UNIVERSIDAD NACIONAL DE SAN CRISTOBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA DE FORMACIÓN PROFESIONAL DE INGENIERÍA DE SISTEMAS

"FRAMEWORK PARA APLICACIONES ORIENTADAS A SERVICIOS DE EMPRESAS CONSULTORAS DE SOFTWARE, LIMA 2015"

Informe del Trabajo Profesional presentado por : Bach. Herbert Jhonathan Mendoza Suxe

Para optar el título profesional de : Ingeniero de Sistemas

Área de investigación : Ingeniería de Software

Ayacucho, Abril del 2015

"FRAMEWORK PARA APLICACIONES ORIENTADAS A SERVICIOS DE EMPRESAS CONSULTORAS DE SOFTWARE, LIMA 2015"

RECOMENDADO : 11 DE AGOSTO DEL 2015

APROBADO : 17 DE SETIEMBRE DEL 2015

Mg. AVELINO T. PALMA GUTIERREZ
PRESIDENTE

Ing. EDITH F. GUEVARA MOROTE
MIEMBRO

Ing. MANUEL A. LAGOS BARZOLA
MIEMBRO

MSc. Ing. EFRAIN E. PORRAS FLORES
MIEMBRO

Ing. FLORO N. YANGALI GUERRA
SECRETARIO DOCENTE

Según el acuerdo constatado en el Acta, levantada el 17 de Setiembre de 2015, en la sustentación del Informe Profesional presentado por el Bachiller en Ingeniería de Sistemas Sr. Herbert Jhonathan MENDOZA SUXE, con el Informe Profesional titulado "FRAMEWORK PARA APLICACIONES ORIENTADAS A SERVICIOS DE EMPRESAS CONSULTORAS DE SOFTWARE, LIMA 2015", fue calificado con la nota de DIECISEIS (16) por lo que se da la respectiva APROBACIÓN.

RECOMENDADO : 11 DE AGOSTO DEL 2015

APROBADO : 17 DE SETIEMBRE DEL 2015

Mg. AVELINO T. PALMA GUTIERREZ
PRESIDENTE

Ing. EDITH F. GUEVARA MOROTE
MIEMBRO

Ing. MANUEL A. LAGOS BARZOLA
MIEMBRO

MSc. Ing. EFRAIN E. PORRAS FLORES
MIEMBRO

Ing. FLORO N. YANGALI GUERRA
SECRETARIO DOCENTE

DEDICATORIA

A mis padres Jorge y Luisa por el soporte y la enseñanza que formaron lo que soy.

A mí querida hermana Gladys, su paciencia y la mucha fuerza que me brinda para ser mejor cada día.

A mis queridas sobrinas Kiara y Daphne que son el motor y sentido de inspiración.

Y por supuesto también a mi sobrino Thiago, mis queridas hermanas Mirelly, Amanda y mi hermano Franklin.

A Medalí por su cariño y apoyo brindado para hacer posible este informe.
A mi pequeño Herbert Jr. con mucho cariño y por hacer interesante cada día.

Con aprecio a Simone y Liam por grandes momentos.

AGRADECIMIENTO

En primer lugar, quiero reconocer el esfuerzo y dedicación de todas las personas que me han colaborado y gracias a ellos hace posible la realización de este Informe.

A mi padres Jorge y Luisa por todo su apoyo, cariño y por acompañarme en los momentos buenos y malos durante todos estos años.

A mis hermanos por ser la fuente de mi inspiración y fortaleza para seguir adelante.

CONTENIDO

RESUMEN

El nacimiento de los Sistemas de Información hace que las organizaciones deseen soporte y alcance de sus objetivos, la arquitectura orientada a servicios es un gran paso natural en el desarrollo de Tecnologías de Información. Hoy en día existe la nube, celulares inteligentes, aplicaciones desktop y cada cual consume un servicio, este se puede definir como un valor un ente que es independiente de tecnologías específicas o lenguajes de programación que use la organización.

El concepto de utilizar servicios crea el inconveniente de cómo implementarlo, pone a la organización enfrente y debe de entender la manera de afrontar los nuevos cambios que tenga con la estructura y los objetivos de la organización, como hacer que la funcionalidad de negocio y TI pueda realizar. Soportar los cambios en la organización y procesos como por ejemplo en las compañías que tienen muchos proveedores, en el sector público que tiene muchos cambios en reglas y regulaciones, entregar la funcionalidad necesaria al negocio a tiempo.

Se ha logrado construir un framework de desarrollo que sea capaz de cubrir los cambios de negocio rápidamente, cubriendo las diferentes funcionalidades de negocio. Usando la tecnología NET de Microsoft, el lenguaje orientado a objetos, el administrador de base de datos SQL Server. El informe considera un marco general de desarrollo de aplicaciones Orientadas a servicios para el año 2015.

Palabras clave: NET, SOA, VB.NET, SOAP, WCF, Data Contract, Service Contract, Service Implementation.

INTRODUCCIÓN

El uso de servicios en el desarrollo de soluciones nos permite implementar aplicaciones que son el soporte a las operaciones diarias, administrar y dar soporte a los procesos mediante las Tecnologías de Información, la necesidad de responder a los cambios es difícil ante la ausencia de un modelo que unifique e integre la arquitectura orientada a servicios.

La organización fundamental de un sistema, está relacionado a cada uno de sus componentes y el principal objetivo es el diseño y la evolución. (Dikmans et al, 2012). SOA define un contrato que especifica las condiciones bajo las cuales el servicio puede ser consumido, para los consumidores, el contrato y la interface son visibles e importantes. Los servicios son fáciles de integrar a los ambientes, la interoperabilidad es el principal punto, el uso de estándares para describir, proveer y accesar a los servicios como XML, WSDL, incrementa las plataformas. Dikmans et al. (2012), afirma que, la interoperabilidad es importante porque se necesita integrar servicios en ambientes heterogéneos en el cual diferentes tecnologías y plataformas son usados. Se usará SOA y el lenguaje de programación VB.NET.

Las razones para realizar el informe es construir un framework de desarrollo de aplicaciones orientado a servicios que ayude con la flexibilidad y la interoperabilidad de responder a nuevos cambios dentro de las organizaciones, mediante SOA, protocolos de transmisión y poder brindar un modelo escalable y robusto. El problema del informe es como diseñar un modelo de desarrollo de aplicaciones orientadas a servicios, usando las mejores prácticas de Microsoft, los protocolos de transmisión, haciendo uso de la tecnología NET de Microsoft, para poder brindar flexibilidad.

Los objetivos del informe son; construir un framework de desarrollo de aplicaciones orientadas a servicios, analizar los mejores patrones de desarrollo y aplicar las mejores prácticas recomendadas por Microsoft.

CAPITULOI

PLANTEAMIENTO DEL INFORME PROFESIONAL

1.1 DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

El desarrollo de sistemas informáticos actualmente hace que sea necesario hacer uso de servicios, donde en el caso que cambie el tema funcional del negocio sólo se haga cambios a nivel del servicio, el desarrollo de estos Sistemas actualmente se hace sin poseer un modelo de desarrollo que integre esta tecnología, con la implementación del modelo lograremos integrar todos los componentes involucrados en la creación de aplicaciones orientadas a servicios en todos los puntos brindando un único modelo y dejando sólo la tarea de la programar la parte funcional del negocio, así cada ambiente de programación pueda dedicarse a la parte funcional y sin tener que preocuparse por el modelo que se usaría.

El modelo orientado a servicios es la base para el desarrollo de los sistemas, estos sistemas orientados a servicios son capaces de consumir el servicio y de ser el caso entregar parámetros, recibir resultado o sólo procesar un método, nuestro modelo nos permitirá reducir costos de diseño del marco de desarrollo.

El objetivo de este informe es construir un framework de desarrollo orientado a servicios, para la creación de aplicaciones orientada a servicios, este modelo permitirá que el desarrollo sea más rápido cumpliendo con los mejoras estándares de desarrollo.

1.2 FORMULACIÓNDEL PROBLEMA DEL INFORME PROFESIONAL

PROBLEMA PRINCIPAL

¿De qué manera construir un framework de aplicaciones orientada a servicios para empresas consultoras de software, Lima 2015?

PROBLEMAS ESPECÍFICOS

a. ¿Cómo desarrollar la descomposición lógica que contribuya a brindar métodos que sean consumidos por los clientes?
b. ¿De qué manera desarrollar la herencia de formularios que contribuye a la implementación de funcionalidades de la aplicación?
c. ¿Cómo desarrollar las capas orientadas a servicios que permita la exposición de métodos para el cliente?

1.3 OBJETIVOS DEL INFORME PROFESIONAL

OBJETIVO GENERAL

Construir un framework para aplicaciones orientadas a servicios para empresas consultoras de software, Lima 2015. Mediante el empleo del lenguaje VB.NET, la arquitectura WCF, el enfoque SOA, el administrador de base de datos SQL Server, con el propósito de disminuir tiempo y costo de desarrollo de aplicaciones orientada a servicios y la finalidad de tener un framework para desarrollo de aplicaciones orientadas a servicios.

OBJETIVOS ESPECÍFICOS

a. Analizar, diseñar, implementar y probar la descomposición lógica que contribuye a brindar métodos que sean consumidos por el cliente con el fin de tener métodos de acceso a datos.
b. Analizar, diseñar, implementar y probar la herencia de formularios que contribuya a la implementación de funcionalidades de la aplicación con la finalidad de tener formularios para ingreso operativo, menús y acceso a reportes.
c. Analizar, diseñar, implementar y probar las capas orientadas a servicios que permite la implementación de métodos para el cliente con la finalidad de exponer los métodos del framework.

CAPITULOII
REVISIÓN DE LA LITERATURA

2.1 FRAMEWORK PARA APLICACIONES ORIENTADAS A SERVICIOS

2.1.1 DESCOMPOSICIÓN LÓGICA

Erl. (2009), afirma que la descomposición lógica más simple permite describir la funcionalidad que estos proveen para una solución de un problema individual común.

Erl. (2009), define que la descomposición es útil por:

a. Representa campos de soluciones testeadas para el diseño de problemas comunes.
b. Organiza el diseño inteligentemente dentro de estándares y formatos fácilmente referenciados.
c. Son generalmente repetidos por más profesionales de TI involucrados con el diseño.
d. Pueden ser usados para asegurar consistencia en como los sistemas son construidos y diseñados.
e. Pueden ser aplicados prioritariamente y subsecuentemente a la implementación de un sistema.
f. Puede ser soportado vía la aplicación de otros diseños de patrones que son parte de la misma colección.

A. VALOR PROVEIDO

Cheng (2010), afirma que para cada servicio, considerar si y por qué se necesita, si un servicio no provee un valor a alguien o algo (organizaciones, departamentos internos, otros sistemas de TI) entonces probablemente no es un buen servicio, o solo una parte del mismo.

B. TRASCENDENTE

Gao (2007), define que debe ser fácil en el futuro poder consumir el servicio, sin embargo la interface del servicio necesita ser transcendente para los consumidores y no tan abstracto o complejo. Si un servicio no es trascendente, los esfuerzos para consumirlos crecen. Los consumidores no serán capaces, los consumidores no serán

capaces o se mostrarán reacios para usar el servicio desde que no entiendan esto o es tan caro usarlo e integrarlo dentro de sus ambientes.

C. IMPLEMENTACION OCULTA

Patack (2011), define que cuando este viene de un servicio, los consumidores no tienen cuidado sobre la implementación bajo el servicio, este es una caja negra para ellos. Los consumidores se centran en el contrato y la interfaz del servicio para decidir si lo usan y ser capaces de consumirlo. En resumen, un servicio y especialmente una interface y un contrato, debería ser descrito y entendible.

Cheng (2010), afirma que ocultando los detalles de la implementación es un acercamiento común a los múltiples paradigmas de programación, a menudo más de SOA en el cual nosotros identificamos diferencias entre el contrato, interface y la implementación. La interfaz del servicio debe ser abstracta u oculta y de alguna manera especifiquen a las organizaciones su trabajo actual. Esto hace más sencillo cambiar, actualizar o cambiar la implementación sin romper la interoperabilidad desde la interface que puede permanecer tal cual esta.

D. CONFIABLE

Gao (2007), afirma que los consumidores necesitan tener fe en el servicio que están consumiendo. Ellos quieren servicios para estar seguros, los servicios deben hacer lo que se suponen que hacen, especialmente en SOA donde los servicios pueden provenir de entornos externos, se piensa de aspectos como seguridad, manejos por defectos, auditorias para incrementar la seguridad del servicio.

E. AISLAMIENTO

Erl (2009), afirma que los servicios solo proveen flexibilidad y pueden ser cambiados fácilmente si sus operaciones son independientes de otras operaciones dentro del mismo u otro servicio, esto es llamado aislamiento, si un cambio en una operación resulta en cambios para muchas otras operaciones que están fuertemente acoplados al cambio de la operación original, entonces perdemos flexibilidad. Las operaciones necesitan construir separadamente bloques que provean capacidades así mismos.

F. INTEROPERABILIDAD

Daim (2014), afirma que los servicios ser fáciles de integrar a nuestro

ambiente, la interoperabilidad es una medida para el monto de esfuerzo que toma usar e invocar servicios. La interoperabilidad es alcanzada usando estándares para describir, proveer y accesar servicios como XML, WS-*, WADL y WSDL. El uso de los estándares incrementa la facilidad de consumir servicios desde múltiples plataformas. Se observa que solo la interfaz necesita ser interoperable, la implementación en si misma puede ser propietaria.

Dikmans et al. (2012), afirma que la interoperabilidad es importante en los siguientes casos:

a. Se necesita integrar servicios heterogéneos en ambientes en los cuales diferentes tecnologías y plataformas son usadas.
b. Nuestro servicio es usado fuera de la organización y no se tiene control o conocimiento sobre las plataformas usadas por nuestros consumidores.
c. Se quiere integrar servicios para empaquetar software u ofrecer servicios como parte de un paquete de software.

Dikmans et al. (2012), afirma que, los principios aislamiento, seguridad, idempotencia pueden ser complicado y difícil para ser aplicados, el diseño del servicio como la granularidad y la reusabilidad desde estos aspectos pueden ser construidos a partir del diseño.

2.1.2 HERENCIA DE FORMULARIOS

Según Hamilton (2002), afirma que la herencia permite derivar una clase hija de una clase padre, el comportamiento de la clase padre es heredado por el hijo como posible extensión, la clase padre típicamente contiene operaciones generales, mientras que la clase hija contiene comportamientos especializados, la relación es conocida como relación.

Fischer et al (2002), afirma que NET Framework es el modelo de programación de código gestionado por Microsoft para la creación de aplicaciones sobre plataforma Windows, proporciona el soporte necesario para los servicios Web, depurar e instalar y consumir servicios Web usando los lenguajes del entorno.

Holzner (2002), afirma que los formularios en VB.NET forman una parte fundamental de desarrollo, se puede usar el control de versiones y controlar los permisos de seguridad. Un ensamblado contiene el código necesario para correr los ejecutables.

2.1.3 CAPAS ORIENTADAS A SERVICIOS

Cheng (2010), afirma que SOA involucra un servicio, este es un ente que es usado para proveer algo a un consumidor. Los servicios pueden ser pagados desde que el productor le agrega valor durante la producción, un servicio puede ser algo tangible como un vaso de leche, un carro, un departamento pero también algo intangible como un viaje o un diagnóstico médico, entonces simplemente un servicio está brindando algo, en resumen un servicio es un término económico que describe lo que las organizaciones y personas producen, vender algo a otro y comprar algo de otro.

Gao (2000), afirma que, En el 2010, el consorcio W3C, acepto el lineamiento de Protocolo de Acceso simple de Objetos. Este XML basado en mensaje forma una transmisión estable para aplicaciones interconectadas mediante HTTP, SOA provee una atractiva alternativa para los traiciónales protocolos como CORBA y DCOM.

Dikmans et al. (2012), establece que hay diferentes esquemas de capas usados en diferentes arquitecturas, marcos de desarrollo. Una capa común en la arquitectura es dividirlo en tres principales capas: la capa de negocio, la capa de información, la capa de tecnología. Otras capas pueden ser dividas dentro de las capas. Para conservar manejable usamos tres capas (Negocio, Información y Tecnología).

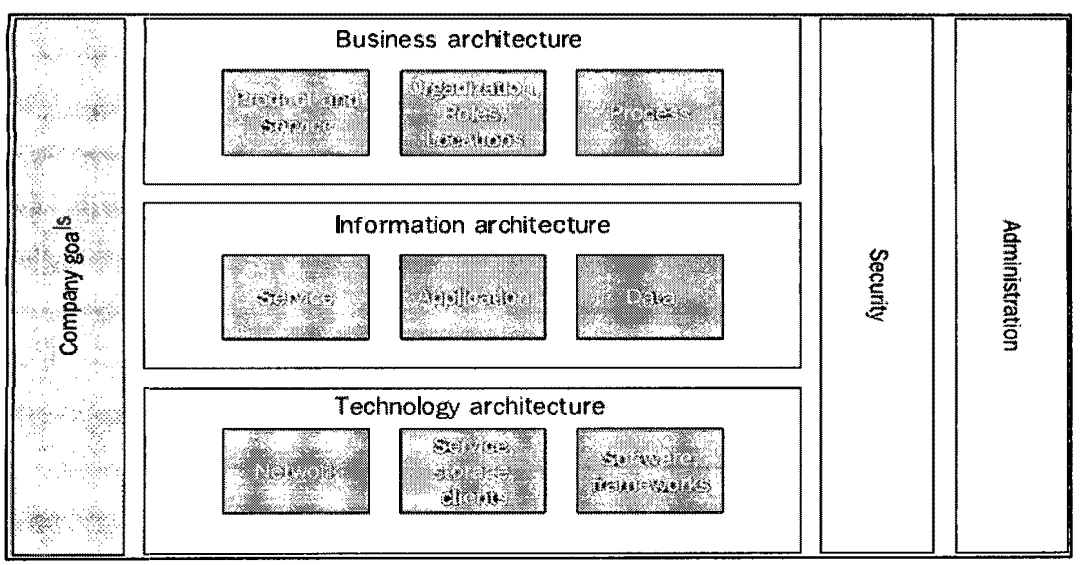

Figura 2.1: Implementación de capas. (Dikmans et al, 2012).

Erl (2009), define que las capas y los aspectos dentro de estas capas pueden ser descritos de diferentes perspectivas: estratégico, táctico u operacional. La vista

seguridad puede ser escrita del punto de vista del área de seguridad. La vista Administración es descrita desde el punto de vista del administrador del sistema. Las otras capas no son escritas desde una perspectiva específica, pero son divididas de acuerdo al tipo de información que es capturado del sistema. Por ejemplo en la capa del negocio, típicamente se encuentra información sobre el proceso de negocio, este puede ser descrito desde diferentes perspectivas, por ejemplo desde la perspectiva de la operación o de la perspectiva del controlador. Lo mismo es verdadero para la información en otras capas.

Liu (2010), menciona que antes de ingresar a la arquitectura orientada a servicios, es necesario definir una arquitectura. La arquitectura es una disciplina que ayuda a las organizaciones a alienar el negocio y las estrategias de la organización a las tecnologías de información. Cada país, industria y comunidad usa una diferente definición.

Dikmas et al. (2012), define a la arquitectura como el fundamento de la organización de un Sistema, personificado en sus componentes, su relación unos con otros y el ambiente, los principios que gobiernan son el diseño y la evolución.
El estándar no toma posición en la pregunta, ¿Que es un sistema?, en el estándar el termino sistema es usado como un contenedor, por ejemplo esto podría referirse a una empresa, un sistema de sistemas, una línea de productos, un servicio, un subsistema, un software.

Gao (2007), afirma que es importante notar que la arquitectura no es igual que la estandarización, esto depende de las estrategias de la compañía o del modelo operacional, cuanta integración y estandarización se necesita y en que procesos, Sistemas y partes de la organización se necesitan. El objetivo de la arquitectura es traducir las estrategias del negocio dentro de una guía apropiada para estandarizar e integrar. La arquitectura es un medio y un fin. Asegurando que las tecnologías de información pueden satisfacer los requerimientos de la organización.

Bieberstein (2008), define que dentro de diferentes perspectivas, hay muchas maneras de romper la organización de un sistema o arquitectura, los sistemas pueden ser divididos dentro de Capas lógicas, vistas.

a. Cuando dividimos cada cosa del negocio, información y tópicos técnicos, estamos

hablando de capas de la arquitectura.

b. Cuando dividimos software en presentación, capa lógica, y acceso a datos, estamos hablando de niveles en el software.
c. Cuando describimos los sistemas diferentemente dependiendo de la perspectiva del negocio, estamos describiendo una vista de la arquitectura desde un punto específico.

A. SERVICIO

Dikmans et al. (2012), menciona que un servicio es una solución lógica, el cual tiene una orientación de servicio para ser aplicado de manera trascendente. Esta orientación diseña principios que distingue una unidad lógica como un servicio comparado a unidades lógicas que pueden existir solamente como objetos o componentes.

Erl (2009), menciona que subsecuentemente al modelo conceptual de un servicio, la orientación a servicios diseña y desarrolla un servicio como un estado independiente de software con específicas características que soporta el logro de las estrategias globales asociadas con la computación orientadas a servicios.

Gao (2007), define que, el concepto de un servicio refiere al componente que provee funcionalidad y es independiente construyendo bloques que pueden ser integrados unos con otros colectivamente para representar una aplicación.

B. CAPACIDAD DE UN SERVICIO

Bieberstein (2008), menciona que cada servicio es asignado con funciones propias en un contexto y es comprendido por un conjunto de funciones o capacidad relacionadas al contexto, sin embargo, un servicio puede ser considerado como un contenedor de capacidades asociadas a propósitos comunes.

Gao (2007), afirma que el termino capacidades de servicio, no tiene implicación como el servicio es implementado, sin embargo este término puede ser especialmente usado durante la modelación de los estados del servicio cuando el diseño físico de un servicio no ha sido terminado. Un servicio existe como un Web Service, Rest, componente, términos como "Métodos del servicio" o "Operaciones del servicio" pueden ser usados.

C. CONSUMIR EL SERVICIO

Erl (2009), afirma que cuando un programa invoca e interacciona con un servicio, se dice que se está consumiendo el servicio, es muy importante entender que este término refiere al rol en tiempo de ejecución que asume un programa cuando está en comunicación con el servicio.

Erl (2009), menciona que cuando se crea una aplicación Windows, que es capaz de usar métodos de un servicio, se dice que la aplicación Windows está consumiendo el servicio. También se puede diseñar un servicio que invoque e interaccione con otros servicios, el cual en otros términos es la composición de servicios, en este caso el servicio actúa temporalmente como Consumidor de un servicio.

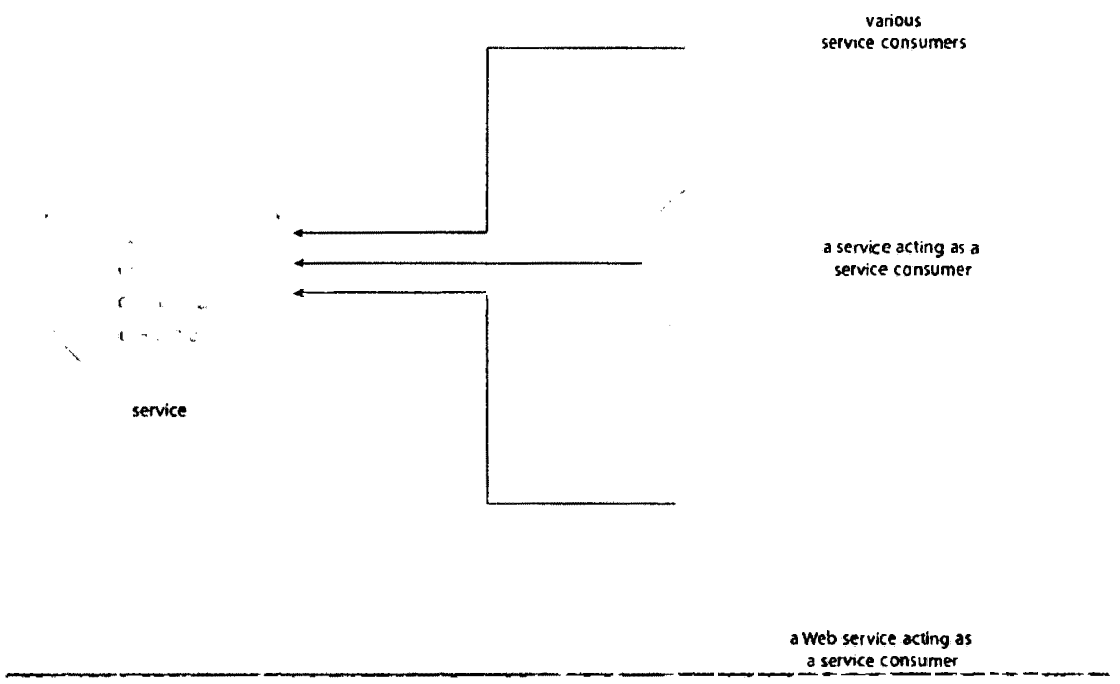

Figura 2.2: Servicio realizar orden, accesado por una variedad de programas actuando como consumidores del servicio. (Erl, 2009).

D. ELEMENTOS DE UN SERVICIO

Bieberstein (2008), define que para cada servicio necesitamos definir ciertas características en orden para que los servicios sean definidos como usables, si el servicio no está bien definido, puede que no sea transparente para todos los consumidores que requieren de los valores que este ofrece, como usar y como compartir

el servicio con todos, esto puede traer confusión y puede causar que los consumidores no lo deseen usar, es por ello que se define los componentes del servicio en tres:

a. Contrato
b. Interface
c. Implementación

Gao (2007), menciona que un Contrato especifica que consumir y que esperar como respuesta en base a los parámetros enviados, la Interface define como usar y accesar al servicio y la Implementación es sobre la realización del servicio. El Contrato y la Interface son visibles desde fuera, la Implementación es la característica más oculta, los consumidores generalmente no conocen de su implementación.

E. ARQUITECTURA ORIENTACIENTADA A SERVICIOS

Bieberstein (2008), define que SOA es un término que representa una nueva generación de una plataforma distribuida, como también involucra muchas cosas, incluyendo su propio diseño de paradigmas y principales diseños, patrones, un modelo de arquitectura distinta, conceptos relacionados y tecnologías.

Erl (2009), define que la computación orientada a servicios construye una plataforma distribuida y agrega nuevas capas, consideraciones de desarrollo y una vasta implementación de tecnologías, muchas de las cuales están basadas en el marco de desarrollo de los Web Services.

Gao (2007) , afirma que la orientación es un paradigma de diseño entendido para la creación de soluciones lógicas, unidades que son individualmente plasmadas tal que colectivamente y repetidamente son utilizadas en soporte de la realización de objetivos estratégicos específicos y los beneficios asociados con La orientación a servicio y la computación orientada a servicios.

Erl (2009), menciona que la solución lógica diseñada acore con la orientación a servicios puede ser calificado con Orientación a servicios, y las unidades de servicio orientadas a referirse como servicio. Como un paradigma de diseño para la computación distribuida, la orientación a servicio puede ser comparado con la orientación a objetos. De hecho tiene muchas raíces en la orientación a objetos y también ha sido influenciado por la industria de desarrollo.

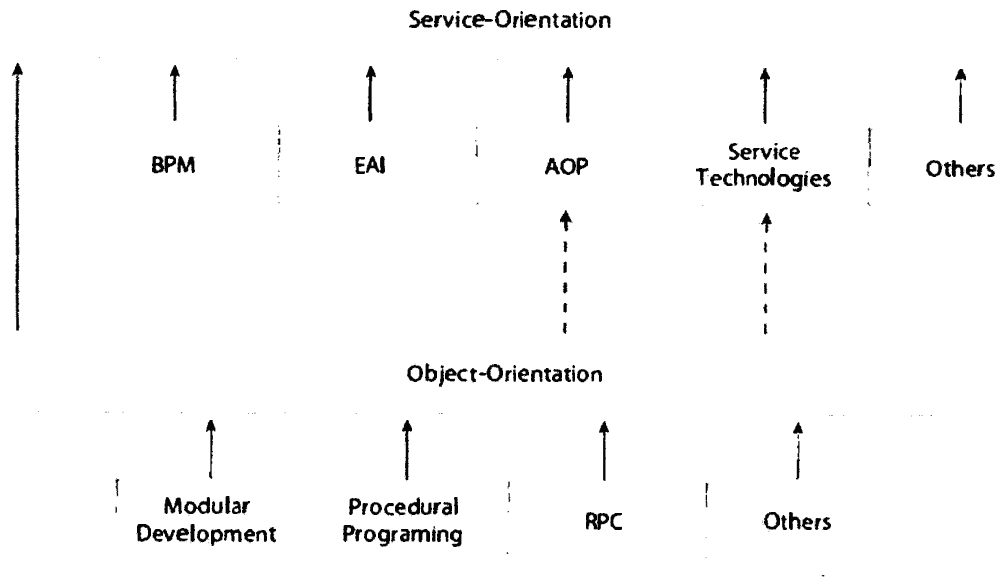

Figura 2.3: Evolución del diseño de paradigmas. (Erl, 2009).

Bieberstein (2008), define que, al implementar SOA muestra una guía satisfactoria para distribuir y compartir de una manera más compleja y realista.

Bieberstein (2008), afirma que la arquitectura orientada a servicios representa un modelo de Arquitectura que tiene como misión mejores la agilidad y el costo efectivo de una empresa mientras se reduce la carga de TI para toda la organización. SOA soporta la orientación a servicios en la realización de objetivos estratégicos, históricamente el término "Orientado a servicios" ha sido usado por los medios como casi el sinónimo con Computación Orientada a servicios.

Dikmans et al. (2012), menciona que la implementación de SOA consiste en la combinación de tecnologías, productos, APIs, extensiones del soporte de infraestructura y varias otras partes. La puesta en práctica de la arquitectura orientada a servicios es única dentro de cada organización, sin embargo es tipificado por la introducción de nuevas tecnologías y plataformas que especifican la creación, ejecución y evolución de soluciones orientados a servicios. Como resultado, construir una arquitectura alrededor de la arquitectura orientada a servicios, establece un entorno amigable para la solución lógica que ha sido diseñado con los principales diseños.

E.1. SERVICIOS BAJO .NET

Cheng (2010), afirma que implementar sistemas distribuidos para proveer

valor al negocio da confianza para los cambios, los procesos de negocio son a menudo soportados por sistemas que corren en diferentes plataformas y tecnologías ambos dentro o fiera de la organización. Los servicios orientados a servicios es un mecanismo que permite a las organizaciones la comunicación fácil entre los sistemas que corren en múltiples plataformas.

Microsoft. (n.d.), menciona que WCF maneja la infraestructura de la comunicación el cual permite crear diversos rangos de aplicaciones con la idea de modelo simplificado. Basado en la noción de WCF. Este contiene las mejoras características de la tecnología distribuida en el desarrollo de sistemas conectados.

E.2. SERVICIOS WINDOWS

Holzner (2002), menciona que los servicios Windows son como otras aplicaciones Windows, pero tienen unas pocas diferencias, primero, no tienen una interface de usuario excepto un panel de control que puede ser abierto haciendo doble clic sobre un icono de notificación. Segundo, su ciclo de vida es diferente que los programas estándares, porque típicamente inicia automáticamente cuando se inicia la computadora y solo se detiene cuando la computadora se apaga, Tercero, usualmente proveen soporte para servicios actuando de lado de las aplicaciones.

Fischer et al. (2002), afirma que en Visual Basic .NET, los servicios Windows están basados en el servicio Base, que provee el soporte que se necesita, para crear un servicio, se necesita sobrescribir los eventos OnStart y OnStop, los cuales son llamados cuando el servicio inicia y finaliza respectivamente. Usualmente, un servicio Windows, corre bajo una cuanta del sistema en Windows, sin embargo se puede instalar el servicio usando una cuenta específica, un simple ejecutable puede contener muchos servicios.

E.3. SERVICIOS WEB

Liu (2010), afirma que los servicios Web son servicio que operan en la Web y pueden ser usado por otras aplicaciones, por ejemplo, cuando se quiere mostrar datos de una fuente de datos en la web en una aplicación Windows, un servicio Web es una perfecta solución, porque la aplicación Windows, puede llamar método en el servicio Web para conseguir los datos. Estos servicios son usados para implementar programación paralela, aplicaciones distribuidas.

E.4. CONSUMO DE SERVICIOS

Patack (2011), afirma que Para poder usar un servicio, se necesita tener agregado una referencia al servicio. Para poder consumir un servicio WCF, existen muchas maneras dependiendo de la opción de hosting, si está usando WCF de lado del cliente, es mucho más productivo porque WCF viene como herramientas para generar clases de Proxy para llamar al servicio WCF. Visual Studio viene con características fáciles para agregar referencias al proyecto y fácilmente generar las clases del proxy.

Patack (2011), define las siguientes opciones para consumir un servicio WCF:

a. Obtener WSDL del servicio y generar proxy para llamar al servicio.
b. Usar la característica Agregar referencia de servicio de Visual Studio 2010 y generar proxy del lado del cliente.
c. Usar la herramienta svcutil.exe para generar las clases proxy.

2.1.4 MÉTODO SOA

Augustyniak (2002), afirma que una estrategia importante de las empresas es responder flexiblemente a los cambios, una de las posibles soluciones para asegurarse que TI es flexible y puede fácilmente cambiar es aplicando una solución de Arquitectura basada en Arquitectura Orientada a servicios (SOA). Arquitectura Orientada a servicios es una referencia que puede ser aplicada en organizaciones y responder rápidamente a cambios sin afectar otras partes, reusando funcionalidades existentes, asegurarse que el cambio es rápido y no se debe iniciar desde el comienzo.

Microsoft (2006), afirma que la Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante servicios Web, una tecnología basada en estándares e independiente de la plataforma.

Dikmans et al. (2012), define las siguientes acciones para diseñar un servicio:

a. Analizar la funcionalidad que ofrece
b. Diseñar la interface
    I. Diseñar las operaciones y la funcionalidad que las operaciones ofrece.
    II. Diseñar los parámetros de las operaciones.
    III. Diseñar los valores de retorno o el efecto de la operación.

IV. Diseñar los casos de prueba de las operaciones.

c. Diseñar el Contrato

I. Define quien está permitido para usar el servicio y quien puede usar que operación.

II. Decide cuan a menudo el servicio está disponible.

III. Define que la carga del servicio debe ser capaz de ser manejado.

IV. Define otros atributos de relevancia de calidad del servicio.

d. Implementación

I. Decide que herramientas o lenguaje están usando para la implementación.

II. Diseña los componentes que se necesita para la implementación.

III. Decide que herramientas usar para implementar las pruebas.

Dikmans et al. (2012), define que, la calidad del servicio es a menudo usado para describir las propiedades no funcionales del servicio como rendimiento, fiabilidad, seguridad, disponibilidad. Los principios del diseño del servicio indica la calidad que el servicio necesita para tener en orden para ser usados en la arquitectura que se está intentando conseguir. Cuando un servicio es pobremente diseñado o pobremente implementado, la arquitectura de nuestra solución probablemente tendrá pocos valores de negocio, lo que se necesita son los diseños principales del servicio que nos ayude a proveer un servicio que cree servicios Reusables, y nos ayude a consumirlo tal como el servicio fue diseñado.

A. BENEFICIOS DE SOA

Microsoft (n.d.), menciona que un negocio se expande y como resultado también crece la demanda para nuevas aplicaciones que puedan facilitar el nuevo modelo de negocio. Este resultado es de la formaciín de muchas ap0licaicones distribuidas que rápidamente se convierten en no mantenibles e inoperables, en este complejo ambiente, aparecen aplicaciones en el vacío que son complicados manejar e interoperar con otras aplicaciones.

Microsoft (n.d.), define que, SOA incrementa el objetivo del negocio, brinda más valor al empleo, integra costos, soporta nuevos dispositivos.

Entonces la informaciín es duplicada, como cada sección opera de manera independiente o en su propio negocio, para apoyar a ello existe esta nueva línea, que involucra automatizar los procesos de negocio entre Sistemas similares.

Este proceso involucra integración y agregación de entidades e información que está presente en diferentes Sistemas, debemos de construir un servicio que actualice los diferentes sistemas.

Holzner (2002), afirma que el negocio es beneficiado conectando los Sistemas y proveyendo una Solución Integral a los clientes, por implementar SOA se simplifica y se mejora la administración, usando SOA las organizaciones son capaces de hacer mejor las cosas, administrar y tomar mejores decisiones, este nivel de visibilidad dentro de la aplicación permite a las compañías identificar y lo más importante hacer cambios a los procesos de negocio existentes brindando oportunidades avanzados para la avance continuo del negocio.

B. EJECUCION DE UN PROYECTO SOA

Erl (2009) , menciona que en el pasado los cambios requeridos en una organización resultaban con obstáculos tecnológicos, por ejemplo reemplazar una parte o un proveedor en teoría era algo sencillo, pero en la realidad requería de mayor esfuerzo, estos esfuerzos pueden envolver en redesarrollar los componentes técnicamente y un mejor rediseño de los procesos de negocio, similarmente, integrando y combinando los procesos de Infraestructura podían llegar a tomar tiempo, el desarrollo de nuevos requerimientos no es posible porque también el costo está involucrado.

Gao (2007), afirma que, SOA provee el significado para flexibilizar de manera apropiada la arquitectura y reducir el costo dentro de la Arquitectura de reinventar.

C. PLATAFORMA SOA DE MICROSOFT

Dikmans et al. (2012), menciona que la Arquitectura orientada a servicios está basada en todos y cada uno de los elementos de la pila de tecnologías de Microsoft, desde las herramientas de desarrollo para crear servicios Web como .NET a productos de Servidor, finalmente las aplicaciones que consumen servicios Web.

D. MODELOS

Bieberstein (2008), menciona que la arquitectura es a menudo expresada usando diagramas y modelos, hay diferentes lenguajes de modelado disponibles. Todos los modelos apuntan a diferentes actores y ámbitos. La siguiente tabla muestra los lenguajes de modelado que pueden ser usados en diferentes capas junto con su

arquitectura objetivo. Dependiendo de la audiencia objetivo, se puede modelar algunos de estos lenguajes para comunicar la arquitectura del sistema.

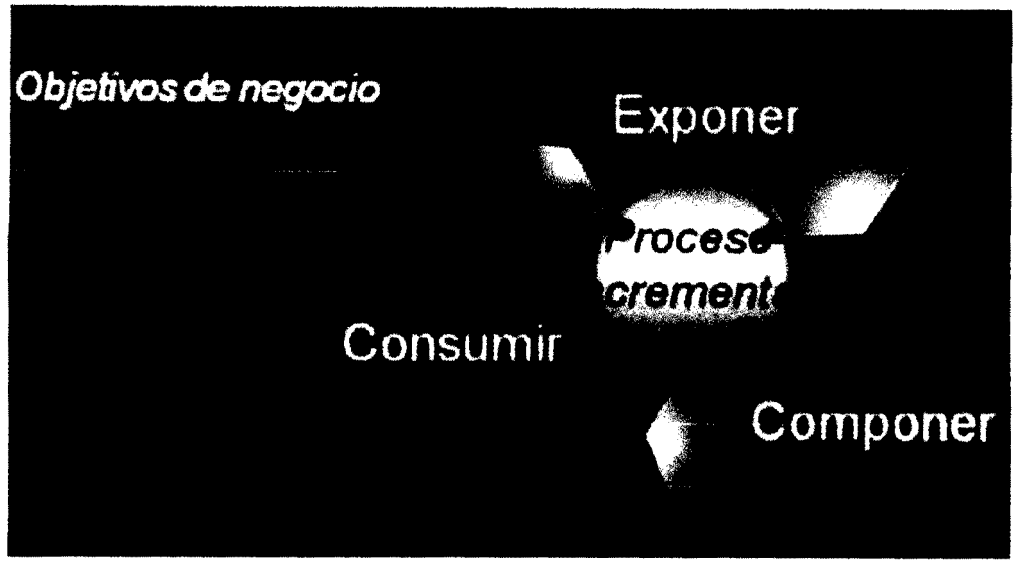

Figura 2.4: Implementación de SOA según Microsoft.
(Dikmans et al, 2012).

D.1. CONTRATOS

Bieberstein (2008), menciona que los contratos restringen las operaciones entre los operadores que están trabajando unos con otros, para servicios de comunicación distribuida, los contratos juegan un rol importante asegurando que el consumidor del servicio pueda cooperar con el servicio provisto correctamente, entonces hacemos referencia a SOA (Arquitectura orientada a servicios), técnicamente hablando SOAP (Protocolo de Acceso simple a Objeto) puede3 ser explicado como un conjunto de componentes que pueden ser invocados, y sus interfaces pueden ser publicados y descubiertas. Desde una perspectiva SOA, con los contratos propiamente definidos, el consumidor del servicio, tiene una idea de cómo trabajar con el servicio sin conocer como el servicio esta implementado.

Patack (2011), afirma que como una plataforma de comunicación unificada, WCF provee soporte completo para el diseño del contrato en varias partes del desarrollo del servicio. WCF incluye ServiceContrat, OperationContract, DataContract, MessageContract, FaultContrat. ServiceContract y OperationContract es usado para representar un servicio y la definición de sus operaciones.

Holzner (2002), afirma que DataContract es usado para representar el acuerdo de los

datos que serán cambiados entre el cliente del servicio y el servidor. Si el servicio quiere tomar control sobre los datos envueltos en la transferencia entre el cliente y el servidor, entonces se debe usar MessageContract para controlar los mensajes del servicio. WCF también provee FaultContract para el servicio sea declarado con errores personalizados para ciertos tipos de operaciones y su contenido por defecto será retornado cuando ocurra un error.

D.2. DATACONTRACT

Marshall (2011), afirma que WCF usa el motor de socialización llamado DataContractSerializer por defecto para serializar y deserializar datos. Si queremos agregar un nuevo tipo de datos complejo (que sea transferido en las operaciones del servicio), se necesita definir este como un tipo DataContract que sea amigable para el motor DataContractSerializer.

Liu (2010), afirma que, los contratos de datos son tipos de datos de un servicio WCF, todos los tipos de datos usados por el servicio WCF deben ser descritos en metadatos para permitir a otras aplicaciones interoperar con el servicio, un contrato de datos puede ser usado en una operación como un parámetro o un tipo de retorno.

D.3. MESSAGE CONTRAT

Dikmans et al. (2012), afirma que DataContract nos puede ayudar a diseñar los tipos de datos usados en un servicio WCF, sin embargo, este solo cubre los datos miembros (variables y parámetros usados en la operación) serializada bajo el mensaje SOAP. Algunas veces se necesita controlar la estructura y formato del mensaje entero SOAP.

Erl (2009), menciona que WCF introduce un concepto MessageContract, el cual ayuda a modelar la estructura y formato del mensaje entero dado en la operación del servicio, actualmente se puede tomar un tipo MessageContract como un tipo especial de DataContract, el cual es marcado por el atributo MessageContractAttirbute. Esta forma nos muestra cómo definir MessageContract para nuestras operaciones WCF y controlar el formato bajo el mensaje SOAP.

D.4. SERVICE CONTRAT

Liu (2010), afirma que Cada servicio (*.svc), tiene un ServiceContract, el cual contiene las operaciones a retornar. Después de especificar el tipo de colección

especifico al agregar la referencia al servicio, el asistente genera las clases del proxy que usa el tipo seleccionado para representar la colección basada en los parámetros o valores de las operaciones del servicio. Podemos usar la utilidad svcutll.exe para generar un proxy del servicio en adición a la herramienta que nos ofrece Visual Studio. Cuando usamos svcutll.exe para la generación del proxy, podemos especificar los tipos de la colección.

D.5. OPERATION CONTRACT

Liu (2010), afirma que la operación es un patrón común en la comunicación distribuida y también uno de los tres soporte en el intercambio de mensaje en WCF, cuando usamos el intercambio de patrones de mensajes, un cliente envía un mensaje usando el intercambio dispara-olvida, esto requiere la confirmación de la distribución.

Patack (2011), afirma que el mensaje se puede perder en la transmisión y nunca alcanzar el servicio, si la operación completa satisfactoriamente y el cliente finaliza, no existe garantía que el remoto endpoint haya recibido el mensaje. En estos casos donde el cliente solo quiere enviar información al lado del servidor, sin tomar cuidado del resultado de ejecución, podemos considerar definir nuestro servicio WCF como un tipo de operación de un solo sentido.

Liu (2010), define, Operation Contrat es definido dentro de un contrato del servicio, este define los parámetros y el tipo de retorno de la operación, una operación puede tomar datos como parámetros, o puede tomar mensajes. Una Operation Contract puede definir el niel como el flujo de transacción, la dirección de operación.

Dikmans et al. (2012), menciona que cuando la operación es marcada como de un solo sentido, en tiempo de ejecución, se detectara esto y la operación del servicio está preparada para este comportamiento, esta operación no retorna un valor y solo pasa parámetros de entrada al servicio, después que el cliente envía el requerimiento del servicio, el cliente esperara la respuesta hasta que el mensaje haya alcanzado al lado del servicio, sin embargo no existe un valor de retorno.

2.1.5 SISTEMA GESTOR DE BASE DE DATOS RELACIONAL

Batini (1994) afirma que los sistemas de gestión de base de datos (DBMS) son paquetes de software para la gestión de las bases de datos; en particular, para

almacenar, manipular y recuperar datos en un computador.

Mannino (2007), afirma que la tecnología de base de datos no sólo mejora las operaciones diarias de las organizaciones sino que también afectan a la calidad de las decisiones, contiene datos sobre muchos aspectos de la vida, preferencias, historial de créditos, hábitos, etc.

D'Andrea (2008), afirma que una base de datos está compuesto de uno o más archivos complejos que almacenan datos en un formato estructurado. Las aplicaciones, para trabajar con base de datos, se comunican a través de un Sistema de Administración y Gestión de datos.

Hansler (1997), afirma que una de las principales funciones de SGBD es garantizar el acceso, la recuperación y actualización de los datos en la base de datos. El SGBD proporciona los mecanismos físicos que permiten a varios usuarios tener acceso de forma rápida y eficiente a diferentes datos relacionados.

D'Andrea (2008), establece que los componentes principales de un proveedor de datos para el acceso a los datos de un origen de datos son los siguientes:

a. Connection: Representa una conexión abierta con el origen de datos, no es una clase derivable.
b. Command: Representa una sentencia SQL o un procedimiento almacenado que ejecuta un comando u orden en un origen de datos.
c. DataReader: Es un mecanismo para leer una secuencia de datos. El objeto solo se puede navegar hacia adelante y no es actualizable.
d. DataAdapter: Representa un conjunto de comandos de acceso a datos y una conexión al origen de datos que se utilizan para cargar el objeto Dataset y posteriormente actualizar el origen de datos.

2.1.6 LENGUAJE DE PROGRAMACIÓN ORIENTADO A OBJETOS

D'Andrea (2008), afirma que un objeto se puede considerar como una caja negra que encierra datos y funciones. Lo que en realidad nos importa es conocer de un objeto que cosas hace y que datos tiene pero no nos importa en absoluto el "como lo hace", el modo en que resuelve sus funciones, solo le interesa al creador del objeto, es

decir al que desarrollo la funcionalidad.

"En la programación orientada a objetos, las funciones deberían hacer una sola cosa, sin embargo, esto no es siempre el caso, para funciones extensas, evaluar si esta realiza más de una tarea, si es así, debes de ser capaz de descomponer la función en múltiples tareas" (Marshall, 2011, p. 10).

"Los cimientos de la orientación a objetos son: abstracción, encapsulación, polimorfismo y herencia" (Fischer et. Al, 2002, p.11).

Según Holzner (2002), los métodos representa la construcción del procedimiento del objeto, por ejemplo un nombre de clase Animal, puede tener muchos métodos dormir y comer, se definen los métodos agregando procedimientos o funciones a la clase.

Según D'Andrea (2008), una de las técnicas fundamentales en la programación orientada a objetos es hacer que cada clase sea responsable de llevar a cabo solo una pequeña cantidad de tareas relacionadas. Si un objeto necesita hacer algo que no es su responsabilidad, lo mejor es crear una nueva clase optimizada para esa tarea. En lugar de añadir código al primer objeto y complicar de esa manera el objeto original.

CAPITULO III

METODOLOGÍA DEL DESARROLLO DEL INFORME PROFESIONAL

3.1 DISEÑO DEL INFORME PROFESIONAL

En el presente informe determinaremos como diseñar un Modelo de Desarrollo de aplicaciones orientadas a servicios, el cual nos ayudare con las futuras implementaciones y que centre la labor del desarrollador en términos funcionales de negocio, se usará el lenguaje VB.NET, Windows Communication Foundation, el paradigma orientado a objetos y el administrador de base de datos SQL Server.

3.2 POBLACIÓN Y MUESTRA

POBLACIÓN.- La población de estudio está compuesta por los métodos orientados a servicios que usan las empresas consultoras de software de la ciudad de Lima en el año 2015.

MUESTRA.- Para determinar la muestra, se tomó un muestreo no probabilístico por conveniencia a juicio de expertos de los métodos orientados a servicios que usan las empresas consultoras de software de la ciudad de Lima en el año 2015, siendo la muestra:

Jefe de Desarrollo: 3

Programadores: 15

3.3 VARIABLES E INDICADORES

DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

PRIMERA VARIABLE

Framework para Aplicaciones Orientadas a servicios.- Es la estructura base definida para el desarrollo de Aplicaciones mediante el modelo orientado a servicios, WCF y cubre la etapa de desarrollo funcional de aplicaciones. Expone los métodos necesarios para consumir métodos, diseñar formularios.

INDICADORES DE LA PRIMERA VARIABLE

Descomposición lógica.- Comprende los métodos de acceso a datos mediante el proveedor de acceso a datos SQL, incluye la manipulación de datos: Insert, Update, Delete, Mergue.

Herencia de formularios.- Es el proceso que permite crear objetos a partir de otros ya definidos, permite heredar en formularios hijos funcionalidades específicas para ingreso operativo, menús y acceso a reportes.

Capas orientadas a servicios.- Es la identificación de los métodos que se expondrán a los clientes como parte del servicio para ser consumidos, incluye message contract, data contract, Service implementation.

DEFINICIÓN OPERACIONAL DE LAS VARIABLES

PRIMERA VARIABLE

X: Framework para aplicaciones orientadas a servicios.

INDICADORES DE LA PRIMERA VARIABLE

X1: Descomposición lógica

X2: Herencia de formularios

X6: Capas orientadas a servicios

3.4 TÉCNICAS E INSTRUMENTOS PARA EL TRATAMIENTO DE INFORMACIÓN

3.4.1 TÉCNICAS PARA RECOLECTAR INFORMACIÓN

Se utilizaron las técnicas de análisis documental, entrevista y encuesta.

Análisis documental: Técnica que fue usada para documentar los tiempos de demora en la programación, acceso a datos en el proceso de programación de aplicaciones orientadas a servicios.

Entrevista: Fue utilizada para recopilar información sobre estándares de programación usando Windows Communication Fondation, dirigida al Jefe de Desarrollo.

Encuesta: Se utilizó para la indagación, exploración y recolección de información sobre

arquitecturas de dasarrollo, mediante preguntas dirigidas a programadores.

3.4.2 INSTRUMENTOS PARA RECOLECTAR INFORMACIÓN

Se ha diseñado el instrumento "Ficha para Análisis Documental", para obtener los tiempos de demora en la programación, acceso a datos, presentado en el Anexo C, tabla N° C.1.

Se ha diseñado el instrumento "Cuestionario al jefe de desarrollo", para la entrevista dirigida al Jefe de Desarrollo, presentado en el Anexo A, Tabla N° A.1. Muestra información sobre estándares de programación.

Se ha diseñado el instrumento "Cuestionario a programadores", para la encuesta a programadores, presentado en el Anexo B, Tabla N° B.1. Muestra la información sobre las arquitecturas de desarrollo, diseño de capas orientados a servicios.

3.4.3 HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

Las herramientas tecnológicas que se utilizan son seleccionadas debido a la interoperabilidad que presenta, soporte continuo y actualización constante, se propone la implementación de un framework para la construcción de aplicaciones orientadas a servicios. Por lo cual seleccionamos las tecnologías según la tabla 3.1.

Nombre	Fabricante	Servicio
Windows7	Microsoft Corporation	Sistema Operativo de Microsoft, sucesor de Windows Vista.
VB.NET	Microsoft Corporation	Lenguaje de Programación orientado a objetos.
Visual Studio 2010	Microsoft Corporation	Entorno Integrado de Programación, permite crear entregar software en variedad de plataformas.
WCF	Microsoft Corporation	Marco para la creación de aplicaciones orientadas a servicios.
ReportViewer	Microsoft Corporation	Es un control que permite diseñar, incrustar reportes en las aplicaciones desarrolladas usando NET Framework.

Microsoft SQL Server	Microsoft Corporation	Es un Sistema de gestión de Base de Datos, basado en el modelo relacional.

Tabla N° 3.1: Herramientas para el desarrollo del Modelo Orientado a servicios.

3.4.4 TÉCNICAS PARA APLICAR SOA

Revisando el marco teórico en el capítulo II, sección 2.1.1, formulamos el proceso, que considera las fases para construir aplicaciones orientadas a servicios, usando SOA, como se muestra en las tablas 3.2 a 3.4.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Definir la capa de Acceso a datos.	Métodos de acceso a datos definidos.	Definir los métodos generales de manipulación de datos que serán usados.	Analista Arquitecto
Obtener la capa de servicio de componentes.	Capa de exposición de contratos.	– Obtener las actividades y agruparlos por funcionalidad. – Conformar los métodos definidos por el servicio.	Analista Arquitecto Programador
Obtener la capa de acceso al servicio.	Capa de implementación de métodos.	Definir la información sintáctica y semántica del servicio. Definir los canales desde donde serán invocados por los clientes.	Analista Arquitecto Programador

Tabla N° 3.2: Análisis del servicio. (Bieberstein, 2008).

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Diseñar las interfaces de métodos.	Operaciones con funcionalidad. Operaciones con parámetros.	– Creación de instancias usandoCLR. – Definir los mensajes que se transmitirán.	Analista Programador
Diseñar el contrato de acceso a los métodos.	Definición de Data Contract y Message Contract.	– Creación de instancias usandoWCF. – Definir la especificación del servicio con WS.	Analista Programador

Tabla N° 3.3: Diseño del servicio. (Dikmans et al., 2012).

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Implementar los formularios operativos.	Formularios maestros para ingreso operativo.	– Realizar herencia visual de formularios. – Agrupar los formularios en tareas especializadas.	Analista Programador
Implementar los menús de acceso.	Menús para acceso a módulos.	– Distribución de menús para acceso a módulos. – Definir de eventos para ser implementados.	Analista Programador
Definirlos métodos para acceder a reportes.	Métodos para acceder a reportes.	Obtener los métodos para acceso a reporte.	Analista Programador

Tabla N° 3.4: Implementación del servicio. (Dikmans et al., 2012).

CAPITULO IV

RESULTADOS DEL INFORME PROFESIONAL

4.1 SOA PARA ANÁLISIS

Según el método SOA para desarrollar aplicaciones orientadas a servicios, desarrolladas en las tablas 3.2 a 3.4, descrito en el capítulo II sección 2.1.4, se obtienen los artefactos de análisis, diseño, implementación. A continuación mostramos los resultados de los artefactos de acuerdo al método SOA.

4.1.1. MÉTODOS DE ACCESO A DATOS DEFINIDOS

Framework.Services.DataAccess

```
Imports System.Linq
Imports System.Collections.Generic
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration

Public Class Sql

#Region "Propiedades"

    ''' <remarks></remarks>
    Private _SetConnectionString As String
    Public Property SetConnectionString() As String
        Get
            Return _SetConnectionString
        End Get
        Set(ByVal value As String)
            _SetConnectionString = value
        End Set
    End Property

    ''' <remarks></remarks>
    Private _sharedConnection As SqlConnection
    Private Property SharedConnection() As SqlConnection
        Get
            If _sharedConnection Is Nothing Then
                _sharedConnection = New SqlConnection(SetConnectionString)
            End If
            Return _sharedConnection
        End Get
        Set(value As SqlConnection)
            _sharedConnection = value
        End Set
    End Property

#End Region

#Region "Constructor"

    Public Sub New()
    End Sub
```

```
#End Region

#Region "Declaracion de Funciones"

#Region "Parameter"

    ''' <param name="name"></param>
    ''' <param name="paramType"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateNullParameter(name As String, paramType As SqlDbType)
As SqlParameter
        Dim parameter As New SqlParameter()

        parameter.SqlDbType = paramType
        parameter.ParameterName = name
        parameter.Value = DBNull.Value
        parameter.Direction = ParameterDirection.Input

        Return parameter

    End Function

    ''' <param name="name"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateNullParameter(name As String) As SqlParameter
        Dim parameter As New SqlParameter()

        parameter.ParameterName = name
        parameter.Value = DBNull.Value
        parameter.Direction = ParameterDirection.Input

        Return parameter
    End Function

    ''' <param name="name"></param>
    ''' <param name="paramType"></param>
    ''' <param name="size"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateNullParameter(name As String, paramType As SqlDbType,
size As Integer) As SqlParameter

        Dim parameter As New SqlParameter()

        parameter.SqlDbType = paramType
        parameter.ParameterName = name
        parameter.Size = size
        parameter.Value = DBNull.Value
        parameter.Direction = ParameterDirection.Input

        Return parameter
    End Function

    ''' <param name="name"></param>
    ''' <param name="paramType"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateOutputParameter(name As String, paramType As
SqlDbType) As SqlParameter
        Dim parameter As New SqlParameter()
```

```
        parameter.SqlDbType = paramType
        parameter.ParameterName = name
        parameter.Direction = ParameterDirection.Output

        Return parameter
    End Function

    ''' <param name="name"></param>
    ''' <param name="paramType"></param>
    ''' <param name="size"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateOutputParameter(name As String, paramType As
SqlDbType, size As Integer) As SqlParameter
        Dim parameter As New SqlParameter()

        parameter.SqlDbType = paramType
        parameter.Size = size
        parameter.ParameterName = name
        parameter.Direction = ParameterDirection.Output

        Return parameter
    End Function

    ''' <param name="name"></param>
    '''<param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateinputOutputParameter(name As String, value As Object)
As SqlParameter
        Dim parameter As New SqlParameter()

        If TypeOf value Is Integer Then
            parameter.SqlDbType = SqlDbType.Int
        Elseif TypeOf value Is Decimal Then
            parameter.SqlDbType = SqlDbType.Decimal
        Elseif TypeOf value Is Long Then
            parameter.SqlDbType = SqlDbType.Bigint
        Elseif TypeOf value Is Short Then
            parameter.SqlDbType = SqlDbType.Smallint
        End If
        parameter.ParameterName = name
        parameter.Value = value
        parameter.Direction = ParameterDirection.InputOutput

        Return parameter
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of
Integer)) As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.Int)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.Int
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
```

```
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of
Short)) As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.SmallInt)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.SmallInt
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of Long))
As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.BigInt)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.Bigint
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of Byte))
As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.Tinyint)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.TinyInt
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of Guid))
```

```
As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.Uniqueidentifier)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.UniqueIdentifier
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of
Boolean)) As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.Bit)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.Bit
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Byte()) As
SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.VarBinary)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.VarBinary
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of
Decimal)) As SqlParameter
        If value Is Nothing Then
            Return CreateNullParameter(name, SqlDbType.Decimal)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.Decimal
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
```

```
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As Nullable(Of Date))
As SqlParameter
        If value Is Nothing Then
            ' If value is null then create a null parameter
            Return CreateNullParameter(name, SqlDbType.DateTime)
        Else
            Dim parameter As New SqlParameter()
            parameter.SqlDbType = SqlDbType.DateTime
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input
            Return parameter
        End If
    End Function

    ''' <param name="name"></param>
    ''' <param name="valueXML"></param>
    ''' <param name="IsXML"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, valueXML As String, IsXML
As Boolean) As SqlParameter
        Dim parameter As New SqlParameter()
        parameter.SqlDbType = SqlDbType.Xml
        parameter.ParameterName = name
        parameter.Value = valueXML
        parameter.Direction = ParameterDirection.Input
        Return parameter
    End Function

    ''' <param name="name"></param>
    ''' <param name="value"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function CreateParameter(name As String, value As String) As
SqlParameter
        If String.Equals(value, "")=False And value = Nothing Then
            Return CreateNullParameter(name, SqlDbType.NVarChar)
        Else
            Dim parameter As New SqlParameter()

            parameter.SqlDbType = SqlDbType.NVarChar
            parameter.ParameterName = name
            parameter.Value = value
            parameter.Direction = ParameterDirection.Input

            Return parameter
        End If
    End Function

#End Region

#Region "Get Data"

    ''' <param name="sqlQuery"></param>
```

```
    ''' <param name="cmdTimeOut"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Private Function GetCommand(sqlQuery As String, ByVal cmdTimeOut As
Nullable(Of Integer)) As SqlCommand
        Dim command As New SqlCommand()

        command.Connection = SharedConnection
        If cmdTimeOut.HasValue Then
            command.CommandTimeout = cmdTimeOut
        End If
        command.CommandType = CommandType.Text
        command.CommandText = sqlQuery

        Return command
    End Function

    ''' <param name="sprocName"></param>
    ''' <param name="cmdtimeOut"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Private Function GetSprocCommand(sprocName As String, ByVal cmdTimeOut As
Nullable(Of Integer)) As SqlCommand
        Dim command As New SqlCommand(sprocName)

        command.Connection = SharedConnection
        If cmdTimeOut.HasValue Then
            command.CommandTimeout = cmdTimeOut
        End If
        command.CommandType = CommandType.StoredProcedure

        Return command
    End Function

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeOut"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function ExecuteScalar(ByVal SQLCommand As String, ByRef Param() As
AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As Nullable(Of
Integer)) As [Object]
        Dim command As New SqlCommand
        Dim resultado As [Object]
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        resultado= command.ExecuteScalar()
        command.Connection.Close()
        Return resultado
    End Function

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
```

```
    ''' <remarks></remarks>
    Public Function GetDataTable(ByVal SQLCommand As String, ByVal Param() As
AccessDB.ParametroDB, isproc As Boolean, ByVal cmdTimeOut As Nullable(Of
Integer)) As DataTable
        Dim returnvalue As DataTable
        Dim command As SqlCommand = PrepareCommand(SQLCommand, Param, isproc,
cmdTimeOut)
        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Dim reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.CloseConnection)
        returnvalue = New DataTable
        returnvalue.Load(reader)
        command.Connection.Close()
        Return returnvalue

    End Function

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetSingleValue(Of T)(ByVal SQLCommand As String, ByRef
Param() As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As
Nullable(Of Integer)) As T
        Dim returnValue As T = Nothing
        Dim command As New SqlCommand

        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.SingleResult)
            If reader.HasRows Then
                reader.Read()
                If Not reader.IsDBNull(0) Then
                    returnValue = DirectCast(reader(0), T)
                End If
                reader.Close()
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return returnValue

    End Function

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeOut"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetSingleInt32(ByVal SQLCommand As String, ByRef Param() As
AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As Nullable(Of
Integer)) As Int32
```

```
        Dim returnValue As Int32 = 0
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.SingleResult)
            If reader.HasRows Then
                reader.Read()
                If NOt reader.IsDBNull(0) Then
                    returnValue = reader.GetInt32(0)
                End If
                reader.Close()
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return returnValue

    End Function

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetSingleString(ByVal SQLCommand As String, ByRef Param()
As AccessDB.ParametroDB, ByVal isProc As BOolean, ByVal CmdTimeOut As
Nullable(Of Integer)) As String
        Dim returnValue As String = NOthing
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, CmdTimeOut)

        If command.ConnectiOn.State <> ConnectiOnState.Open Then
            Command.COnnection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(COmmandBehaviOr.SingleResult)
            If reader.HasRows Then
                reader.Read()
                If Not reader.IsDBNull(0) Then
                    returnValue = reader.GetString(0)
                End If
                reader.Close()
            End If
        End Using
        If isProC Then
            Param = OutputValues(cOmmand, Param)
        End If
        command.Connection.Close()
        Return returnValue
    End Function

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
```

```
    ''' <remarks></remarks>
    Public Function GetStringList(ByVal SQLCommand As String, ByRef Param() As
AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As Nullable(Of
Integer)) As List(Of String)
        Dim returnList As List(Of String) = Nothing
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.CloseConnection)
            If reader.HasRows Then
                returnList = New List(Of String)()
                While reader.Read()
                    If Not reader.IsDBNull(0) Then
                        returnList.Add(reader.GetString(0))
                    End If
                End While
                reader.Close()
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return returnList

    End Function

    ''' <typeparam name="T"></typeparam>
    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetSingle(Of T As Class)(ByVal SQLCommand As String, ByRef
Param() As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As
Nullable(Of Integer)) As T
        Dim dto As T = Nothing
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.SingleRow)
            If reader.HasRows Then
                reader.Read()
                Dim mapper As IDataMapper = New
DataMapperFactory().GetMapper(GetType(T))
                dto = DirectCast(mapper.GetData(reader), T)
                reader.Close()
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return dto
```

```
    End Function

    ''' <typeparam name="T"></typeparam>
    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetList(Of T As Class)(ByVal SQLCommand As String, ByRef
Param() As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As
Nullable(Of Integer)) As List(Of T)
        Dim dtoList As New List(Of T)()
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.CloseConnection)
            If reader.HasRows Then
                Dim mapper As IDataMapper = New
DataMapperFactory().GetMapper(GetType(T))
                While reader.Read()
                    Dim dto As T = Nothing
                    dto = DirectCast(mapper.GetData(reader), T)
                    dtoList.Add(dto)
                End While
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return dtoList

    End Function

    ''' <typeparam name="T"></typeparam>
    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetListSingle(Of T As Class)(ByVal SQLCommand As String,
ByRef Param() As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal
cmdTimeOut As Nullable(Of Integer)) As List(Of T)
        Dim dtoList As New List(Of T)()
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.SingleRow)
            If reader.HasRows Then
                Dim mapper As IDataMapper = New
DataMapperFactory().GetMapper(GetType(T))
                While reader.Read()
```

```
                    Dim dto As T = Nothing
                    dto = DirectCast(mapper.GetData(reader), T)
                    dtoList.Add(dto)
                End While
                reader.Close()
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return dtoList

    End Function

    ''' <typeparam name="T"></typeparam>
    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeout"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function GetListSingleResult(Of T As Class)(ByVal SQLCommand As
String, ByRef Param() As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal
cmdTimeOut As Nullable(Of Integer)) As List(Of T)
        Dim dtoList As New List(Of T)()
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.SingleResult)
            If reader.HasRows Then
                Dim mapper As IDataMapper = New
DataMapperFactory().GetMapper(GetType(T))
                While reader.Read()
                    Dim dto As T = Nothing
                    dto = DirectCast(mapper.GetData(reader), T)
                    dtoList.Add(dto)
                End While
                reader.Close()
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        command.Connection.Close()
        Return dtoList
    End Function

    ''' <summary>
    ''' retorna lista evaluada con cast si es diferente tipo
    ''' </summary>
    ''' <typeparam name="T"></typeparam>
    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeOut"></param>
    ''' <returns></returns>
    ''' <remarks>
    ''' </remarks>
```

```
    Public Function GetListWithCast(Of T As Class)(ByVal SQLCommand As String,
ByRef Param() As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal
cmdTimeOut As Nullable(Of Integer)) As List(Of T)
        Dim dtoList As New List(Of T)()
        Dim command As New SqlCommand
        command = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        If command.Connection.State <> ConnectionState.Open Then
            command.Connection.Open()
        End If
        Using reader As SqlDataReader =
command.ExecuteReader(CommandBehavior.CloseConnection)
            If reader.HasRows Then
                Dim mapper As IDataMapper = New
DataMapperFactory().GetMapper(GetType(T))
                While reader.Read()
                    Dim dto As T = Nothing
                    dto = DirectCast(mapper.GetDataWithCast(reader), T)
                    dtoList.Add(dto)
                End While
            End If
        End Using
        If isProc Then
            Param = OutputValues(command, Param)
        End If
        Command.Connection.Close()
        Return dtoList

    End Function

#End Region



#End Region

#Region "Declaracion de Procedimientos"

    ''' <param name="command"></param>
    ''' <param name="IsLocalTransaction"></param>
    ''' <remarks></remarks>
    Private Sub ExecuteNonQuery(command As SqlCommand, Optional ByVal
IsLocalTransaction As Boolean = False)
        Dim transaction As SqlTransaction = Nothing

        Try
            If command.Connection.State <> ConnectionState.Open Then
                command.Connection.Open()
            End If

            'Si es Transaccional
            If IsLocalTransaction = True Then
                transaction =
command.Connection.BeginTransaction("TransactionProcess")
                command.Transaction = transaction
                command.ExecuteNonQuery()
                transaction.Commit()
            Else
                command.ExecuteNonQuery()
            End If
        Catch e As Exception
            If IsLocalTransaction = True Then
                transaction.Rollback()
```

```
            End If

            Throw New Exception(e.Message, e)
        Finally
            command.Connection.Close()
        End Try
    End Sub

    ''' <param name="commandParameters"></param>
    ''' <param name="parameterValues"></param>
    ''' <remarks></remarks>
    Private Overloads Shared Sub AssignParameterValues(ByVal
commandParameters() As SqlParameter, ByVal parameterValues() As Object)
        Dim i As Integer
        Dim j As Integer

        If (commandParameters Is Nothing) AndAlso (parameterValues Is Nothing)
Then
            ' Do nothing if we get no data
            Return
        End If

        'We must have the same number of values as we pave parameters to put
them in
        If commandParameters.Length <> parameterValues.Length Then
            Throw New ArgumentException("Parameter count does not match
Parameter Value count.")
        End If

        'Value array
        j = commandParameters.Length - 1

        For i = 0 To j
            ' If the current array value derives from IDbDataParameter, then
assign its Value property
            If TypeOf parameterValues(i) Is IDbDataParameter Then
                Dim paraminstance As IDbDataParameter =
CType(parameterValues(i), IDbDataParameter)
                If (paramInstance.Value Is Nothing) Then
                    commandParameters(i).Value = DBNull.Value
                Else
                    commandParameters(i).Value = paramInstance.Value
                End If
            Elseif (parameterValues(i) Is Nothing) Then
                commandParameters(i).Value = DBNull.Value
            Else
                commandParameters(i).Value = parameterValues(i)
            End If
        Next
    End Sub

    ''' <param name="command"></param>
    ''' <param name="connection"></param>
    ''' <param name="transaction"></param>
    ''' <param name="commandType"></param>
    ''' <param name="commandText"></param>
    ''' <param name="commandParameters"></param>
    ''' <param name="mustCloseConnection"></param>
    ''' <remarks></remarks>
    Private Shared Sub PrepareCommand(ByVal command As SqlCommand, _
                                      ByVal connection As SqlConnection, _
                                      ByVal transaction As SqlTransaction,
```

```
                                        ByVal commandType As CommandType, _
                                        ByVal commandText As String, _
                                        ByVal commandParameters() As
SqlParameter, _
                                        ByRef mustCloseConnection As Boolean)

        If (command Is Nothing) Then
            Throw New ArgumentNullException("command")
        End If
        If (commandText Is Nothing OrElse commandText.Length = 0) Then
            Throw New ArgumentNullException("commandText")
        End If

        ' If the provided connection is not open, we will open it
        If connection.State <> ConnectionState.Open Then
            connection.Open()
            mustCloseConnection = True
        Else
            mustCloseConnection = False
        End If

        'Associate the connection with the command
        command.Connection = connection

        ' Set the command text (stored procedure name or SQL statement)
        command.CommandText = commandText

        ' If we were provided a transaction, assign it.
        If Not (transaction Is Nothing) Then
            If transaction.Connection Is Nothing Then
                Throw New ArgumentException("The transaction was rollbacked or
commited, please provide an open transaction.", "transaction")
            End If
            command.Transaction = transaction
        End If

        ' Set the command type
        command.CommandType = commandType

        'Attach the command parameters if they are provided
        If Not (commandParameters Is Nothing) Then
            AttachParameters(command, commandParameters)
        End If
        Return

    End Sub

    ''' <param name="command"></param>
    ''' <param name="commandParameters"></param>
    ''' <remarks></remarks>
    Private Shared Sub AttachParameters(ByVal command As SqlCommand, ByVal
commandParameters() As SqlParameter)
        If (command Is Nothing) Then
            Throw New ArgumentNullException("command")
        End If
        If (Not commandParameters Is Nothing) Then
            Dim p As SqlParameter
            For Each p In commandParameters
                If (Not p Is Nothing) Then
                    'Check for derived output value with no value assigned
                    If (p.Direction = ParameterDirection.InputOutput OrElse
p.Direction = ParameterDirection.Input) AndAlso p.Value Is Nothing Then
                        p.Value = DBNull.Value
```

```
                    End If
                    command.Parameters.Add(p)
                End If
            Next p
        End If
    End Sub

    ''' <param name="SQLCommand"></param>
    ''' <param name="Parameters"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeOut"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Private Function PrepareCommand(ByVal SQLCommand As String, ByVal
Parameters() AS AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut
As Nullable(Of Integer)) As SqlCommand
        Dim cmd As SqlCommand
        If isProc Then
            cmd = GetSprocCommand(SQLCommand, cmdTimeOut)
            If Not IsNothing(Parameters) Then
                With cmd.Parameters
                    For Each param As AccessDB.ParametroDB In Parameters
                        If param.isOuput AndAlso param.Size = 0 AndAlso
IsNothing(param.Value) Then
                            .Add(CreateOutputParameter(param.Name,
param.Value))
                        Elseif param.isOuput AndAlso param.Size > 0 AndAlso
IsNothing(param.Value) Then
                            .Add(CreateOutputParameter(param.Name, param.Value,
param.Size))
                        ElseIf param.isOuput AndAlso IsNumeric(param.Value)
Then
                            .Add(CreateinputOutputParameter(param.Name,
param.Value))
                        Elseif Not param.isOuput And Not param.isXml Then
                            If TypeOf param.Value Is Integer Then
                                .Add(CreateParameter(param.Name,
CInt(param.Value)))
                            ElseIf TypeOf param.Value Is Boolean Then
                                .Add(CreateParameter(param.Name,
CBool(param.Value)))
                            Elseif TypeOf param.Value Is Byte Then
                                .Add(CreateParameter(param.Name,
CByte(param.Value)))
                            ElseIf TypeOf param.Value Is Date Then
                                .Add(CreateParameter(param.Name,
CDate(param.Value)))
                            ElseIf TypeOf param.Value Is Decimal Then
                                .Add(CreateParameter(param.Name,
CDec(param.Value)))
                            ElseIf TypeOf param.Value Is Long Then
                                .Add(CreateParameter(param.Name,
CLng(param.Value)))
                            ElseIf TypeOf param.Value Is Short Then
                                .Add(CreateParameter(param.Name,
CShort(param.Value)))
                            Elseif TypeOf param.Value Is Guid Then
                                .Add(CreateParameter(param.Name,
CType(param.Value, Guid)))
                            Elseif TypeOf param.Value Is String Then
                                .Add(CreateParameter(param.Name,
CStr(param.Value)))
                            ElseIf TypeOf param.Value Is Byte() Then
```

```
                                .Add(CreateParameter(param.Name,
DirectCast(param.Value, Byte())))
                            ElseIf Not IsNothing(param.Name) Then
                                .Add(CreateNullParameter(param.Name))
                            End If
                        Else
                            .Add(CreateParameter(param.Name, CStr(param.Value),
True))
                        End If
                    Next
                End With
            End If
        Else
            cmd = GetCommand(SQLCommand, cmdTimeOut)
        End If
        Return cmd

    End Function

    ''' <param name="command"></param>
    ''' <param name="parameters"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Private Function OutputValues(ByVal command As SqlCommand, ByVal
parameters() As AccessDB.ParametroDB) As AccessDB.ParametroDB()
        If Not IsNothing(parameters) Then
            For x = 0 To parameters.Length - 1
                If parameters(x).isOuput Then
                    parameters(x).Value =
command.Parameters(parameters(x).Name).Value
                End If
            Next
        End If
        Return parameters
    End Function

#End Region

#Region "Llamada de procedimientos publicos"

    ''' <param name="SQLCommand"></param>
    ''' <param name="Param"></param>
    ''' <param name="isProc"></param>
    ''' <param name="cmdTimeOut"></param>
    ''' <param name="IsLocalTransactional"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function ExecuteNonQuery(ByVal SQLCommand As String, ByRef Param()
As AccessDB.ParametroDB, ByVal isProc As Boolean, ByVal cmdTimeOut As
Nullable(Of Integer), _
                                    Optional ByVal IsLocalTransactional As
Boolean = False) As AccessDB.ParametroDB()
        Dim cmd As New SqlCommand
        cmd = PrepareCommand(SQLCommand, Param, isProc, cmdTimeOut)

        Dim Coneccion As SqlConnection = SharedConnection

        ExecuteNonQuery(cmd, IsLocalTransactional)
        If isProc Then
            Param = OutputValues(cmd, Param)
        End If
        Return Param
    End Function
```

```
#End Region

End Class
```

Figura 4.1: Codigo fuente que define los métodos de acceso a datos.

4.1.2. CAPA DE EXPOSICÓN DE CONTRATOS

Project.ServiceContract

```
Imports System.ServiceModel
Imports Project.MessageContract

<ServiceContract()> _
Public Interface ISCSOA

#Region "General"

    <OperationContract()> _
    Function RetornaEstado() As Boolean

    <OperationContract()> _
    Function SCO_SeguridadUsuario(ByVal request As MC_SeguridadUsuarioRequest)
As MC_SeguridadUsuarioResponse

    <OperationContract()> _
    Function SCO_SeguridadModulo(ByVal request As MC_SeguridadModuloRequest) As
MC_SeguridadModuloResponse

    <OperationContract()> _
    Function SCO_SeguridadGrupoUsuario(ByVal request As
MC_SeguridadGrupoUsuarioRequest) As MC_SeguridadGrupoUsuarioResponse

    <OperationContract()> _
    Function SCO_ParametroSQL(ByVal request As MC_ParametroSQLRequest) As
MC_ParametroSQLResponse

    <OperationContract()> _
    Function SCO_GENCodigo(ByVal request As MC_GENCodigoRequest) As
MC_GENCodigoResponse

    <OperationContract()> _
    Function SCO_GENGrupoCodigo(ByVal request As MC_GENGrupoCodigoRequest) As
MC_GENGrupoCodigoResponse

    <OperationContract()> _
    Function SCO_GENModulo(ByVal request As MC_GENModuloRequest) As
MC_GENModuloResponse

    <OperationContract()> _
    Function SCO_GENSistema(ByVal request As MC_GENSistemaRequest) As
MC_GENSistemaResponse

    <OperationContract()> _
    Function SCO_GENLogTransaccional(ByVal request As
MC_GENLogTransaccionalRequest) As MC_GENLogTransaccionalResponse

#End Region

#Region "Adicionales"

    <OperationContract()> _
    Function SCO_Correo(ByVal request As MC_CorreoRequest) As MC_CorreoResponse
```

```
    <OperationContract()> _
    Function SCO_Servidor(ByVal request As MC_ServidorRequest) As
MC_ServidorResponse

#End Region

End Interface
```

Figura 4.2: Interfaz que define los contratos del servicio.

SIService.vb

```
Imports Project.ServiceContract
Imports Project.MessageContract
Imports Microsoft.Practices.EnterpriseLibrary.ExceptionHandling
Imports Project.BusinessLogic
Imports Project.DataContract
Imports Framework.Services.Exception

<ServiceModel.ServiceBehavior(InstanceContextMode:=ServiceModel.InstanceContext
Mode.PerCall)> _
Public Class SIService
    Implements ISCSOA

#Region "Adicionales"

    Public Function SCO_ParametroSQL(ByVal request As
MessageContract.MC_ParametroSQLRequest) As
MessageContract.MC_ParametroSQLResponse Implements
ServiceContract.ISCSOA.SCO_ParametroSQL
        Dim resp As New MC_ParametroSQLResponse

        Try
            Dim BL As New BL_ParametroSQL

            Select Case request.Operacion
                Case DC_ParametroSQLOperation.ParametroSQLEjecutarSentencia
                    resp.ParametroSQL =
SI_ParametroSQLTranslator.ToDataContract(BL.ParametroSQLEjecutarSentencia(SI_Pa
rametroSQLTranslator.ToBusiness(request.ParametroSQL)))
                Case
DC_ParametroSQLOperation.ParametroSQLEjecutarSentenciaTexto
                    resp.ParametroSQL =
SI_ParametroSQLTranslator.ToDataContract(BL.ParametroSQLEjecutarSentenciaTexto(
SI_ParametroSQLTranslator.ToBusiness(request.ParametroSQL)))
                Case DC_ParametroSQLOperation.EjecutarSentenciaListadoDatatable
                    resp.ParametroSQL =
SI_ParametroSQLTranslator.ToDataContract(BL.EjecutarSentenciaListadoDatatable(S
I_ParametroSQLTranslator.ToBusiness(request.ParametroSQL)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_SeguridadModulo(ByVal request As
```

```
MessageContract.MC_SeguridadModuloRequest) As
MessageContract.MC_SeguridadModuloResponse Implements
ServiceContract.ISCSOA.SCO_SeguridadModulo
        Dim resp As New MC_SeguridadModuloResponse

        Try
            Dim BL As New BL_SeguridadModulo

            Select Case request.Operacion
                Case
DC_SeguridadModuloOperation.SeguridadModuloSelPermisoxRegistroSistema
                    resp.SeguridadModuloList =
SI_SeguridadModuloTranslator.ToDataContract(BL.SeguridadModuloSelPermisoxRegist
roSistema(SI_SeguridadModuloTranslator.ToBusiness(request.SeguridadModulo)))
                Case
DC_SeguridadModuloOperation.SeguridadModuloSelxSistemaModuloPadre
                    resp.SeguridadModuloList =
SI_SeguridadModuloTranslator.ToDataContract(BL.SeguridadModuloSelxSistemaCodigo
Padre(SI_SeguridadModuloTranslator.ToBusiness(request.SeguridadModulo)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceinterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_SeguridadUsuario(ByVal request As
MessageContract.MC_SeguridadUsuarioRequest) As
MessageContract.MC_SeguridadUsuarioResponse Implements
ServiceContract.ISCSOA.SCO_SeguridadUsuario
        Dim resp As New MC_SeguridadUsuarioResponse

        Try
            Dim BL As New BL_SeguridadUsuario

            Select Case request.Operacion
                Case DC_SeguridadUsuarioOperation.SeguridadUsuarioSelxRegistro
                    resp.SeguridadUsuario =
SI_SeguridadUsuarioTranslator.ToDataContract(BL.SeguridadUsuarioSelxRegistro(SI
_SeguridadUsuarioTranslator.ToBusiness(request.SeguridadUsuario)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceinterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function
```

```
    Public Function SCO_Correo(ByVal request As
MessageContract.MC_CorreoRequest) As MessageContract.MC_CorreoResponse
Implements ServiceContract.ISCSOA.SCO_Correo
        Dim resp As New MC_CorreoResponse

        Try
            Dim BL As New BL_Correo

            Select Case request.Operacion
                Case DC_CorreoOperation.EnviarCorreo

BL.EnviarCorreo(SI_CorreoTranslator.ToBusiness(request.Correo))
                Case DC_CorreoOperation.EnviarCorreos

BL.EnviarCorreos(SI_CorreoTranslator.ToBusiness(request.CorreoList))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCo_Servidor(ByVal request As
MessageContract.MC_ServidorRequest) As MessageContract.MC_ServidorResponse
Implements ServiceContract.ISCSOA.SCO_Servidor
        Dim resp As New MC_ServidorResponse
        Try

            Select Case request.Operacion
                Case DC_ServidorOperation.FechaHoraSel
                    resp.FechaHora = DateTime.Now

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_SeguridadGrupoUsuario(ByVal request AS
MessageContract.MC_SeguridadGrupoUsuarioRequest) As
MessageContract.MC_SeguridadGrupoUsuarioResponse Implements
ServiceContract.ISCSOA.SCO_SeguridadGrupoUsuario
        Dim resp As New MC_SeguridadGrupoUsuarioResponse

        Try
            Dim BL As New BL_SeguridadGrupoUsuario
```

```
            Select Case request.Operacion
                Case
DC_SeguridadGrupoUsuarioOperation.PermisoUsuarioSelxSistemaGrupo
                    resp.SeguridadGrupoUsuarioList =
SI_SeguridadGrupoUsuarioTranslator.ToDataContract(BL.PermisoUsuarioSelxSistemaG
rupo(SI_SeguridadGrupoUsuarioTranslator.ToBusiness(request.SeguridadGrupoUsuari
o)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceinterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

#End Region

#Region "General"

    Public Function RetornaEstado() As Boolean Implements ISCSOA.RetornaEstado
        Dim resp As Boolean = True

        Try
            'validaciones adicionales
            resp = True
        Catch ex As Exception
            resp = False
            Dim rethrow = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_GENSistema(ByVal request As MC_GENSistemaRequest) As
MC_GENSistemaResponse Implements ISCSOA.SCO_GENSistema
        Dim resp As New MC_GENSistemaResponse

        Try
            Dim BL As New BL_GENSistema

            Select Case request.Operacion
                Case DC_GENSistemaOperation.GENSistemains

BL.GENSistemains(SI_GENSistemaTranslator.ToBusiness(request.Sistema),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENSistemaOperation.GENSistemaAct

BL.GENSistemaAct(SI_GENSistemaTranslator.ToBusiness(request.Sistema),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENSistemaOperation.GENSistemaElim

BL.GENSistemaElim(SI_GENSistemaTranslator.ToBusiness(request.Sistema),
```

```
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENSistemaOperation.GENSistemaSel
                    resp.Sistema =
SI_GENSistemaTranslator.ToDataContract(BL.GENSistemaSel(SI_GENSistemaTranslator
.ToBusiness(request.Sistema)))
                Case DC_GENSistemaOperation.GENSistemaSelTodo
                    resp.SistemaList =
SI_GENSistemaTranslator.ToDataContract(BL.GENSistemaSelTodo)
                Case DC_GENSistemaOperation.GENSistemaSelArbol
                    resp.SistemaList =
SI_GENSistemaTranslator.ToDataContract(BL.GENSistemaSelArbol)
                Case DC_GENSistemaOperation.GENSistemaActXGrupo

BL.GENSistemaActXGrupo(SI_GENSistemaTranslator.ToBusiness(request.SistemaListNu
evo), SI_GENSistemaTranslator.ToBusiness(request.SistemaListActualizado),
SI_GENSistemaTranslator.ToBusiness(request.SistemaListEliminado),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_GENModulo(ByVal request As MC_GENModuloRequest) As
MC_GENModuloResponse Implements ISCSOA.SCO_GENModulo
        Dim resp As New MC_GENModuloResponse

        Try
            Dim BL As New BL_GENModulo

            Select Case request.Operacion
                Case DC_GENModuloOperation.GENModuloIns
                    resp.IdentityGenerado =
BL.GENModuloIns(SI_GENModuloTranslator.ToBusiness(request.Modulo),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENModuloOperation.GENModuloAct

BL.GENModuloAct(SI_GENModuloTranslator.ToBusiness(request.Modulo),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENModuloOperation.GENModuloElim

BL.GENModuloElim(SI_GENModuloTranslator.ToBusiness(request.Modulo),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENModuloOperation.GENModuloSel
                    resp.Modulo =
SI_GENModuloTranslator.ToDataContract(BL.GENModuloSel(SI_GENModuloTranslator.To
Business(request.Modulo)))
                Case DC_GENModuloOperation.GENModuloSelxSistema
                    resp.ModuloList =
SI_GENModuloTranslator.ToDataContract(BL.GENModuloSelxSistema(SI_GENModuloTrans
lator.ToBusiness(request.Modulo)))
                Case DC_GENModuloOperation.GENModuloSelArbolxSistema
                    resp.ModuloList =
SI_GENModuloTranslator.ToDataContract(BL.GENModuloSelArbolxSistema(SI_GENModulo
```

```
Translator.ToBusiness(request.Modulo)))
                Case DC_GENModuloOperation.GENModuloSelTodo
                    resp.ModuloList =
SI_GENModuloTranslator.ToDataContract(BL.GENModuloSelTodo())
                Case DC_GENModuloOperation.GENModuloSelxSistemaRecursivo
                    resp.ModuloList =
SI_GENModuloTranslator.ToDataContract(BL.GENModuloSelxSistemaRecursivo(SI_GENMo
duloTranslator.ToBusiness(request.Modulo)))
                Case DC_GENModuloOperation.GENModuloSelArbolMenuxSistema
                    resp.ModuloList =
SI_GENModuloTranslator.ToDataContract(BL.GENModuloSelArbolMenuxSistema(SI_GENMo
duloTranslator.ToBusiness(request.Modulo)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_GENCodigo(ByVal request As MC_GENCodigoRequest) As
MC_GENCodigoResponse Implements ISCSOA.SCO_GENCodigo
        Dim resp As New MC_GENCodigoResponse

        Try
            Dim BL As New BL_GENCodigo

            Select Case request.Operacion
                Case DC_GENCodigoOperation.GENCodigoIns
                    resp.IdentityGenerado =
BL.GENCodigoins(SI_GENCodigoTranslator.ToBusiness(request.Codigo),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENCodigoOperation.GENCodigoAct

BL.GENCodigoAct(SI_GENCodigoTranslator.ToBusiness(request.Codigo),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENCodigoOperation.GENCodigoElim

BL.GENCodigoElim(SI_GENCodigoTranslator.ToBusiness(request.Codigo),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENCodigoOperation.GENCodigoSel
                    resp.Codigo =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSel(SI_GENCodigoTranslator.To
Business(request.Codigo)))
                Case DC_GENCodigoOperation.GENCodigoSelxSistema
                    resp.CodigoList =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSelxSistema(SI_GENCodigoTrans
lator.ToBusiness(request.Codigo)))
                Case DC_GENCodigoOperation.GENCodigoSelxSistemaGrupoCodigo
                    resp.CodigoList =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSelxSistemaGrupoCodigo(SI_GEN
CodigoTranslator.ToBusiness(request.Codigo)))
                Case DC_GENCodigoOperation.GENCodigoActXGrupo

BL.GENCodigoActXGrupo(SI_GENCodigoTranslator.ToBusiness(request.CodigoListNuevo
), SI_GENCodigoTranslator.ToBusiness(request.CodigoListActualizado),
```

```
SI_GENCodigoTranslator.ToBusiness(request.CodigoListEliminado),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENCodigoOperation.GENCodigoSelxGrupoCodigo
                    resp.CodigoList =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSelxGrupoCodigo(SI_GENCodigoT
ranslator.ToBusiness(request.Codigo)))
                Case
DC_GENCodigoOperation.GENCodigoSelHijosxCodigoPadreRecursivo
                    resp.CodigoList =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSelHijosxCodigoPadreRecursivo
(SI_GENCodigoTranslator.ToBusiness(request.Codigo)))
                Case DC_GENCodigoOperation.GENCodigoSelxCodigoPadre
                    resp.CodigoList =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSelxCodigoPadre(SI_GENCodigoT
ranslator.ToBusiness(request.Codigo)))
                Case
DC_GENCodigoOperation.GENCodigoSelNotificacionArbolLocacion
                    resp.CodigoList =
SI_GENCodigoTranslator.ToDataContract(BL.GENCodigoSelNotificacionArbolLocacion(
SI_GENCodigoTranslator.ToBusiness(request.Codigo)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceInterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_GENGrupoCodigo(ByVal request As
MessageContract.MC_GENGrupoCodigoRequest) As
MessageContract.MC_GENGrupoCodigoResponse Implements
ServiceContract.ISCSOA.SCO_GENGrupoCodigo
        Dim resp As New MC_GENGrupoCodigoResponse

        Try
            Dim BL As New BL_GENGrupoCodigo

            Select Case request.Operacion
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoAct

BL.GENGrupoCodigoAct(SI_GENGrupoCodigoTranslator.ToBusiness(request.GrupoCodigo
), SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoElim

BL.GENGrupoCodigoElim(SI_GENGrupoCodigoTranslator.ToBusiness(request.GrupoCodig
o), SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoins
                    resp.IdentityGenerado =
BL.GENGrupoCodigoIns(SI_GENGrupoCodigoTranslator.ToBusiness(request.GrupoCodigo
), SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoSel
                    resp.GrupoCodigo =
SI_GENGrupoCodigoTranslator.ToDataContract(BL.GENGrupoCodigoSel(SI_GENGrupoCodi
goTranslator.ToBusiness(request.GrupoCodigo)))
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoSelTodo
                    resp.GrupoCodigoList =
```

```
SI_GENGrupoCodigoTranslator.ToDataContract(BL.GENGrupoCodigoSelTodo())
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoActXGrupo

BL.GENGrupoCodigoActXGrupo(SI_GENGrupoCodigoTranslator.ToBusiness(request.Grupo
CodigoListNuevo),
SI_GENGrupoCodigoTranslator.ToBusiness(request.GrupoCodigoListActualizado),
SI_GENGrupoCodigoTranslator.ToBusiness(request.GrupoCodigoListEliminado),
SI_GENLogTransaccionalTranslator.ToBusiness(request.Log))
                Case DC_GENGrupoCodigoOperation.GENGrupoCodigoSelxUsoExclusivo
                    resp.GrupoCodigoList =
SI_GENGrupoCodigoTranslator.ToDataContract(BL.GENGrupoCodigoSelxUsoExclusivo(SI
_GENGrupoCodigoTranslator.ToBusiness(request.GrupoCodigo)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceinterfacePolicy)
            If (rethrow) Then
                Throw
            End If
        End Try

        Return resp
    End Function

    Public Function SCO_GENLogTransaccional(ByVal request As
MC_GENLogTransaccionalRequest) As MC_GENLogTransaccionalResponse Implements
ISCSOA.SCO_GENLogTransaccional
        Dim resp As New MC_GENLogTransaccionalResponse

        Try
            Dim BL As New BL_GENLogTransaccional

            Select Case request.Operacion
                Case DC_GENLogTransaccionalOperation.GENLogTransaccionalIns

BL.GENLogTransaccionalIns(SI_GENLogTransaccionalTranslator.ToBusiness(request.L
og))
                Case
DC_GENLogTransaccionalOperation.GENLogTransaccionalSelxPeriodos
                    resp.LogList =
SI_GENLogTransaccionalTranslator.ToDataContract(BL.GENLogTransaccionalSelxPerio
dos(SI_GENLogTransaccionalTranslator.ToBusiness(request.Log)))
                Case
DC_GENLogTransaccionalOperation.GENLogTransaccionalSelxSistemaPeriodos
                    resp.LogList =
SI_GENLogTransaccionalTranslator.ToDataContract(BL.GENLogTransaccionalSelxSiste
maPeriodos(SI_GENLogTransaccionalTranslator.ToBusiness(request.Log)))
                Case
DC_GENLogTransaccionalOperation.GENLogTransaccionalSelxSistemaPeriodosTipo
                    resp.LogList =
SI_GENLogTransaccionalTranslator.ToDataContract(BL.GENLogTransaccionalSelxSiste
maPeriodosTipo(SI_GENLogTransaccionalTranslator.ToBusiness(request.Log)))

                Case Else
                    Throw New Exception("Operacion del servicio no valida")
            End Select
        Catch ex As Exception
            Dim rethrow As Boolean = ExceptionPolicy.HandleException(ex,
ConstanteExcepcion.Politicas.ServiceinterfacePolicy)
            If (rethrow) Then
```

```
                Throw
            End If
        End Try

        Return resp
    End Function

#End Region

End Class
```

Figura 4.3: Clase que implementa los contratos del servicio.

4.1.3. CAPA DE IMPLEMENTACIÓN DE MÉTODOS

```
Imports Project.ServiceAccess.REFProxy

Public Class SA_GENCodigoService
    Implements ISA_GENCodigoService

#Region "Procedimientos"

    ''' <summary>
    ''' actualiza un registro
    ''' </summary>
    ''' <param name="item"></param>
    ''' <param name="log"></param>
    ''' <remarks>
    ''' 2014-10-26        Herbert Mendoza              creacion
    ''' </remarks>
    Public Sub GENCodigoAct(ByVal item As DC_GENCodigo, ByVal log As
DC_GENLogTransaccional) Implements ISA_GENCodigoService.GENCodigoAct

        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Log = log
        req.Operacion = DC_GENCodigoOperation.GENCodigoAct

        proxySIO = SAProxy.CrearProxySIO
        proxySIO.SCO_GENCodigo(req)

    End Sub

    ''' <summary>
    ''' elimina un registro
    ''' </summary>
    ''' <param name="item">idcodigo</param>
    ''' <param name="log"></param>
    ''' <remarks>
    ''' 2014-10-26        Herbert Mendoza              creacion
    ''' </remarks>
    Public Sub GENCodigoElim(item As DC_GENCodigo, ByVal log As
DC_GENLogTransaccional) Implements ISA_GENCodigoService.GENCodigoElim

        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO

        req = New MC_GENCodigoRequest
```

```
        req.Codigo = item
        req.Log = log
        req.Operacion = DC_GENCodigoOperation.GENCodigoElim

        proxySIO = SAProxy.CrearProxySIO
        proxySIO.SCO_GENCodigo(req)

    End Sub

    ''' <summary>
    ''' Realiza la ejecución de multiples Operaciones de mantenimiento
    ''' </summary>
    ''' <param name="ListNuevo"></param>
    ''' <param name="ListActualizado"></param>
    ''' <param name="ListEliminado"></param>
    ''' <param name="log"></param>
    ''' <remarks>
    ''' 2014-10-07        Herbert Mendoza              creacion
    ''' </remarks>
    Public Sub GENCodigoActXGrupo(ListNuevo As REFProxy.DC_GENCodigoList,
ListActualizado As REFProxy.DC_GENCodigoList, ListEliminado As
REFProxy.DC_GENCodigoList, log As REFProxy.DC_GENLogTransaccional) Implements
ISA_GENCodigoService.GENCodigoActXGrupo
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO

        req = New MC_GENCodigoRequest

        req.CodigoListNuevo = ListNuevo
        req.CodigoListActualizado = ListActualizado
        req.CodigoListEliminado = ListEliminado
        req.Log = log
        req.Operacion = DC_GENCodigoOperation.GENCodigoActXGrupo

        proxySIO = SAProxy.CrearProxySIO
        proxySIO.SCO_GENCodigo(req)

    End Sub

#End Region

#Region "Funciones"

    ''' <summary>
    ''' inserta un registro
    ''' </summary>
    ''' <param name="item"></param>
    ''' <param name="log"></param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-26        Herbert Mendoza              creacion
    ''' </remarks>
    Public Function GENCodigoins(item As DC_GENCodigo, ByVal log As
DC_GENLogTransaccional) As Integer Implements ISA_GENCodigoService.GENCodigoins
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As Integer

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Log = log
        req.Operacion = DC_GENCodigoOperation.GENCodigoIns
```

```
        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).IdentityGenerado

        Return resp
    End Function

    ''' <summary>
    ''' Selecciona un registro
    ''' </summary>
    ''' <param name="item">idcodigo</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-26        Herbert Mendoza              creacion
    ''' </remarks>
    Public Function GENCodigoSel(item As DC_GENCodigo) As DC_GENCodigo
Implements ISA_GENCodigoService.GENCodigoSel
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As DC_GENCodigo

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Operacion = DC_GENCodigoOperation.GENCodigoSel

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).Codigo

        Return resp
    End Function

    ''' <summary>
    ''' selecciona registros por grupocodigo y sistema
    ''' </summary>
    ''' <param name="item">idsistema, idgrupocodigo</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-06        Herbert Mendoza              creacion
    ''' </remarks>
    Public Function GENCodigoSelxSistemaGrupoCodigo(item As DC_GENCodigo) As
REFProxy.DC_GENCodigoList Implements
ISA_GENCodigoService.GENCodigoSelxSistemaGrupoCodigo
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As DC_GENCodigoList

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Operacion = DC_GENCodigoOperation.GENCodigoSelxSistemaGrupoCodigo

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).CodigoList

        Return resp
    End Function

    ''' <summary>
    ''' selecciona registros por sistema
    ''' </summary>
    ''' <param name="item">idSistema</param>
    ''' <returns></returns>
```

```
    ''' <remarks>
    ''' 2014-10-26        Herbert Mendoza              creacion
    ''' </remarks>
    Public Function GENCodigoSelxSistema(item As DC_GENCodigo) As
REFProxy.DC_GENCodigoList Implements ISA_GENCodigoService.GENCodigoSelxSistema
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As DC_GENCodigoList

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Operacion = DC_GENCodigoOperation.GENCodigoSelxSistema

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).CodigoList

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos hijos por codigo padre. consulta recursiva
    ''' </summary>
    ''' <param name="item">idcodigopadre</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04        Herbert Mendoza      creacion
    ''' </remarks>
    Public Function GENCodigoSelHijosxCodigoPadreRecursivo(item As
REFProxy.DC_GENCodigo) As REFProxy.DC_GENCodigoList Implements
ISA_GENCodigoService.GENCodigoSelHijosxCodigoPadreRecursivo
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As DC_GENCodigoList

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Operacion =
DC_GENCodigoOperation.GENCodigoSelHijosxCodigoPadreRecursivo

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).CodigoList

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos por grupocodigo
    ''' </summary>
    ''' <param name="item">idgrupocodigo</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04       Herbert Mendoza           creacion
    ''' </remarks>
    Public Function GENCodigoSelxGrupoCodigo(item As REFProxy.DC_GENCodigo) As
REFProxy.DC_GENCodigoList Implements
ISA_GENCodigoService.GENCodigoSelxGrupoCodigo
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As DC_GENCodigoList

        req= New MC_GENCodigoRequest
```

```
        req.Codigo = item
        req.Operacion = DC_GENCodigoOperation.GENCodigoSelxGrupoCodigo

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).CodigoList

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos por codigo padre
    ''' </summary>
    ''' <param name="item">idcodigopadre</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04        Herbert Mendoza      creacion
    ''' </remarks>
    Public Function GENCodigoSelxCodigoPadre(item AS REFProxy.DC_GENCodigo) As
REFProxy.DC_GENCodigoList Implements
ISA_GENCodigoService.GENCodigoSelxCodigoPadre
        Dim req AS MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp AS DC_GENCodigoList

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Operacion = DC_GENCodigoOperation.GENCodigoSelxCodigoPadre

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).CodigoList

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos por locacion
    ''' </summary>
    ''' <param name="item">idcodigopadre</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04        Herbert Mendoza      creacion
    ''' </remarks>
    Public Function GENCodigoSelNotificacionArbolLocacion(item As
REFProxy.DC_GENCodigo) As REFProxy.DC_GENCodigoList Implements
ISA_GENCodigoService.GENCodigoSelNotificacionArbolLocacion
        Dim req As MC_GENCodigoRequest
        Dim proxySIO As ISCSIO
        Dim resp As DC_GENCodigoList

        req = New MC_GENCodigoRequest

        req.Codigo = item
        req.Operacion =
DC_GENCodigoOperation.GENCodigoSelNotificacionArbolLocacion

        proxySIO = SAProxy.CrearProxySIO
        resp = proxySIO.SCO_GENCodigo(req).CodigoList

        Return resp
    End Function
#End Region
```

```
End Class
```

Figura 4.4: Métodos de acceso al servicio, GENModulo.

4.2 SOA PARA DISEÑO

4.2.1. OPERACIONES CON FUNCIONALIDAD

Project.BusinessLogic: GENCodigo

```
Imports Project.BusinessEntity
Imports System.Transactions

Public Class BL_GENCodigo
    Inherits BLBase

#Region "Atributos"
#End Region

#Region "Procedimientos"

    ''' <summary>
    ''' actualiza un registro
    ''' </summary>
    ''' <param name="item"></param>
    ''' <param name="log"></param>
    ''' <remarks>
    ''' 2014-10-12        Herbert Mendoza              creación
    ''' </remarks>
    Public Sub GENCodigoAct(ByVal item As BE_GENCodigo, ByVal log As
BE_GENLogTransaccional)
        Me.DA_GENCodigo.GENCodigoAct(item, log)
    End Sub

    ''' <summary>
    ''' elimina un registro
    ''' </summary>
    ''' <param name="item">idcodigo</param>
    ''' <param name="log"></param>
    ''' <remarks>
    ''' 2014-10-26        Herbert Mendoza              creacion
    ''' </remarks>
    Public Sub GENCodigoElim(ByVal item As BE_GENCodigo, ByVal log As
BE_GENLogTransaccional)
        Me.DA_GENCodigo.GENCodigoElim(item, log)
    End Sub

    ''' <summary>
    ''' Realiza la ejecución de multiples operaciones de mantenimiento
    ''' </summary>
    ''' <param name="ListNuevo"></param>
    ''' <param name="ListActualizado"></param>
    ''' <param name="ListEliminado"></param>
    ''' <param name="log"></param>
    ''' <remarks>
    ''' 2014-10-07        Herbert Mendoza              creacion
    ''' </remarks>
    Public Sub GENCodigoActXGrupo(ByVal ListNuevo As List(Of BE_GENCodigo),
                                  ByVal ListActualizado As List(Of
BE_GENCodigo),
                                  ByVal ListEliminado As List(Of BE_GENCodigo),
                                  ByVal log As BE_GENLogTransaccional)
```

```
        Using txScope As New TransactionScope(TransactionScopeOption.Required)
            For Each Item As BE_GENCodigo In ListEliminado
                GENCodigoElim(Item, log)
            Next

            For Each Item As BE_GENCodigo In ListActualizado
                GENCodigoAct(Item, log)
            Next

            For Each Item As BE_GENCodigo In ListNuevo
                GENCodigoIns(Item, log)
            Next

            txScope.Complete()
        End Using
    End Sub

#End Region

#Region "Funciones"

    ''' <summary>
    ''' inserta un registro
    ''' </summary>
    ''' <param name="item"></param>
    ''' <param name="log"></param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-12        Herbert Mendoza              creación
    ''' </remarks>
    Public Function GENCodigoIns(ByVal item As BE_GENCodigo, ByVal log As
BE_GENLogTransaccional) As Integer
        Dim resp As Integer

        resp = Me.DA_GENCodigo.GENCodigoIns(item, log)

        Return resp
    End Function

    ''' <summary>
    ''' selecciona un registro
    ''' </summary>
    ''' <param name="item">idcodigo</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-12        Herbert Mendoza              creación
    ''' </remarks>
    Public Function GENCodigoSel(item As BE_GENCodigo) As BE_GENCodigo
        Dim resp As BE_GENCodigo

        resp = Me.DA_GENCodigo.GENCodigoSel(item)

        Return resp
    End Function

    ''' <summary>
    ''' selecciona registros por sistema
    ''' </summary>
    ''' <param name="item">idsistema</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-12        Herbert Mendoza              creación
    ''' </remarks>
```

```
    Public Function GENCodigoSelxSistema(ByVal item As BE_GENCodigo) As List(Of
BE_GENCodigo)
        Dim resp As List(Of BE_GENCodigo)

        resp = Me.DA_GENCodigo.GENCodigoSelxSistema(item)

        Return resp
    End Function

    ''' <summary>
    ''' selecciona registros por grupocodigo y sistema
    ''' </summary>
    ''' <param name="item">idsistema, idgrupocodigo</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-12        Herbert Mendoza              creación
    ''' </remarks>
    Public Function GENCodigoSelxSistemaGrupoCodigo(ByVal item As BE_GENCodigo)
As List(Of BE_GENCodigo)
        Dim resp As List(Of BE_GENCodigo)

        resp = Me.DA_GENCodigo.GENCodigoSelxSistemaGrupoCodigo(item)

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos por grupocodigo
    ''' </summary>
    ''' <param name="item">idgrupocodigo</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04       Herbert Mendoza        creacion
    ''' </remarks>
    Public Function GENCodigoSelxGrupoCodigo(ByVal item As BE_GENCodigo) As
List(Of BE_GENCodigo)
        Dim resp As List(Of BE_GENCodigo)

        resp = Me.DA_GENCodigo.GENCodigoSelxGrupoCodigo(item)

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos hijos por idcodigopadre. consulta recursiva
    ''' </summary>
    ''' <param name="item">idcodigopadre</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04     Herbert Mendoza     creacion
    ''' </remarks>
    Public Function GENCodigoSelHijosxCodigoPadreRecursivo(ByVal item As
BE_GENCodigo) As List(Of BE_GENCodigo)
        Dim resp As List(Of BE_GENCodigo)

        resp = Me.DA_GENCodigo.GENCodigoSelHijosxCodigoPadreRecursivo(item)

        Return resp
    End Function

    ''' <summary>
    ''' lista de codigos por codigo padre
    ''' </summary>
```

```
    ''' <param name="item">idcodigopadre</param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04      Herbert Mendoza          creacion
    ''' </remarks>
    Public Function GENCodigoSelxCodigoPadre(ByVal item As BE_GENCodigo) As
List(Of BE_GENCodigo)
        Dim resp As List(Of BE_GENCodigo)

        resp = Me.DA_GENCodigo.GENCodigoSelxCodigoPadre(item)

        Return resp
    End Function

    ''' <summary>
    ''' lista los correos electrónicos para notificación por locación
    ''' </summary>
    ''' <param name="item"></param>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-04      Herbert Mendoza          creacion
    ''' </remarks>
    Public Function GENCodigoSelNotificacionArbolLocacion(ByVal item As
BE_GENCodigo) As List(Of BE_GENCodigo)
        Dim resp As List(Of BE_GENCodigo)

        resp = Me.DA_GENCodigo.GENCodigoSelNotificacionArbolLocacion(item)

        Return resp
    End Function

#End Region

End Class
```

Figura 4.5: Funcionalidad de entidad GENModulo.

4.2.2. OPERACIONES CON PARÁMETROS

Project.DataContract: ParametroSQL

```
Imports System.Runtime.Serialization

<Serializable()> _
<DataContract()> _
Public Class DC_ParametroSQL

#Region "Propiedades"

    Private _Sentencia As String
    ''' <summary>
    ''' sentencia
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Sentencia() As String
        Get
            Return _Sentencia
        End Get
        Set(ByVal value As String)
            _Sentencia= value
        End Set
```

```
    End Property

    Private _CadenaConexion As String
    ''' <summary>
    ''' cadena de conexion
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property CadenaConexion() As String
        Get
            Return _CadenaConexion
        End Get
        Set(ByVal value As String)
            _CadenaConexion = value
        End Set
    End Property

    Private _Valor As Decimal
    ''' <summary>
    ''' valor
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor() As Decimal
        Get
            Return _Valor
        End Get
        Set(ByVal value As Decimal)
            _Valor= value
        End Set
    End Property

    Private _ValorTexto As String
    ''' <summary>
    ''' valor de tipo texto que retorna la ejecucion de la sentencia
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property ValorTexto() As String
        Get
            Return _ValorTexto
        End Get
        Set(ByVal value As String)
            _ValorTexto = value
        End Set
    End Property

    Private _Proveedor As String
    ''' <summary>
    ''' proveedor
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Proveedor() As String
        Get
```

```
            Return _Proveedor
        End Get
        Set(ByVal value As String)
            _Proveedor= value
        End Set
    End Property

    Private _ResultadoConsulta As DataTable
    ''' <summary>
    ''' Resultado usado para devolver objetos de tipo DataTablae
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property ResultadoConsulta() As DataTable
        Get
            Return _ResultadoConsulta
        End Get
        Set(ByVal value As DataTable)
            _ResultadoConsulta = value
        End Set
    End Property

    Private _EvaluaTipoOrigenDestino As Boolean = False
    ''' <summary>
    ''' indica si se evalua el tipo origen con el tipo destino para un cast.
default false
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember(EmitDefaultValue:=False)> _
    Public Property EvaluaTipoOrigenDestino() As Boolean
        Get
            Return _EvaluaTipoOrigenDestino
        End Get
        Set(ByVal value As Boolean)
            _EvaluaTipoOrigenDestino = value
        End Set
    End Property

#End Region

End Class
```

Figura 4.6: Operaciones con parámetros SQL.

4.2.3. DEFINICIÓN DE DATA CONTRACT Y MESSAGE CONTRACT

Project.DataContract

```
Imports System.Runtime.Serialization

<DataContract()> _
Public Class DC_GENCodigo

#Region "Propiedades"

#Region "Entidad"

    Private _IDCodigo As Integer
    ''' <summary>
    ''' id del codigo
```

```
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property IDCodigo() As Integer
        Get
            Return _IDCodigo
        End Get
        Set(ByVal value As Integer)
            _IDCodigo = value
        End Set
    End Property

    Private _IDGrupoCodigo As Integer
    ''' <summary>
    ''' id del grupo del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property IDGrupoCodigo() As Integer
        Get
            Return _IDGrupoCodigo
        End Get
        Set(ByVal value As Integer)
            _IDGrupoCodigo = value
        End Set
    End Property

    Private _IDSistema As String
    ''' <summary>
    ''' id del sistema
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property IDSistema() As String
        Get
            Return _IDSistema
        End Get
        Set(ByVal value As String)
            _IDSistema = value
        End Set
    End Property

    Private _DescripcionCorta As String
    ''' <summary>
    ''' descripcion corta del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property DescripcionCorta() As String
        Get
            Return _DescripcionCorta
        End Get
        Set(ByVal value As String)
            _DescripcionCorta = value
        End Set
```

```
    End Property

    Private _DescripcionLarga As String
    ''' <summary>
    ''' descripcion larga del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property DescripcionLarga() As String
        Get
            Return _DescripcionLarga
        End Get
        Set(ByVal value As String)
            _DescripcionLarga = value
        End Set
    End Property

    Private _Secuencia As Byte
    ''' <summary>
    ''' secuencia del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Secuencia() As Byte
        Get
            Return _Secuencia
        End Get
        Set(ByVal value As Byte)
            _Secuencia= value
        End Set
    End Property

    Private _Valor01 As String
    ''' <summary>
    ''' valor 1 del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor01() As String
        Get
            Return _Valor01
        End Get
        Set(ByVal value As String)
            _Valor01 = value
        End Set
    End Property

    Private _Valor02 As String
    ''' <summary>
    ''' valor 2 del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor02() As String
        Get
```

```
            Return _Valor02
        End Get
        Set(ByVal value As String)
            _Valor02 = value
        End Set
    End Property

    Private _Valor03 As String
    ''' <summary>
    ''' valor 3 del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor03() As String
        Get
            Return _Valor03
        End Get
        Set(ByVal value As String)
            _Valor03 = value
        End Set
    End Property

    Private _Valor04 As String
    ''' <summary>
    ''' valor 4 del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor04() As String
        Get
            Return _Valor04
        End Get
        Set(ByVal value As String)
            _Valor04 = value
        End Set
    End Property

    Private _Valor05 As String
    ''' <summary>
    ''' valor 5 para uso de codigo
    ''' </summary>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor05() As String
        Get
            Return _Valor05
        End Get
        Set(ByVal value As String)
            _Valor05 = value
        End Set
    End Property

    Private _Valor06 As String
    ''' <summary>
    ''' valor 6 para uso de codigo
    ''' </summary>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor06() As String
```

```
        Get
            Return _Valor06
        End Get
        Set(ByVal value As String)
            _Valor06 = value
        End Set
    End Property

    Private _Valor07 As String
    ''' <summary>
    ''' valor 7
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor07() As String
        Get
            Return _Valor07
        End Get
        Set(ByVal value As String)
            _Valor07 = value
        End Set
    End Property

    Private _Valor08 As String
    ''' <summary>
    ''' valor 8
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Valor08() As String
        Get
            Return _Valor08
        End Get
        Set(ByVal value As String)
            _Valor08 = value
        End Set
    End Property

    Private _Color As Integer?
    ''' <summary>
    ''' color del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Color() As Integer?
        Get
            Return _Color
        End Get
        Set(ByVal value As Integer?)
            _Color= value
        End Set
    End Property

    Private _CodRef01 As String
    ''' <summary>
    ''' codigo 1 de referencia del id
    ''' </summary>
```

```
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property CodRef01() As String
        Get
            Return _CodRef01
        End Get
        Set(ByVal value As String)
            _CodRef01 = value
        End Set
    End Property

    Private _IDCodigoPadre As Nullable(Of Integer)
    ''' <summary>
    ''' id del codigo padre
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property IDCodigoPadre() As Nullable(Of Integer)
        Get
            Return _IDCodigoPadre
        End Get
        Set(ByVal value As Nullable(Of Integer))
            _IDCodigoPadre = value
        End Set
    End Property

    Private _Activo As Boolean
    ''' <summary>
    ''' indica si esta activo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Activo() As Boolean
        Get
            Return _Activo
        End Get
        Set(ByVal value As Boolean)
            _Activo= value
        End Set
    End Property

    Private _FechaCrea As DateTime
    ''' <summary>
    ''' fecha de creacion
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property FechaCrea() As DateTime
        Get
            Return _FechaCrea
        End Get
        Set(ByVal value As DateTime)
            _FechaCrea = value
        End Set
    End Property
```

```
    Private _UsuarioCrea As String
    ''' <summary>
    ''' usuario de creacion
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property UsuarioCrea() As String
        Get
            Return _UsuarioCrea
        End Get
        Set(ByVal value As String)
            _UsuarioCrea = value
        End Set
    End Property

    Private _FechaAct As Nullable(Of Date)
    ''' <summary>
    ''' fecha de ultima actualizacion
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property FechaAct() As Nullable(Of Date)
        Get
            Return _FechaAct
        End Get
        Set(ByVal value As Nullable(Of Date))
            _FechaAct = value
        End Set
    End Property

    Private _UsuarioAct As String
    ''' <summary>
    ''' usuario de ultima actualizacion
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property UsuarioAct() As String
        Get
            Return _UsuarioAct
        End Get
        Set(ByVal value As String)
            _UsuarioAct = value
        End Set
    End Property

#End Region

#Region "Adicionales"

    Private _Codigo As String
    ''' <summary>
    ''' ADICIONAL: codigo de id
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
```

```
    <DataMember()> _
    Public Property Codigo() As String
        Get
            Return _Codigo
        End Get
        Set(ByVal value As String)
            _Codigo = value
        End Set
    End Property

    Private _CodigoPadre As String
    ''' <summary>
    ''' ADICIONAL: codigo id del padre
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property CodigoPadre() As String
        Get
            Return _CodigoPadre
        End Get
        Set(ByVal value As String)
            _CodigoPadre = value
        End Set
    End Property

    Private _Descripcion As String
    ''' <summary>
    ''' ADICIONAL: descripcion del codigo
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Descripcion() As String
        Get
            Return _Descripcion
        End Get
        Set(ByVal value As String)
            _Descripcion = value
        End Set
    End Property

    Private _Nivel As Integer
    ''' <summary>
    ''' ADICIONAL: nivel del id
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property Nivel() As Integer
        Get
            Return _Nivel
        End Get
        Set(ByVal value As Integer)
            _Nivel= value
        End Set
    End Property

    Private _ReservaTipoID As Integer
    ''' <summary>
```

```
    ''' ADICIONAL: Tipo de generacion de ID
    ''' </summary>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property ReservaTipoID() As Integer
        Get
            Return _ReservaTipoID
        End Get
        Set(ByVal value As Integer)
            _ReservaTipoID = value
        End Set
    End Property

    Private _ReservaRangoMin As Integer
    ''' <summary>
    ''' ADICIONAL: rango min de reserva de id
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <DataMember()> _
    Public Property ReservaRangoMin() As Integer
        Get
            Return _ReservaRangoMin
        End Get
        Set(ByVal value As Integer)
            _ReservaRangoMin = value
        End Set
    End Property

#End Region

#End Region

End Class
```

Figura 4.7: Definición de DataContract y MessageContract.

Project.MessageContract.Request

```
Imports System.ServiceModel
Imports Project.DataContract

<MessageContract(IsWrapped:=True)> _
Public Class MC_GENCodigoRequest

    <MessageBodyMember()> _
    Public Operacion As DC_GENCodigoOperation

    <MessageBodyMember()> _
    Public Codigo As DC_GENCodigo

    <MessageBodyMember()> _
    Public CodigoList As DC_GENCodigoList

    <MessageBodyMember()> _
    Public Log As DC_GENLogTransaccional

    <MessageBodyMember()> _
    Public CodigoListNuevo As DC_GENCodigoList

    <MessageBodyMember()> _
    Public CodigoListActualizado As DC_GENCodigoList

    <MessageBodyMember()> _
```

```
    Public CodigoListEliminado AS DC_GENCodigoList

End Class
```

Figura 4.8: Request MessageContract de la Entidad GENCodigo.

Project.MessageContract.Response

```
Imports System.ServiceModel
Imports Project.DataContract

<MessageContract(IsWrapped:=True)> _
Public Class MC_GENCodigoResponse

    <MessageBodyMember()> _
    Public Codigo As DC_GENCodigo

    <MessageBodyMember()> _
    Public CodigoList As DC_GENCodigoList

    <MessageBodyMember()> _
    Public IdentityGenerado As Integer

End Class
```

Figura 4.9: Response MessageContract de Entidad GENCodigo.

4.3 SOA PARA IMPLEMENTACIÓN

4.3.1. FORMULARIOS MAESTROS PARA INGRESO OPERATIVO

frmMaestro.vb

```
Imports Microsoft.Practices.Unity

Public Class frmMaestro
    Inherits System.Windows.Forms.Form

#Region "Atributos"

    Protected ContainerService As IUnityContainer
    Protected SesionService As Services.Session.ISesion
    Protected MenuSeleccionado As String = String.Empty
    Protected _TipoAcceso As Acceso.eTipoAcceso = Acceso.eTipoAcceso.Escritura

#End Region

#Region "Constructor"

    Sub New()
        InitializeComponent()

        SesionService = New Services.Session.Sesion
    End Sub

    Sub New(ByVal container As IUnityContainer)
        InitializeComponent()

        Me.ContainerService = container
        Me.SesionService = container.Resolve(Of Services.Session.ISesion)()
    End Sub

#End Region

#Region "Propiedades"
```

```
    ''' <summary>
    ''' Propiedad que almacena un objeto de la Opción del Menu.
    ''' Esta propiedad podra ser usado de manera independiente para cada
formulario
    ''' que herede de WinFormMaster, asimismo permitira almacenar cualquier
tipo de dato.
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Property MenuItemSeleccionado() As Object
        Get
            If Me.SesionService.RetornarValor(Of Object)(Me.Name) Is Nothing
Then
                Return MenuSeleccionado
            End If
            Return SesionService.RetornarValor(Of Object)(Me.Name)
        End Get
        Set(ByVal value As Object)
            SesionService.RegistraValor(Of Object)(Me.Name, value)
        End Set
    End Property

    ''' <summary>
    ''' Propiedad que determina el tipo de acceso que se tiene sobre el
formulario
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Property TipoAcceso() As Acceso.eTipoAcceso
        Get
            Return _TipoAcceso
        End Get
        Set(ByVal value As Acceso.eTipoAcceso)
            _TipoAcceso = value
        End Set
    End Property

    ''' <summary>
    ''' Propiedad que determina si el formulario tiene actualizaciones
pendientes de almacenar.
    ''' </summary>
    ''' <remarks></remarks>
    Private _ActualizacionPendiente As Boolean
    Public Property ActualizacionPendiente() As Boolean
        Get
            Return _ActualizacionPendiente
        End Get
        Set(ByVal value As Boolean)
            _ActualizacionPendiente = value
        End Set
    End Property

    Private _MDI As System.Windows.Forms.Form
    ''' <summary>
    ''' Referencia al formulario MDI al cual esta vinculado el formulario
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Property MDI() As System.Windows.Forms.Form
```

```
        Get
            Return _MDI
        End Get
        Set(ByVal value As System.Windows.Forms.Form)
            _MDI = value
        End Set
    End Property


#End Region

#Region "Procedimientos"

    ''' <summary>
    ''' Procedimiento sobreescrebible, que permitira adicionar funcionalidad de
aplicar accesos sobre controles que
    ''' contienen funcionalidad del negocio.
    ''' </summary>
    ''' <remarks>
    ''' 2014-10-10        Herbert Mendoza             Creación
    ''' </remarks>
    Overridable Sub _AplicarSeguridad()

    End Sub

    ''' <summary>
    ''' Procedimiento sobreescrebible, que permitira adicionar funcionalidad de
aplicar la seguridad de aprobación.
    ''' </summary>
    ''' <param name="AplicarSeguridad">Indica si desea activar o
bloquear</param>
    ''' <remarks>
    ''' 2014-10-16        Herbert Mendoza             Creación
    ''' </remarks>
    Overridable Sub _SeguridadDeAprobacion(ByVal AplicarSeguridad As Boolean)

    End Sub

#End Region

#Region "Funciones"

    Overridable Function _Salir() As Boolean

    End Function

#End Region

End Class
```

Figura 4.10: Implementación de formulario operativo frmMaestro

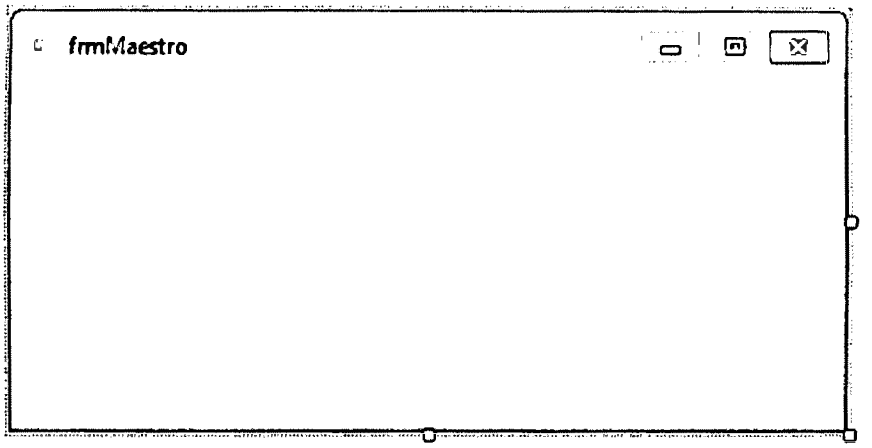

Figura 4.11: Formulario operativo frmMaestro.

frmMasterMaestro.vb

```
Imports Infragistics.Win
Imports Microsoft.Practices.EnterpriseLibrary.ExceptionHandling
Imports Microsoft.Practices.Unity
Imports Framework.Services.Exception
Imports Framework.Services.Session
Imports Project.ServiceAccess.SIOProxy
Imports System.IO
Imports Project.Shared.Code

Public Class frmMasterMaestro
    Inherits Framework.WindowsForms.Master.frmMaestro

    Public Enum TipoMensaje
        NINGUNO= 0
        TIPO_OK = 1
        TIPO_INFORMACION = 2
        TIPO_WARNING = 3
        TIPO_ERROR = 4
    End Enum

#Region "Declaracion de Variables"
#End Region

#Region "Propiedades"

    Private _ModuloSeleccionado As DC_GENModulo
    ''' <summary>
    ''' Propiedad que devuleve la Clase DC_GENModulo de la opción seleccionado
en el Treeview
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-01        Herbert Mendoza              creación
    ''' </remarks>
    Public ReadOnly Property ModuloSeleccionado() As DC_GENModulo
        Get
            _ModuloSeleccionado = Nothing

            If MenuItemSeleccionado IsNot Nothing Then

                If MenuItemSeleccionado.GetType =
GetType(UltraWinTree.UltraTreeNode) Then
```

```
                    If DirectCast(MenuItemSeleccionado,
UltraWinTree.UltraTreeNode).Tag IsNot Nothing Then

                        _ModuloSeleccionado =
DirectCast(DirectCast(MenuitemSeleccionado, UltraWinTree.UltraTreeNode).Tag,
DC_GENModulo)

                    End If

                End If

            End If

            Return _ModuloSeleccionado
        End Get
    End Property

    ''' <summary>
    ''' Propiedad que devuelve el formulario MDI del Modelo
    ''' </summary>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-01        Herbert Mendoza              creación
    ''' </remarks>
    Public ReadOnly Property FormularioMDI() As frmMDI
        Get
            Return If(MyBase.MDI IsNot Nothing, DirectCast(MyBase.MDI, frmMDI),
Nothing)
        End Get
    End Property

#End Region

#Region "Constructor"

    Public Sub New()
        MyBase.New()
        InitializeComponent()
        Me.SesIonService = New Sesion
    End Sub

    Public Sub New(ByVal container As IUnityContainer)
        MyBase.New(container)
        InitializeComponent()
    End Sub

#End Region

#Region "Procedimientos"

    ''' <summary>
    ''' procedimiento para exportar las grillas a excel.
    ''' se carga cada grilla en una hoja excel
    ''' </summary>
    ''' <remarks>
    ''' 20140127          Herbert Mendoza          creacion
    ''' </remarks>
    Public Overridable Sub ExportarExcel()
        Dim Ruta As String
        Dim Aplicacionxls As Microsoft.Office.Interop.Excel.Application
        Dim grillas As List(Of UltraWinGrid.UltraGrid)
        Dim grilla As UltraWinGrid.UltraGrid
        Dim workbook As Infragistics.Excel.Workbook
```

```
        Dim worksheet As Infragistics.Excel.Worksheet
        Dim grdExporter As New UltraWinGrid.ExcelExport.UltraGridExcelExporter
        Dim FileExt As String

        'captura grillas en pantalla
        grillas= Procedimiento.ObtenerControles(Of UltraWinGrid.UltraGrid)(Me)

        'abre excel si hay grillas
        If grillas IsNot Nothing AndAlso grillas.Count > 0 Then

            Aplicacionxls = New Microsoft.Office.Interop.Excel.Application

            workbook = New Infragistics.Excel.Workbook

            If Aplicacionxls.DefaultSaveFormat =
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook Then

workbook.SetCurrentFormat(Infragistics.Excel.WorkbookFormat.Excel2007)
                FileExt = "xlsx"
            Else

workbook.SetCurrentFormat(Infragistics.Excel.WorkbookFormat.Excel97To2003)
                FileExt = "xls"
            End If

            'exporta cada grilla encontrada
            For Each grilla In grillas
                worksheet = workbook.Worksheets.Add(grilla.Name)
                grdExporter.Export(grilla, worksheet)
            Next

            'define la ruta
            Ruta= ObtenerNombreArchivo(Path.GetTempPath,
ModuloSeleccionado.Descripcion, FileExt)

            'graba archivo
            workbook.Save(Ruta)

            'abre excel
            Aplicacionxls.Workbooks.Open(Ruta)
            Aplicacionxls.Visible = True

            grdExporter.Dispose()

        End If

    End Sub

    ''' <summary>
    ''' Obtiene el nombre a Exportar
    ''' </summary>
    ''' <param name="Ruta">Ruta donde se creara el archivo</param>
    ''' <param name="Nombre">Nombre con el que se creara el archivo</param>
    ''' <remarks>
    ''' 20140127          Herbert Mendoza         creacion
    ''' </remarks>
    Public Overridable Function ObtenerNombreArchivo(ByVal Ruta As String,
ByVal Nombre As String, Optional ByVal Extension As String = "xls") As String
        Dim blnArchivo As Boolean = True
        Dim RutaArchivo As String = ""
        Dim NombreTemp As String = ""
        Dim i As Int32
```

```
        i = 1
        NombreTemp = Nombre
        While blnArchivo
            RutaArchivo = String.Format("{0}\{1}.{2}", Ruta, NombreTemp,
Extension)
            RutaArchivo = RutaArchivo.Replace("\\", "\")
            If System.IO.File.Exists(RutaArchivo) Then
                NombreTemp = Nombre & "(" & CStr(i) & ")"
            Else
                blnArchivo = False
            End If
            i += 1
        End While

        Return RutaArchivo
    End Function

#End Region

#Region "Funciones"

    ''' <summary>
    ''' para revisar datos adicionales antes de permitir la aprobacion
    ''' lanza error si hay rompe alguna validacion
    ''' </summary>
    ''' <remarks>
    ''' 2014-10-16         Herbert Mendoza                    creacion
    ''' </remarks>
    Public Overridable Function ValidarGrabarAprobacion() As Boolean
        Return True
    End Function

    ''' <summary>
    ''' preparar log para uso de modulo
    ''' </summary>
    ''' <returns></returns>
    ''' <remarks>
    ''' 2014-10-16         Herbert Mendoza                    creacion
    ''' </remarks>
    Public Overridable Function PreparaLog() As DC_GENLogTransaccional
        Dim log As DC_GENLogTransaccional
        log = New DC_GENLogTransaccional
        log.IDSistema = ModuloSeleccionado.IDSistema
        log.NombreEquipo = SesionService.NombreEquipoUsuario
        log.SesionWindows = SesionService.DominioWindowsUsuario
        log.CodigoUsuario = SesionService.CodigoUsuario
        log.NombreUsuario = SesionService.NombreUsuario
        log.Origen = Me.Name
        log.FechaDato = Now
        log.RegistrarLog = ModuloSeleccionado.GrabarLog
        Return log
    End Function


#End Region

#Region "Eventos"

    Private Sub frmMasterMaestro_Load(sender As Object, e As System.EventArgs)
Handles Me.Load

        Try
```

```
UltraWinGrid.Resources.Customizer.SetCustomizedString("DataErrorCellUpdateUnabl
eToUpdateValue", "El valor ingresado es incorrecto")

        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try

    End Sub

#End Region

End Class
```

Figura 4.12: Implementación de formulario operativo frmMasterMaestro.

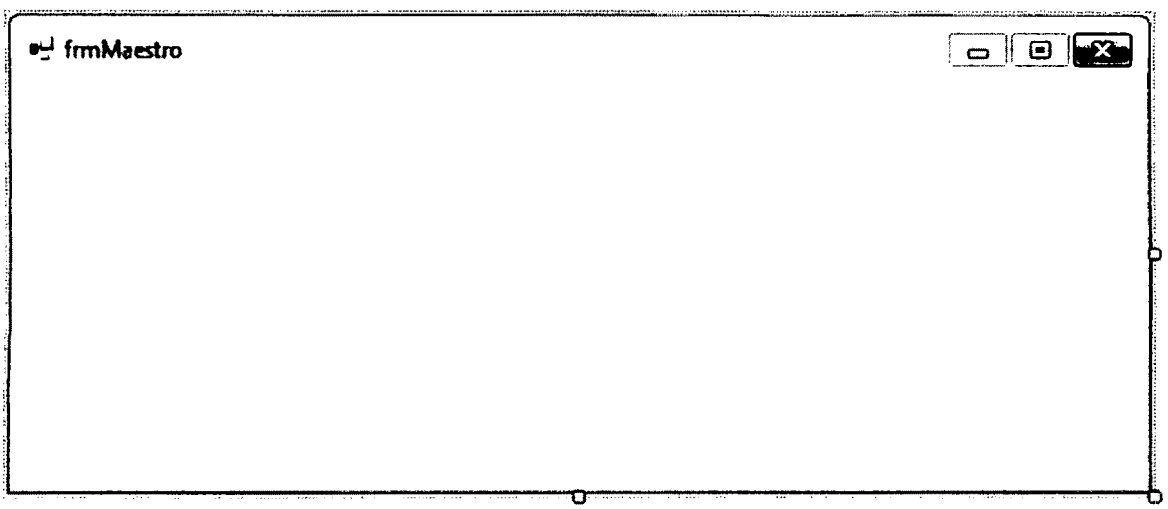

Figura 4.13: Formulario operativo frmMasterMaestro.

frmMasterMantenimiento.vb

```
Imports Infragistics.Win
Imports Microsoft.Practices.EnterpriseLibrary.ExceptionHandling
Imports Microsoft.Practices.Unity
Imports Framework.Services.Exception
Imports Framework.Services.Session
Imports Project.ServiceAccess.REFProxy
Imports Project.Shared
Imports Project.Shared.Code

Public Class frmMasterMantenimiento
    Inherits frmMasterMaestro

#Region "Declaracion de Variables"

#End Region

#Region "Propiedades"

    Private _ValidarPermisosEnFormularioHijo As Boolean
    ''' <summary>
    ''' Indicador para permitir evento clic de botones en la barra de
herramientas y validarlos en formularios hijos
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Property ValidarPermisosEnFormularioHijo() As Boolean
```

```
        Get
            Return _ValidarPermisosEnFormularioHijo
        End Get
        Set(ByVal value As Boolean)
            _ValidarPermisosEnFormularioHijo = value
        End Set
    End Property

#Region "Barra de Estado"

    Public WriteOnly Property BarraEstado_CreadoPor() As String
        Set(ByVal value As String)
            sbBarraEstado.Panels("CreadoPor").Text = value
            sbBarraEstado.Panels("CreadoPor").Visible = Not
String.IsNullOrEmpty(value)
        End Set
    End Property

    Public WriteOnly Property BarraEstado_ActualizadoPor() As String
        Set(ByVal value As String)
            sbBarraEstado.Panels("ActualizadoPor").Text = value
            sbBarraEstado.Panels("ActualizadoPor").Visible = Not
String.IsNullOrEmpty(value)
        End Set
    End Property

    Public WriteOnly Property BarraEstado_Mensaje As String
        Set(value As String)
            sbBarraEstado.Panels("Mensaje").Text = value
            sbBarraEstado.Panels("Mensaje").Visible = Not
String.IsNullOrEmpty(value)
        End Set
    End Property

    Public ReadOnly Property BarraEstado_BarraProgreso() As
UltraWinStatusBar.UltraStatusPanel
        Get
            Return sbBarraEstado.Panels("BarraProgreso")
        End Get
    End Property

    ''' <summary>
    ''' Mensaje del número de registros
    ''' </summary>
    ''' <value></value>
    ''' <remarks></remarks>
    Public WriteOnly Property BarraEstado_NroRegistros As String
        Set(value As String)
            sbBarraEstado.Panels("NroRegistros").Text = value
            sbBarraEstado.Panels("NroRegistros").Visible = Not
String.IsNullOrEmpty(value)
        End Set
    End Property

#End Region


#Region "Barra de Herramientas"

    Public ReadOnly Property BotonBuscar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Buscar").SharedProps
        End Get
```

```
    End Property

    Public ReadOnly Property BotonEliminar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Eliminar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonGrabar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Grabar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonCargar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Cargar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonExportarExcel() As
UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("ExportarExcel").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonAyuda() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Ayuda").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonSalir() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Salir").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonNuevo() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Nuevo").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonProcesar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Procesar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonCopiar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Copiar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonModificar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Modificar").SharedProps
        End Get
    End Property
```

```
    Public ReadOnly Property BotonDeshacer() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Deshacer").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonPegar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Pegar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonAsignar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Asignar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonDesasignar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Desasignar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonActualizar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Actualizar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonAgregar() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("Agregar").SharedProps
        End Get
    End Property

    Public ReadOnly Property BotonAgregarHijo() As UltraWinToolbars.SharedProps
        Get
            Return tbmBarra.Tools("AgregarHijo").SharedProps
        End Get
    End Property

#End Region



#End Region

#Region "Constructor"

    Public Sub New()
        MyBase.New()
        InitializeComponent()
        Me.SesionService = New Sesion
    End Sub

    Public Sub New(ByVal container As IUnityContainer)
        MyBase.New(container)
        InitializeComponent()
    End Sub

#End Region
```

```
#Region "Declaracion de Procedimientos"

    ''' <summary>
    ''' limpiar la barra de estado
    ''' </summary>
    ''' <remarks></remarks>
    Public Overridable Sub LimpiarBarraEstado()

        BarraEstado_CreadoPor = String.Empty
        BarraEstado_ActualizadoPor = String.Empty
        BarraEstado_Mensaje = String.Empty
        BarraEstado_BarraProgreso.Visible = False
        BarraEstado_NroRegistros = String.Empty
        BarraEstado_Notificacion(TipoMensaje.NINGUNO, String.Empty)
    End Sub

    Public Overridable Sub LimpiarPantalla()

    End Sub

    Public Overridable Sub CargarCombos()

    End Sub

    Public Overridable Function ValidaGrabar() As Boolean

    End Function

    Public Overridable Sub Grabar()

    End Sub

    Public Overridable Function ValidaEliminar() As Boolean

    End Function

    Public Overridable Sub Eliminar()

    End Sub

    Public Overridable Function ValidaCargar() As Boolean

    End Function

    Public Overridable Sub Cargar()

    End Sub

    Public Overridable Function ValidaBuscar() As Boolean

    End Function

    Public Overridable Sub Buscar()

    End Sub

    Public Overridable Sub CargarDatos()

    End Sub

    Public Overridable Function ValidaNuevo() As Boolean

    End Function
```

```
Public Overridable Sub Nuevo()
End Sub
Public Overridable Function ValidaProcesar() As Boolean
End Function
Public Overridable Sub Procesar()
End Sub
Public Overridable Function ValidaCopiar() As Boolean
End Function
Public Overridable Sub Copiar()
End Sub
Public Overridable Function ValidaModificar() As Boolean
End Function
Public Overridable Sub Modificar()
End Sub
Public Overridable Function ValidaDeshacer() As Boolean
End Function
Public Overridable Sub Deshacer()
End Sub
Public Overridable Function ValidaPegar() As Boolean
End Function
Public Overridable Sub Pegar()
End Sub
Public Overridable Function ValidaActualizar() As Boolean
End Function
Public Overridable Sub Actualizar()
End Sub
Public Overridable Function ValidaAgregar() As Boolean
End Function
Public Overridable Sub Agregar()
End Sub
Public Overridable Function ValidaAgregarHijo() As Boolean
```

```
    End Function

    Public Overridable Sub AgregarHijo()

    End Sub

    Public Overridable Function ValidaAsignar() As Boolean

    End Function

    Public Overridable Sub Asignar()

    End Sub

    Public Overridable Function ValidaDesasignar() As Boolean

    End Function

    Public Overridable Sub Desasignar()

    End Sub

    ''' <summary>
    ''' procedimiento para mostrar la ayuda correspondiente a la pantalla
    ''' </summary>
    ''' <remarks></remarks>
    Public Overridable Sub Ayuda()
        If Not String.IsNullOrEmpty(ModuloSeleccionado.RutaAyuda) Then
            Process.Start(ModuloSeleccionado.RutaAyuda)
        End If
    End Sub

    ''' <summary>
    ''' procedimientos para salir de la pantalla
    ''' </summary>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Overrides Function _Salir() As Boolean
        Dim Contenedor As Object
        Dim TabContenedor As UltraWinTabControl.UltraTabPageControl = Nothing

        'obtiene objeto padre del formulario, si existe
        Contenedor= Me.Parent

        'si formulario esta dentro de tab
        If Contenedor IsNot Nothing AndAlso Contenedor.GetType =
GetType(UltraWinTabControl.UltraTabPageControl) Then
            TabContenedor = DirectCast(Contenedor,
UltraWinTabControl.UltraTabPageControl)
        End If

        'si detecta cambios pendientes de grabar
        If ActualizacionPendiente Then

            'solicita confirmar cambios o descartarlos
            If MessageBox.Show("Se estan realizando cambios, ¿Desea salir de
todas maneras?", "Salir", MessageBoxButtons.YesNo, MessageBoxicon.Question,
MessageBoxDefaultButton.Button2) = Windows.Forms.DialogResult.Yes Then

                'cerrar contenedor del formulario
                If TabContenedor IsNot Nothing Then
                    FormularioMDI.splitMDIDer.Collapsed = True
                    FormularioMDI.splitMDIDer.Visible = True
```

```
                TabContenedor.Tab.Text = String.Empty
                TabContenedor.Tab.Visible = False

                TabContenedor.Tab.Close()
                TabContenedor.Tab.Dispose()
            End If


            'cerrar formulario
            Me.Close()
            Me.Dispose()

            Return True

        Else
            Return False

        End If

    Else
        'no hay cambios pendientes

        'cerrar contenedor del formulario
        If TabContenedor IsNot Nothing Then
            FormularioMDI.splitMDIDer.Collapsed = True
            FormularioMDI.splitMDIDer.Visible = True

            TabContenedor.Tab.Text = String.Empty
            TabContenedor.Tab.Visible = False

            TabContenedor.Tab.Close()
            TabContenedor.Tab.Dispose()

        End If

        'cerrar formulario
        Me.Close()
        Me.Dispose()

    End If

    Return True

End Function

''' <summary>
''' procedimiento para mostrar los campos de auditoria de un registro
''' </summary>
''' <param name="sender"></param>
''' <param name="e"></param>
''' <remarks>
''' 2014-10-22  Herbert Mendoza                creacion
''' </remarks>
Public Overridable Sub AfterRowActivateGrilla(sender As Object, e As
System.EventArgs)

    Dim filaActiva As UltraWinGrid.UltraGridRow
    Dim grilla As UltraWinGrid.UltraGrid
    Dim UsuarioCrea As Object
    Dim FechaCrea As Object
    Dim UsuarioAct As Object
    Dim FechaAct As Object
```

```
        grilla= DirectCast(sender, UltraWinGrid.UltraGrid)
        filaActiva = grilla.ActiveRow

        UsuarioCrea = Nothing
        FeChaCrea = Nothing
        UsuarioAct = Nothing
        FechaAct = Nothing

        If filaActiva.Band.Columns.Exists("UsuarioCrea") Then
            UsuarioCrea = filaActiva.Cells("UsuarioCrea").Value
        End If
        If filaActiva.Band.Columns.Exists("FechaCrea") Then
            FechaCrea = filaActiva.Cells("FechaCrea").Value
        End If
        If filaActiva.Band.Columns.Exists("UsuarioAct") Then
            UsuarioAct = filaActiva.Cells("UsuarioAct").Value
        End If
        If filaActiva.Band.Columns.Exists("FechaAct") Then
            FechaAct = filaActiva.Cells("FechaAct").Value
        End If

        If filaActiva.IsDataRow Then
            CamposDeAuditoriaEnBarraDeEstado(UsuarioCrea, FechaCrea,
UsuarioAct, FechaAct)
        Else
            LimpiarBarraEstado()
        End If

    End Sub

    ''' <summary>
    ''' muestra los campos de auditoria de una registro seleccionado en la
barra de estado
    ''' </summary>
    ''' <param name="UsuarioCrea">usuario que crea el registro</param>
    ''' <param name="FeChaCrea">fecha en que se crea el registro</param>
    ''' <param name="UsuarioAct">ultimo usuario que actualiza el
registro</param>
    ''' <param name="FeChaAct">ultima fecha que se actualiza el
registro</param>
    ''' <remarks></remarks>
    Public Overridable Sub CamposDeAuditoriaEnBarraDeEstado(UsuarioCrea As
Object, FechaCrea As Object, UsuarioAct As Object, FechaAct As Object)
        Dim UsuarioCreaFinal As String
        Dim FechaCreaFinal As String
        Dim UsuarioActFinal As String
        Dim FechaActFinal As String
        UsuarioCreaFinal = If(UsuarioCrea IsNot Nothing AndAlso Not
IsDBNull(UsuarioCrea), UsuarioCrea.ToString, String.Empty)
        FechaCreaFinal = If(FechaCrea IsNot Nothing AndAlso Not
IsDBNull(FechaCrea) AndAlso Date.TryParse(FechaCrea, Nothing) AndAlso
Date.MinValue <> FechaCrea,
Date.Parse(FechaCrea).ToString(Constante.General.Pantalla.FormatoFecha.yyyyMMdd
HHmmss_03), String.Empty)
        UsuarioActFinal = If(UsuarioAct IsNot Nothing AndAlso Not
IsDBNull(UsuarioAct), UsuarioAct.ToString, String.Empty)
        FeChaActFinal = If(FechaAct IsNot Nothing AndAlso Not
IsDBNull(FechaAct) AndAlso Date.TryParse(FechaAct, Nothing) AndAlso
Date.MinValue <> FechaAct,
Date.Parse(FechaAct).ToString(Constante.General.Pantalla.FormatoFecha.yyyyMMddH
Hmmss_03), String.Empty)
```

```
        BarraEstado_CreadoPor = UsuarioCreaFinal &
If(String.IsNullOrEmpty(FechaCreaFinal), String.Empty, " - " & FechaCreaFinal)
        BarraEstado_ActualizadoPor = UsuarioActFinal &
If(String.IsNullOrEmpty(FechaActFinal), String.Empty, " - " & FechaActFinal)

    End Sub

    ''' <summary>
    ''' Muestra en barra de estado mensaje de acuerdo al tipo de mensaje
    ''' </summary>
    ''' <param name="tipo">Tipo de mensaje (error, información, OK,
alerta)</param>
    ''' <param name="Mensaje">Texto que aparecerá en la barra de estado</param>
    ''' <param name="TiempoVisualizacion">Tiempo en segundos que estará
disponible el mensaje por default 8 seg; 0 significa que estará visible hasta
limpiarlo manualmente</param>
    ''' <remarks></remarks>
    Public Overridable Sub BarraEstado_Notificacion(ByVal tipo As TipoMensaje,
Mensaje As String, Optional TiempoVisualizacion As Uinteger = 8)
        sbBarraEstado.Panels("Mensaje").Appearance.Image = Nothing
        sbBarraEstado.Panels("Mensaje").Appearance.BackColor = DefaultBackColor
        Select Case tipo
            Case TipoMensaje.TIPO_ERROR
                sbBarraEstado.Panels("Mensaje").Appearance.Image =
My.Resources.Error_icon
                sbBarraEstado.Panels("Mensaje").Appearance.BackColor =
Color.Pink
            Case TipoMensaje.TIPO_INFORMACION
                sbBarraEstado.Panels("Mensaje").Appearance.Image =
My.Resources.Information_icon
                sbBarraEstado.Panels("Mensaje").Appearance.BackColor =
Color.White
            Case TipoMensaje.TIPO_OK
                sbBarraEstado.Panels("Mensaje").Appearance.Image =
My.Resources.ok_icon
                sbBarraEstado.Panels("Mensaje").Appearance.BackColor =
Color.LightGreen
            Case (TipoMensaje.TIPO_WARNING)
                sbBarraEstado.Panels("Mensaje").Appearance.Image =
My.Resources.warning_icon
                sbBarraEstado.Panels("Mensaje").Appearance.BackColor =
Color.LightYellow
        End Select
        BarraEstado_Mensaje = Mensaje
        'Se activa timer si se muestra mensaje de notificación
        If tipo<> TipoMensaje.NINGUNO AndAlso Not
String.IsNullOrWhiteSpace(Mensaje) AndAlso TiempoVisualizacion > 0 Then
            timerMensaje.Interval = TiempoVisualizacion * 1000
            timerMensaje.Enabled = True
            timerMensaje.Stop()
            timerMensaje.Start()
        End If
    End Sub
    ''' <summary>
    ''' Setea la cantidad de registros consultados en la barra de estado
    ''' </summary>
    ''' <param name="NroRegistros"></param>
    ''' <remarks></remarks>
    Public Overridable Sub BarraEstado_Registros(ByVal NroRegistros As Integer)
        If NroRegistros > 0 Then
            BarraEstado_NroRegistros = NroRegistros.FormatoCadena +
IIf(NroRegistros = 1, "registro.", "registros.")
        End If
```

```
    End Sub
#End Region

#Region "Declaracion de Eventos"

    Private Sub tbmBarra_ToolClick(sender As System.Object, e As
Infragistics.Win.UltraWinToolbars.ToolClickEventArgs) Handles
tbmBarra.ToolClick

        Try
            Cursor= Cursors.WaitCursor

            Select Case e.Tool.Key

                Case "Buscar"
                    If ValidaBuscar() Then
                        Buscar()
                    End If

                Case "Cargar"
                    If ValidaCargar() Then
                        Cargar()
                    End If
                Case "Eliminar"
                    If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                        If ValidaEliminar() Then
                            Eliminar()
                        End If
                    Else
                        MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxicon.Information)
                    End If

                Case "Grabar"
                    If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                        If ValidaGrabar() Then
                            Grabar()
                        End If
                    Else
                        MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                    End If

                Case "Nuevo"
                    If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                        If ValidaNuevo() Then
                            Nuevo()
                        End If
                    Else
                        MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                    End If

                Case "Procesar"
                    If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                        If ValidaProcesar() Then
                            Procesar()
                        End If
                    Else
```

```
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "Copiar"
                If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                    If ValidaCopiar() Then
                        Copiar()
                    End If
                Else
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "Modificar"
                If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                    If ValidaModificar() Then
                        Modificar()
                    End If
                Else
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "Deshacer"
                If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                    If ValidaDeshacer() Then
                        Deshacer()
                    End If
                Else
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "Pegar"
                If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                    If ValidaPegar() Then
                        Pegar()
                    End If
                Else
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "Agregar"
                If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                    If ValidaAgregar() Then
                        Agregar()
                    End If
                Else
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "AgregarHijo"
                If ValidarPermisosEnFormularioHijo OrElse TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura Then
                    If ValidaAgregarHijo() Then
```

```
                        AgregarHijo()
                    End If
                Else
                    MessageBox.Show("Usted no tiene permiso para ejecutar
esta opcion.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
                End If

            Case "ACtualizar"
                If ValidaActualizar() Then
                    Actualizar()
                End If
            Case "ExportarExcel"
                ExportarExcel()
            Case "Ayuda"
                Ayuda()
            Case "Salir"
                _Salir()
            Case "Asignar"
                If ValidaAsignar() Then
                    Asignar()
                End If
            Case "Desasignar"
                If ValidaDesasignar() Then
                    Desasignar()
                End If
        End Select

    Catch ex As ExCeption
        ExCeptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
    Finally
        Cursor= Cursors.Default
    End Try

End Sub

Private Sub frmMasterOperativo_Load(sender As Object, e As
System.EventArgs) Handles Me.Load
    Try
        'ocultar si no tiene permiso de escritura
        For Each tool In tbmBarra.Tools
            Select Case tool.Key
                Case "BusCar", "Actualizar", "ExportarExcel", "Ayuda",
"Salir"
                    tool.SharedProps.Visible = True
                Case Else
                    tool.SharedProps.Visible = (TipoAcceso =
Framework.WindowsForms.Master.Acceso.eTipoAcceso.Escritura)
            End Select
        Next

    Catch ex As Exception
        ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
    End Try
End Sub

#End Region

End Class
```

Figura 4.14: Implementación de formulario operativo frmMasterMantenimiento.vb.

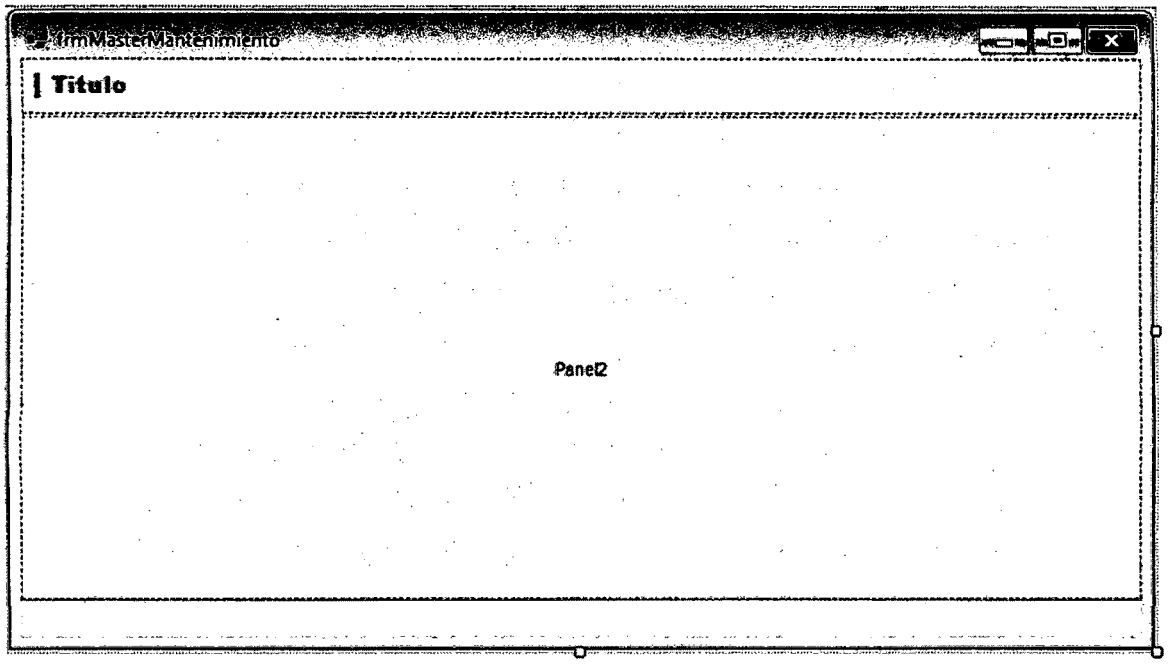

Figura 4.15: Formulario operativo frmMasterMantenimiento.

frmMasterMantenimientoGrilla.vb

```
Imports Infragistics.Win
Imports Microsoft.Practices.Unity
Imports Framework.Services.Exception
Imports Framework.Services.Session
Imports Microsoft.Practices.EnterpriseLibrary.ExceptionHandling
Imports Project.Shared
Imports Project.Shared.Code

Public Class frmMasterMantenimientoGriIla
    Inherits frmMasterMantenimiento

#Region "Declaracion de Variables"

#End Region

#Region "Propiedades"

#End Region

#Region "Constructor"

    Public Sub New()
        MyBase.New()
        InitializeComponent()
        Me.SesionService = New Sesion
    End Sub

    Public Sub New(ByVal container As IUnityContainer)
        MyBase.New(container)
        InitializeComponent()
    End Sub

#End Region

#Region "Procedimientos"

    Public Overrides Function ValidaBuscar() As Boolean
```

```
        Return True
    End Function

    Public Overrides Sub Buscar()
        Cursor= Cursors.WaitCursor

        LimpiarPantalla()

        CargarDatos()

        ActualizacionPendiente = False

        Cursor= Cursors.Default
    End Sub

    Public Overrides Sub LimpiarPantalla()
        grdDetalle.DataSource = Nothing

        BotonBuscar.Enabled = True
        BotonEliminar.Enabled = False
        BotonGrabar.Enabled = False
        BotonExportarExcel.Enabled = False

        LimpiarBarraEstado()

        ActualizacionPendiente = False

    End Sub

    Public Overrides Function ValidaEliminar() As Boolean
        Dim resultado As Boolean = True
        'valida seleccion nde fila
        If grdDetalle.Selected.Rows.Count < 1 Then
            resultado= False
            MessageBox.Show("Debe seleccionar al menos un registro para
eliminar.", "Informacion", MessageBoxButtons.OK, MessageBoxIcon.Information)
        End If

        Return resultado
    End Function

    Public Overrides Sub Eliminar()
        For Each fila In grdDetalle.Selected.Rows
            If fila.IsDataRow Then
                'marcar como eliminado

fila.Cells(Constante.General.Pantalla.Grilla.ColumnaFlagMantenimiento).Value =
Constante.General.Pantalla.eMantenimiento.Eliminacion
                'ocultar
                fila.Hidden = True
                'actualiza flag
                ActualizacionPendiente = True
                'quitar seleccion
                fila.Selected = False

                BotonEliminar.Enabled = False
                BotonGrabar.Enabled = True

            End If
        Next
    End Sub

#End Region
```

```
#Region "Eventos"

    Private Sub grdDetalle_AfterCellUpdate(sender As Object, e As
Infragistics.Win.UltraWinGrid.CellEventArgs) Handles grdDetalle.AfterCellUpdate
        Try
            If e.Cell.Row.IsDataRow AndAlso Not FormularioMDI Is Nothing
AndAlso Not FormularioMDI.fljAprobacion.EstaAprobado Then
                BotonGrabar.Enabled = True
                ActualizacionPendiente = True
                If
e.Cell.Row.Cells(Constante.General.Pantalla.Grilla.ColumnaFlagMantenimiento).Va
lue = Constante.General.Pantalla.eMantenimiento.Vista Then

e.Cell.Row.Cells(Constante.General.Pantalla.Grilla.ColumnaFlagMantenimiento).Va
lue = Constante.General.Pantalla.eMantenimiento.Actualizacion
                End If
            End If
        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub

    Private Sub grdDetalle_AfterRowActivate(sender As Object, e As
System.EventArgs) Handles grdDetalle.AfterRowActivate
        Try
            AfterRowActivateGrilla(sender, e)
        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub

    Private Sub grdDetalle_AfterRowinsert(sender As Object, e As
Infragistics.Win.UltraWinGrid.RowEventArgs) Handles grdDetalle.AfterRowInsert
        Try
            If Not FormularioMDI Is Nothing AndAlso Not
FormularioMDI.fljAprobacion.EstaAprobado Then
                'actuliza columna flag

e.Row.Cells(Constante.General.Pantalla.Grilla.ColumnaFlagMantenimiento).Value =
Constante.General.Pantalla.eMantenimiento.Nuevo
                'actualiza flag
                ActualizacionPendiente = True
                BotonGrabar.Enabled = True
            End If
        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub


    Private Sub grdDetalle_BeforeSelectChange(sender As Object, e As
Infragistics.Win.UltraWinGrid.BeforeSelectChangeEventArgs) Handles
grdDetalle.BeforeSelectChange
        Try
            If Not FormularioMDI Is Nothing AndAlso Not
FormularioMDI.fljAprobacion.EstaAprobado Then
                BotonEliminar.Enabled = (e.NewSelections.Rows.Count > 0)
            End If
        Catch ex As Exception
```

```
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub

    Private Sub grdDetalle_CellChange(sender As Object, e As
Infragistics.Win.UltraWinGrid.CellEventArgs) Handles grdDetalle.CellChange
        Try
            If grdDetalle.ActiveCell IsNot Nothing AndAlso
grdDetalle.ActiveRow.IsDataRow AndAlso Not FormularioMDI Is Nothing AndAlso Not
FormularioMDI.fljAprobacion.EstaAprobado Then
                If grdDetalle.ActiveCell.Column.CellActivation =
UltraWinGrid.Activation.AllowEdit AndAlso grdDetalle.ActiveCell.Row.Activation
= UltraWinGrid.Activation.AllowEdit AndAlso grdDetalle.ActiveCell.Activation =
UltraWinGrid.Activation.AllowEdit Then
                    BotonGrabar.Enabled = True
                    ActualizacionPendiente = True
                End If
            End If
        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub

    Private Sub grdDetalle_InitializeLayout(sender As Object, e As
Infragistics.Win.UltraWinGrid.InitializeLayoutEventArgs) Handles
grdDetalle.InitializeLayout
        Try
            e.Layout.Override.AllowDelete = DefaultableBoolean.False
            e.Layout.Override.AllowAddNew =
UltraWinGrid.AllowAddNew.TemplateOnBottom
            e.Layout.Override.AllowUpdate = DefaultableBoolean.True
            e.Layout.Override.AllowGroupBy = DefaultableBoolean.False
            e.Layout.Override.GroupByRowinitialExpansionState =
UltraWinGrid.GroupByRowInitialExpansionState.Collapsed
            e.Layout.Override.GroupByRowExpansionStyle =
UltraWinGrid.GroupByRowExpansionStyle.Disabled
            e.Layout.Override.ExpansionIndicator =
UltraWinGrid.ShowExpansionIndicator.Never
            e.Layout.Override.AllowRowFiltering = DefaultableBoolean.True
            e.Layout.Override.FilterUIType =
UltraWinGrid.FilterUIType.FilterRow
            e.Layout.Override.HeaderClickAction =
UltraWinGrid.HeaderClickAction.SortMulti
            e.Layout.MaxBandDepth = 1
            e.Layout.ViewStyle = UltraWinGrid.ViewStyle.SingleBand
            e.Layout.Override.AllowMultiCellOperations =
UltraWinGrid.AllowMultiCellOperation.Copy +
UltraWinGrid.AllowMultiCellOperation.Paste
            e.Layout.Override.WrapHeaderText = DefaultableBoolean.True

        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub

    Private Sub grdDetalle_KeyPress(sender As Object, e As
System.Windows.Forms.KeyPressEventArgs) Handles grdDetalle.KeyPress
        Try
            If grdDetalle.ActiveCell IsNot Nothing AndAlso
grdDetalle.ActiveRow.IsDataRow AndAlso Not FormularioMDI Is Nothing AndAlso Not
```

```
FormularioMDI.fljAprobacion.EstaAprobado Then
                If grdDetalle.ActiveCell.Column.CellActivation =
UltraWinGrid.Activation.AllowEdit AndAlso grdDetalle.ActiveCell.Row.Activation
= UltraWinGrid.Activation.AllowEdit AndAlso grdDetalle.ActiveCell.Activation =
UltraWinGrid.Activation.AllowEdit Then
                    BotonGrabar.Enabled = True
                    ActualizacionPendiente = True
                End If
            End If
        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        End Try
    End Sub

    Private Sub frmMasterMantenimientoGrilla_Load(sender As Object, e As
System.EventArgs) Handles Me.Load
        Try
            CargarCombos()

            LimpiarPantalla()

        Catch ex As Exception
            ExceptionPolicy.HandleException(Procedimiento.PreparaExcepcion(ex,
Me.Name), ConstanteExcepcion.Politicas.PresentationPolicy)
        Finally
        End Try
    End Sub

#End Region

End Class
```

Figura 4.16: Implementación de formulario operativo rmMasterMantenimientoGrilla.vb.

4.3.2. MENÚS PARA ACCESO A MÓDULOS

```
Imports System.ComponentModel
Imports System.Windows.Forms
Imports System.Collections.Specialized

Public Class ucMenuToolBars

#Region "Property"

#Region "Visible/Enabled/Texto de Botones del Formulario"

    <Category("Framework")> _
    Public Property _MenuControl_BackColor() As System.Drawing.Color
        Get
            Return tsMenuTop.BackColor
        End Get
        Set(ByVal value As System.Drawing.Color)
            tsMenuTop.BackColor = value
        End Set
    End Property

    <Category("Framework")> _
    Public Property _MenuControl_Visible() As Boolean
        Get
            Return tsMenuTop.Visible
        End Get
        Set(ByVal value As Boolean)
```

```
            tsMenuTop.Visible = value
        End Set
    End Property

    <Category("Framework")> _
    Public Property _ControlHabilitado() As Boolean
        Get
            Return tsMenuTop.Enabled
        End Get
        Set(ByVal value As Boolean)
            tsMenuTop.Enabled = value
        End Set
    End Property

    'Boton Buscar
    <Category("Framework")> _
    Public ReadOnly Property _BotonBuscar() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbBuscar
        End Get
    End Property

    'Boton Detalle
    <Category("Framework")> _
    Public ReadOnly Property _BotonDetalle() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbDetalle
        End Get
    End Property

    'Boton Nuevo
    <Category("Framework")> _
    Public ReadOnly Property _BotonNuevo() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbNuevo
        End Get
    End Property

    'Boton Modificar
    <Category("Framework")> _
    Public ReadOnly Property _BotonModificar() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbModificar
        End Get
    End Property

    'Boton Eliminar
    <Category("Framework")> _
    Public ReadOnly Property _BotonEliminar() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbEliminar
        End Get
    End Property


    'Boton Grabar
    <Category("Framework")> _
    Public ReadOnly Property _BotonGrabar() As
```

```
System.Windows.Forms.ToolStripButton
        Get
            Return tsbGrabar
        End Get
    End Property

    'Boton CargarDatos
    <Category("Framework")> _
    Public ReadOnly Property _BotonCargarDatos() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbCargarDatos
        End Get
    End Property

    'Boton Exportar
    <Category("Framework")> _
    Public ReadOnly Property _BotonExportar() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbExportar
        End Get
    End Property

    'Boton Ayuda
    <Category("Framework")> _
    Public ReadOnly Property _BotonAyuda() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbAyuda
        End Get
    End Property

    'Boton Salir
    <Category("Framework")> _
    Public ReadOnly Property _BotonSalir() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbSalir
        End Get
    End Property

    'Boton Deshacer
    <Category("Framework")> _
    Public ReadOnly Property _BotonDeshacer() As
System.Windows.Forms.ToolStripButton
        Get
            Return tsbDeshacer
        End Get
    End Property

#End Region

#End Region

#Region "Manejadores de Eventos"

    ' Define un event handler delegate
    Public Delegate Sub EventHandler(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.ToolStripitemClickedEventArgs)
    'Define un event handler delegate para los nuevos botones en Runtime
    Public Delegate Sub EventHandlerMenu(ByVal sender As System.Object, ByVal e
As EventArgs)
```

```
    'Declara un Evento tipo Delegado
    Public Event EventHandlerDelegate As EventHandler
    'Declara un Evento tipo Delegado  para los nuevos botones en Runtime
    Public Event EventHandlerDelegateMenu As EventHandlerMenu

    ''' <class>onEventHandlerDelegate</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Redifine el Evento.
    ''' </summary>
    ''' <remarks></remarks>

    Public Overridable Sub onEventHandlerDelegate(ByVal e As
System.Windows.Forms.ToolStripItemClickedEventArgs)
        RaiseEvent EventHandlerDelegate(Me, e)
    End Sub

#End Region

#Region "Costructor"

    Sub New()
        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call.
    End Sub

#End Region

#Region "Procedimientos"

    ''' <class>ValidarBuscar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarBuscar que es llamado del FrameWork.WindowsForms.Forms
para redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarBuscar() As Boolean

    End Function

    ''' <class>Buscar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <Summary>
    '''  Metodo Buscar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </Summary>
    ''' <remarks></remarks>
    Public Sub _Buscar()

    End Sub

    ''' <class>Detalle</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Detalle que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
```

```
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarDetalle() As Boolean

    End Function

    ''' <class>Detalle</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Detalle que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Detalle()

    End Sub

    ''' <class>ValidarNuevo</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarNuevo que es llamado del FrameWork.WindowsForms.Forms
para redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarNuevo() As Boolean

    End Function

    ''' <class>Nuevo</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Nuevo que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Nuevo()
        tsbBuscar.Visible = False
        tsbDetalle.Visible = False
        tsbNuevo.Visible = False
        tsbModificar.Visible = False
        tsbEliminar.Visible = False
        tsbGrabar.Visible = True
        tsbDeshacer.Visible = True
        tsbCargarDatos.Visible = False
        tsbExportar.Visible = True
        tsbAyuda.Visible = True
        tsbSalir.Visible = True
    End Sub

    ''' <class>IniciarMenuFormulario</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo IniciarMenuFormulario que es llamado del
FrameWork.WindowsForms.Forms para redireccionarlo al Cliente para Sobreescribir
el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _IniciarMenuFormulario()
        tsbBuscar.Visible = True
```

```
        tsbDetalle.Visible = False
        tsbNuevo.Visible = True
        tsbModificar.Visible = False
        tsbEliminar.Visible = False
        tsbGrabar.Visible = False
        tsbDeshacer.Visible = True
        tsbCargarDatos.Visible = False
        tsbExportar.Visible = True
        tsbAyuda.Visible = True
        tsbSalir.Visible = True
    End Sub

    ''' <class>ValidarModificar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarModificar que es llamado del
FrameWork.WindowsForms.Forms para redireccionarlo al Cliente para Sobreescribir
el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarModificar() As Boolean

    End Function

    ''' <class>Modificar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Modificar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Modificar()
        tsbBuscar.Visible = False
        tsbDetalle.Visible = False
        tsbNuevo.Visible = False
        tsbModificar.Visible = False
        tsbEliminar.Visible = False
        tsbGrabar.Visible = True
        tsbDeshacer.Visible = True
        tsbCargarDatos.Visible = False
        tsbExportar.Visible = True
        tsbAyuda.Visible = True
        tsbSalir.Visible = True
    End Sub

    ''' <class>Modificar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Modificar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _VistaPreviaModificar()
        tsbBuscar.Visible = False
        tsbDetalle.Visible = False
        tsbNuevo.Visible = True
        tsbModificar.Visible = True
        tsbEliminar.Visible = True
        tsbGrabar.Visible = False
        tsbDeshacer.Visible = True
```

```
        tsbCargarDatos.Visible = False
        tsbExportar.Visible = True
        tsbAyuda.Visible = True
        tsbSalir.Visible = True
    End Sub

    ''' <class>ValidarEliminar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarEliminar que es llamado del FrameWork.WindowsForms.Forms
para redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarEliminar() As Boolean

    End Function

    ''' <class>Eliminar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Eliminar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Eliminar()
        tsbBuscar.Visible = True
        tsbDetalle.Visible = False
        tsbNuevo.Visible = True
        tsbModificar.Visible = False
        tsbEliminar.Visible = False
        tsbGrabar.Visible = False
        tsbDeshacer.Visible = True
        tsbCargarDatos.Visible = False
        tsbExportar.Visible = True
        tsbAyuda.Visible = True
        tsbSalir.Visible = True
    End Sub

    ''' <class>Eliminar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Eliminar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Deshacer()
        _IniciarMenuFormulario()
    End Sub

    ''' <class>ValidarGrabar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarGrabar que es llamado del FrameWork.WindowsForms.Forms
para redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarGrabar() As Boolean

    End Function
```

```
    ''' <class>Grabar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Grabar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Grabar()
        tsbBuscar.Visible = True
        tsbDetalle.Visible = False
        tsbNuevo.Visible = True
        tsbModificar.Visible = False
        tsbEliminar.Visible = False
        tsbGrabar.Visible = False
        tsbDeshacer.Visible = True
        tsbCargarDatos.Visible = False
        tsbExportar.Visible = True
        tsbAyuda.Visible = True
        tsbSalir.Visible = True
    End Sub

    ''' <class>ValidarCargarDatos</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarCargarDatos que es llamado del
FrameWork.WindowsForms.Forms para redireccionarlo al Cliente para Sobreescribir
el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarCargarDatos() As Boolean

    End Function

    ''' <class>CargarDatos</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Cargaroatos que es llamado del FrameWork.WindowsForms.Forms
para redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _CargarDatos()

    End Sub

    ''' <class>ValidarExportar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo ValidarExportar que es llamado del FrameWork.WindowsForms.Forms
para redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _ValidarExportar() As Boolean

    End Function

    ''' <class>Exportar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
```

```
    ''' <summary>
    '''  Metodo Exportar que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Exportar()

    End Sub

    ''' <class>Ayuda</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Ayuda que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub _Ayuda()

    End Sub

    ''' <class>_PreguntarEliminarRegistrar</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo PreguntarSalirDelSistema que es llamado del
FrameWork.WindowsForms.Forms para redireccionarlo al Cliente para Sobreescribir
el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _PreguntarEliminarRegistrar() As Boolean
        If System.Windows.Forms.MessageBox.Show("¿Desea eliminar el registro?",
"Eliminar Registro", MessageBoxButtons.YesNo, MessageBoxicon.Question) =
DialogResult.Yes Then
            Return True
        Else
            Return False
        End If
    End Function

    ''' <class>PreguntarDeshacerFuncionalidad</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-26</date>
    ''' <summary>
    '''  Metodo PreguntarDeshacerFuncionalidad que es llamado del
FrameWork.WindowsForms.Forms para redireccionarlo al Cliente para Sobreescribir
el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _PreguntarDeshacerFuncionalidad() As Boolean
        If System.Windows.Forms.MessageBox.Show("¿Desea deshacer los cambios
realizados?", "Deshacer Funcionalidad", MessageBoxButtons.YesNo,
MessageBoxicon.Question) = DialogResult.Yes Then
            Return True
        Else
            Return False
        End If
    End Function

    ''' <class>Salir</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
```

```
    ''' Metodo Salir que es llamado del FrameWork.WindowsForms.Forms para
redireccionarlo al Cliente para Sobreescribir el Código.
    ''' </summary>
    ''' <remarks></remarks>
    Public Function _Salir() As Boolean
        If System.Windows.Forms.MessageBox.Show("¿Desea salir de la Opción?",
"Salir de la Opción", MessageBoxButtons.YesNo, MessageBoxicon.Question) =
DialogResult.Yes Then
            Return True
        Else
            Return False
        End If
    End Function

#End Region

#Region "Funciones"

#End Region

#Region "Eventos de Controles"

#Region "tsMenuTop"

    Private Sub tsMenuTop_ItemClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.ToolStripitemClickedEventArgs) Handles
tsMenuTop.ItemClicked
        RaiseEvent EventHandlerDelegate(sender, e)
    End Sub

    Private Sub tsMenuTop_Clicked(sender AS System.Object, e As EventArgs)
Handles tsMenuTop.Click
        RaiseEvent EventHandlerDelegateMenu(sender, e)
    End Sub

#End Region

#End Region

End Class
```

Figura 4.17: Menus de acceso a módulos

4.3.3. MÉTODOS PARA ACCEDER A REPORTES

```
Imports System.Windows.Forms
Imports System.Collections.Specialized
Imports Microsoft.Practices.Unity

Public Class frmReporte
    Inherits Framework.WindowsForms.Master.frmMaestro

#Region "Constructor"

    Sub New()
        ' This call is required by the designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call.
    End Sub

    Sub New(ByVal container AS IUnityContainer)

        MyBase.New(container)
```

```
        InitializeComponent()

    End Sub

#End Region

#Region "Declaracion de Eventos"

    Private Sub frmReporte_Load(sender As Object, e As System.EventArgs)
Handles Me.Load
        ucMenuFormulario._BotonBuscar.Visible = False
        ucMenuFormulario._BotonDetalle.Visible = False
        ucMenuFormulario._BotonNuevo.Visible = False
        ucMenuFormulario._BotonModificar.Visible = False
        ucMenuFormulario._BotonEliminar.Visible = False
        ucMenuFormulario._BotonGrabar.Visible = False
        ucMenuFormulario._BotonDeshacer.Visible = False
        ucMenuFormulario._BotonExportar.Visible = False
    End Sub

#End Region

#Region "Declaracion de Procedimientos"

    ''' <summary>
    '''
    ''' </summary>
    ''' <param name="Ruta"></param>
    ''' <param name="Reporte"></param>
    ''' <param name="param"></param>
    ''' <remarks></remarks>
    Overridable Sub _EjecutarReporte(Optional ByVal Ruta As String = "", _
                                     Optional ByVal Reporte As String = "", _
                                     Optional ByVal param As Hashtable =
Nothing)
        ReporteSeleccionConceptos(Ruta, Reporte, param)
    End Sub

    ''' <summary>
    '''
    ''' </summary>
    ''' <param name="Ruta"></param>
    ''' <param name="Reporte"></param>
    ''' <param name="param"></param>
    ''' <remarks></remarks>
    Private Sub ReporteSeleccionConceptos(ByVal Ruta As String, _
                                          ByVal Reporte As String, _
                                          ByVal param As Hashtable)
        Try
            Me.Cursor= Cursors.WaitCursor
            rvReporte.ProcessingMode =
Microsoft.Reporting.WinForms.ProcessingMode.Remote

            rvReporte.ServerReport.ReportServerUrl = New Uri(Ruta)
            rvReporte.ServerReport.ReportPath = Reporte

            If Not param Is Nothing AndAlso param.Count > 0 Then
                If SetParametrosSeleccionConceptos(param) = False Then
                    Throw New Exception("Ocurrió un error al Asignar los
parametros al Reporte. ")
                End If
            End If
```

```
            rvReporte.RefreshReport()
        Catch ex As Exception
            Throw
        Finally
            Me.Cursor= Cursors.Default
        End Try
    End Sub


    ''' <class>CargarDatos</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo CargarDatos que será Reemplazable por el Codigo del Cliente
    ''' </summary>
    ''' <remarks></remarks>
    Overridable Sub _CargarDatos()
        ucMenuFormulario._CargarDatos()
    End Sub

    ''' <class>Ayuda</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Ayuda que será Reemplazable por el Codigo del Cliente
    ''' </summary>
    ''' <remarks></remarks>
    ''' <changehistory>
    ''' Date           Name            Description
    ''' ----------------------------------------
    ''' (MM/DD/YYYY)  (name)          (Description)
    ''' </changehistory>
    Overridable Sub _Ayuda()
        ucMenuFormulario._Ayuda()
    End Sub

    ''' <class>_Salir</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Metodo Salir que será Reemplazable por el Codigo del Cliente
    ''' </summary>
    ''' <remarks></remarks>
    Overrides Function _Salir() As Boolean
        If ucMenuFormulario._Salir Then
            Return True
        Else
            Return False
        End If
    End Function
    ''' <class>_CrearBotonMenu</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-22</date>
    ''' <summary>
    '''  Metodo _CrearBotonMenu que será Reemplazable por el Codigo del Cliente
    ''' </summary>
    ''' <remarks></remarks>
    Overridable Sub _CrearBotonMenu(ByVal aParam As SortedList)
        If Not aParam Is Nothing And aParam.Count > 0 Then
            Me.ucMenuFormulario.SuspendLayout()
            Me.SuspendLayout()
            Dim tb(aParam.Count - 1) As ToolStripButton
```

```
            For i = 0 To aParam.Count - 1
                tb(i) = New ToolStripButton
                tb(i).AutoSize = True
                tb(i).DisplayStyle = ToolStripitemDisplayStyle.ImageAndText
                tb(i).Name = "BotonMenuPersonalizado" & i
                If String.IsNullOrEmpty(aParam.GetKey(i).ToString.Trim) = False
Then
                    tb(i).Text = aParam.GetKey(i).ToString
                Else
                    tb(i).Text = "Boton " & i
                End If
                If String.IsNullOrEmpty(aParam.GetByIndex(i).ToString.Trim) =
False Then
                    tb(i).Image = CType(aParam.GetByIndex(i),
System.Drawing.Image)
                Else
                    tb(i).Image = My.Resources.BotonPersonalizado.ToBitmap
                End If
                AddHandler tb(i).Click, AddressOf
ucMenuFormularioAdicionarBotones_EventHandlerDelegate
            Next

            Me.ucMenuFormulario.tsMenuTop.Items.AddRange(tb)
            Me.ucMenuFormulario.tsMenuTop.ResumeLayout(False)
            Me.ucMenuFormulario.tsMenuTop.PerformLayout()
            Me.ResumeLayout(False)
            Me.PerformLayout()
        End If

    End Sub

    ''' <class>_EventoBotonMenuNuevo</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-0722</date>
    ''' <summary>
    '''  Metodo _EventoBotonMenuNuevo que será Reemplazable por el Codigo del
Cliente
    ''' </summary>
    ''' <remarks></remarks>
    Overridable Sub _EventoBotonMenuNuevo(ByVal index As Integer)
    End Sub
#End Region

#Region "Declaracion de Funciones"

    Private Function SetParametrosSeleccionConceptos(ByVal param As Hashtable)
As Boolean
        Dim htParam As Hashtable =
CollectionsUtil.CreateCaseInsensitiveHashtable()
        Dim parameters As Microsoft.Reporting.WinForms.ReportParameter() = New
Microsoft.Reporting.WinForms.ReportParameter(param.Count - 1) {}
        Dim contador As Byte= 0

        htParam = param.Clone

        If htParam Is Nothing Then
            Return False
        Else
            For Each entry As DictionaryEntry In htParam
                If String.IsNullOrEmpty(entry.Key.ToString.Trim) And
String.IsNullOrEmpty(entry.Value.ToString.Trim) Then
                    entry.Key = String.Empty
                    entry.Value = String.Empty
```

```
                End If

                parameters(contador) = New
Microsoft.Reporting.WinForms.ReportParameter(entry.Key.ToString,
entry.Value.ToString)
                contador= contador+ 1
            Next
        End If

        rvReporte.ServerReport.SetParameters(parameters)
        Return True
    End Function

#End Region

#Region "Personalizados"

    ''' <class>ucMenuFormularioAdicionarBotones_EventHandlerDelegate </class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-22</date>
    ''' <summary>
    '''  Metodo ucMenuFormularioAdicionarBotones_EventHandlerDelegate que será
responsable de simular el Evento del Control
    ''' </summary>
    ''' <remarks></remarks>
    Overridable Sub ucMenuFormularioAdicionarBotones_EventHandlerDelegate(ByVal
sender As System.Object, ByVal e As EventArgs) Handles
ucMenuFormulario.EventHandlerDelegateMenu

    End Sub

    ''' <class>ucMenuFormulario_EventHandlerDelegate</class>
    ''' <author>Herbert Mendoza</author>
    ''' <date>2014-10-28 1:37:22 PM</date>
    ''' <summary>
    '''  Delegado ucMenuFormulario_EventHandlerDelegate que realiza la
Simulación de llamada al Cliente
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub ucMenuFormulario_EventHandlerDelegate(ByVal sender As
System.Object, ByVal e As System.Windows.Forms.ToolStripitemClickedEventArgs)
Handles ucMenuFormulario.EventHandlerDelegate
        Dim ctrl As Control= TryCast(sender, Control)

        Select Case DirectCast(ctrl,
System.Windows.Forms.ToolStrip).Items.IndexOf(e.ClickedItem)
            Case 8
                _CargarDatos()
            Case 9
                _Ayuda()
            Case 10
                _Salir()
            Case Else
                _EventoBotonMenuNuevo(DirectCast(ctrl,
System.Windows.Forms.ToolStrip).Items.IndexOf(e.Clickeditem))
        End Select
    End Sub

#End Region
End Class
```

Figura 4.18: Menus de acceso a reportes.

CAPITULO V

CONCLUSIONES

5.1 CONCLUSIONES

a. Según el marco teórico en el capítulo II, sección 2.1.1, las técnicas para realizar la descomposición lógica desarrollada en el capítulo III, sección 3.4.4, tabla N° 3.2 y los artefactos obtenidos en el capítulo IV, definir la capa de acceso a datos en la figura N° 4.1; la capa de métodos del servicio en la figura N° 4.4. Se concluye que el framework brinda métodos de acceso a datos.

b. Según el marco teórico en el capítulo II, sección 2.1.2, las técnicas para realizar la herencia de formularios desarrollada en el capítulo III, sección 3.4.4, tabla N° 3.4 y los artefactos obtenidos en el capítulo IV, implementar los formularios operativos en la figura N° 4.10, figura N° 4.12 y en la figura N° 4.14; implementar los menús de acceso en la figura N° 4.17. Se concluye que el framework entrega formularios preparados para tareas específicas especializadas.

c. Según el marco teórico en el capítulo II, sección 2.1.3, las técnicas para aplicar la arquitectura orientada a servicios desarrollada en el capítulo III, sección 3.4.4, tabla N° 3.3 y los artefactos obtenidos en el capítulo IV, diseñar las interfaces de métodos en la figura N° 4.2; implementar el contrato en la figura N° 4.3. Se concluye que las capas del servicio exponen métodos del framework y serán consumidos por los usuarios.

BIBLIOGRAFIA

1. Augustyniak, M. (2002). *NET XML Web Services in 24 Hours. Indiana, United States of America*: Sams Publishing.
2. Batini, C., Ceri, S. y Navathe, S. B. (1994*). Diseño conceptual de base de datos. Un enfoque de entidades-interrelaciones. Boston, Estados Unidos: Addison-Wesley Iboamericana.*
3. Bieberstein, N., Laird, R., Keith, J. y Mitra, T. (2008). *Executing SOA: A Practical Guide for the Service-Oriented Architect. United States of America: IBM Press.*
4. Cheng, S. (2010). *Microsoft Windows Communication Foundation 4.0 Cookbook for Developing SOA Applications. Birmingham, Reino Unido: Packt Publishing.*
5. D'Andrea, E. (2008). Visual Basic 2008. *Curso de Iniciación. Barcelona, España: Inforbook's.*
6. Daim, T., Neshati, R., Russell, W., Eastham, J. (2014). *Technology Development. Beaverton, United States of America: Springer.*
7. Dikmans, L., Luttikhuizen, R. (2012). *SOA Made Simple. Birmingham, Reino Unido: Packt Publishing Ltd.*
8. Erl, T. (2009). *SOA Design Patterns. Boston, United States of America: Prentice Hall.*
9. Faison, T. (2006). *Event-Based Programming: Taking Events to the Limit. New York, United States of America: Apress.*
10. Fischer, T., Slate, J., Stromquist, P. y Wu, C. (2002). *Professional Design Patterns in VBNET: Building Adaptable Applications. New York, United States of America: Apress.*
11. Gao, T. (2007). *The complete Reference to Professional SOA with Visual Studio 2005 (C# & VB 2005) .NET 3.0. United States of America: Lulu Press.*
12. Hamilton, P. (2002). *Object-Oriented Programming with Visual Basic .NET. United States of America: O'Reilly & Associates.*
13. Hansler, G. y Hansen, J. (1997). *Diseño y administración de base de datos (2ª Ed.). Madrid, España: Prentice Hall.*

14. Holzner, S. (2002). *Visual Basic .NET Black Book, New York, United States of America: Al vavano.*

15. Leiter, C., Wood, D., Cierkowski, M. Boettger, A. (2009). *Beginning Microsoft SQL Server 2008 Administration. Indianapolis, Canadá: Wiley Publishing.*

16. Liu, M. (2010). *WCF 4.0 Multi-tier Services Development with LINQ to Entities, Birmingham, Reino Unido: Packt Publishing.*

17. MacDonald, M. (2012). *Pro WPF 4.5 in VB. Engelska: Apress.*

18. Mannino, M. (2007). *Database Design, Application Development, and Administration. United States of America: McGraw-Hill.*

19. Marshall, D. (2011). *Parallel Programming with Microsoft® Visual Studio® 2010 Step by Step. California, United States of America: O'Reilly Media.*

20. Microsoft Corporation. (2003). *Whitepaper: La arquitectura SOA de Microsoft® aplicada al mundo real.*

21. Microsoft. (n.d.). *Service Oriented Architecture: Right on Track. Obtenida el 10 de Agosto del 2014, de http://download.microsoft.com*

22. Patack, N. (2011). *Pro WCF 4: Practical Microsoft SOA Implementation, Second Edition. New York, United States of America: Apress.*

23. Ruiz, F., Serrano, M. (n.d.). *Programación con Visual Basic .NET- Obtenida el 06 de Setiembre del 2014, de http://http://alarcos.esi.uclm.es/*

24. Tagliaferri, M. (2001). *Visual Basic .NET! I didn't know You Could Do That, United States of America: Sybex Inc.*

25. Universidad Carlos III de Madrid. (n.d). *Diseño basado en componentes. Obtenida el 01 de Setiembre del 2014, de http://http://ocw.uc3m.es.*

26. Wiener, R. (2009). WCF: *A Case Study Involving a Distributed Client/Server Game, Journal of Object Technology. Zurich: ETH.*

ANEXO A

CUESTIONARIO AL JEFE DE DESARROLLO

1. ¿Conoce estándares de programación para aplicaciones orientadas a servicios? ¿Cuáles?

   .....

   .....

   .....

   .....

2. ¿Cómo realiza la actualización funcional en sus aplicaciones?

   .....

   .....

3. ¿Aplica la arquitectura Windows Communication Foundation?

   .....

   .....

4. ¿Implementa aplicaciones con Patterns and Practices?

   .....

   .....

5. ¿Cuántos incidentes/requerimientos funcionales realizan?

   .....

Tabla N° A.1: Cuestionario al jefe de desarrollo

ANEXOB

CUESTIONARIO A PROGRAMADORES

1. ¿Qué paradigma de programación aplica?

.....

.....

.....

2. ¿Describe los patrones de desarrollo que utiliza en sus aplicaciones?

.....

.....

3. ¿Cómo implementa aplicaciones orientada a Servicios?

.....

.....

.....

4. ¿Aplica Unitiy Container, Utility a sus aplicaciones?

.....

.....

5. ¿De qué manera realiza el acceso a datos?

A) En tres capas
B) Desde un componente
C) Usando dos capas
D) Otro:.....

6. ¿Realiza herencia visual de formularios?

A) Sí
E) No
F) Desconoce

7. ¿En qué capa realiza la implementación funcional?

A) En el servicio
G) En el aplicativo
H) Otro:.....

Tabla N° B.1: Cuestionario a programadores

ANEXOC
ANÁLISIS DOCUMENTAL

FECHA										
CASO DE USO	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10
HORAS	10 8 6 4 2 1									
APELLIDO Y NOMBRES:										

Tabla N° C.1: Gráfica de programación de acceso a datos