamada al servidor desde el cliente.

CurrentPageLoad: Este método se ejecuta cada vez que el evento registrado se activa. El método tiene el objetivo principal de registrar la función serverFunction y la función clientFunction.

GetCallbackResult: Este método es implementado debido a la interfaz IcallbackEventHandler, su objetivo principal es la de devolver un valor a la función javascript del cliente.

RaiseCallbackEvent: Este método es implementado debido a la interfaz IcallbackEventHandler, su objetivo principal es la de recibir los valores de la función javascript para su posterior tratamiento.

En la Tabla 27 se muestra el código de JsonDeserializer.cs

```
public class JsonDeserializer
{
    public static Params deserialize(string jsonData)
    {
        return new
JavaScriptSerializer().Deserialize<Params>(jsonData);
    }
}
```

Tabla 27: JsonDeserializer.cs

Fuente: Elaboración Propia, 2014

Deserialize: Recibe un string (que viene en formato JSON) y lo deserializa para convertirlo a un objeto Params que es una propiedad creada que contiene una lista con todos los argumentos que el JSON recibe.

7.1.3 Demostración

En esta sección se muestra la forma en que se debe usar las librerías creadas.

Antes que nada, se debe hacer referencia a la dll que permite hacer las llamadas a los métodos y funciones necesarias. Para esto se realizan los siguientes pasos:

Seleccionar Referencias con el click derecho seleccionar agregar referencia (Add Reference). Ver Figura 50.

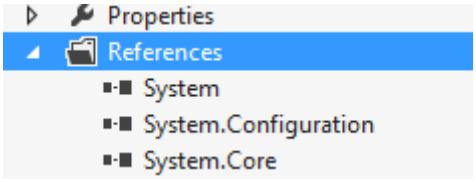

Figura 50: Referencias

Fuente: Elaboración Propia, 2014

Buscar la DLL Callback.dll y seleccionar Aceptar (OK). Ver Figura 51.

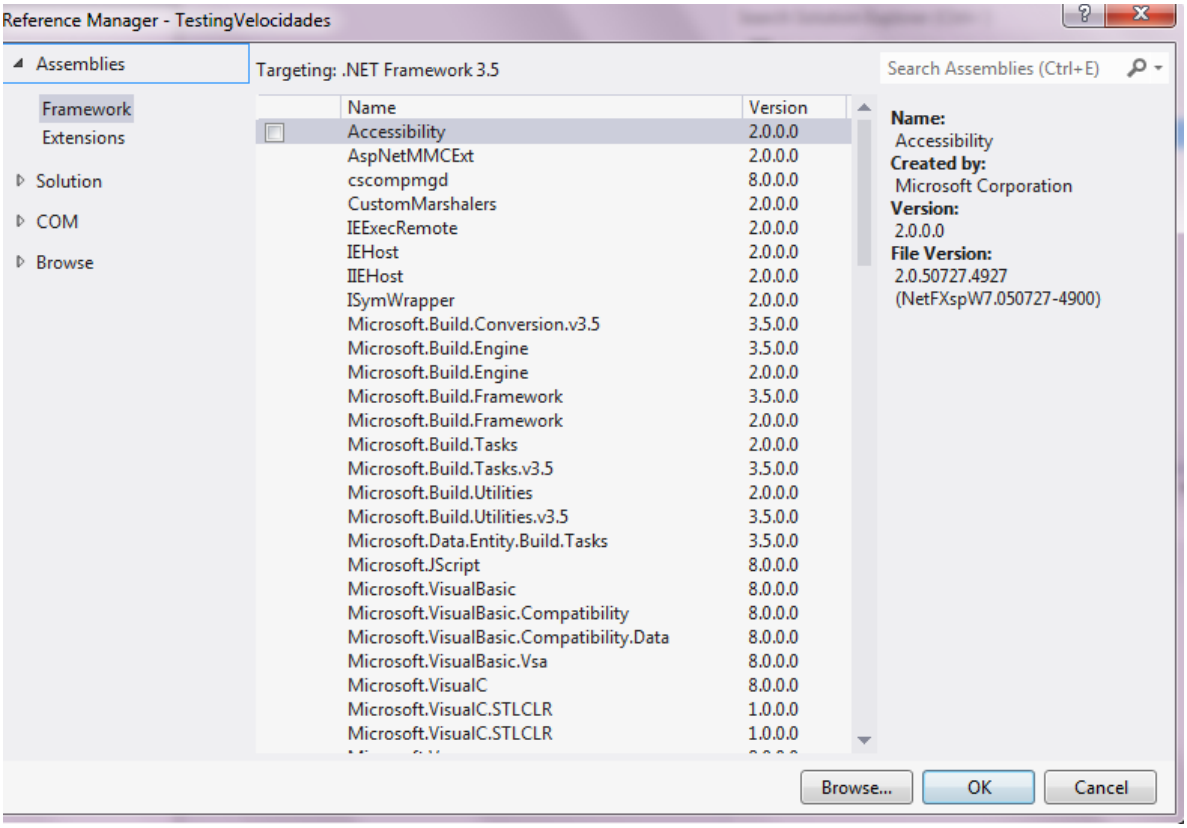

Figura 51: Agregar Referencias

Fuente: Elaboración Propia, 2014

También es necesario agregar el archivo ScriptCallback al proyecto, Con toda esta configuración inicial, ya se pueden seguir los siguientes pasos para utilizar la librería.

En la Tabla 28 se muestra el código demo en el lado del cliente.

```
function funcionDemo(arg) {
            iCallbackFunction({
                namespace: "NameSpace",
                clase: "Clase",
                method: "Metodo",
                success: function (arg) { success_Postback(arg); },
                error: function (error) { error_Postback(arg); },
                data: (new Array(arg1, arg2, arg3))
            });
        }

        function success_Postback(res) {
            //hacer algo
        }

        function error_Postback(res) {
            //hacer algo
        }
```

Tabla 28: Demo Lado del Cliente

Fuente: Elaboración Propia, 2014

Se utiliza la notación literal de JavaScript para acceder a la llamada de la función, se envían los siguientes parámetros:

nameSpace: Es el espacio de nombres (NameSpace) de la ubicación de la clase a la cual se está tratando de acceder.

Clase: Es el nombre de la clase que contiene el método al cual tratamos de acceder.

Method: Es el nombre del método al que tratamos de acceder.

Success: Es la función que se ejecuta si no hay ningún error.

Error: Es la función que se ejecuta si se encuentra algún error.

Data: Contiene un arreglo con todas las variables que se deben enviar al método. La cantidad de argumentos que recibe debe ser la misma cantidad que el método al que se llama.

success_Postback: Esta función se ejecuta si todo ha salido bien en el código, es decir que no se encontraron excepciones. El nombre de esta función puede variar a gusto del usuario.

error_Postback: Esta función se ejecuta si se encontró alguna excepción. El nombre de esta función puede variar a gusto del usuario.

En el código behind, se debe colocar el contenido de la Tabla 29.

```
protected override void OnPreInit(EventArgs e)
        {
            Callback cb = new Callback();
        }
```

Tabla 29: OnPreInit

Fuente: Elaboración Propia, 2014

OnPreInit: Se debe crear una instancia de la clase Callback, de esta forma registramos el evento necesario para ejecutar llamadas asíncronas. Ver Tabla 30.

```
private string Metodo(string arg1, string arg2, string arg3)
        {
            return arg1 + arg2 + arg3;
        }
```

Tabla 30: Método

Fuente: Elaboración Propia, 2014

Este es el método al que se está tratando de acceder, notar que recibe tres parámetros y que son exactamente la misma cantidad que los argumentos enviados.

7.2 Pruebas

7.2.1 Artefactos de pruebas

7.2.1.1 Módulos del programa

En esta sección se muestran los módulos que se pretenden probar, además de las especificaciones de las pruebas a realizar en cada uno. Cabe notar, que cada módulo representa un componente del sistema. En la Tabla 31 se muestra los módulos del programa.

Modulo	Pruebas	Descripción
ScriptCallback.js	Funcionales	Este tipo de prueba asegura el trabajo apropiado de los requerimientos funcionales, incluyendo la entrada de datos, procesamiento y obtención de resultados.
Callback.cs	Funcionales	

Tabla 31: Módulos del Programa

Fuente: Elaboración Propia, 2014

7.2.2 Características a ser probadas

En esta sección se encuentran las características de la herramienta a ser probadas con un caso de estudio específico. Ver Tabla 32.

Característica	Descripción	Módulo
Requerimientos Funcionales	Se debe utilizar el documento de casos de uso para tener claro los casos de éxito y fallo, y si la herramienta cumple con ellos.	Los módulos donde se puede probar esta característica son: ScriptCallback.js Callback.cs

Tabla 32: Características a Ser Probadas

Fuente: Elaboración Propia, 2014

7.2.3 Aproximación

En esta sección se exponen los tipos de pruebas a utilizar para la herramienta, cada una de ellas presenta un formato, el cual se va registrar los resultados.

7.2.4 Pruebas Funcionales

El requerimiento es aprobado si este cumple con lo que está escrito en la especificación de requerimientos. En la Tabla 33 se muestra la infromacion para las pruebas funcionales.

NOMBRE	Pruebas Funcionales	IDENTIFICADOR	FT01
ACTIVIDADES	Análisis de requerimientos del sistema		
TIEMPO ESTIMADO	45 – 60 minutos por unidad		
MÉTODOS O HERRAMIENTAS	Visual Studio		
ENTREGABLES	Lista de chequeo sobre el cumplimiento del requerimiento, ¿realiza lo que el requerimiento describe?		

Tabla 33: Pruebas Funcionales

Fuente: Elaboración Propia, 2014

7.2.5 Procesos de Prueba

Cada cuadro está asociado a un requerimiento. En las Tablas 34, 35, 36, 37, 38, 39, 40, 41 y 42 se muestran los casos de pruebas a realizar:

NOMBRE	Permitir la creación de métodos propios en el lado del servidor	PRUEBAS	P1
PROPÓSITO	El sistema debe permitir la comunicación del cliente con el servidor mediante otro método que no sea RaiseCallbackEvent, este otro método puede tener cualquier nombre definido previamente por el programador.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		

Tabla 34: Procesos de Prueba – Creación de Métodos Propios

Fuente: Elaboración Propia, 2014

NOMBRE	Eliminar la utilización de un método que devuelva el resultado del servidor al cliente	PRUEBAS	P2
PROPÓSITO	Actualmente se utiliza el método GetCallbackResult para devolver información al cliente, se debe evitar esto y se debe utilizar el mismo método creado por el programador para tener la comunicación con el cliente.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		

Tabla 35: Procesos de Prueba – Eliminación del Método que Devuelve el Resultado del servidor

Fuente: Elaboración Propia, 2014

NOMBRE	Permitir la implementación de los métodos del servidor en otras clases	PRUEBAS	P3
PROPÓSITO	Actualmente los métodos que se comunican con el cliente desde el servidor, siempre deben ir en el WebForm donde se programa la llamada asíncrona. Ahora se debe permitir colocar estos métodos en cualquier clase, incluso en cualquier otro namespace.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1.		

	Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.

Tabla 36: Procesos de Prueba – Implementación de los Métodos del Servidor

Fuente: Elaboración Propia, 2014

NOMBRE	Permitir enviar más de un argumento al servidor	PRUEBAS	P4
PROPÓSITO	El método RaiseCallbackEvent solo recibe un parámetro, se desea que ahora se puedan enviar múltiples parámetros al servidor.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4) y más de 1 parámetro. Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		
Notas:	Realizar pruebas con: 0 argumentos 1 argumento 5 argumentos 10 argumentos 25 argumentos 50 argumentos 100 argumentos 200 argumentos		

Tabla 37: Procesos de Prueba – Enviar Más de Un Argumento al Servidor

Fuente: Elaboración Propia, 2014

NOMBRE	Permitir enviar argumentos que sean tanto números como cadenas de texto	PRUEBAS	P5
PROPÓSITO	Actualmente solo se puede enviar un parámetro al servidor y este necesariamente tiene que ser de tipo string. Se desea que además del tipo string, se pueda enviar también argumentos de tipo numérico (int, double, decimal, float).		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		

Tabla 38: Procesos de Prueba – Enviar Argumentos

Fuente: Elaboración Propia, 2014

NOMBRE	Tener una función estándar para comunicarse con el servidor	PRUEBAS	P6
PROPÓSITO	En el PageLoad se define la función que se comunica con el servidor, ahora lo que se desea es que se cree una función estándar que sea la que se comunique con el servidor y que esta no sea declarada en el PageLoad.		

PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.

Tabla 39: Procesos de Prueba – Función Estándar

Fuente: Elaboración Propia, 2014

NOMBRE	Poder elegir que función va a recibir la información del servidor	PRUEBAS	P7
PROPÓSITO	Actualmente en el PageLoad se define la función que recibe la información del servidor, ahora se desea que la función sea definida desde el cliente.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		

Tabla 40: Procesos de Prueba – Función que Recibe la Información

Fuente: Elaboración Propia, 2014

NOMBRE	Controlar el manejo de errores	PRUEBAS	P8
PROPÓSITO	No se manejan las excepciones que se puedan encontrar en el servidor, se desea cambiar esto y tener un control de errores.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor y provocar una excepción. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		

Tabla 41: Procesos de Prueba – Controlar el Manejo de Errores

Fuente: Elaboración Propia, 2014

NOMBRE	Poder elegir la función que se ejecuta si se encuentra algún error	PRUEBAS	P9
PROPÓSITO	Se desea poder elegir que función se debe ejecutar cada vez que se encuentra una excepción en el servidor.		
PRERREQUISITOS	Agregar la dll iCallback.dll y hacer referencia a ella en el proyecto. Agregar el archivo ScriptCallback.js y hacer referencia a él.		
PASOS	Crear una función javascript (de cualquier nombre). Dentro de la función realizar una llamada a iCallbackFunction(). Crear un botón y asociar la función javascript del punto 1. Crear un método personalizado en el servidor y provocar una excepción. Colocar el nombre del namespace (del punto 4), nombre de la clase (del punto 4), nombre del método (del punto 4). Agregar en OnPreInit una instancia de la clase Callback(). Compilar la aplicación Hacer click en el botón.		

Tabla 42: Procesos de Prueba – Función de Error

Fuente: Elaboración Propia, 2014

7.2.6 Reporte de Pruebas

En la Tabla 43 se muestra el reporte de pruebas.

TestName	Status	Observación
Permitir la creación de métodos propios en el lado del servidor	PASSED	No se encontró ningún problema.
Eliminar la utilización de un método que devuelva el resultado del servidor al cliente	PASSED	Se comprobó que ahora solo se utiliza un método para recibir y devolver la información.
Permitir la implementación de los métodos del servidor en otras clases	PASSED	Se comprobó que los métodos no necesariamente deben estar en la aspx.cs.
Permitir enviar más de un argumento al servidor	PASSED	Se probó con: 0, 1, 5, 10, 25, 50, 100 y 200 argumentos. Se detectó que a partir del argumento 100 el rendimiento de la aplicación decae.
Permitir enviar argumentos que sean tanto números como cadenas de texto	PASSED	Se comprobó que se pueden enviar tipos numéricos y cadenas de texto.
Tener una función estándar para comunicarse con el servidor	PASSED	Se comprobó el uso de la función estándar.
Poder elegir que función va a recibir la información del servidor	PASSED	Se comprobó el uso de una función que recibe los datos del servidor.
Controlar el manejo de errores	PASSED	Se comprobó el manejo de errores.
Poder elegir la función que se ejecuta si se encuentra algún error	PASSED	Se comprobó que se puede usar una función propia para controlar los errores.

Tabla 43: Reporte de Pruebas

Fuente: Elaboración Propia, 2014

CAPÍTULO VIII: GESTIÓN DEL PROYECTO

8.1 Estudio de Factibilidad

En esta sección se ven las posibilidades técnicas, económicas y legales del proyecto para que se lleve a cabo.

8.1.1 Viabilidad técnica

Aquí se describen los recursos tecnológicos necesarios para el desarrollo del proyecto y/o funcionamiento. La Tabla 44 muestra la viabilidad técnica.

	Recurso	Cantidad
Hardware	PC de escritorio Intel Core I3. 2GB Ram. 120GB de disco Duro.	2
Software	Visual Studio 2008	2
	Visual Studio 2010	2
	Visual Studio 2012	2
	Rational Rose	2
	Rational Requisite Pro	2
	Librería Jquery	2
Muebles, inmuebles y otros medios de comunicación	Conexión eléctrica	2
	Conexión a internet	2

Tabla 44: Viabilidad Técnica

Fuente: Elaboración Propia, 2013

8.1.2 Viabilidad económica

Se considera que un proyecto es viable económicamente cuando, una vez alcanzada su capacidad de producción plena, es capaz de obtener de su actividad, una vez deducidos todos los costes, un excedente (beneficio) suficiente para hacer al coste de la deuda, la remuneración de lo invertido y la financiación.

El proyecto tiene una duración estimada de seis meses, por lo que se considera para las estimaciones económicas los cálculos de la Tabla 45.

	Recursos	Mes						Totales
		Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	
Recursos Humanos	Analista programador	500	500	500	500	500	500	
	Analista Programador	500	500	500	500	500	500	
	Costo total recursos humanos	1000	1000	1000	1000	1000	1000	6000
Hardware	1 PC de Escritorio	3300	0	0	0	0	0	
	1 PC de Escritorio	3300	0	0	0	0	0	
	Costo total Hardware	6600	0	0	0	0	0	6600
Software	Visual Studio 2005 Express	0	0	0	0	0	0	
	Visual Studio 2008 Express	0	0	0	0	0	0	
	Visual Studio 2010	0	0	0	0	0	0	
	Visual Studio 2012 Express	0	0	0	0	0	0	
	Librería JQuery	0	0	0	0	0	0	
	Costo total Software	0	0	0	0	0	0	0
Muebles, inmuebles y otros medios de comunicación	Conexión eléctrica	80	80	80	80	80	80	
	Conexión a internet	120	120	120	120	120	120	
	Costo total de muebles…	200	200	200	200	200	200	1200
Total								13800

Tabla 45: Viabilidad Económica 1

Fuente: Elaboración Propia, 2013

Para recuperar el dinero invertido, se piensa en dar capacitaciones y soporte a las empresas o usuarios que utilicen la dll propuesta. Con una estimación promedio de diez personas al mes que necesiten soporte y un costo de 250 nuevos soles por el servicio, se ha realizado la siguiente tabla.

Tomar en cuenta para la Tabla 46 que el costo por cliente es igual a 250 nuevos soles con lo cual para calcular el total del mes se multiplican la cantidad de 10 clientes esperados por 250 soles.

	Año 1											
	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
N° Clientes (Pesimista)	1	1	1	1	1	1	1	1	1	1	1	1
N° Clientes (Esperado)	10	10	10	10	10	10	10	10	10	10	10	10
N° Clientes (Optimista)	20	20	20	20	20	20	20	20	20	20	20	20
Total x Mes (Pesimista)	250	250	250	250	250	250	250	250	250	250	250	250
Total x Mes (Esperado)	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500
Total x Mes (Optimista)	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
Ganancias (Pesimista)	250	500	750	1000	1250	1500	1750	2000	2250	2500	2750	3000
Ganancias (Esperado)	2500	5000	7500	10000	12500	15000	17500	2000	22500	25000	27500	30000
Ganancias (Optimista)	5000	10000	15000	20000	25000	30000	35000	40000	45000	50000	55000	60000

Tabla 46: Viabilidad Económica 2

Fuente: Elaboración Propia, 2013

La Tabla 47 muestra el flujo de fondos para los diez primeros periodos.

Período	Flujo de Fondos
0	-13800
1	2500
2	2500
3	2500
4	2500
5	2500
6	2500
7	2500
8	2500
9	2500
10	2500

Tabla 47: Flujo de Fondos

Fuente: Elaboración Propia, 2013

La Tabla 48 el resultado del VAN y TIR.

TIR	12.57%	
VAN	S/.	597.56

Tabla 48: VAN y TIR

Fuente: Elaboración Propia, 2013

Según el grafico de las tablas, podemos observar que se necesitan 09 meses para que se recupere lo invertido, debido a que al mes se obtienen 2500 soles. También se puede observar que el valor de la tasa es menor y el cálculo del VAN es mayor a 0, indicándonos que el proyecto es rentable.

8.1.3 Viabilidad legal

La creación de programas usando el lenguaje C# y su compilador es gratuito, lo que tiene un costo, es el uso del IDE. En la Tabla 49 se muestran los términos de uso en Visual Studio 2013:

DERECHOS DE INSTALACIÓN Y USO.
Instalación y Uso. Un usuario puede instalar y usar en sus dispositivos cualquier número de copias del software con el fin de diseñar, desarrollar y probar sus programas.
Restricciones a la Distribución. No podrá:
• utilizar las marcas comerciales de Microsoft en los nombres de sus programas o de forma que sugiera que sus programas proceden de Microsoft o están respaldados por Microsoft;
• incluir Código Distribuible en programas malintencionados, engañosos o ilegales;

Tabla 49: Viabilidad Legal: Derechos de Instalación y Uso

Fuente: Elaboración Propia, 2013

Microsoft coloca de forma gratuita la distribución y comercialización de software producido bajo el IDE Visual Studio.

8.2 Organización del proyecto

8.2.1 Organigrama del proyecto

En la Figura 52 se muestra el organigrama del proyecto.

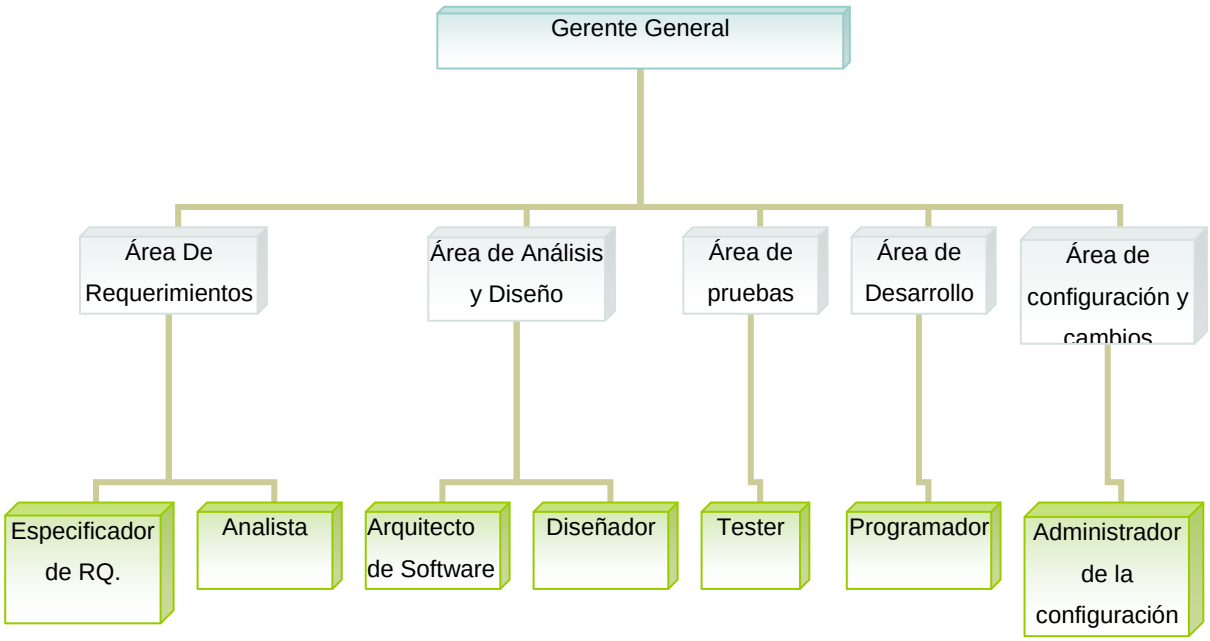

Figura 52: Organigrama del Proyecto

Fuente: Elaboración Propia, 2013

8.2.2 EDT del proyecto

El EDT del proyecto permite saber los entregables que tendrá esta tesis. Se ha considerado la fase de gestión del proyecto, que es la que permite organizar el proyecto, en esta fase se formula el proyecto, se realizan los pasos de planeación, seguimiento y control y se finaliza con el cierre del proyecto, es decir la aceptación final de que el proyecto ha concluido correctamente.

Luego se tienen las fases de Modelado del negocio, requerimientos, Análisis y diseño, pruebas, desarrollo y finaliza con la implementación.

El objetivo del EDT es saber los entregables que se incorporaran en el cronograma. Se puede apreciar el EDT en la Figura 53.

Figura 53: EDT del

Fuente: Elaboración Propia, 2013

8.3 Estimación y Ejecución del proyecto

En esta sección se muestra el cronograma y la estimación de costos del proyecto.

8.3.1 Cronograma de la ejecución del proyecto

En la Figura 54 se muestra el cronograma del proyecto.

	WBS		Task Name	Duration	Start	Finish
1	**1**	✓	**Mejoramiento de ICallbackEventHandler mediante una herramienta basada en Reflection y JavaScript**	**489 days**	**Tue 01/01/13**	**Fri 14/11/14**
2	1.1	✓	INICIO	0 days	Tue 01/01/13	Tue 01/01/13
3	**1.2**	✓	**Elaboracion del Plan de Tesis**	**14 days**	**Tue 01/01/13**	**Fri 18/01/13**
4	**1.2.1**	✓	**Introduccion**	**5 days**	**Tue 01/01/13**	**Tue 08/01/13**
5	1.2.1.1	✓	Marco Situacional	1 day	Tue 01/01/13	Tue 01/01/13
6	1.2.1.2	✓	Problematizacion	1 day	Wed 02/01/13	Wed 02/01/13
7	1.2.1.3	✓	Objetivos	1 day	Thu 03/01/13	Thu 03/01/13
8	1.2.1.4	✓	Importancia	1 day	Fri 04/01/13	Fri 04/01/13
9	1.2.1.5	✓	Metodología	1 day	Sat 05/01/13	Sat 05/01/13
10	1.2.1.6	✓	Hipótesis	1 day	Sun 06/01/13	Sun 06/01/13
11	1.2.1.7	✓	Sumario o Esquema	1 day	Mon 07/01/13	Mon 07/01/13
12	1.2.1.8	✓	Hito Introduccion Fianalzado	0 days	Tue 08/01/13	Tue 08/01/13
13	**1.2.2**	✓	**Marco Teórico**	**2 days**	**Wed 09/01/13**	**Thu 10/01/13**
14	1.2.2.1	✓	Antecedentes	1 day	Wed 09/01/13	Wed 09/01/13
15	1.2.2.2	✓	Buscar Información	1 day	Thu 10/01/13	Thu 10/01/13
16	1.2.2.3	✓	Hito Marco Teórico Finalizado	0 days	Thu 10/01/1	Thu 10/01/13
17	**1.2.3**	✓	**Aspectos Operativos**	**0 days**	**Sat 12/01/13**	**Mon 14/01/13**
18	1.2.3.1	✓	Cronograma de Trabajo	1 day	Sat 12/01/13	Sat 12/01/13
19	1.2.3.2	✓	Fuentes de Información	1 day	Sun 13/01/13	Sun 13/01/13
20	1.2.3.3	✓	Hito Aspectos Operativos Finalizado	0 days	Mon 14/01/13	Mon 14/01/13
21	**1.2.4**	✓	**Revision del Plan de Tesis**	**3 days**	**Mon 07/01/13**	**Wed 09/01/13**
22	1.2.4.1	✓	Reunion con el Asesor	1 day	Tue 15/01/13	Tue 15/01/13
23	1.2.4.2	✓	Corregir observaciones del asesor	1 day	Wed 16/01/13	Wed 16/01/13
24	1.2.4.3	✓	Hito Revision del Plan de Tesis Finalizado	1 day	Thu 17/01/13	Thu 17/01/13

Figura 54: Cronograma de la Ejecución del Proyecto (Continúa)

25	**1.2.5**	✓	**Presentacion del Plan de Tesis**	**1 day**	**Fri 18/01/13**	**Fri 18/01/13**
26	1.2.5.1	✓	Entregar el Plan de Tesis	1 day	Fri 18/01/13	Fri 18/01/13
27	1.2.5.2	✓	Hito Presentación del Plan de Tesis Finalizado	0 days	Fri 18/01/13	Fri 18/01/13
28	**1.3**	✓	**Elaboracion del Proyecto de Tesis**	**465 days**	**Sat 02/02/13**	**Fri 14/11/14**
29	1.3.1	✓	Capitulo I: Vision del Proyecto	1 day	Sat 02/02/13	Sat 02/02/13
30	1.3.2	✓	Antecedentes del Problema	1 day	Sun 03/02/13	Sun 03/02/13
31	1.3.3	✓	Fundamentacion del Problema	1 day	Mon 04/02/13	Mon 04/02/13
32	1.3.4	✓	Objetivos del Proyecto	1 day	Tue 05/02/13	Tue 05/02/13
33	1.3.5	✓	Importancia	1 day	Fri 08/02/13	Fri 08/02/13
34	1.3.6	✓	Alcance	1 day	Fri 15/02/13	Fri 15/02/13
35	1.3.7	✓	Viabilidad	1 day	Wed 20/02/13	Wed 20/02/13
36	1.3.8	✓	Revision del Capitulo I	1 day	Thu 28/03/13	Thu 28/03/13
37	1.3.9	✓	Corregir Observaciones del Capitulo I	5 days	Sun 31/03/13	Thu 04/04/13
38	1.3.10	✓	Hito Capitulo I Finalizado	0 days	Mon 04/03/13	Mon 04/03/13
39	**1.3.11**	✓	**Capitulo II: Marco Teórico**	**18 days**	**Sun 05/05/13**	**Wed 29/05/13**
40	1.3.11.	✓	Fundamentos Basico	5 days	Sun 05/05/13	Thu 09/05/13
41	1.3.11.	✓	Revision del capitulo II	1 day	Wed 15/05/13	Wed 15/05/13
42	1.3.11.	✓	Corregir Observaciones del Capitulo II	10 days	Thu 16/05/13	Wed 29/05/13
43	1.3.11.	✓	Hito Capitulo II Finalizado	0 days	Wed 29/05/13	Wed 29/05/13
44	**1.3.12**	✓	**Capitulo III: Estado del Arte**	**25 days**	**Wed 29/05/13**	**Tue 02/07/13**
45	1.3.12.	✓	Softwares existentes	25 days	Wed 29/05/13	Tue 02/07/13
46	1.3.12.	✓	Revision del Capitulo III	1 day	Mon 03/06/13	Mon 03/06/13
47	1.3.12.	✓	Corregir Observaciones del Capitulo III	15 days	Tue 04/06/13	Mon 24/06/13
48	1.3.12.	✓	Hito Capitulo III Finalizado	0 days	Mon 24/06/13	Mon 24/06/13

44	**1.3.12**	✓	**Capitulo III: Estado del Arte**	**25 days**	**Wed 29/05/13**	**Tue 02/07/13**
45	1.3.12.	✓	Softwares existentes	25 days	Wed 29/05/13	Tue 02/07/13
46	1.3.12.	✓	Revision del Capitulo III	1 day	Mon 03/06/13	Mon 03/06/13
47	1.3.12.	✓	Corregir Observaciones del Capitulo III	15 days	Tue 04/06/13	Mon 24/06/13
48	1.3.12.	✓	Hito Capitulo III Finalizado	0 days	Mon 24/06/13	Mon 24/06/13
49	**1.3.13**	✓	**Capitulo IV: Modelado del Negocio**	**34 days**	**Mon 24/06/13**	**Thu 08/08/13**
50	1.3.13.	✓	Casos de Uso del Negocio	5 days	Mon 24/06/13	Fri 28/06/13
51	1.3.13.	✓	Diagrama de Actividades del Negocio	5 days	Fri 28/06/13	Thu 04/07/13
52	1.3.13.	✓	Revision del Capitulo IV	1 day	Thu 04/07/13	Thu 04/07/13
53	1.3.13.	✓	Corregir Observaciones del Capitulo IV	25 days	Fri 05/07/13	Thu 08/08/13
54	**1.3.14**	✓	**Presentacion del Entregable I**	**19 days**	**Mon 12/08/13**	**Thu 05/09/13**
55	1.3.14.	✓	Revisar Entregable I	1 day	Mon 12/08/13	Mon 12/08/13
56	1.3.14.	✓	Recepcion de Observaciones	1 day	Thu 15/08/13	Thu 15/08/13
57	1.3.14.	✓	Corregir Observaciones	15 days	Fri 16/08/13	Thu 05/09/13
58	1.3.14.	✓	Hito Presentacion del Entregable I Finalizado	0 days	Thu 05/09/13	Thu 05/09/13
59	**1.3.15**	✓	**Capitulo V: Requerimientos del Proyecto**	**19 days**	**Thu 05/09/13**	**Tue 01/10/13**
60	1.3.15.	✓	Requerimientos de Software	1 day	Thu 05/09/13	Thu 05/09/13
61	1.3.15.	✓	Casos de Uso del Sistema	4 days	Fri 06/09/13	Wed 11/09/13
62	1.3.15.	✓	Modelo Conceptual del Sistema	4 days	Thu 12/09/13	Tue 17/09/13
63	1.3.15.	✓	Benchmarking	2 days	Wed 18/09/13	Thu 19/09/13
64	1.3.15.	✓	Prototipos de la solucion	2 days	Fri 20/09/13	Mon 23/09/13
65	1.3.15.	✓	Revisión del Capitulo V	1 day	Tue 24/09/13	Tue 24/09/13
66	1.3.15.	✓	Corregir Observaciones del Capitulo V	5 days	Wed 25/09/13	Tue 01/10/13
67	1.3.15.	✓	Hito Capítulo V Finalizado	0 days	Tue 01/10/13	Tue 01/10/13

Figura 54: Cronograma de la Ejecución del Proyecto (Continúa)

68	**1.3.16**	✓	**Capítulo VI: Arquitectura**	**18 days**	**Thu 03/10/13**	**Mon 28/10/13**
69	1.3.16.	✓	Realizacion de los Casos de Uso mas significativos	1 day	Thu 03/10/13	Thu 03/10/13
70	1.3.16.	✓	Modelo de Datos	1 day	Fri 04/10/13	Fri 04/10/13
71	1.3.16.	✓	Modelo de Despliegue	1 day	Sat 05/10/13	Sat 05/10/13
72	1.3.16.	✓	Modelo de Componentes	1 day	Sun 06/10/13	Sun 06/10/13
73	1.3.16.	✓	Revision del Capitulo VI	1 day	Mon 07/10/13	Mon 07/10/13
74	1.3.16.	✓	Corregir Observaciones del Capitulo VI	15 days	Tue 08/10/13	Mon 28/10/13
75	1.3.16.	✓	Hito Capitulo VI Finalizado	0 days	Mon 28/10/13	Mon 28/10/13
76	**1.3.17**	✓	**Capítulo VII: Desarrollo y Pruebas**	**239 days**	**Mon 28/10/13**	**Thu 25/09/14**
77	**1.3.17.**	✓	**CUS 1: Utilizar Función Estándar**	**74 days**	**Mon 28/10/13**	**Thu 06/02/14**
78	1.3.17.	✓	Desarrollar CUS 1	60 days	Mon 28/10/13	Fri 17/01/14
79	1.3.17.	✓	Test Case CUS 1	15 days	Fri 17/01/14	Thu 06/02/14
80	**1.3.17.**	✓	**CUS 2: Utilizar Metodo Personalizable**	**74 days**	**Fri 07/02/14**	**Wed 21/05/14**
81	1.3.17.	✓	Desarrollar CUS 2	60 days	Fri 07/02/14	Thu 01/05/14
82	1.3.17.	✓	Test Case CUS 2	15 days	Thu 01/05/14	Wed 21/05/14
83	**1.3.18**	✓	**Capitulo VIII: Gestion del Proyecto**	**126 days**	**Fri 23/05/14**	**Fri 14/11/14**
84	1.3.18.	✓	Organizacion del Proyecto	1 day	Fri 23/05/14	Fri 23/05/14
85	1.3.18.	✓	Estimacion y Ejecución del proyecto	1 day	Sat 24/05/14	Sat 24/05/14
86	1.3.18.	✓	Gestion de Riesgos del Proyecto	1 day	Sun 25/05/14	Sun 25/05/14
87	1.3.18.	✓	Plan de Cambios en el Proyecto	1 day	Mon 26/05/14	Mon 26/05/14
88	1.3.18.	✓	Revision del Capitulo VIII	1 day	Tue 27/05/14	Tue 27/05/14
89	1.3.18.	✓	Corregir Observaciones del Capitulo VIII	15 days	Tue 27/05/14	Mon 16/06/14
90	1.3.18.	✓	Hito Capitulo VIII Finalizado	0 days	Tue 27/05/14	Tue 27/05/14

90	1.3.18.	✓	Hito Capitulo VIII Finalizado	0 days	Tue 27/05/14	Tue 27/05/14
91	**1.3.18.**	✓	**Conclusiones, Bibliografia**	**3 days**	**Tue 27/05/14**	**Thu 29/05/14**
92	1.3.18.	✓	Elaborar Conclusiones y Bibliografia	1 day	Tue 27/05/14	Tue 27/05/14
93	1.3.18.	✓	Revisar Conclusiones y Bibliografia	1 day	Wed 28/05/14	Wed 28/05/14
94	1.3.18.	✓	Corregir Conclusiones y Bibliografia	1 day	Thu 29/05/14	Thu 29/05/14
95	1.3.18.	✓	Hito Conclusiones y Bibliografia Finalizado	1 day	Thu 29/05/14	Thu 29/05/14
96	**1.3.18.**	✓	**Presentacion Final del Proyecto de Tesis**	**76 days**	**Thu 10/07/14**	**Thu 23/10/14**
97	1.3.18.	✓	Revisar Entregable Final	15 days	Thu 10/07/14	Wed 30/07/14
98	1.3.18.	✓	Recepcion de Observaciones	1 day	Fri 15/08/14	Fri 15/08/14
99	1.3.18.	✓	Corregir Observaciones	50 days	Fri 15/08/14	Thu 23/10/14
100	1.3.18.	✓	Hito Presentacion del Entregable Finalizado	0 days	Thu 23/10/14	Thu 23/10/14
101	1.3.18.	✓	**Sustentacion del Proyecto de Tesis**	1 day	Fri 14/11/14	Fri 14/11/14
102	1.3.18.	✓	FIN	0 days	Fri 14/11/14	Fri 14/11/14

Figura 54: Cronograma de la Ejecución del Proyecto

Fuente: Elaboración Propia, 2014

CONCLUSIONES

Durante esta etapa de elaboración del proyecto, se han llegado a las siguientes conclusiones.

Se llegó a cumplir con el objetivo de reducir a cuatro pasos la implementación de ICallbackEventHandler y además cada paso es más óptimo.

El proyecto permite realizar páginas Web, utilizando la interfaz iCallbackEventHandler de Microsoft de una manera más fácil, permitiendo a los programadores familiarizarse rápidamente con las funciones de JavaScript de lado del cliente y los métodos de C# de lado del servidor.

Se puede usar varios métodos personalizables en la aplicación, ocultando a los programadores la complejidad de usar los métodos RaiseCallBackEvent y GetCallBackResult, además se encapsula la lógica interna que hace las llamadas asíncronas.

Se puede crear varios parámetros de diferentes tipos en los métodos personalizables.

Se reduce el tiempo y la dificultad en implementar varias llamadas asíncronas.

Se puede controlar las formas de controlar las excepciones, para que el sistema sea más robusto y tolerante a fallos

Se utiliza en el Front-End una función estándar "ICallBackFunction" para que tenga una comunicación con el servidor.

Además, con Reflection se puede encontrar cualquier detalle sobre un objeto assembly, además de poder invocar métodos y obtener o establecer valores de una propiedad, todo esto en tiempo de ejecución. También, su correcto y adecuado uso nos permite diseñar, crear, usar excepciones de forma más eficiente. Para módulos o componentes de librerías que incluyen excepciones personalizadas, el implementar correctamente las propiedades de

Exception, facilita el uso de nuestro producto a programadores suministrándole información completa sobre la falla o error que ha generado el uso incorrecto del componente en alguna sección de código.

RECOMENDACIONES

Dentro de un proyecto diferente como lo fue éste, siempre se desea que haya una mejora continua del mismo; por lo tanto se recomienda a futuros estudiantes que tengan interés en el proyecto para que puedan complementar el sistema con más aportes para facilitar aún más el uso de esta Interfaz, debido a que esta interfaz está presente en muchos controles de asp.net y en muchas tecnologías de Microsoft.

GLOSARIO DE TÉRMINOS

.CS	Extensión usada para indicar que es un documento que contiene en su interior el lenguaje c#.
.Net	Es un framework de Microsoft.
AJAX	Es una técnica de desarrollo web, que permite hacer una consulta al servidor y mostrar la respuesta en pantalla sin necesidad de recargar la página.
ASP	Es una tecnología de Microsoft del tipo “lado del servidor” para páginas web generadas dinámicamente.
Body	Es la etiqueta de un documento HTML, en la cual se colocan etiquetas que mostraran la información que el usuario ve en pantalla.
C#	Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.
CSS	Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.
CUN	Perteneciente al UML, el caso de uso del negocio es usado para describir el negocio de una empresa u organización.
CUS	Es un conjunto de requerimientos funcionales, es utilizado para describir a un sistema informático.
DOM	Es una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en

documentos.

ECMAScript	Es una especificación de lenguaje de programación publicada por ECMA International.
EDT	Es una descomposición jerárquica orientada a los entregables que tiene un proyecto.
Framework	Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, con base a la cual otro proyecto de *software* puede ser más fácilmente organizado y desarrollado.
Function	Para efectos de esta tesis, se refiere a un método javascript.
Head	Es una etiqueta perteneciente a un documento HTML.
Helper	En la programación, es una clase de ayuda que permite hacer código reutilizando uno ya existente.
HTTP	Es el protocolo usado en cada transferencia de la WWW.
iFrame	Es un elemento HTML que permite insertar o incrustar un documento HTML dentro de un documento HTML principal.
JavaScript	Es un lenguaje de programación interpretado.
Json	Es un formato ligero para el intercambio de datos.
Postback	Termino que se refiere al hecho de recargar una página web
Tester	Persona que realiza pruebas a los programas computacionales.

SIGLARIO

AJAX	Asynchronous JavaScript And XML
ASP	Active Server Pages
BD	Base de datos
C#	C Sharp
CSS	Cascading Style Sheets
CUN	Caso de uso del negocio
CUS	Caso de uso del sistema
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
EDT	Estructura de descomposición de trabajo
FCL	Framework Class Library
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MathML	Mathematical Markup Language

SGML	Estándar Gerealized Markup Language
SQL	Structured query language
SSJS	Server-side JavaScript
UML	Unified Modeling Language
VS	Visual Studio
W3C	World Wide Web Consortium
XHTML	eXtensible Hypertext Markup Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

REFERENCIAS BIBLIOGRÁFICAS

LIBROS

Albahari, J., & Albahari, B. (2010). *C# 4.0 in a nutshell.* Sebastopol: O'reilly.

Arsenovski, D. (2009). *Refactoring in C# & ASP.NET.* Indiana: Wiley Publishing.

Chaffer, J., & Swedberg, K. (2013). *Learning JQuery Four Edition.* Birningham: Packt Publishing Ltd. Livery Place 35 Livery Street Birningham.

Deitel, H., & Deitel, P. (2010). *C# for Programmers (4th Edition).* Boston: Prentice Hall.

Deitel, P., & Deitel, H. (2010). *Javascript for programmers.* Crawfordsville: Prentice Hall.

Fawcett, J., Ayers, D., & Quin, L. (2012). *Beginning XML.* Indiana: Wrox.

Ferguson, R., & Heilmann, C. (2013). *Beginning Javascript with DOM Scripting and AJAX.* Nueva York: Apress.

Freeman, E., & Robson, E. (2014). *Head First JavaScript Programming.* Sebastopol: O'Reilly Media.

Garimella, K., lees, M., & Williams, B. (2009). *Book: BPM Basics for dummies.* Indianapolis: Wiley Publishing.

MacDonald, M., Freeman, A., & Szpuszta, M. (2010). *Pro ASP.NET 4 in C# 2010, 4th Edition.* Nueva York: Apress.

Sawyer McFarland, D. (2012). *JavaScript & Jquery : The Missing Manual, Second Edition.* Sebastopol: O'Reilly.

Solis, D. (2012). *Illustrated C# 2012.* New York: Apress.

Suehring, S. (2013). *Javascript Step by Step Third Edition.* Sebastopol: O'Reilly.

PAPERS

Hanakawa, N. (2013). *An Intelligent Web Browser Plug-In for Automatic Translation to Ajax Approach*. Obtenido de http://www.techrepublic.com/resource-library/whitepapers/an-intelligent-web-browser-plug-in-for-automatic-translation-to-ajax-approach/

Jeon, J., & Lee, S. (2007). *Position Paper: Toward a Mobile Rich Web*. Obtenido de http://www.w3.org/2007/06/mobile-ajax/papers/etri.jeon.MobileAJAX-PositionPaper-r5.pdf

Okur, S., Hartveld, D., & Arie van, D. (2014). *A Study and Toolkit for Asynchronous Programming in C#*. Obtenido de http://dig.cs.illinois.edu/papers/ICSE14_Semih.pdf