UNIVERSIDAD CATÓLICA DE SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENÍERIA DE SISTEMAS

"Desarrollo de un Componente de Software de Seguridad para el Monitoreo y Control de Información en un Equipo Informático bajo la Plataforma COM+"

Tesis Presentada por el Bachiller:

Igor Francis Calla Suxo

Para optar el Título Profesional de:

Ingeniero de Sistemas

AREQUIPA- PERÚ

2013

AGRADECIMIENTOS

A Dios por permitirme estar en este mundo,

A mi Familia y amigos por el gran apoyo brindado,

A mi asesora la Ing. *Karina Rosas Paredes por su guía y apoyo.*

DEDICATORIA:

Este proyecto está dedicado en primer lugar a mis padres por ser las personas más importantes en mi vida, por haberme brindado su apoyo incondicional en cada momento y por ser un ejemplo a seguir y de los cuales me siento muy orgulloso; a esa persona que Dios me permitió conocer Hinna L.C por haber compartido conmigo momentos inolvidables y haberme brindado los ánimos necesarios para alcanzar mis objetivos.

PRESENTACIÓN

Sra. Directora del Programa Profesional de Ingeniería de Sistemas.

Sres. Miembros del Jurado.

De conformidad con las disposiciones del Reglamento de Grados y Títulos del Programa Profesional de Ingeniería de Sistemas, pongo a vuestra consideración el presente trabajo de investigación titulado:

"Desarrollo de un Componente de Software de Seguridad para el Monitoreo y Control de Información en un Equipo Informático bajo la Plataforma COM+"

Calla Suxo, Igor Francis

RESUMEN

En el presente trabajo de tesis se plantea el desarrollo de un componente de software dentro de un contexto basado en seguridad, para que interactúe con el usuario, monitoreando y controlando las acciones del mismo mediante los permisos de cada usuario que ha iniciado sesión, brindando así seguridad frente al robo y mal manejo de la información de un equipo informático en las pequeñas y medianas empresas (Pymes), esto debido a que se tiene un vacio en el control de seguridad a nivel de máquina. El desarrollo de software basado en componentes (DSBC), es una tecnología que ha empezado a demostrar que ofrece ventajas en tiempo de desarrollo y reducción de costos en el proceso de desarrollo de software. Los desarrollos tradicionales de aplicaciones incurren en altos costos y en una inversión de tiempo extensa. El iniciar un desarrollo de software desde cero es un reto muy grande, incluso para una empresa que pueda soportar este proceso. Esto sesgaría el desarrollo de software a las grandes empresas, y no le daría cabida a las pequeñas y medianas empresas (Pymes) que desean adquirir tecnologías o construir sus propias soluciones. Las empresas pequeñas han estado siempre al margen del desarrollo de software; solo hasta la década pasada se les permitió la introducción a este campo. Esta introducción se originó ante la demanda de estas por buscar sus propios productos para dejar de depender de aquellos que las grandes empresas ponían en el mercado. Por otra parte, las empresas buscaban la reducción de costos en la tecnología (hardware, software e infraestructura de comunicación). Por eso el propósito en esta tesis es realizar la construcción de este componente de software que hará uso del lenguaje estandarizado

por ECMA C Sharp (C#) bajo la metodología UP (Proceso Unificado) y la de la plataforma COM+ (Component Object Model) ya que esta es una infraestructura de servicios que soporta la ejecución de componentes. Usaremos las ventajas que nos proporcionan los servicios web, y crearemos un host service usando las clases para servicios que nos proporciona el framework .NET haciendo uso de WCF (Windows Communication Foundation), Tomaremos como dominio del mismo los sistemas operativos de Windows desde la versión XP hasta la más reciente Windows 7, haciendo uso de la librería API de Windows sobre la arquitectura de 32 bits la cual nos dará mayor control sobre los sistemas operativos a probar, además se realizará comunicación entre procesos, así como la creación dinámica de objetos teniendo como finalidad y objetivo principal el lograr el control de una estación de trabajo restringiendo permisos de acuerdo al nivel de usuario que ingrese en la máquina pudiendo dejar copiar archivos o bloquear los mismos protegiendo así nuestra información de cualquier extraño brindando un componente que pueda ser utilizado por la comunidad para resolver exigencias reales en cuanto a monitoreo, control y seguridad de información, para esto el componente cubrirá ciertas características de control y hará el uso de un servicio que guardará información relevante sobre el usuario en un log de eventos que podremos visualizar diariamente, para así evitar el robo de información que es el principal activo de una empresa, o el uso indebido de las aplicaciones instaladas en el equipo de trabajo según el tipo de usuario, teniendo un acceso personalizado, además se comprobará el uso del componente en diferentes entornos que interactúen con el escritorio, se tiene como fin secundario que este proyecto se convierta en un proyecto de código abierto que continúe su crecimiento

y evolución a través de las aportaciones de los interesados en este tipo de componentes para lo cual se liberará el mismo haciendo uso de herramientas de software libre que permitan reutilizar y modificar el código del componente a desarrollar. Se presentará el prototipo de este componente que dará solución a un problema que se da actualmente y que es “la falla en seguridad de la información y el uso de aplicaciones indebidas” brindando así una solución eficiente. El resultado final de este trabajo será la elaboración de un componente que cumpla con los objetivos trazados al momento de realizar la propuesta de su desarrollo, y esto se probará a más detalle en el capítulo III y IV del presente trabajo de tesis.

ABSTRACT

In this thesis proposes the development of a software component within a security-based context, to interact with the user, monitoring and controlling the actions of the same through the permissions for each user that is logged, providing security against theft and mishandling of information a computer in small and medium enterprises (PYMEs), this because it has a gap in the security check at the machine level. The development of component-based software (CBSD) is a technology that has begun to show that offers advantages in development time and cost reduction in the software development process. The traditional application developments incur high costs and extensive time investment. The software development start from scratch is a big challenge, even for a company that can support this process. This would skew the software development for large companies, and would not give room for small and medium enterprises (PYMEs) that want to acquire technologies or build their own solutions. Small businesses have always been outside the software development, until the last decade only allowed the introduction to this field. This entry was originated with the demand for these products to seek their own to stop relying on those big companies put on the market. Moreover, companies seeking cost reductions in technology (hardware, software and communication infrastructure). So the purpose of this thesis is to build this software component that will make use of language standardized by ECMA C Sharp (C #) under the methodology UP (Unified Process) and the platform COM + (Component Object Model) as this is a service infrastructure that supports the execution of components. We will use the advantages we provide

web services, and create a service host using the classes for the services it provides. NET Framework using WCF (Windows Communication Foundation), will take control of it as Windows operating systems since version XP to latest Windows 7, using library Windows API on 32-bit architecture which will give us greater control over operating systems to test, and interprocess communication will take place and the dynamic creation of objects having aim and main objective is to gain control of a workstation restricting permissions according to the user level to enter the machine can copy files or block leave them protecting our information from any stranger offering a component that can be used by community to address real needs in terms of monitoring, control and information security, for this component will cover certain characteristics of control and make use of a service that will store relevant information about the user in a log of events that we see daily, to avoid information theft is the main asset of a company, or misuse of the applications installed on the computer work by type of user, having personalized access also be checked using the component in different environments to interact with the desktop, it is secondary purpose that this project becomes an open source project to continue its growth and evolution through the contributions of those interested in this type of components which will be released using the same free software tools that allow reuse and modify the code of the component to be developed. It will present the prototype of this component that will solve a problem that occurs now and is "the failure of information security and the use of misapplication" thus providing an efficient solution. The end result of this work will be the development of a component that meets the goals set at the time of the

proposed development, and this will prove more detail in Chapter III and IV of this thesis.

INTRODUCCIÓN

La presente tesis tiene como objetivo desarrollar un componente de software de seguridad para el monitoreo y control de acciones de un usuario basándonos en el uso del lenguaje C# bajo la plataforma COM+ todo esto orientado a las pequeñas y medianas empresas (PYMES), el proceso para el desarrollo del mismo se encuentra dividido en 4 capítulos, los cuales que se detallan a continuación:

El Capítulo I presenta el planteamiento del problema, su descripción y justificación, variables y sus indicadores, los alcances y limitaciones, asimismo, los objetivos generales y específicos.

El Capítulo II corresponde al marco teórico, en el cual se presenta toda la información teórica sobre la que se ha basado la resolución del problema.

El Capítulo III detalla el análisis y diseño para la construcción del componente, contemplando la elección de la tecnología a utilizar, análisis del dominio, diseño estructural y la definición funcional final, mediante el Lenguaje Unificado de Modelado (UML) haciendo uso del Proceso Unificado Rational (RUP), se describirá el desarrollo del componente de seguridad para el control y monitoreo haciendo uso de servicios de Windows, siguiendo cada uno de los pasos establecidos en la construcción del mismo y destacando también las características más relevantes en cuanto a su implementación.

El Capítulo IV corresponde a la evaluación del componente desarrollado, considerando encuestas a expertos y pruebas de seguimiento del componente y las aplicaciones que interactúen con el mismo para el posterior análisis de resultados. Finalmente se presentan las conclusiones y recomendaciones correspondientes de la tesis presentada.

ÍNDICE

ÍNDICE DE FIGURAS

CAPITULO II

CAPITULO III

ÍNDICE DE TABLAS

CAPITULO III

CAPITULO IV

ÍNDICE DE GRÁFICOS

CAPITULO I: PLANTEAMIENTO TEÓRICO

1.1 Identificación del Problema

1.1.1. Título Descriptivo del proyecto

"Desarrollo de un componente de software de seguridad para el monitoreo y control de información en un equipo informático bajo la plataforma COM+".

1.1.2. Descripción del problema

1.1.2.1. Definición del problema

Cuando nos referimos a la información que es el principal activo de una empresa, este tiene una importancia fundamental para el funcionamiento y quizá incluso sea decisiva para la perduración de la empresa u organización por ello es necesario hablar del problema de seguridad de la información a nivel de máquina, la seguridad es evitar que alguien extraiga información indebida desde una computadora o por alguno de sus periféricos, por ello, partamos de que cada usuario debe tener la información necesaria al nivel de acceso al cual fue delimitado, por esto la auditoría de sistemas es una buena práctica que debe desarrollarse en las instituciones, a efectos de resguardar la información en su integridad y confidencialidad, pero al realizar una auditoría interna puede traer consigo el no profesionalismo de no reportar las actividades hechas por los usuarios. El manejo de los

puertos de entrada/salida trae consigo el copiado de información relevante, si ésta no es controlada para determinar qué información fue manipulada, se corre el riesgo y la incertidumbre que se esté copiando información sin medida de control, que no nos garantice que fue con algún propósito licito.

1.1.3. Área científica a la que corresponde el problema

Área Científica: Ingeniería del software.

Línea: Arquitectura del software.

1.1.4. Tipo y Nivel de Investigación

Tipo: Aplicada.

Nivel: Descriptiva.

1.1.5. Objetivos

1.1.5.1. Objetivo General

"Desarrollar un componente de software de seguridad para el monitoreo y control de información de un equipo informático bajo la plataforma COM+".

1.1.5.2. Objetivos Específicos

- Crear un componente de software de seguridad que nos permita la reutilización del mismo basándonos en modelos comprobados.

- Utilizar la arquitectura de servicios WCF (Windows Communication Foundation) para la comunicación entre procesos del componente de software propuesto.
- Hacer uso de Herramientas para desarrollo basadas en Software Libre GPL o GNU.
- Desarrollar clases y librerías reutilizables que sean un punto de partida para el desarrollo de este tipo de componentes.
- Demostrar la utilidad y facilidad de uso del componente en diferentes plataformas de sistemas operativos.

1.2. Formulación de la Hipótesis

Dado que se tiene el problema de seguridad de información a nivel de máquina, “Es posible mejorar el monitoreo, control y seguridad de información de una estación de trabajo restringiendo permisos protegiéndola de cualquier extraño brindando el desarrollo de un componente de software de seguridad basado en un marco teórico que permita implementar este tipo de aplicaciones de una manera más eficiente, mucho más sencilla y más rentable aprovechando las características de la plataforma de Modelo de Objetos de Componentes (COM+) y las clases del Framework .NET haciendo uso de servicios WFC, así como de herramientas que nos proporciona el software libre para que pueda ser utilizado por la comunidad para resolver exigencias reales en cuanto a la falla en seguridad de la información y el uso de aplicaciones indebidas”.

1.3. Variables del Estudio

1.3.1. Variables Independientes

- Software Libre.
- Modelo de Objetos de Componentes.
- Plataforma .Net.

☑ **Indicadores**

- Tiempo de implementación.
- Tiempo de respuesta.

1.3.2. Variables Dependientes

- El Componente de Software a crear en la propuesta.

☑ **Indicadores**

- Utilidad.
- Eficiencia.
- Seguridad.
- Sencillez

1.4. Técnica de Recolección de Datos

- Investigación teórica, encuestas a expertos y revisión documental.

1.5. Solución propuesta

1.5.1. Justificación

Para solucionar el problema se plantea el desarrollo de un componente de seguridad basado en la plataforma COM+, ya que estos en la actualidad forman parte de la piedra angular de la ingeniería del software moderna ya

que el desarrollo y uso de componentes de software ganó aceptación rápidamente debido a su capacidad para promover la reutilización de los mismos.

Los componentes de software se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados, en éste caso problemas relacionados a los procesos de monitoreo y control de seguridad de la información en una empresa como son las PYMES, entre otras cosas el diseño del componente de software orientados al modelo de objeto de componentes(COM+), permite una mayor portabilidad y un mejor tiempo de respuesta, en comparación con el desarrollo tradicional de los sistemas de seguridad de software que son más costosos y que trabajan en un dominio de sistemas operativos limitado, además la configuración flexible del componente, permitirá también la reutilización del código que lo generó de una manera más provechosa. La utilización de estos viene siendo muy exitosa en muchos dominios. Las empresas del dominio antes mencionado (PYMES), necesitan un control automatizado de todos sus movimientos referentes al tratamiento de la información de los distintos usuarios para poder tener un mejor control en la administración de los mismos por lo que se considera que la implementación de un componente de software de seguridad ayudaría enormemente en dicha labor, que es la de auditoría de la información.

1.6. Descripción de la solución

Se desarrollara un componente de seguridad para el monitoreo y control de un equipo informático utilizando la plataforma de Modelo de Objetos de Componentes (COM+) haciendo uso de herramientas que nos proporciona el Software Libre , lo cual traería también como consecuencia un mejoramiento en la eficiencia y eficacia en la auditoria de procesos que se realizan en las empresas (PYMES).

1.7. Alcances y Limitaciones

1.7.1. Alcances:

- Una mejor auditoria de procesos en cuanto a la manipulación de información en un dominio especifico.
- Aumentar la seguridad de los procesos a ejecutar en un equipo informático en las distintas áreas de las empresas (en este caso las Pymes).
- Portabilidad e Independencia en cuanto a la plataforma, haciendo que el componente se pueda ejecutar de manera trasparente en cualquier sistema operativo de Windows.
- Utilización de herramientas de software libre que nos permita tener un costo cero sobre el mantenimiento del componente de software de seguridad a desarrollar ya que el componente será independiente del IDE de desarrollo.

1.7.2. Limitaciones:

- No está orientado a grandes empresas donde se requiere el uso de componentes más sofisticados.

1.8. Antecedentes Investigativos:

A continuación se hace referencia a dos proyectos los cuales se enfocan en las siguientes temáticas: “seguridad, bloqueo de puertos, monitoreo y control de accesos”.

1.8.1. USB Blocker

Este software es una solución que nos permite implementar la protección de los puertos USB pero desde un punto centralizado. Para ello requiere, evidentemente, que tengamos un dominio bien configurado, pues funciona mediante las políticas de este, siendo compatible con el Active Directory de Windows. Se puede incluso definir una lista de ordenadores en los que no se aplicarán esas políticas evitando que se puedan conectar dispositivos USB de almacenamiento, haciendo más difícil el robo de información o la entrada de malware.

1.8.2. Devicelock

Este software hace que cada vez que un usuario desea acceder un dispositivo, se intercepte esta solicitud a nivel del núcleo del sistema operativo. Dependiendo del tipo de dispositivo y la interfaz de conexión (por ejemplo, USB), DeviceLock verifica los derechos del usuario en la

Lista de Control de Acceso (Access Control List) adecuada. Si el usuario no dispone de derechos para acceder al dispositivo, se muestra un error de “acceso denegado”. La verificación de acceso puede producirse en dos niveles: el “nivel de interfaz” (puerto) y el “nivel de tipo”. Algunos dispositivos se verifican en ambos niveles, mientras que otros sólo en un nivel. Ambos ejemplos no están orientados al uso de componentes y solo funcionan en ciertos sistemas operativos de Windows.

CAPITULO II: MARCO TEÓRICO

2.1. Reseña histórica de Componentes de Software

Como respuesta a la necesidad de buscar alternativas que permitan hacer más fáciles las tareas de desarrollo e integración de software, uno de los más recientes modelos de desarrollo es el de componentes. Aspectos como el reuso y la interoperabilidad, muy importantes en la actualidad, son característicos de este enfoque, y este puede visualizarse como un modelo de desarrollo bastante prometedor, la historia del software de componentes es algo compleja, esto se debe más que todo a que Microsoft ha tendido a cambiar su enfoque y su estructura cada pocos años, dando nuevos nombres y nuevas marcas a las tecnologías, inicialmente se opto por el desarrollo de una tecnología para documentos combinados llamada OLE (Object Linking and Embedding), la cual fue construida encima de controles conocidos como Dynamic Data Exchange (Intercambio dinámico de datos) y Visual Basic Extension (VBX) de Visual Basic 1.0. En 1992, Microsoft introdujo Windows 3.1 y con este OLE llegó a los ordenadores de escritorio, sin embargo el fabricante otorgó a algunas partes de OLE relativas a Internet el nuevo nombre de ActiveX, un símbolo que a la larga abarcaba gradualmente a todas las tecnologías de componentes. Entonces OLE ocupó de nuevo el papel de tecnología de documentos combinados (para lo que fue creado originalmente) y fue utilizada principalmente en Microsoft Office. En 1995 llegó el tiempo de COM, que se implementó por primera vez en Windows 95, luego en 1996, al aumentar la

importancia de las redes y de las aplicaciones conectables en red, Microsoft anunció DCOM como su alternativa a la arquitectura CORBA (Common Object Request Broker Architecture). DCOM apareció por primera vez en Windows NT 4.0, con herramientas de ayuda para crear aplicaciones cliente/servidor que abarcaban tanto la red corporativa como Internet. En septiembre de 1998, Microsoft cambió de nuevo a COM el nombre de todo su Framework. Posteriormente cuando se introdujo Windows 2000, Microsoft modifico de nuevo el COM consiguiendo que la organización de Tecnología de Información de una empresa pudiera ejecutarlo en grupos o conjuntos de componentes gestionados por el MTS (Microsoft Transaction Server). Un componente creado adecuadamente podía ser reutilizado realizando llamadas a su rutina de re-inicialización sin necesidad de descargarlo de la memoria, los componentes podían también ser “llamados” desde otro ordenador, lo cual antes sólo era posible con DCOM, por lo que Microsoft prescindió en gran medida de DCOM como concepto independiente, sin embargo, todavía quedaba un nuevo paso: Microsoft lanzaba su iniciativa .Net, un framework que casi sustituyó por completo a la tecnología COM, aunque la compatibilidad retroactiva es limitada. De hecho, la tecnología .Net puede utilizar un objeto COM implementando lo que se conoce como un wrapper, .Net incluye software y sistemas operativos de aplicación para clientes “inteligentes”, dispositivos inteligentes y servicios web que pueden ser combinados con otros e incluso ser utilizados directamente con aplicaciones de clientes inteligentes, una infraestructura de servidor y un entorno (Visual Studio .Net) que soporta

directamente cierto número de lenguajes de desarrollo a través del CLR(Common Language Runtime de .Net). A pesar de haber sido relativamente sustituidas por .Net, las tecnologías COM y DCOM siguen vivas, siendo que en el Service Pack 2 para Windows XP Microsoft implementó dos cambios importantes a DCOM. Así, introduce restricciones a nivel de todo el ordenador que ofrecen una comprobación de acceso adicional, a través de una lista de control de accesos cada vez que es activado, "llamado" o puesto en marcha un servidor COM. Es por esto que actualmente se puede crear soluciones .Net que interactúen con componentes COM y DCOM de manera transparente al usuario final.

2.2. Concepto de Componente

La palabra componente nos hace pensar en una unidad o elemento con propósito bien definido que, trabajando en conjunto con otras, puede ofrecer alguna funcionalidad compleja. Transportando este concepto al contexto de ingeniería de software, específicamente de desarrollo basado en componentes, un componente es la pieza elemental de este enfoque de desarrollo, de esta forma, a partir de componentes existentes pueden llegarse a construir aplicaciones completas. En términos formales, y de acuerdo con [MAR96], un componente es una pieza de software que cumple con dos características: no depende de la aplicación que la utiliza y, se puede emplear en diversas aplicaciones. Según se cita [SZY97], los asistentes a la ECOOP96 definen a un componente como una unidad de composición con interfaces definidas contractualmente y dependencias contextuales explícitas solamente.

Existen varias definiciones de componentes realizadas por expertos que han sido los encargados del desarrollo de esta metodología, ellos han tomado como base la metodología de la programación orientada objetos y el modelado a través de UML:

• Un componente es una parte no trivial, casi independiente, y reemplazable de un sistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. Un componente se conforma y provee la realización física por medio de un conjunto de interfaces [KRU98].

• Un componente de software en tiempo de ejecución es un paquete dinámicamente vinculado con uno o varios programas manejados como una unidad y que son accedidos mediante interfaces bien documentadas que pueden ser descubiertos en tiempo de ejecución [GAR98].

• Un componente de software es una unidad de composición con interfaces contractualmente especificadas y explícitas sólo con dependencias dentro de un contexto. Un componente de software puede ser desplegado independientemente y es sujeto a la composición de terceros [SZY98-B].

• Un componente de negocio representa la implementación de software del concepto de un negocio “autónomo” o un proceso de negocio. Que consiste de artefactos de software necesarios para expresar, implementar y poner en marcha el concepto de elemento reusable de un sistema más grande de negocios [KOZ98].

Estas definiciones no son mutuamente excluyentes, por el contrario, se complementan y construyen el significado no sólo de componente, también el significado del desarrollo basado en componentes. Sin embargo más allá de su definición existen algunas características claves para que un elemento pueda ser catalogado como componente:

Identificable: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.

Auto contenido: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.

Puede ser remplazado por otro componente: Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.

Con acceso solamente a través de su interfaz: Debe asegurar que estas no cambiaran a lo largo de su implementación.

Sus servicios no varían: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

Bien Documentado: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.

Es genérico: Sus servicios deben servir para varias aplicaciones.

Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.

Independiente de la plataforma: Hardware, Software, S.O [MON03].

Un componente de software puede desplegarse independientemente y estar sujeto a composición por terceros. El concepto de componentes para el desarrollo de software no es un concepto nuevo; para muchos autores simplemente es la evolución de la metodología orientada a objetos [FUE03-A]. De hecho, muchas de las características de los componentes para el desarrollo parten de la idea del diseño orientado a objetos. Pero la historia del desarrollo de software basado en componentes proviene aún desde más atrás. Uno de los logros de la revolución industrial fue el desarrollo por componentes, surgiendo a partir de la necesidad de estandarizar los elementos de los productos realizados en línea, como los automóviles [GON04]. El verdadero valor del software de componentes para el desarrollo de aplicaciones es que permite utilizar módulos binarios, es decir, codificación en lenguaje de máquina y no bibliotecas de código fuente como sucedía con la mayoría de los entornos de desarrollo. Los desarrollos tradicionales de aplicaciones incurren en altos costos y en una inversión de tiempo extensa. El iniciar un desarrollo de software desde cero es un reto muy grande, incluso para una empresa que pueda soportar este proceso. Esto sesgaría el desarrollo de software a las grandes empresas, y no le daría cabida a las pequeñas y medianas empresas que desean adquirir tecnologías o construir sus propias soluciones. Las empresas pequeñas han

estado siempre al margen del desarrollo de software; solo hasta la década pasada se les permitió la introducción a este campo. Esta introducción se originó ante la demanda de estas por buscar sus propios productos para dejar de depender de aquellos que las grandes empresas ponían en el mercado. Por otra parte, las empresas buscaban la reducción de costos en la tecnología (Hardware, Software e Infraestructura de Comunicación). Los desarrolladores de aplicaciones han establecido reglas de procedimiento, programación orientada a objetos y bibliotecas de software reutilizable en un intento por conseguir una mayor interoperabilidad y eficiencia, normalmente y si pueden evitarlo, los desarrolladores rara vez crean aplicaciones a partir de cero, y lo que intentan es aprovechar la infraestructura de hardware, software y herramientas ya existentes, así como componentes de software previamente creados, todo ello para controlar el tiempo y el coste del desarrollo así como del despliegue de las aplicaciones. Aunque estos métodos adoptan nombres muy diferentes, se designan normalmente como “software de componentes”, por lo que podemos considerar por ejemplo a los servicios web como una forma más distribuida del software de componentes. Teniendo en cuenta que Windows se ha convertido en el sistema operativo principal de muchos ordenadores de escritorio y que Microsoft ha decidido extender ese monopolio al mundo de los servidores, no es sorprendente que haya ofrecido durante años su propio repertorio de productos y estándares para software de componentes teniendo entre los más antiguos, y por lo tanto más conocidos y ampliamente utilizados el COM y su sucesor con capacidad para redes conocido como DCOM (Distributed COM).

2.3. Concepto de Desarrollo de Software Basado en Componentes

Lo más importante que se debe tener en cuenta al definir el concepto de DSBC (Desarrollo de Software Basado en Componentes), es el de diferenciar las características existentes entre el DSBC y el paradigma de objetos, la Programación Orientada a Objetos (POO) es cuando el software puede ser desarrollado de acuerdo a un modelo mental de un objeto real o imaginado, mientras que el DSBC dice que el software debe ser desarrollado a partir de componentes prefabricados. El DSBC busca, dentro de otros objetivos, reducir el tiempo de trabajo, el esfuerzo que requiere implementar una aplicación y los costos del proyecto y de esta forma, incrementar el nivel de productividad de los grupos desarrolladores y minimizar los riesgos globales sin incurrir en gastos exorbitantes. De esta manera, las pequeñas empresas pueden tener una mayor confiabilidad a la hora de realizar una inversión tecnológica. Otra ventaja es poder integrar lo mejor de las tecnologías para desarrollar una aplicación de manera personalizada, a la medida de las necesidades de los clientes. Esto permite a los desarrolladores y a la empresa adquirir las tecnologías que más se adapten a sus necesidades, y no incurrir en gastos de licenciamiento o soporte y actualización de las grandes soluciones, ya que muchas de estas tecnologías son gratis y existen bajo la premisa de Freeware y GNU (General Public License), lo cual añade otra gran ventaja. Otro punto a tomar en cuenta es que el DSBC, pertenece al paradigma de programación de sistemas abiertos, los cuales son extensibles y tienen una interacción con componentes heterogéneos que ingresan o abandonan el sistema de forma dinámica, es decir que los componentes pueden ser

remplazados, por otros independientemente de su arquitectura y de su desarrollo[FUE03-B]. De esta manera se puede hacer un paralelo entre el mundo percibido por el hombre y el DSBC desde un punto de vista menos formal y con una perspectiva del mundo cotidiano donde se tienen sistemas abiertos y cerrados. Por ejemplo, el ser humano es un sistema abierto y realiza interacciones con el medio (otro sistema), la mayoría de los sistemas en el mundo percibido por el hombre son abiertos y necesitan una interacción con el medio para poder sobrevivir como en el caso del hombre (alimentos, aire, relaciones, etc.). Esta interacción es la que permite que crezca el sistema (extensión), dándole la posibilidad de mejorar con el tiempo y lograr una estabilidad (evolución para el caso del hombre).

Existen ya varios modelos que permiten entender los componentes de software, para poder encontrar los componentes que determinada aplicación necesita. Uno de estos modelos desarrollado por Anneliese Andrews y Sudipto Ghosh, en Colorado State University [AND02] plantéa la posibilidad de entender un componente por medio de tres perspectivas, estas son:

- **Dominio:** Representa en términos generales todos los aspectos del problema del usuario relacionado de forma directa con la funcionalidad que apoya el componente.

- **Programa:** Este enfoque es el que más varia de componente a componente, ya que muestra en forma más detallada la información técnica del componente

como la estructura de los archivos de información, definiciones de las bases de datos, la definición de la Interface de datos, los tipos de parámetros, información acerca de la invocación de los métodos del componente, etc.

• **Situación:** En este punto “El conocimiento del diseño y comportamiento de las entidades necesita ser especificado” [AND02], la información que se determina en este punto es útil para igualar los requerimientos con las capacidades funcionales y no funcionales del componente; aquí se deben determinar aspectos como: Estructura de la arquitectura física y conceptual, flujos de Información que el componente transfiere, usa y transforma y tipos de algoritmos.

Ya fueron presentados los términos: la definición de componente, sus características y las variables por las cuales puede ser comprendido un componente. Ahora bien, ¿Cómo se integran los componentes dentro del ciclo de vida de desarrollo de software basado en componentes? La siguiente figura ilustra el ciclo de vida del software basado en componentes desde una perspectiva global [XIA02]:

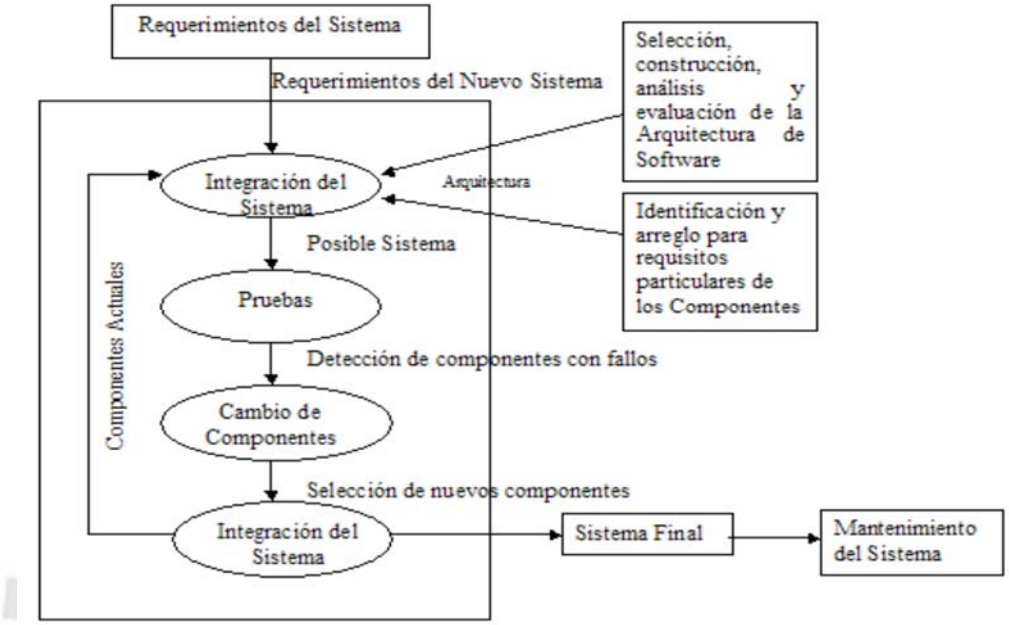


Figura 2.1: Modelo del ciclo de vida de DSBC.

Fuente: Elaboración Propia.

A continuación mostramos una breve descripción de cada una de las actividades que involucra el modelo del ciclo de vida para DSBC:

• **Análisis de Requerimientos:** En esta etapa del ciclo de vida los procesos y las necesidades del negocio se descubren y se expresan en los casos de uso.

• **Selección, construcción, análisis y evaluación de la Arquitectura de Software:** "La arquitectura del software define un sistema en términos de componentes computacionales y la interacción entre ellos" [XIA02]. Estas tareas podrán ser realizadas con éxito si se exponen las propiedades externas de los componentes que harán parte de la aplicación y las relaciones entre ellos.

• **Identificación y arreglo para requisitos particulares del Componente:** En esta actividad, los componentes deben ser seleccionados por los requerimientos funcionales y de calidad que satisfaga cada componente. Luego de haber sido identificados los componentes que serán integrados al sistema, se debe evaluar si el componente necesita ser sujeto a alguna modificación.

• **Integración del Sistem**a: En esta actividad se debe examinar, evaluar y determinar cómo va a ser la comunicación y la coordinación entre los componentes que harán parte del sistema. Luego debe ensamblarse el sistema y proseguir con una serie de pruebas que determinarán si los componentes seleccionados son los adecuados.

• **Pruebas:** Posterior a la integración de los componentes se debe proceder con las pruebas, esto implica evaluar el funcionamiento de los componentes que fueron integrados en el sistema, si algún componente demuestra no estar funcionando de forma correcta se debe pensar en la posibilidad de reemplazarlo o modificarlo para luego proceder con la reintegración.

• **Mantenimiento:** En el período del mantenimiento, se lleva a cabo un proceso similar al desarrollado en la POO, esto es vigilar el correcto funcionamiento del sistema, corregir fallas en el comportamiento, etc.

2.4.Clasificación de Componentes

2.4.1. ¿Cómo clasificar componentes?

La clasificación de componentes es un proceso extenso, ya que un componente involucra en sí mismo varios atributos relevantes. En este segmento se expone una serie de variables que podrían considerarse criterios de clasificación de componentes:

2.4.1.1. Tamaño

El tamaño de los componentes puede ser medido por medio de las métricas utilizadas en diseño orientado a objetos. Esto significa que la medición del tamaño de un componente puede ser medido a través de:

• Líneas de Código (LDC).

• Orientadas a Función.

“Si un componente resulta ser demasiado grande en tamaño, el proceso de aseguramiento de calidad del mismo será más complicado y exigente para el desarrollador” [XIA02].

2.4.1.2. Complejidad

En algunas ocasiones, son utilizadas métricas de tamaño para evaluar la complejidad, pero es recomendable hacer uso de otro tipo de métricas. “Si un componente es demasiado trivial no podrá sacársele mayor provecho en su reutilización, y si el componente es demasiado complejo será difícil asegurar su calidad” [XIA02].

2.4.1.3. Métricas de Complejidad

Las métricas más utilizadas para medir la complejidad de los componentes, combina uno de los métodos más reconocidos para medir la complejidad de métodos o algoritmos desarrollado por Tomas McCabe, "Complejidad Ciclomática", este método mide el número de decisiones lógicas en un segmento de código. A continuación nombramos cuatro diferentes técnicas para medir la complejidad de un componente:

- **CPC (Component Plain Complexity)**: Mide la complejidad del componente por medio de la suma de clases, clases abstractas e interfaces, la complejidad de clases y métodos.
- **CSC (Component Static Complexity)**: Se centra en la estructura interna del componente por medio de una visión estática del mismo, utilizando variables como la relación entre las clases y el peso de cada relación.
- **CDC (Component Dynamic Complexity):** Se centra en el número de mensajes que pasan dentro del componente por medio de una visión dinámica, evaluando variables como la frecuencia en el intercambio de mensajes entre clases y la complejidad de los mensajes.
- **CCC (Component Cyclomatic Complexity):** Esta medida de complejidad es utilizada cuando el componente ya ha sido finalizado. Utiliza como variables el código desarrollado, la suma de las clases, interfaces, métodos definidos en cada una de las interfaces.

- A diferencia del método CCC, los métodos anteriores CPC, CSC, CDC utilizan para sus cálculos los diagramas de clase, la interacción de los diagramas y el diagrama de componentes.

2.4.1.4. Mantenibilidad

"La mantenibilidad de un sistema es la facilidad con la cual puede ser modificado frente a cambios en el ambiente, requerimientos funcionales o especificaciones funcionales" [BEN02]. Para los componentes de software esta es una característica de vital importancia, ya que su propia naturaleza los obliga a tener la capacidad de adaptarse a diferentes entornos.

Generalmente, las métricas que se utilizan para medir la mantenibilidad utilizan variables que miden los atributos, el comportamiento y el flujo de trabajo, entre otros.

2.4.1.5. Reusabilidad

La reusabilidad de un componente se puede medir a partir de dos diferentes perspectivas, estas son [MIN02]:

- Cómo puede un componente ser reutilizado: Este tipo de medida tiene en cuenta las siguientes variables: El número de cada método de interface que puede proveer funciones en común entre varias aplicaciones en un dominio, el número de métodos declarados en la interface que pertenecen al componente.
- Cómo es re - usado un componente en una aplicación particular: Las variables que se miden para este objetivo en particular son: Las líneas de código

re - utilizadas por el componente en una aplicación, el total de líneas entregados en la aplicación. La combinación de estas dos variables resulta el porcentaje de funcionalidad que aporta el componente dentro de toda la aplicación.

2.4.1.6. Frecuencia de Reuso

El número de veces que ha sido utilizado un componente dentro de distintas aplicaciones, es sin lugar a dudas el mejor indicador de frecuencia de re– uso. Cabe anotar que este atributo puede ser solo medido en componentes que ya han sido expuestos al mercado.

2.4.1.7. Confiabilidad

Es la probabilidad de falló en el funcionamiento del componente dentro de cierto escenario operacional.

2.5.¿Cómo evaluar la calidad de un componente?

"La calidad dentro de la Ingeniería de Sistemas se define como la totalidad de aspectos y características de un producto o servicio que tiene que ver con su habilidad de satisfacer las necesidades declaradas o implícitas" [BER02].

La calidad de los componentes de software puede ser el factor decisivo para la selección de los componentes que entrarán a ser parte de un nuevo sistema, es decir, si dos componentes son candidatos para hacer parte de una aplicación y muestran la misma funcionalidad, el nivel de calidad que implemente cada uno de ellos será el elemento sobre el cual se base la decisión final. Algunos de los

elementos que pueden dar al usuario una idea del nivel de calidad de un componente son [XIA02]:

- Funcionalidad.
- Interface.
- Usabilidad.
- Pruebas.
- Seguridad.

2.6.Reutilización de Componentes de Software

La reutilización de componentes de software es un proceso inspirado en la manera en que se producen y ensamblan componentes en la ingeniería de sistemas físicos.

La aplicación de este concepto al desarrollo de software no es nueva. Las librerías de subrutinas especializadas, comúnmente utilizadas desde la década de los setenta, representan uno de los primeros intentos por reutilizar software.

Existen variadas definiciones del término Reutilización de Software. Algunas de estas definiciones son las siguientes:

- Según Somerville [SOM00], “la reutilización es un enfoque de desarrollo [de software] que trata de maximizar el uso recurrente de componentes de software existentes”.
- Según Sametinger [SAM97], “la reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo”.

- Según Sodhi et al. [SOD99], "la reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes".

Estas tres definiciones consideran la reutilización como un enfoque o proceso de desarrollo de software. Su principal diferencia radica en el objeto de reutilización (componente, software y activo). Tal como lo plantean Sodhi et al. [SOD99], la reutilización de software va más allá de la reutilización de piezas de software. Ella involucra el uso de otros elementos de software, tales como algoritmos, diseños, arquitecturas de software, documentación y especificaciones de requerimientos.

Con base en el análisis de estas definiciones podemos establecer nuestra propia definición en los siguientes términos:

"La reutilización de software es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la especificación, análisis, diseño, implementación y pruebas de una aplicación o sistema de software".

Varios estudios han demostrado la efectividad de la reutilización del software. Sametinger [SAM97], en particular, describe algunos de estos indicadores:

- Entre el 40 y 60% del código fuente de una aplicación es reutilizable en otra similar.

- Aproximadamente el 60% del diseño y del código de aplicaciones administrativas es reutilizable.
- Aproximadamente el 75% de las funciones son comunes a más de un programa.
- Sólo el 15% del código encontrado en muchos sistemas es único y novedoso a una aplicación específica. El rango general de uso recurrente potencial está entre el 15% y el 85%.

A partir de estos indicadores es fácil deducir el impacto y la importancia que tiene la reutilización de componentes en el proceso de desarrollo de software; particularmente, en tres de las variables más importantes de la Ingeniería de Software: el costo, tiempo y esfuerzo requerido para desarrollar un producto de software. La aplicación apropiada de la reutilización en un proyecto de software conduce, indiscutiblemente, a una reducción significativa de los valores de estas tres variables. Otros beneficios importantes son el incremento de la calidad del software producido, el aumento de la productividad de los grupos de desarrollo y la reducción del riesgo global del proyecto.

2.7.Activos Reutilizables y Componentes de Software

En el contexto de Ingeniería de Software, un *Activo Reutilizable* es un producto diseñado expresamente para ser empleado de forma recurrente en el desarrollo de muchos sistemas y aplicaciones. Ejemplos de activos reutilizables son:

algoritmos, patrones de diseño, esquemas de base de datos, arquitecturas de software, especificaciones de requerimientos, de diseño y de prueba, entre otros.

En los últimos años, como resultado de presiones crecientes sobre la industria del software orientado a reducir drásticamente el costo y tiempo de desarrollo de sistemas y aplicaciones, sin afectar los niveles de calidad del producto, ha surgido un nuevo activo reutilizable denominado *Componente de Software*. Se han propuesto numerosas definiciones a este término, entre las cuales destacan las siguientes:

- Según Philippe Krutchen de Rational Rose [KRU98], un componente es una parte no trivial, casi independiente y reemplazable de un sistema que cumple una función dentro del contexto de una arquitectura bien definida. Un componente cumple con un conjunto de interfaces y provee la realización física de ellas.
- Según Clemens Szyperski [SZY02], un componente de software es una unidad de composición con interfaces especificadas contractualmente y solamente dependencias explícitas de contexto. Un componente de software puede ser desplegado independientemente y está sujeto a composición por terceros.
- Según Herzum y Sims [HER00], un componente es un artefacto de software auto contenido y claramente identificable que describe ó ejecuta funciones específicas; que tiene, además, una interfaz claramente

establecida, una documentación apropiada y un *status* de uso recurrente bien definido.

- Según el CBDi Forum [CDBI99], un componente es una pieza de software que describe y/o libera un conjunto de servicios que son usados sólo a través de interfaces bien definidas.

Más recientemente, el Instituto de Ingeniería de Software (SEI, *Software Engineering Institute*) de la **Universidad Carnegie-Mellon** formuló una definición con el propósito de consolidar las diferentes opiniones acerca de lo que debía ser un componente de software. Según el SEI [BACH00], **un componente es "una implementación opaca de funcionalidad, sujeta a composición por terceros y que cumple con un modelo de componentes"**. Con respecto al primer aspecto, un componente se considera una implementación opaca debido a que su distribución predominantemente es en formato binario y sus consumidores lo utilizan como una "caja negra" a través de su interfaz. Dicho aspecto está alineado con el principio de encapsulamiento de la programación orientada a objetos [BUD94], [JOY98].

Por otra parte, la composición por terceros implica que los componentes son intrínsecamente reutilizables debido a que un sistema basado en componentes puede ser ensamblado con relativa facilidad por un integrador empleando componentes suministrados por múltiples proveedores independientes. Finalmente, la coordinación e interacción entre componentes exige un marco regulatorio estandarizado (modelo de componentes) que establece la

infraestructura de software requerida (*framework*) y las convenciones y restricciones de diseño de los mismos.

Tal como lo refleja la definición anterior, un componente de software puede ser visto desde dos perspectivas distintas, como:

1. Implementación: los componentes se pueden ensamblar y desplegar para crear sistemas y aplicaciones que se ejecutan en un computador.

2. Abstracción de arquitectura: los componentes expresan las reglas de diseño que impone el modelo de componentes.

Están disponibles diversas tecnologías de componentes típicamente asociadas con infraestructuras de procesamiento distribuido (Enterprise JavaBeans [SUN01], CORBA Component Model [OMG02] y .NET [THAI03]) y aplicaciones de escritorio (Controles ActiveX [CHAP96] y JavaBeans [SUN97]).

Una de las características más importantes de los componentes es que son reutilizables. Para ello los componentes deben satisfacer como mínimo el siguiente conjunto de características:

- **Identificable:** un componente debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.

- **Accesible sólo a través de su interfaz:** el componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.
- **Sus servicios son invariantes:** las operaciones que ofrece un componente, a través de su interfaz, no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz.
- **Documentado:** un componente debe tener una documentación adecuada que facilite su búsqueda en repositorios de componentes, evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte. Adicionalmente, para favorecer su reutilización es deseable que un componente sea:
- **Genérico:** sus servicios pueden ser usados en una gran variedad de aplicaciones.
- **Auto contenido:** es conveniente que un componente dependa lo menos posible de otros componentes para cumplir su función de forma tal que pueda ser desarrollado, probado, optimizado, utilizado, entendido y modificado individualmente.
- **Mantenido:** es deseable que un componente (como toda pieza de software) esté inmerso en un proceso de mejoramiento continuo que le garantice al integrador nuevas versiones que incluyan correctivos, optimizaciones y

nuevas características. Esto contribuye a que dicho componente sea seleccionado con mayor frecuencia para formar parte de sistemas de software.

- **Independiente de la plataforma (hardware y sistema operativo), del lenguaje de programación y de las herramientas de desarrollo:** existen diversas plataformas de cómputo de uso frecuente (Windows/Intel, Solaris/Sparc, OSX/PPC, Linux/Intel) y es deseable que un componente pueda ejecutarse en todas ellas.
- Asimismo, ya que existe una amplia gama de lenguajes de programación y herramientas de desarrollo, es natural que encontremos componentes escritos empleando lenguajes y herramientas de la preferencia del programador, por lo tanto es deseable que dichas preferencias no limiten el uso de los componentes.
- **Puede ser reutilizado dinámicamente:** puede ser cargado en tiempo de ejecución en una aplicación.
- C**ertificado:** el componente puede ser certificado por una agencia de software independiente o mediante la aplicación de modelos de auto-certificación que le permiten al comprador del componente determinar la calidad del software adquirido [MOR01].
- **Accedido uniformemente sin importar su localidad**: la forma de invocar los servicios ofrecidos por los componentes debiese ser independiente de su ubicación (local o remota). Para ello el modelo de componentes debería

estar basado en tecnologías de procesamiento distribuido tales como CORBA [OMG02], RMI [SUN02] y .NET Remoting [THAI03].

2.8. La Interfaz de un Componente

La interfaz define el conjunto de operaciones que un componente puede realizar; estas operaciones son también llamadas servicios o responsabilidades. Las interfaces proveen un mecanismo para interconectar componentes y controlar las dependencias entre ellos.

La naturaleza de la interfaz varía dependiendo del lenguaje programación empleado para implementar el componente. Los lenguajes orientados a objetos (Smalltalk-80 [GOL89], C++ [STRO00] y Java [STE00]) soportan alguna forma de interfaz, que por lo general están separadas de las implementaciones. En lenguajes procedimentales (Pascal [NAN95] y FORTRAN [SMOR94]) las interfaces se definen a través de declaraciones de funciones o procedimientos y el uso de variables globales.

Algunos lenguajes neutrales de especificación de interfaces han sido desarrollados tales como IDL (*Interface Definition Language*) [OMG02] de OMG (*Object Management Group*).

En general, una interfaz de programación de aplicaciones (API, *Application Programming Interface*) es una especificación, en un lenguaje de programación, de las propiedades de un módulo de software. Los clientes del módulo sólo deben

depender exclusivamente de las propiedades definidas por el API de forma explícita.

Algunas tecnologías (Enterprise JavaBeans [SUN01]) exigen que sus componentes implementen dos tipos de interfaces:

1. **Interfaz de negocio**: que refleja el rol del componente en el sistema.
2. **Interfaz de infraestructura**: es impuesta por el modelo de componentes con el fin de permitirle al *framework* interactuar con el componente.

Por otra parte, nótese que las interfaces convencionales definen la firma de las operaciones (nombre de la operación, tipo y orden de los argumentos, y la manera como se devuelven los resultados) que provee un componente. Las operaciones también se conocen como propiedades funcionales. Sin embargo, estas interfaces no expresan adecuadamente propiedades del componente relativas a, por ejemplo, su desempeño, precisión, disponibilidad, latencia, seguridad, entre otras. Dichas propiedades se conocen como propiedades extra funcionales [DIG99].

Es útil diferenciar los tipos de propiedades de los componentes. Por ejemplo, Beugnard et al. [BEU99] define cuatro tipos de propiedades relacionadas con:

1. Sintaxis: corresponden a las propiedades funcionales expresadas explícitamente a través de la interfaz del componente.

2. Comportamiento: definen las condiciones que deben cumplir los valores de entrada (precondiciones) y salida (post-condiciones) de las operaciones.

3. Sincronización: expresan aspectos de concurrencia.

4. Calidad de Servicio: contempla atributos tales como tiempo de respuesta, uso de memoria, precisión, confiabilidad, facilidad de mantenimiento y reutilización, entre otros. Se han realizado algunos intentos para que las interfaces expresen mejor las propiedades extra funcionales, tales como el lenguaje de programación Eiffel [MEY91] y el método formal Object-Z [DUK95] para propiedades de comportamiento, Object Calculus [LAN97] para propiedades de sincronización y la notación NoFun [FRAN97] para las propiedades de calidad de servicio.

Los párrafos anteriores sólo describen a las interfaces como una manera de especificar el flujo unidireccional de dependencia que tiene un cliente con respecto a un componente. Sin embargo, es mejor decir que un cliente y un componente dependen el uno del otro; un cliente depende de la forma en que un componente provee sus servicios, y un componente depende de cómo los clientes utilizan los servicios que éste ofrece. Esta interdependencia ha llevado a acuñar el término Contrato de Interfaz [SAM97], [BACH00] en la literatura de investigación acerca sistemas basados en componentes.

2.9. Frameworks y Modelos de Componentes

Existe cierta confusión en la literatura referente a la terminología de modelos y *frameworks* de componentes. Sin embargo, hay consenso acerca de que los sistemas basados en componentes confían en estándares y convenciones bien

definidas (modelo de componentes) y en una infraestructura de soporte (*framework* de componentes).

Los modelos de componentes especifican las reglas de diseño que deben obedecer los componentes. Estas reglas de diseño mejoran la composición, aseguran que las calidades de servicio se logren en todo el sistema, y que los componentes se puedan desplegar fácilmente tanto en entornos de desarrollo como de producción. Los modelos de componentes imponen estándares y convenciones sobre:

- **Tipos de Componentes:** Un tipo de componente puede ser definido en términos de las interfaces que implementa. Los tipos diferentes de componentes pueden desempeñar diferentes roles en el sistema, y participar en distintos tipos de esquemas de interacción.

- **Esquemas de Interacción:** especifican cómo los componentes son localizados, cuáles protocolos de comunicación son utilizados, y cómo se satisfacen las calidades de servicio (seguridad, transacciones, alta disponibilidad). El modelo de componentes puede describir cómo interactúan los componentes entre sí y cómo interactúan dichos componentes con el *framework*.

- **Asociación (*bindings*) de recursos**: El proceso de composición de componentes es simplemente enlazar un componente a uno o más recursos. Un recurso es un servicio ofrecido por un *framework* o por otro componente desplegado en ese *framework*.

Un modelo de componentes describe cuáles recursos están disponibles a los componentes, y cómo y cuándo se asocian estos componentes a éstos recursos.

Por otra parte, los *frameworks* de componentes proporcionan servicios que soportan y hacen cumplir el modelo componentes asociados. El *framework* maneja recursos compartidos por los componentes, y proporciona mecanismos subyacentes que permiten la comunicación (interacción) entre ellos. Los *frameworks* son activos y actúan directamente sobre componentes, por ejemplo administrando su ciclo de vida (comenzar, suspender, reasumir, o terminar la ejecución de un componente), y otros recursos.

Existen muchos ejemplos de *frameworks* de componentes, entre éstos Enterprise JavaBeans (EJB) [SUN01] y Visual Basic Framework (VBF) de Microsoft [CHAP96] son de los más representativos.

La especificación de EJB define un *framework* de *servidores* y *contenedores* para dar soporte al modelo de componentes. Los *Servidores EJB* son responsables de proporcionar servicios de sistemas tales como persistencia, transacciones y seguridad, mientras que los *Contenedores EJB* son responsables de manejar el ciclo de vida del componente. Por su parte, VBF es quizás el *framework* más popular para el desarrollo de aplicaciones de escritorio. Se concentra en la composición visual de componentes (llamados VBXs) más que en tener un entorno que garantice la calidad de servicio de éstos. VBF incluye al intérprete de Visual Basic (para ejecutar scripts y hacer

composición) y el Modelo de Objetos de Componentes (COM, Component Object Model) [CDBI99] (encargado de los servicios de despliegue y comunicación).

Un mercado robusto de componentes de software requiere modelos y *frameworks* estandarizados. Sin embargo, la experiencia ha demostrado que distintos dominios de aplicación tienen diferentes requisitos de funcionalidad y calidad de servicio (latencia, seguridad y disponibilidad). Esto sugiere la necesidad de tener una variedad modelos y *frameworks* de componentes para cada dominio. EJB, CCM y .NET se han abocado al dominio de aplicaciones empresariales (ERP, BPM, e-commerce y sistemas financieros) el cual es lo suficientemente grande y coherente para definir estándares de modelos y *frameworks* de componentes. Sin embargo, existen iniciativas que están abordando otros dominios de aplicación. Las más notorias son las promovidas por OMG para el control de tráfico aéreo, análisis de secuencias bio-moleculares, mapas genómicos, infraestructura de clave pública, entre otras. Adicionalmente, WaterBeans [WALL00] y SIMyO [ARE01] son iniciativas relacionadas con tratamiento de agua, y modelado y optimización, respectivamente.

2.10. Mecanismos de Composición de Software

Bajo el modelo de desarrollo de software basado en componentes, las nuevas aplicaciones se construyen mediante la integración o composición de componentes. Sametinger [SAM97] define la composición de software como “el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces (de composición)". Nótese

que se hace especial énfasis en las interfaces como elementos fundamentales para lograr la composición de componentes.

La composición puede concebirse como una relación cliente-servidor entre dos componentes (Figura 2.2). El componente cliente solicita un servicio (operación) del componente servidor, el cual ejecuta la operación solicitada y devuelve los resultados al cliente. El servidor produce un resultado que es consumido por el cliente.

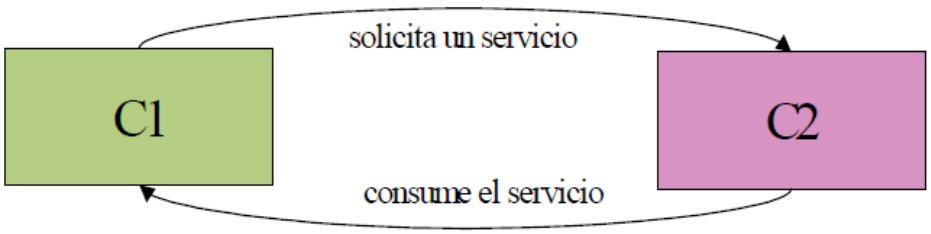


Figura 2.2. Interacción entre componentes.

- Fuente: Elaboración Propia.

Además de los componentes, los *frameworks* también se consideran entidades sujetas a composición. En consecuencia, existen tres clases principales de interacción en los sistemas basados en componentes [DIG99]:

1. Componente-Componente (C-C): permite la interacción entre componentes. De este tipo de interacción se obtiene la funcionalidad de la aplicación, de forma tal que los contratos que especifican este tipo de interacción pueden ser clasificados como Contratos a Nivel de Aplicación.

2. Componente-*Framework* (C-F): posibilita las interacciones entre el *framework* y sus componentes. Dicha interacción permite que el *framework* administre los recursos de los componentes. Los contratos que especifican estas interacciones pueden ser clasificados como Contratos a Nivel de Sistema.

3. *Framework-Framework* (F-F): posibilita las interacciones entre *framewoks* y permiten la composición de componentes desplegados en *framewoks* heterogéneos. Estos contratos pueden ser clasificados como Contratos de Interoperabilidad.

La forma de materializar la composición entre componentes depende de los mecanismos especificados por su modelo de programación.

Típicamente, los modelos de componentes se basan en tecnologías orientadas a objetos, por lo tanto los mecanismos de composición emplean algunas características tales como relaciones entre clases (especialización, agregación, asociación y uso) [JOY98], polimorfismo y enlace dinámico [BUD94].

Adicionalmente, dichos mecanismo de composición típicamente se describen mediante el uso de patrones de diseño [GAM95], [MAR02].

Las tecnologías de componentes no distribuidos, típicamente asociadas con aplicaciones de escritorio (Controles ActiveX [CHAP96] y JavaBeans [SUN97]), hacen uso extensivo de características orientadas a objetos dentro de sus mecanismos de composición. Por el contrario, en la composición de

componentes distribuidos (Enterprise JavaBeans [SUN01], CORBA Component Model [OMG02] y .NET [THAI03]) principalmente se emplean relaciones de uso, asociación y agregación.

2.11. El Proceso de Desarrollo

Las preguntas que se intentan responder en este punto son: ¿Cómo se desarrolla un componente? y ¿cómo se crea una aplicación que reutilice componentes existentes?, Sommerville**[SOM00]** clasifica los procesos de desarrollo de software basados en la reutilización de componentes en dos categorías:

a) Desarrollo de componentes: Este proceso implica la adaptación o desarrollo de componentes con el propósito expreso de ser reutilizados en futuras aplicaciones. Su objetivo es producir repositorios de activos que puedan ser reutilizados en el desarrollo de software. Para ser reutilizables, estos componentes deben satisfacer las características descritas en la sección.

b) Desarrollo de software con reutilización de componentes: Es un proceso en el cual el desarrollo de una nueva aplicación involucra la reutilización de un conjunto de componentes existentes. Este enfoque maximiza la reutilización de componentes de software existentes y reduce el número de componentes que requieren ser desarrollados en su totalidad (desde cero). Para ser exitoso, este proceso demanda dos condiciones mínimas:

i) La existencia de repositorios o bases de componentes reutilizables y

ii) que los componentes sean confiables y actúen de acuerdo a sus especificaciones.

El modelo de procesos descrito por Sametinger [SAM97] provee, sin embargo, una visión mucho más completa y amplia del desarrollo de software basado en componentes. Este modelo, denominado *ciclo de vida gemelo* (*twin life cycle*), divide el proceso de desarrollo de software en dos grandes bloques paralelos, tal como se ilustra en la Figura 2.3 El primer bloque, conocido como Ingeniería de Dominios, contempla los procesos necesarios para desarrollar activos de software reutilizables en un dominio particular. El segundo bloque es denominado Ingeniería de Aplicaciones. Su propósito es el desarrollo de aplicaciones basado en la reutilización de activos de software producidos a través de la Ingeniería de Dominios.

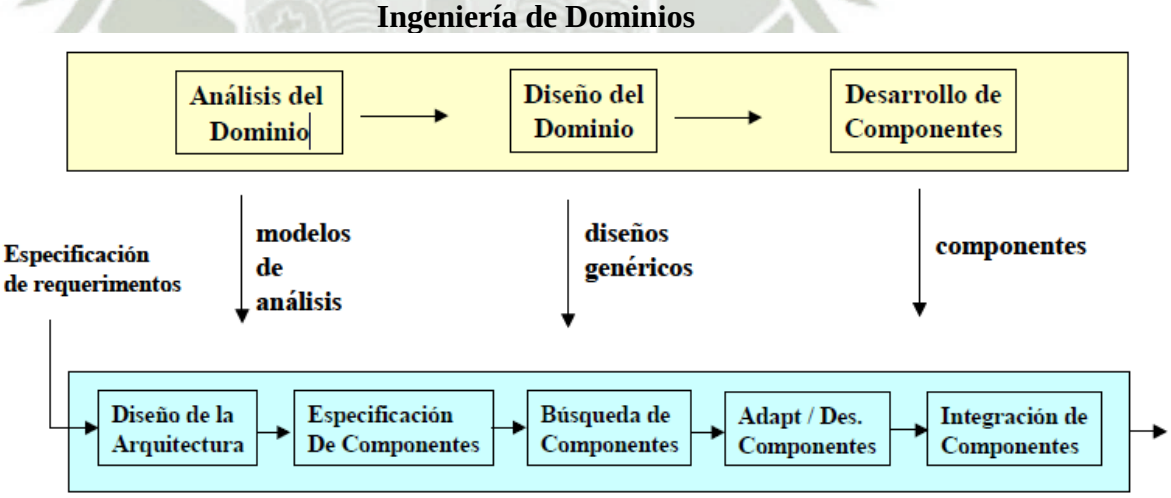


Figura 2.3 El modelo de procesos gemelos para el desarrollo de software basado en componentes.
Fuente: Elaboración Propia.

Un modelo alternativo al modelo de Ingeniería de Aplicaciones es el modelo WATCH [MON00], [MON03]. Este modelo combina los procesos más relevantes de la Ingeniería de Software Orientada a Objetos con el modelo de Ingeniería de Aplicaciones del ciclo de vida gemelo. Una característica importante de este modelo es la integración de los procesos gerenciales con los procesos técnicos del desarrollo de software basado en componentes.

Esta integración facilita la labor del líder del proyecto, al describir cómo se debe llevar a cabo una gestión del proyecto integrada a los procesos técnicos del desarrollo de software.

La estructura del método WATCH se ilustra en la figura 2.4. Esta estructura emplea la metáfora de un reloj de pulsera para describir el orden de ejecución de los procesos técnicos de desarrollo de aplicaciones, indicando además cómo los procesos gerenciales controlan o coordinan el orden de ejecución de los procesos técnicos. Los procesos gerenciales están ubicados en el centro de la estructura para indicar explícitamente que ellos programan, dirigen y controlan el proceso de desarrollo. Los procesos técnicos están ubicados en el entorno siguiendo la forma que tiene el dial de un reloj. Ello indica que el orden de ejecución de las fases técnicas se realiza en el sentido de las agujas del reloj. Los procesos gerenciales pueden, sin embargo, invertir el orden de ejecución para repetir algunas de las fases anteriores.

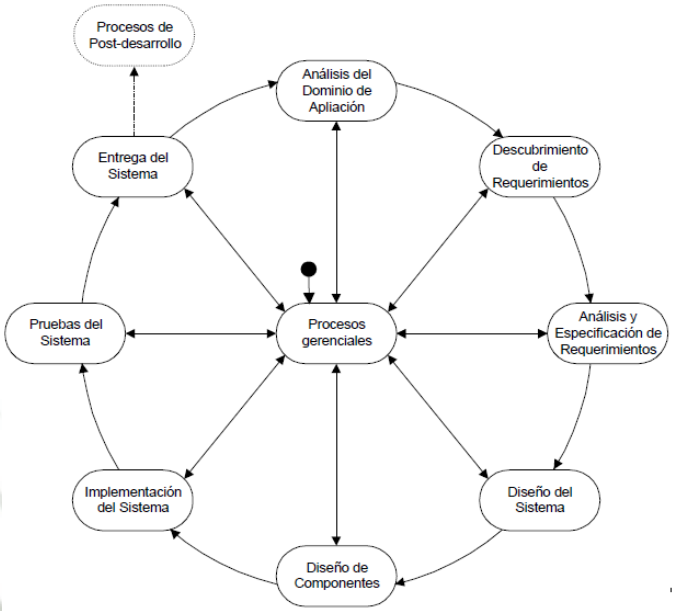


Figura 2.4. El modelo de procesos.

Fuente: WATCH [MON00], [MON03].

Las tres primeras fases del modelo son similares a los modelos de procesos tradicionales. La fase de Análisis del Contexto permite que el grupo de desarrollo adquiera un conocimiento adecuado del dominio o contexto del sistema en desarrollo. Las fases de Descubrimiento, Análisis y Especificación de Requerimientos se encargan de identificar las necesidades y requerimientos de los usuarios, así como analizarlos, especificarlos gráficamente y documentarlos.

La fase de diseño del sistema establece y describe la arquitectura del software. Describe cada uno de los componentes que requieren tal estructura y cómo esos componentes se interconectan para interactuar. El grupo de desarrollo debe, a partir de esta arquitectura, determinar cuáles componentes se pueden

reutilizar y cuáles requieren ser desarrollados en su totalidad. Los componentes reutilizados deben ser adaptados, para satisfacer los requerimientos del sistema; mientras que los componentes nuevos, deben ser diseñados, codificados y probados separadamente durante la fase de Implementación. Las Pruebas del sistema permiten detectar errores en la integración de los componentes. Finalmente, la fase de Entrega se encarga de la instalación, adiestramiento de usuarios y puesta en operación del sistema.

2.12. Tipos de Componentes

2.12.1. Framework:

Los frameworks de componentes proporcionan servicios que soportan un modelo de componentes [MON03]. Estos modelos de componentes son patrones que permiten interactuar entres si de acuerdo al problema que resuelven y permiten la extensibilidad del framework y su funcionalidad, estos son aplicados a dominios específicos, lo cual hace que el framework también lo sea, se dice que un framework es una aplicación incompleta y genérica, sobre la cual las aplicaciones que se implementan (las que soporta) son las que le dan el estado final definiendo una funcionalidad especifica y propia.

Las principales ventajas que ofrecen los frameworks son la reducción del costo en el proceso de desarrollo de aplicaciones software para dominios específicos, y la mejora de la calidad del producto final. [FUE03-B]. Existen tres tipos de Frameworks, los Frameworks de caja blanca exigen al usuario un conocimiento muy profundo de su estructura interna y pueden llevar

aparejados los problemas que acarrea la herencia, como pueden ser las anomalías de la herencia [MAT93]. Por otro lado, los Frameworks de caja negra enfrentan problemas de la programación orientada a componentes, como son la composición tardía [SZY98-A] o la clarividencia (previsión de los posibles usos futuros del Framework). Esto significa que cualquiera de los dos necesita de un entendimiento técnico del framework sobre el cual se quiere trabajar y una previa evaluación de sus características principales y su descripción. La elección de un framework es un proceso de ingeniería de software y de este depende el producto final que se quiera desarrollar. La mala elección de éste conllevará a un desarrollo complejo y fuera del domino. Es necesario integrar los conceptos de DSBC y la ingeniería de dominio para llegar a un desarrollo satisfactorio [PRES97].

Utilizar un Framework no es una tarea que se debe tomar a la ligera, es necesario la utilización y condensación de conocimientos para que este pueda ser utilizado como un componente. No es necesario conocer todos los tipos de Frameworks existentes, pero sí es necesario saber cómo estos ofrecen sus servicios. La arquitectura de los Frameworks se basa en el uso de patrones de software, los cuales simplifican su construcción y permiten su extensión [DES98], los patrones no solo se convierten en los constructores del framework sino que constituyen la manera de interactuar con este. La manera más común de desarrollar una aplicación sobre un framework es a través de los patrones, los cuales por medio de sus interfaces nos permiten interactuar con el y adaptarlo a la aplicación.

2.12.2. Bussines Component

Los componentes de negocio, son aquellos componentes especializados en prestar alguna clase de servicio, enfocado a un dominio en particular [FER02]. Los componentes de negocio son aquellos componentes que generan el valor agregado y se enfocan a las necesidades de los clientes. Un componente de negocio es aquel que realiza o provee al cliente de la funcionalidad necesaria en su aplicación para resolver sus necesidades y cumplir con sus requerimientos.

Este tipo de componentes se sitúan por lo general en el último escalafón dentro del desarrollo basado en componentes, ya que estos son los que darán la funcionalidad propia del negocio a la aplicación, estos componentes están soportados por lo general bajo un framework en particular, el cual permite que el componente de negocio se desempeñe y cumpla con su objetivo, lo cual implica que además de las características propias de un componente de software, tenga también las características adicionales del framework sobre el cual está soportado. Un componente de negocio no le da particularidad a una aplicación, ni realiza por sí solo la funcionalidad de una aplicación; es la integración entre el framework y por lo general varios componentes de negocio, los que le dan la particularidad de todo un desarrollo. Los componentes de negocio permiten interactuar con otros a través de protocolos ya estandarizados inherentes al framework sobre el cual están soportados, como por ejemplo J2EE [DES98].

Los componentes de negocio necesitan de un nivel de especificación más allá del establecido por el framework o los patrones. Los niveles de especificación de los componentes de negocio resulta útil, ya que cada nivel se enfoca a un aspecto de la especificación del negocio y direcciona diferentes roles en el desarrollo como desarrollador de componentes, documentador, jefe de proyecto, etc. Esto ayuda en la búsqueda del componente necesario que se adapte al dominio y prevea la funcionalidad que se busca, desde diferentes niveles de conocimientos.

Existen siete distintos niveles, cada uno con un aspecto en específico direccionado al negocio:

1.- Nivel de Marketing: Características de la organización y el negocio, condiciones técnicas.

2.- Nivel de tarea: Ayuda a las tareas del negocio que corresponden al dominio de la aplicación. Propósito de la aplicación.

3.- Nivel de terminología: Definición de conceptos del dominio de la aplicación.

4.- Nivel de Calidad: Criterios de calidad, procedimientos y categorías de medidas, niveles de servicio.

5.- Nivel de interacción: Secuencias de dependencias a través de servicios del mismo componente de negocio y de diferentes componentes de negocio.

6.- Nivel de Comportamiento: Pre y Pos condición, invariantes.

7.- Nivel de Interfaces: Servicios, parámetros, tipos de datos y excepciones. [FER02].

Por medio de estos niveles la búsqueda de un componente de negocio resulta más rápida y fácil, acelerando el proceso de DSBC.

2.13. Tecnologías de componentes estandarizadas:

Los componentes son unidades bien definidas, que adquirieron sus características principales de sus orígenes en el desarrollo de software de la POO y el UML. Aunque la infraestructura tecnológica del DSBC se ha desarrollado, son pocas las tecnologías y productores de software que se han estandarizado [BRO98]. Estas tecnologías han enriquecido el DSBC y, de hecho, hacen posible que nazca el concepto. Aunque la investigación en el área del DSBC es relativamente nueva, existen día a día investigaciones que tratan de unificar criterios y estandarizar las tecnologías, que permitan unificar criterios y madurar el DSBC. Una de las principales organizaciones y pionera desde hace mucho tiempo en el tema de componentes es la ORGANIZACIÓN OMG, (Object Management Group) esta es una organización que agrupa a más de 400 empresas entre vendedoras de software y compañías relacionadas con la tecnología de objetos. Su bandera es la especificación multiplataforma MDA, y basada en modelos de especificación como MOF, UML, XML y CWM, esta cuenta con su propia plataforma middleware CORBA. [OMG04]. Su importancia radica en ser una organización sin ánimo de lucro que busca la unificación de criterios que ayuden al DSBC. Esta se constituye como un organismo centralizado que ayuda al arbitramiento de ideas en este tema.

2.13.1. CORBA

Es un estándar abierto para la aplicación de la interoperabilidad y está respaldada por la OMG. CORBA maneja al detalle la interoperabilidad entre componentes y permite la comunicación entre aplicaciones independientemente de su ubicación y diseño. CORBA es bien utilizado en el desarrollo de aplicaciones distribuidas y en el DSBC, ya que ofrece una programación consistentemente distribuida y un ambiente en tiempo de ejecución para muchos lenguajes de programación, sistemas operativos y redes distribuidas [XIA02]. CORBA es aceptado casi como un estándar por los vendedores de tecnología y no como una plataforma independiente.

2.13.2. COM

Uno de los software de componentes con más experiencia es COM (Component Object Model), una arquitectura independiente del lenguaje desarrollada por Microsoft en los años noventa, el cual permite a los desarrolladores crear aplicaciones a partir de componentes de software binario reutilizables, y en un mundo de redes interconectadas y de servicios web, las cargas de proceso son cada vez más divididas entre clientes y servidores separados los unos de los otros, tanto en distancia como en funcionalidad, esto significa que es esencial que los ordenadores sean capaces de comunicarse entre sí, utilizando como información datos de entrada generados por otros ordenadores, esto no se trata algo nuevo ya que no es más que la consecuencia de cómo se vienen desarrollando las aplicaciones durante los últimos años.

COM define un interfaz de programación de aplicación (API) que permite a diversos componentes interactuar entre sí. Mientras utilizasen una infraestructura binaria especificada por Microsoft, podían ser escritos en diferentes lenguajes y aún así interoperar unos con otros una vez que hubieran sido compilados.

COM permite el desarrollo de servicios de aplicación utilizando documentos combinados, controles a medida, scripting interaplicaciones, transferencia de datos y otros tipos de interacciones de software.

2.13.3. DCOM

Distributed COM es la extensión para ambientes distribuidos de COM (Component Object Model), el cual provee de un protocolo de comunicación con el cual componentes distribuidos pueden interactuar y comunicarse independientemente de su ubicación [XIA02]. El aporte de DCOM a las tecnologías Microsoft se ve en su más reciente producto MS.NET.

2.13.4. Enterprise Java Beans

Modelo de componentes basado en arquitectura cliente servidor. Esta plataforma ofrece una solución multiplataforma, de fácil reutilización, integración universal con otros componentes además de la maquina virtual de java, seguridad transaccional, entre otras. Estas características la han hecho reconocida en el mundo de los programadores y del DSBC [XIA02]. La

tecnología java ha estado a la vanguardia del DSBC y constituye una referencia clave en este tema.

2.13.5. Microsoft .NET Framework:

Esta es una nueva plataforma para construir e integrar, servicios orientados a la aplicación. Este framework es la solución para aplicaciones que operan bajo un ambiente Windows simplificando el lenguaje de desarrollo y accediendo en tiempo de ejecución a las funcionalidades del framework. [RIC02]. La infraestructura Microsoft es una de las más reconocidas en el mundo y MS.NET constituye una oportunidad de nuevas soluciones a partir de DSBC.

2.14. Sistemas de Monitoreo

- Es un proceso permanente para verificar sistemáticamente que las actividades o procesos planificados se llevan a cabo según lo esperado.

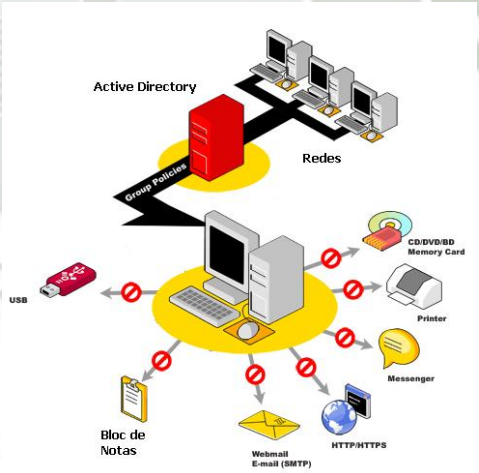


Figura 2.5 – Bloqueo de dispositivos de entrada externos.

Fuente: Elaboración Propia.

2.14.1. Las Metas del Monitoreo

- Producir alertas cuando se produzca algo que consideremos importante, como por ejemplo el MSN Messenger muestra alertas en el momento que un usuario aparece como conectado.
- Determinar la ubicación y el nombre de la computadora.
- Brindar la información necesaria al administrador o al encargado.
- Determinar qué realizaron los usuarios cuando estuvieron frente a uno de los computadores.

2.14.2. ¿Quién debe realizar el monitoreo y la evaluación?

A. Monitoreo formal interno y externo

1. Monitoreo Interno (conducido por los actores locales).-

   En el monitoreo interno deberían participar interesados de estrategia locales; pero pueden surgir conflictos de intereses.

2. Monitoreo y Evaluación Externo (conducida por organismos autónomos reconocidos).- Una opinión imparcial y un análisis realizado por expertos independientes pueden hacer una contribución significativa.
3. Vinculando el Monitoreo Interno y Externo (Las evaluaciones independientes debieran complementarse con las internas). Es probable que las evaluaciones autónomas sean más valiosas si son acordadas y comisionadas por los múltiples interesados en la estrategia, y si son construidas en base a evaluaciones internas, en lugar de que éstas sean impuestas desde afuera.

B. Monitoreo y Evaluación Participativos (MEP)

La expresión “monitoreo y evaluación participativos” es útil para describir un amplio rango de prácticas. Para el propósito de las estrategias de desarrollo sostenible, el MEP debe ser interpretado como métodos de monitoreo que desarrollan alianzas entre múltiples interesados para un monitoreo eficiente y efectivo.

2.14.3. ¿Cuándo debe llevarse a cabo el monitoreo y la evaluación?

La evaluación debe comenzar desde el principio del proceso de la estrategia, para establecer una línea de base. Pero, como el monitoreo y la evaluación forman parte de un método de mejoramiento continuo para la toma de decisiones, estos deben ser actividades regulares e integradas en lugar de ser eventos esporádicos y separados. El beneficio de una evaluación regular es que alienta a los participantes a repensar las prioridades, reorganizar objetivos y reprogramar su curso de acción.

2.14.4. Tipos De Monitoreo

A. Monitoreo de la Red: El cual detecta problemas de rendimiento de la red antes de que supongan costosos tiempos de inactividad.

B. Monitoreo de Servidores: Mejora la disponibilidad y el rendimiento de su infraestructura de servidores.

C. Monitoreo de Aplicaciones: Identifica problemas de rendimiento de las Aplicaciones antes de que tengan impacto en los usuarios finales.

El proyecto que se realiza en el presente trabajo de investigación se enfoca a los siguientes tipos de monitoreo:

- **Monitoreo del Uso:** Aquí se le informa al administrador lo que los usuarios pueden haber realizado luego de haber accesado al sistema, pudiendo ser actividades que los usuarios no estaban permitidos a realizar, identificados durante dicho monitoreo, e informados a la autoridad oficial pertinente, usualmente, un administrador de red. En el caso de que el usuario no concuerde con el monitoreo, se pueden tomar decisiones como que un usuario no pueda acceder más al sistema.
- **Monitoreo de Usuarios:** De vez en cuando habrá ocasiones donde un administrador querrá saber exactamente lo que los usuarios hacen en su sistema. Un monitor de usuarios es un programa que realiza un control oculto sobre las acciones que los usuarios realizan en determinadas computadoras.

2.14.5. Características de los Sistemas de Monitoreo

A. Análisis de las necesidades actuales de seguridad y monitoreo: Es el mismo para todas las compañías, es decir, la salvaguarda permanente de la información. En términos generales, la orientación de la meta de este tipo de proyectos es preventiva o correctiva, por lo que las acciones que se llevaran a cabo tendrán diferentes niveles de espera, procedimientos específicos y soluciones diversas.

B. Determinación de nivel de seguridad requerido: En este punto es esencial partir de la elaboración de un mapa de riesgos para establecer el nivel real la problemática actual y la naturaleza y avance del riesgo en la organización.

C. Estudio de vulnerabilidades y amenazas dentro de la empresa: Cada empresa cuenta con su infraestructura IT, diferente de otras, por lo que el compendio de amenazas a su seguridad cambia considerablemente. No obstante, existen algunos puntos en común en la problemática actual de la seguridad informática.

D. Definición de políticas de seguridad e implementación de un sistema de monitoreo: Poner en marcha de inmediato, las debidas acciones y políticas que permitan garantizar que los recursos informáticos de una compañía estén disponibles para cumplir sus propósitos.

2.14.6. Principales Beneficios del Monitoreo

- Cuida la información 7 días 24 horas al día.
- Expertos están vigilando la información todo el tiempo.
- Permite identificar e incluso, bloquear incidentes potenciales.
- Ofrece las mejores recomendaciones y soporte para los incidentes.
- Ayuda en la determinación de políticas de seguridad y requisitos reguladores.

2.15. XML

Según [WIK11c], XML Extensible Markup Language (Lenguaje de Marcas Extensible), es un metalenguaje extensible de etiquetas desarrollado por el

World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

2.15.1. Validez de un Documento XML

Según [WIK11c], cada aplicación de XML, es decir cada lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento. Esta relación entre elementos se especifica en un documento externo o definición (expresada como DTD (Document Type Definition) o como XSchema).

Definición de Tipo de Documento (DTD): Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD.

XML Schema Definition (XSD): Es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa. Se consigue así, una percepción del tipo de documento con un nivel alto de abstracción.

2.16. Web Service (Servicios Web)

Según [WIK11a], un Web Service es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferente y ejecutada sobre cualquier plataforma pueden utilizar los Web Services para intercambiar datos en redes de ordenadores como Internet.

Según [ATS11], un Web Service es un programa accesible a través de la red que expone su funcionalidad a través de mensajes SOAP sobre Https publicando sus características mediante WSDL.

2.17. Arquitectura Orientada a Servicios (SOA)

Según [WIK11b], La Arquitectura Orientada a Servicios (Service-Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario. En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Web Services (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

2.18. WCF (Windows Communication Foundation)

Windows Communication Foundation (WCF), es parte del .NET Framework (Incluido en las versiones 3.0 y 3.5), es una plataforma que los desarrolladores pueden usar para construir aplicaciones orientadas al servicio. El término orientado a las aplicaciones de servicio es una composición de una colección de servicios que se comunican solo para intercambiar mensajes.

2.18.1. Configuración de Servicio WCF

Todas las estructuras WCF tienen un contracto el cual es consumido para definir los siguientes tres núcleos del contrato.

- **Contrato del servicio (Service contract).-** El cual define que operaciones del servicio están disponibles para ser invocados mediante mensajes de requerimientos que son enviados al servicio desde el cliente.
- **Contrato de data (Data contract).-** Define la estructura de data incluido en los payloads de los flujos de mensajes dentro y fuera del servicio.
- **Contrato del mensaje (Message contract).-** Permite controlar las cabeceras que aparecen en los mensajes y como los mensajes son estructurados.

Los tres tipos de núcleo de los contractos WCF mapean directamente para corresponder estándares de Web Services en la siguiente forma:

- Contrato de Servicio, mapea a un WSDL.

- Contrato de Data, mapea a un XSD.
- Contrato de Mensajes, mapea al protocolo SOAP.

Estos mecanismos de definición de contratos residen en un namespace (espacio de nombres) llamado System.ServiceModel, Microsoft como parte de su visión de modelado y Dominio de lenguajes específicos (DSLs) mira estas definiciones de contrato de lenguajes como modelos basados en .NET que pueden ser usados como alternativas a sus contrapartes basados en estándares.

CAPITULO III: DESARROLLO DEL COMPONENTE

3.1 Selección de Tecnologías

3.1.1 Herramientas de Desarrollo

Alternativas:

- Lenguaje de Programación Desktop C# 3.0 (Plataforma .Net).
- Lenguaje scripting PHP (Plataforma Web).

La selección se redujo solamente a dos opciones debido a la experiencia del desarrollador con estos lenguajes. La tecnología escogida es la plataforma .NET utilizando el lenguaje de programación C# 3.0, debido a las ventajas que presenta sobre scripting PHP:

- Las aplicaciones hechas en C# .NET no se leen secuencialmente como en PHP que es un lenguaje interpretado línea a línea, sino que se compilan, logrando un incremento de velocidad de respuesta del servidor (en sucesivas invocaciones).
- Otro aspecto a tomar en cuenta es el framework con que cuenta la plataforma .Net ya que tiene una extraordinaria compatibilidad con XML y los Servicios Web ya que viene con clases que son de gran utilidad para el manejo de estas tecnologías estándares. En PHP se debe recurrir a librerías adicionales para la implementación y consumo de Servicios Web, y el manejo de archivos XML presenta todavía algunos errores cuando el manejo de información es de manera intensiva.

3.1.2 Plataforma

Partiendo de la selección de C# como herramienta de desarrollo, se tiene dos alternativas para la plataforma a emplearse:

Lenguaje de Programación	Framework	Sistema Operativo
C# .NET	Net Framework 2.0	Windows XP SP2
C# .NET	Net Framework 3.0	Windows 7 SP1
C# .NET	Mono Framework 1.1 + XSP2/Apache 2.X con mod Mono 1.9.1	Linux Ubuntu 10.0

Tabla 3.1 Cuadro Comparativo de posibles tecnologías a usar.

Fuente: Elaboración Propia.

En este punto la experiencia del desarrollador fue determinante al escoger la segunda opción como plataforma. A esto se suma el hecho de que algunos métodos que tiene el Framework .NET 3.0 y son los candidatos de uso, no han sido implementados aún en Mono 1.9.1 y en el Framework 2.0.

3.1.3 Repositorios de Datos

Se partió en la utilización de dos enfoques de almacenamiento de datos.

- Documentos XML (para archivos de configuración).
- Modelo de datos Relacional (para datos de usuarios).

La elección de documentos XML y la utilización de un modelo de datos relacional fueron determinadas por lo siguiente:

- El **estándar XML** ofrece esquemas de validación (DTD y XSD), lenguajes de consulta (XPATH, XQUERY, etc.) e interfaces de programación (DOM, JDOM, etc.). Todos estos aspectos se encuentran soportados por el Framework .NET 3.0, además contamos con la flexibilidad y portabilidad de los documentos XML.
- La utilización de servicios web para transmitir datos (en formato XML) entre programas y servicios.
- En cuanto al **modelo de datos relacional** nos permite tener un mejor control de nuestros datos ya que tenemos el manejo de índices, secuencias, procedimientos almacenados, seguridad, transacciones e integridad referencial de datos, en este aspecto nos inclinamos por el uso del gestor de base de datos libre PostgreSQL 8.2 (basándonos en la experiencia del programador).

3.1.4 Servicios Web (Web Services)

La elección natural fue la de los Servicios Web, ya que son un estándar para la comunicación aprobado por OASIS y el W3C.

3.1.5 Windows Communication Foundation (WCF)

Es una plataforma que los desarrolladores pueden usar para construir aplicaciones orientadas al servicio. El término orientado a las aplicaciones de servicio es una composición de una colección de servicios que se comunican solo para intercambiar mensajes.

3.1.6 Fat Clients y Thin Clients

Los posibles tipos de clientes a emplear:

- Thin Client.
- Fat Client.

Un **Thin Client** es una aplicación accesible a través de un Web Browser, donde la lógica se ubica del lado de servidor, esto será usado para los web services.

Un **Fat Client** es una aplicación residentes en el equipo, se conocen generalmente con el nombre de aplicaciones de escritorio (desktop), este será usado para crear el componente y el modulo de administración del mismo. El objetivo de integrar ambos clientes es para permitir la interacción de los mismos aprovechando el uso de los servicios de Windows.

3.1.7 Selección de Patrón de Diseño y Arquitectura

Se emplea el **patrón MVC (modelo-vista-controlador)** por ser un patrón ampliamente extendido y aplicado en el desarrollo de aplicaciones Web y componentes. Esta propuesta de componente parte del concepto de tener participantes (pudiendo estar dispersos geográficamente) que exponen Web Services, los que a su vez encapsulan cierta funcionalidad limitada y que al ser integrados ofrecen una funcionalidad mayor al usuario final. Teniendo en cuenta este concepto la que mejor se ajusta es la **Arquitectura Orientada a Servicios (SOA).**

3.2 Arquitectura Propuesta

Se propone la siguiente arquitectura basada en el framework 3.0 de .net.

Figura 3.1: Arquitectura Propuesta para el proyecto.
Fuente: Elaboración Propia.

3.3 Análisis del Dominio

3.3.1 Dominio

El dominio de esta propuesta está en el desarrollo de un componente de software de seguridad para el monitoreo y control de información en un equipo informático bajo la plataforma COM+ a través de Web Services usando Fat y Thin Clients, aplicando el patrón MVC bajo la arquitectura SOA.

3.3.2 Requerimientos Generales

- **Manual de Usuario para el implementador**: El manual contiene los pasos necesarios para administrar el componente aplicando la propuesta, además de cómo definir los archivos de configuración utilizados, así como otros empleados por el componente para su funcionamiento.
- **Utilización de Estándares:**
  - **XML:** Usado para el almacenamiento de datos de los documentos de trabajo.
  - **HTTP:** Protocolo usado en cada transacción sobre Web.
  - **SOAP:** Protocolo utilizado en los Web Services.
  - **WSDL:** Describe la interfaz publica a los Web Services.
- **Registro cronológico de las actividades:** Todas las actividades realizadas creadas bajo esta propuesta de componente se registran en el log de eventos de Windows. Cada archivo registra la fecha y la hora de cada incidencia además de datos adicionales relacionados a cada actividad.

3.4 Análisis y Diseño

En este capítulo se explicará la metodología a utilizar para el análisis y diseño del sistema, los **requerimientos funcionales y no funcionales** del sistema y se presentarán los diagramas de casos de usos, diagramas de clases, diagramas de secuencias, diagramas de paquetes y diagramas de componentes, finalmente se analizará el diseño del componente paso a paso.

3.4.1 Metodología a Utilizar

Durante el desarrollo del software propuesto en este tema de tesis se siguió una metodología orientada a objetos basada en el Proceso Unificado de Desarrollo de Software (RUP).

[LAR05] RUP es una metodología de desarrollo de software iterativo y cuyo proceso de modelamiento está basado en UML (Unified Modeling Language) que es una notación de modelamiento estándar.

[RUM99] RUP tiene dos estructuras o dimensiones. Estructura dinámica o dimensión tiempo del proceso muestran como el proceso se expresa en términos de ciclos, fases, iteraciones e hitos que se despliegan a través del ciclo de vida del proyecto. La estructura dinámica del proyecto en cuatro fases:

1. **Incepción:** Establece un claro conocimiento del proyecto a realizar así como de los requisitos y se establece el alcance del proyecto.
2. **Elaboración:** Se preocupa de las tareas técnicas como diseño, implementación, pruebas y la arquitectura.

3. **Construcción:** Hace más que la implementación, pasa de una arquitectura a la primera versión operacional del sistema.
4. **Transición:** Se asegura que el sistema cubra las necesidades de los usuarios.

En este punto del ciclo de vida se enfoca en el funcionamiento del producto, la configuración y la instalación. Estructura estática describe cómo los elementos del proceso, actividades, disciplinas, artefactos y roles, son lógicamente agrupados dentro del flujo básico de trabajo.

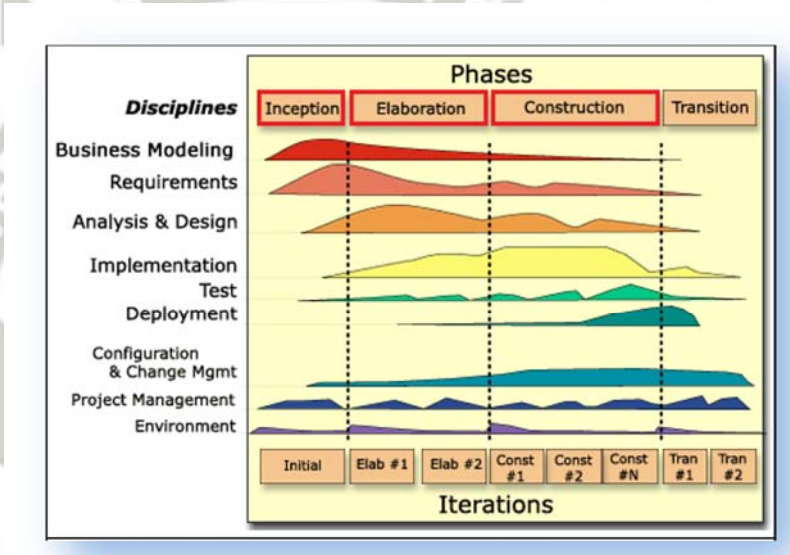


Figura 3.2: Ciclo de vida de un proyecto.
Fuente: http://www.consolida-it.com/marco.htm.

[LAR05] Artefacto es un producto final o parcial que es producido y utilizado durante el proyecto. Un artefacto puede ser un documento, un modelo o un elemento del modelo.

Está metodología fue seleccionada para el desarrollo de la solución propuesta por los siguientes motivos:

- Permite desarrollar un marco de trabajo para el desarrollo exitoso de grandes proyectos de software.
- RUP promueve el desarrollo de software iterativo y se adecua al tipo de proyecto propuesto.

RUP (Rational Unified Process) consta de 4 fases: Concepción, Elaboración, Construcción, y Transición; además consta de 9 disciplinas: Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Test, Instalación, Administración de Configuración y Cambio, Administración de Proyecto y Ambiente. Sin embargo el presente proyecto solo abarcará hasta la tercera fase, pues en el alcance no se ha definido procesos de implantación del sistema a desarrollarse, es por esto que se seguirá la metodología RUP sólo hasta la fase de Construcción.

3.4.2 Identificación de requerimientos

A continuación se hará una descripción de los usuarios y los requerimientos funcionales y no funcionales del sistema.

3.4.2.1 Descripción de los usuarios

3.4.2.1.1 Perfiles de usuario

Los usuarios del componente se clasificarán en 2 perfiles:

- **Administrador.-** Será el encargado de dar la configuración y mantenimiento al sistema.
- **Usuario del Sistema.-** Será el usuario final.

3.4.2.2 Requerimientos funcionales y no funcionales

En este punto veremos los requerimientos funcionales y no funcionales que tendrá el componente que se caracterizara principalmente por ser orientado a objetos gracias a que estará hecho en el lenguaje C#, además de estar desarrollado bajo la arquitectura Modelo-Vista-Controlador.

3.4.2.2.1 Requerimientos Funcionales

RF01	Tener un conjunto de clases y librerías el en núcleo (Core) que permitan controlar nuestro componente de seguridad.
RF02	Dar mantenimiento al sistema que se creara en base al Prototipo Propuesto.
RF03	Tener una validación de usuarios (Login) y también un control de usuarios del sistema para acceso al modulo de Administración.
RF04	Permitir controlar los Servicios asociados al Componente mediante un Administrador de Servicios de Windows.
RF05	Validar accesos a nivel de usuarios, teniendo en cuenta el control y validación de aplicaciones a ejecutar.
RF06	Independencia en relación al SGBD (Sistema Gestor de Base de Datos) siendo transparente e independiente haciendo uso de Dlls (librería de enlace dinámico) de conexión nativas y un componente de Base de Datos.
RF07	Independencia entre Sistemas Operativos compatibles con la Plataforma.

Tabla 3.2 Requisitos Funcionales Componente.
Fuente: Elaboración Propia.

3.4.2.2.2 Requerimientos No-Funcionales

RNF01	Rendimiento
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	La comunicación con los clientes, se realizara mediante protocolos con WCF basados en estándares que permitan cumplir los demás requisitos del sistema.
Comentarios	Ninguno.

Tabla 3.3 Requisitos No-Funcionales Del Componente I.
Fuente: Elaboración Propia.

RNF02	Seguridad de acceso por código y por Base de Datos
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	Se especifican elementos que protegerán al software de accesos, solo para personas autorizadas, y se restringirá l acceso a la información.
Comentarios	Ninguno.

Tabla 3.4 Requisitos No-Funcionales Del Componente II.
Fuente: Elaboración Propia.

RNF03	Portabilidad a nivel de Plataforma Compatible
Objetivos asociados	Ninguno
Requisitos asociados	Tener instalada la versión del framework .net
Descripción	El sistema deberá ser fácilmente portable y trabajara en cualquier sistema operativo que contenga un sistema operativo Windows.
Comentarios	Ninguno.

Tabla 3.5 Requisitos No-Funcionales del Componente III.
Fuente: Elaboración Propia.

3.4.3 Análisis de la solución

3.4.3.1 Casos de uso

[RUM99] Los casos de uso son una manera formal de capturar y expresar la interacción y el diálogo entre los usuarios del sistema, llamados actores, y el propio sistema. [CON03] Los casos de uso expresan lo que el sistema debería hacer, sin preocuparse en el cómo lo hará.

3.4.3.2 Actores

[KEN01] Un actor representa un rol de un usuario que interactúa con el sistema. A continuación se describe a los actores del sistema propuesto:

ACT-1	**Usuario**
Descripción:	Ese actor se refiere a los usuarios del componente que serán los encargados de administrar los servicios asociados al componente de seguridad instanciándolo.
Versión:	Versión 1.0
Observación/Comentarios	Ninguno

Tabla 3.6 Definición de autores – Actor usuario.
Fuente: Elaboración propia.

ACT-2	**Sistema**
Descripción:	Ese actor se refiere al Componente a desarrollar, el cual recibirá las salidas de los procesos iniciados por **ACT-1** y enviara las respuestas a las solicitudes iniciadas por **ACT-1.**
Versión:	Versión 1.0
Observación/Comentarios	Ninguno

Tabla 3.7 Definición de autores – Actor usuario.
Fuente: Elaboración propia.

3.4.3.3 Dominio del Componente

El Dominio del componente está basado en la seguridad de sistemas para PC usando la arquitectura patrón Modelo-Vista-Controlador y algunas herramientas de Software Libre.

3.4.3.4 Componente de Seguridad

OBJ-01	Reutilizamos el componente BD
Descripción	Tener la capacidad de utilizar componentes COM+
Estabilidad	Alta
Comentarios	Se reutilizan el Componente de BD para la aplicación.

Tabla 3.8 Objetivo 1 del Componente.
Fuente: Elaboración Propia.

OBJ-02	Uso de librerías de enlace dinámico(Dlls) y arquitectura MVC
Descripción	Se debe contar con una arquitectura que soporte perfectamente el patrón de arquitectura Modelo-Vista-Controlador haciendo el uso de las características del lenguaje de programación seleccionado.
Estabilidad	Alta
Comentarios	Ninguno

Tabla 3.9 Objetivo 2 del Componente.
Fuente: Elaboración Propia.

OBJ-03	Api de Windows
Descripción	Se debe de permitir el uso de Librerías de todo tipo, siempre y cuando sean de uso libre.
Estabilidad	Alta
Comentarios	Ninguno

Tabla 3.10 Objetivo 3 Del Componente.
Fuente: Elaboración Propia.

3.4.4 Diagramas de Casos de Uso

3.4.4.1 CDU-01 Caso de Uso Principal

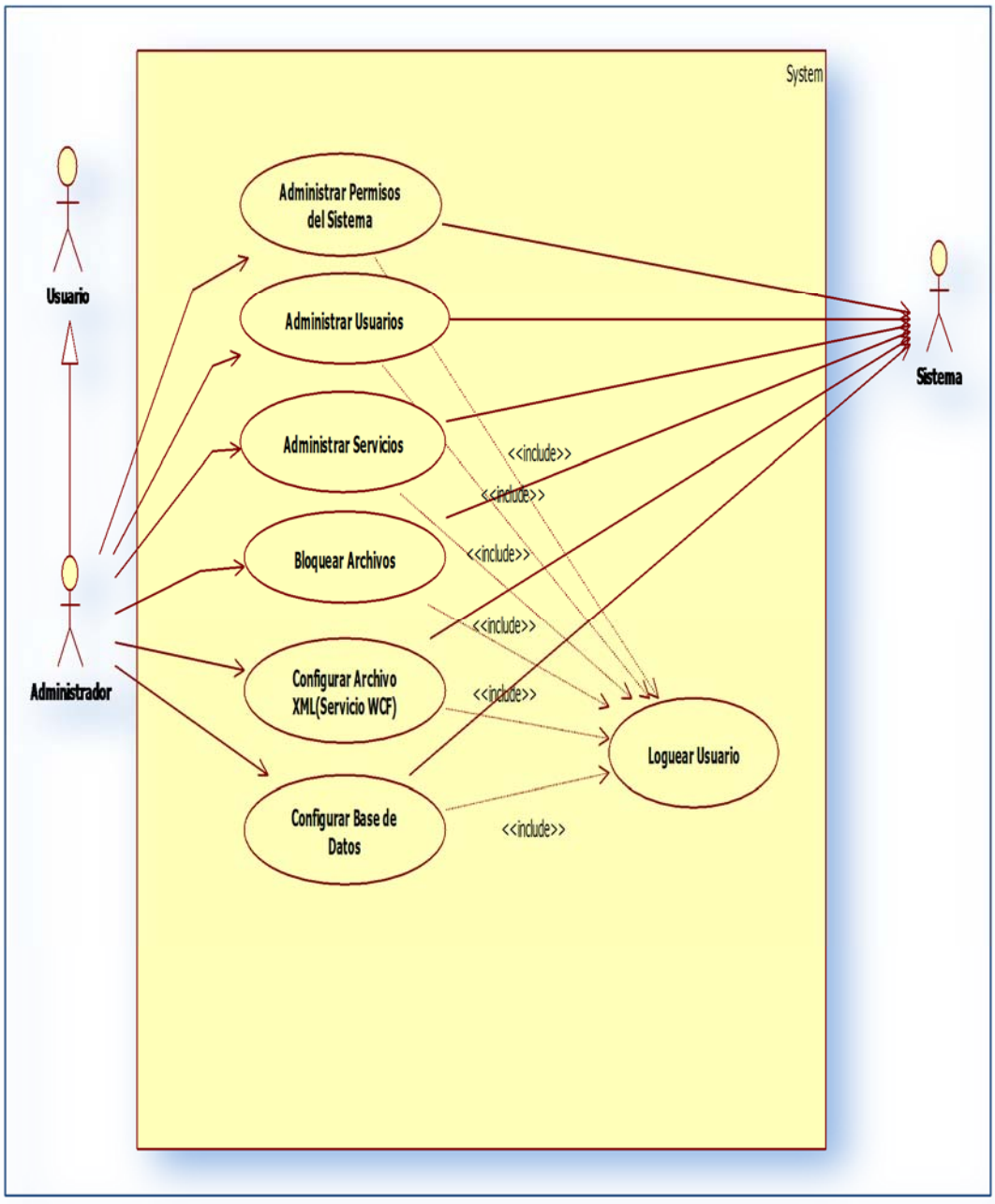


Figura 3.3 – Diagramas de Casos de Uso CDU- 01-Principal.

Fuente: Elaboración Propia.

3.4.4.2 CDU-02- Administrar Servicios

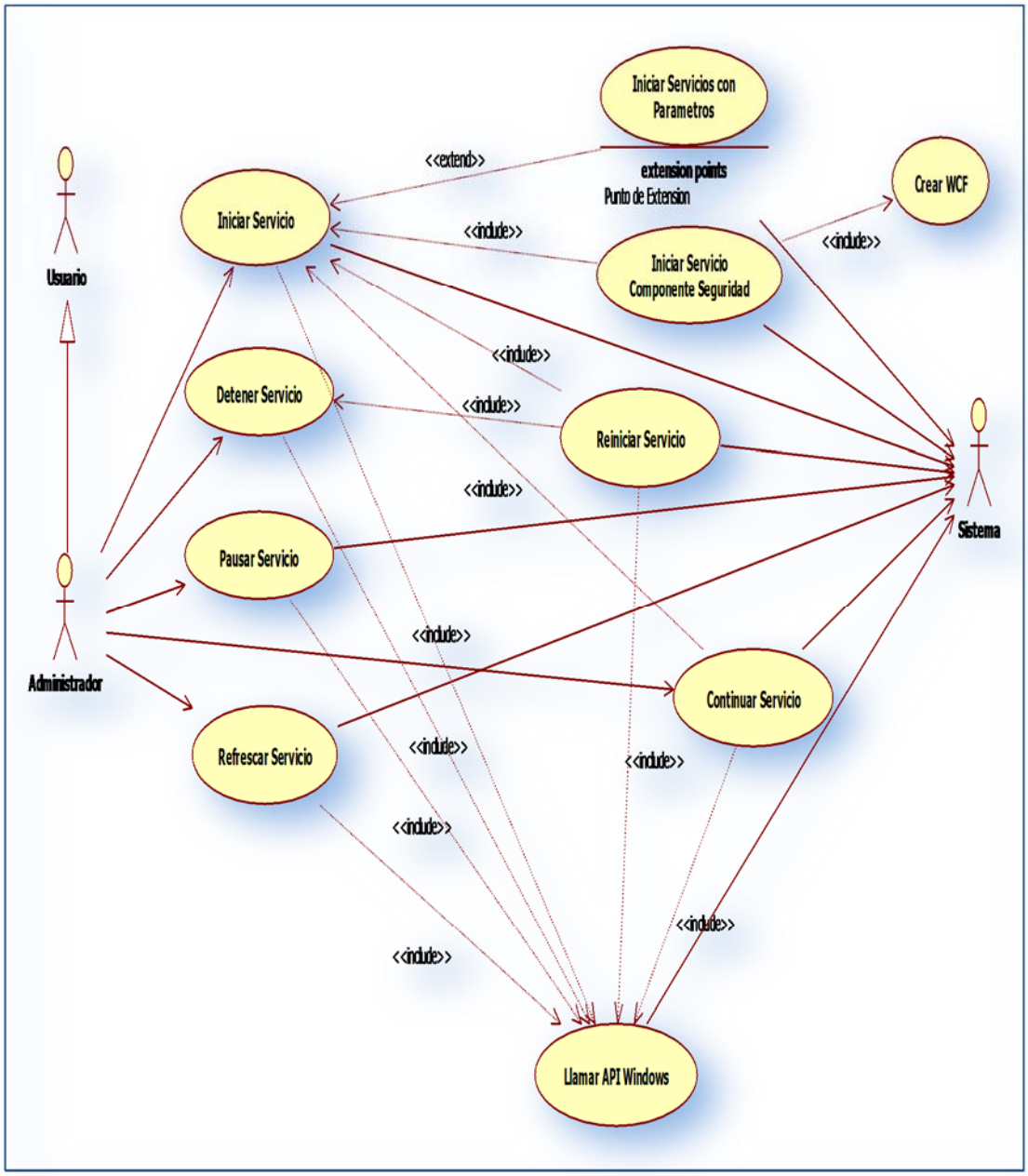


Figura 3.4 – Diagramas de Casos de Uso CDU-02-Administrar Servicios.

Fuente: Elaboración Propia.

3.4.4.3 CDU-03 –Administrar Usuarios

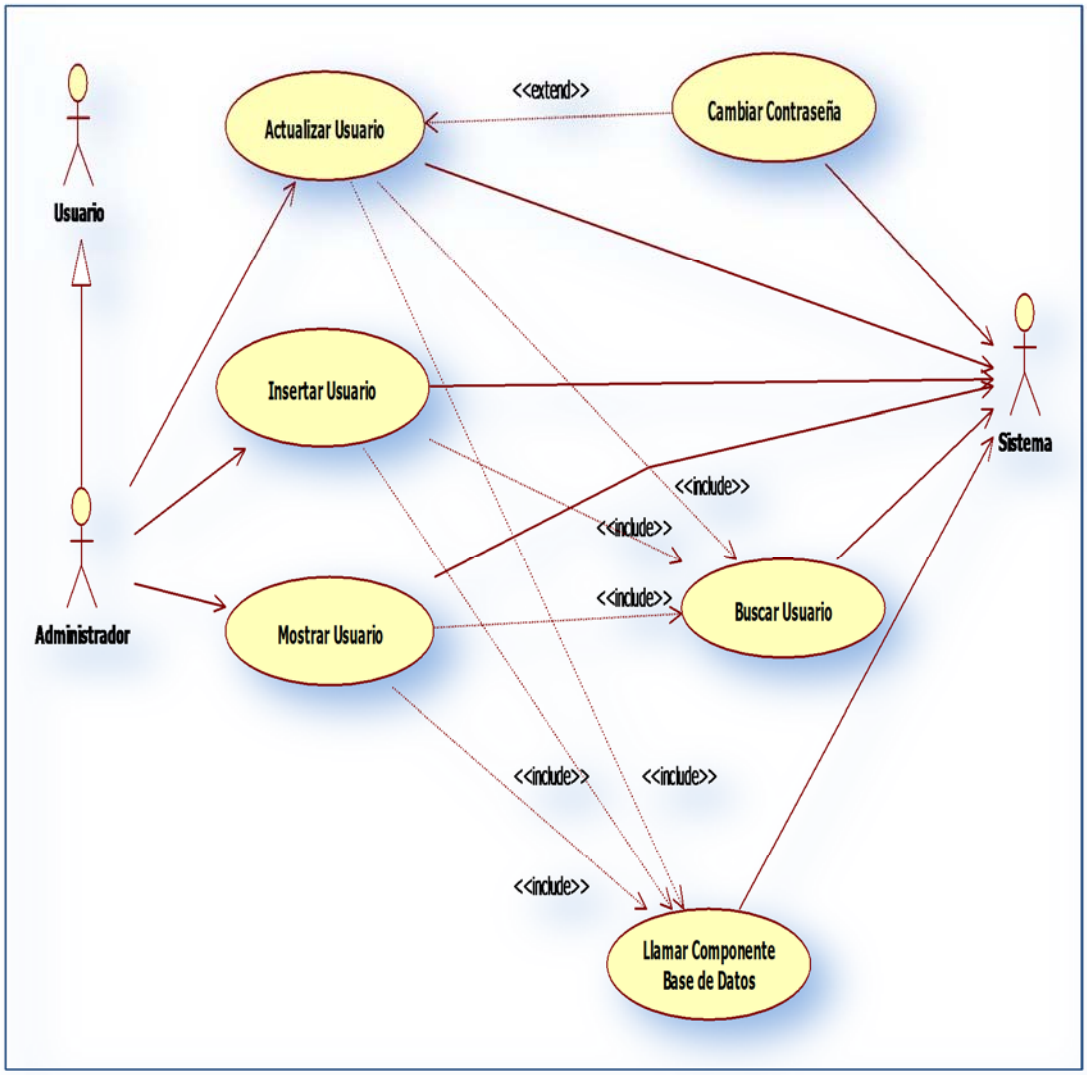


Figura 3.5 – Diagramas de Casos de Uso CDU-03-Administrar Usuarios.

Fuente: Elaboración Propia.

3.4.4.4 CDU-04 –Bloquear Archivos

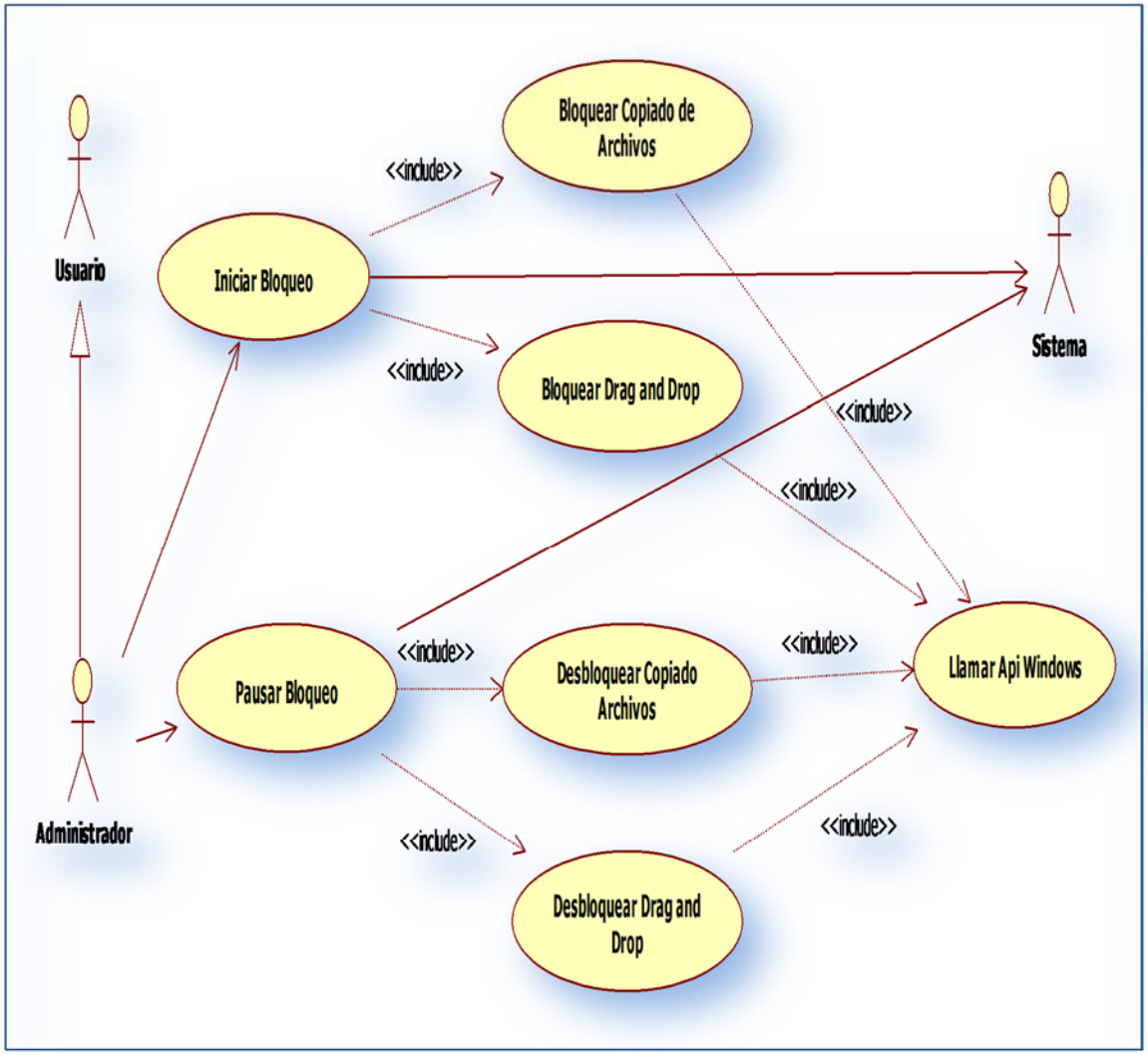


Figura 3.6 Diagramas de Casos de Uso CDU-04-Bloquear Archivos.

Fuente: Elaboración Propia.

3.4.4.5 CDU-05 – Configuración INI (Base de Datos)

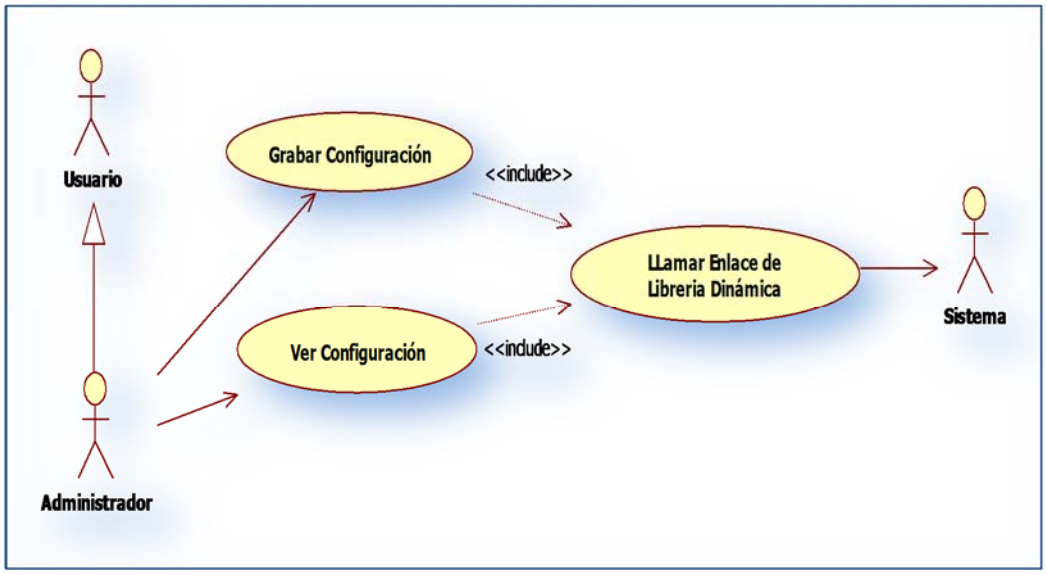


Figura 3.7 – Diagramas de Caso de Uso CDU-05-Configuracion INI (Base De Datos).

Fuente: Elaboración Propia.

3.4.4.6 CDU-06-Configuración XML (Servicio WCF)

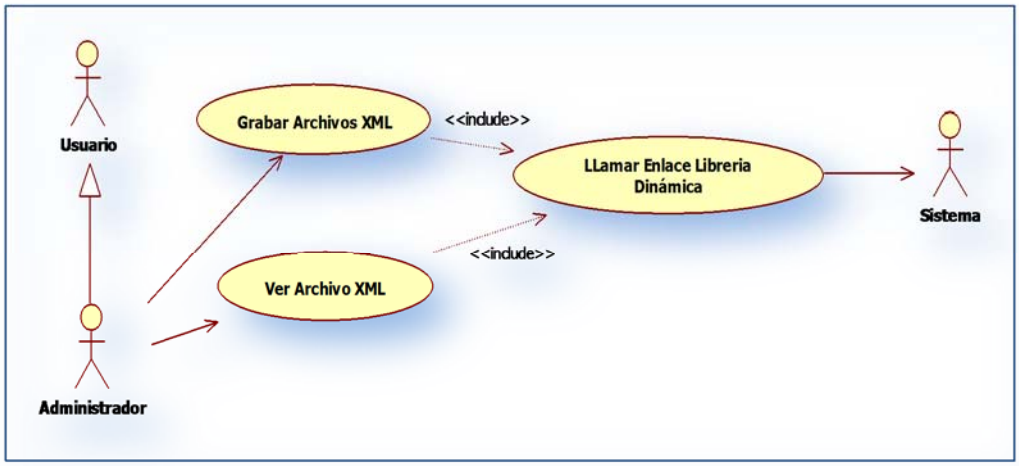


Figura 3.8 -Diagramas de Caso de Uso CDU-06-Configuracion XML (Servicio WCF).
Fuente: Elaboración Propia.

3.4.4.7 Especificación de Casos de Uso

3.4.4.7.1 Administrar Servicios

3.4.4.7.1.1 Descripción

Casos de Uso	**Administrar Servicios.**	**[CDU-02]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para administrar los servicios del sistema.	
Tipo de Caso de Uso	Primario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y encontrarse en la pantalla principal de administración.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
1.- El usuario hace click en el botón Refrescar Servicios. 3.- Buscamos el servicio que queremos iniciar y damos click en el botón iniciar.	2.- El Sistema muestra en el formulario principal todos los servicios Windows del Sistema dentro de una lista. 4.- El Sistema inicia el servicio que hemos especificado.	
Flujos Alternativos		
Acción: del Actor(es)	**Acción: del Sistema**	
3.1.- El usuario da click en iniciar servicio y no ocurre nada.	3.2.- El Sistema no puede iniciar el servicio porque este no se encuentra instalado o no se tienen permisos de administrador.	
Post-condiciones	El servicio seleccionado se inicia, llamando al programa ejecutable del componente de seguridad para que se inicie mediante el uso de un Web Service.	

Tabla 3.11 Caso de Uso Administrar Servicios.

Fuente: Elaboración Propia.

3.4.4.7.1.2 Pantallas que describen el Caso de Uso

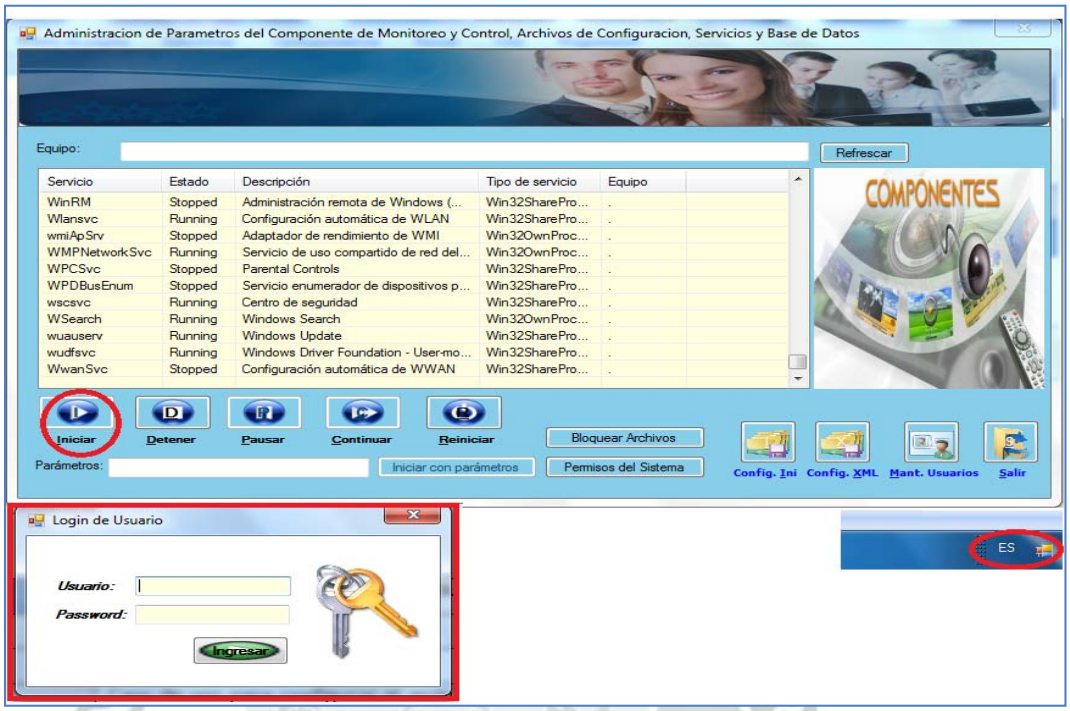


Figura 3.9 Pantalla de Administración de Servicios.

Fuente: Elaboración Propia.

3.4.4.7.2 Administrar Usuarios

3.4.4.7.2.1 Descripción

Casos de Uso	**Administrar Usuarios.**	**[CDU-03]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para administrar usuarios del sistema.	
Tipo de Caso de Uso	Primario.	
Ref. Cruzada	Ninguna.	

Pre-condiciones	Estar correctamente logueado en el sistema y ver la pantalla principal.
Flujo Normal de Eventos	
Acción: del Actor(es)	**Acción: del Sistema**
1.- El Usuario hace click en el mantenimiento de usuarios. 3.- El Usuario modifica, inactiva o ingresa los campos necesarios y da click en el botón guardar.	2.- El Sistema muestra la pantalla de configuración con los respectivos campos editables. 4.- El Sistema valida los campos y graba el registro si los datos son correctos.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
4.1.- En caso de que la validación no sea exitosa entonces se piden los datos nuevamente. 4.2.- El usuario reingresa los datos según las validaciones de la pantalla y da click en el botón guardar.	4.3.- El sistema validara si los campos enviados son correctos, si fuera así muestra una pantalla de grabación exitosa.
Post-condiciones	Grabación de datos exitosa en la base de datos en la tabla usuarios.

Tabla 3.12 Caso de Uso Administrar Usuarios.

Fuente: Elaboración Propia.

3.4.4.7.2.2 Pantallas que describen el Caso de Uso

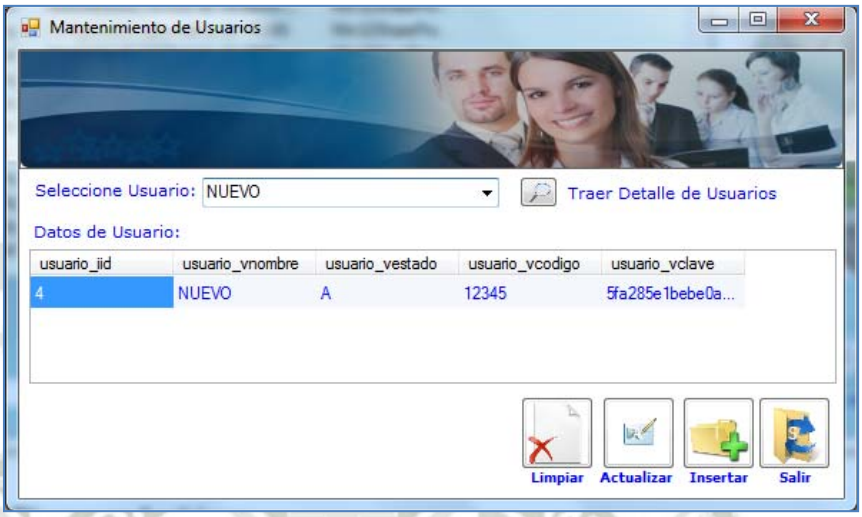


Figura 3.10 Pantalla de Mantenimiento de Usuarios del Sistema.

Fuente: Elaboración Propia.

3.4.4.7.3 Bloquear Archivos

3.4.4.7.3.1 Descripción

Casos de Uso	**Bloquear Archivos.**	**[CDU-04]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para bloquear archivos para que estos no puedan ser copiados fuera del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y estar en la pantalla principal de administración.	
Flujo Normal de Eventos		

Acción: del Actor(es)	**Acción: del Sistema**
2.- El Usuario hace click en el botón bloquear archivos. 4.- El Usuario intenta copiar algún archivo y no podrá ya que se encuentra bloqueado.	1.- El Sistema muestra la pantalla principal. 3.- El Sistema muestra la pantalla de bloqueo de archivos y apenas se muestra bloquea los archivos para que estos no puedan ser copiados, llamando a la librería de enlace dinámico, ClipboardControlLib.dll.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
3.1.- El usuario da click al botón pausa para detener el bloqueo de archivos.	3.2.- El sistema llama a la librería dinámica, ClipboardControlLib.dll y detiene el bloqueo de archivos permitiendo copiarlos.
Post-condiciones	El usuario no puede copiar ningún archivo del sistema.

Tabla 3.13 Caso de Uso Bloquear Archivos.

Fuente: Elaboración Propia.

3.4.4.7.3.2 Pantallas que describen el Caso de Uso

Figura 3.11 Pantalla de Bloqueo de Archivos.

Fuente: Elaboración Propia.

3.4.4.7.4 Configuración INI (Base de Datos)

3.4.4.7.4.1 Descripción

Casos de Uso	**Configuración INI (Base de Datos).**	**[CDU-05]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para configurar el archivo de inicio de la base de datos del sistema.	
Tipo de Caso de Uso	Primario.	
Ref. Cruzada	Ninguna.	

Pre-condiciones	Estar correctamente logueado en el sistema.
Flujo Normal de Eventos	
Acción: del Actor(es)	**Acción: del Sistema**
1.- El Usuario hace click en el botón Config. INI para configurar el archivo de base de datos. 3.- El Usuario modifica los campos necesarios y da click en el botón guardar.	2.- El Sistema muestra la pantalla de configuración con los respectivos campos editables. 4.- El Sistema valida los campos y graba el registro en el archivo ConfigBD.ini haciendo uso de la librería Nini.dll para configuración de archivos.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
2.1.- El usuario da click al botón Ver Archivo INI para ver el archivo de base de datos que se ha configurado.	2.2.- El sistema llama al archivo de configuración y utiliza el programa notepad.exe para mostrarlo
Post-condiciones	El archivo ConfigBD.ini queda grabado con los nuevos parámetros.

Tabla 3.14 Caso de Uso Configuración INI (Base de Datos).

Fuente: Elaboración Propia.

3.4.4.7.4.2 Pantallas que describen el Caso de Uso

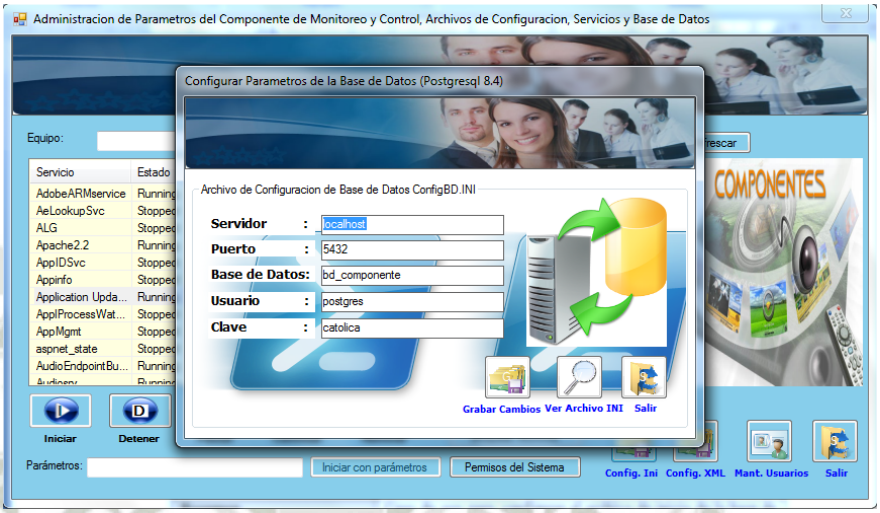


Figura 3.12 Pantalla de Configuración Base de Datos del Componente.

Fuente: Elaboración Propia.

3.4.4.7.5 Configuración XML (Servicio WCF)

3.4.4.7.5.1 Descripción

Casos de Uso	**Configuración XML (Servicio WCF)**	**[CDU-06]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para configurar los parámetros del archivo del componente de seguridad del sistema.	
Tipo de Caso de Uso	Primario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y en la pantalla principal de administración.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
1.- El Usuario hace click en la opción configurar XML.	2.- El sistema muestra la pantalla para configurar el archivo XML del componente.	

3.- El Usuario modifica cualquiera de las cajas de texto que son: ProcesoaVigilar, CreditoDiario, DiasaVigilar, Tiempo Bloqueo, Usuarios a Vigilar. 4.- El Usuario da click al botón grabar cambios.	5.- El sistema valida los campos ingresados y conecta al componente Nini.dll para guardar los datos mediante el método de guardado asignado. 6.- Si la grabación fue exitosa el sistema muestra una pantalla al usuario indicando que la grabación se realizo correctamente.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
2.1.- El usuario da click al botón Ver Archivo INI para ver el archivo de base de datos que se ha configurado.	2.2.- El sistema llama al archivo de configuración y utiliza el programa notepad.exe para mostrarlo.
Post-condiciones	Los cambios son guardados correctamente en el archivo XML.

Tabla 3.15 Caso de Uso Configuración XML (Servicio WCF).

Fuente: Elaboración Propia.

3.4.4.7.5.2 Pantallas que describen el Caso de Uso

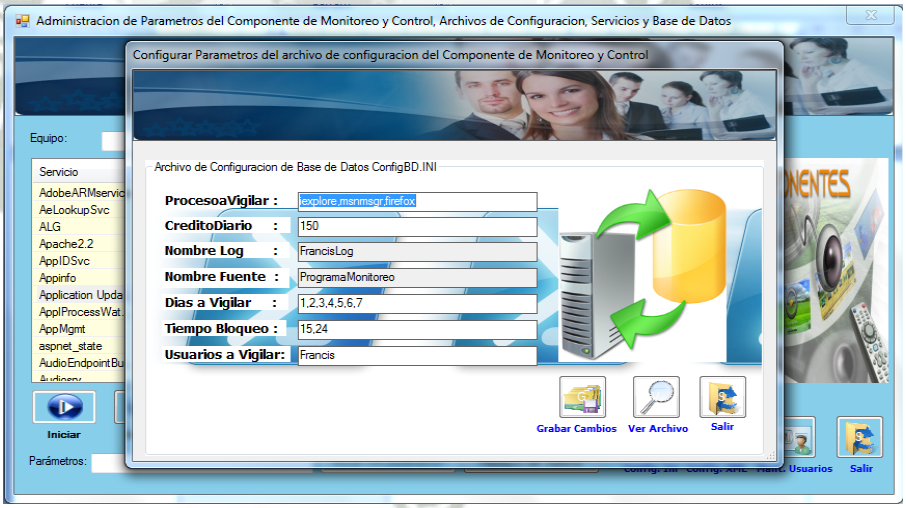


Figura 3.13 Pantalla de Configuración de Archivo XML del Componente.

Fuente: Elaboración Propia.

3.4.4.7.6 Administrar Permisos Del Sistema

3.4.4.7.6.1 Descripción

Casos de Uso	**Administrar Permisos del Sistema.**	**[CDU-07]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para administrar los permisos del sistema.	
Tipo de Caso de Uso	Primario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y en la pantalla principal de administración.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
1.- El Usuario da click en el botón permisos del sistema del formulario principal. 3.- El Usuario modifica las directivas de seguridad del sistema, algunas requieren que el sistema se reinicie. 5.- El Usuario da click al botón reiniciar del sistema.	2.- El Sistema muestra la pantalla de directivas locales del sistema. 4.- El Sistema pide que se reinicie el sistema si el usuario realiza algún cambio en las directivas de seguridad del sistema. 6.- El sistema se reinicia, confirmando los cambios.	
Flujos Alternativos		
Acción: del Actor(es)	**Acción: del Sistema**	
2.1.- El usuario da click en el botón Permisos del Sistema del formulario principal.	2.2.- El sistema no encuentra el archive **gpedit.msc** y manda una pantalla de error al usuario.	
Post-condiciones	Se graban las modificaciones en los permisos del sistema según las directivas locales del mismo.	

Tabla 3.16 Caso de Uso Administrar Permisos del Sistema.

Fuente: Elaboración Propia.

3.4.4.7.6.2 Pantallas que describen el Caso de Uso

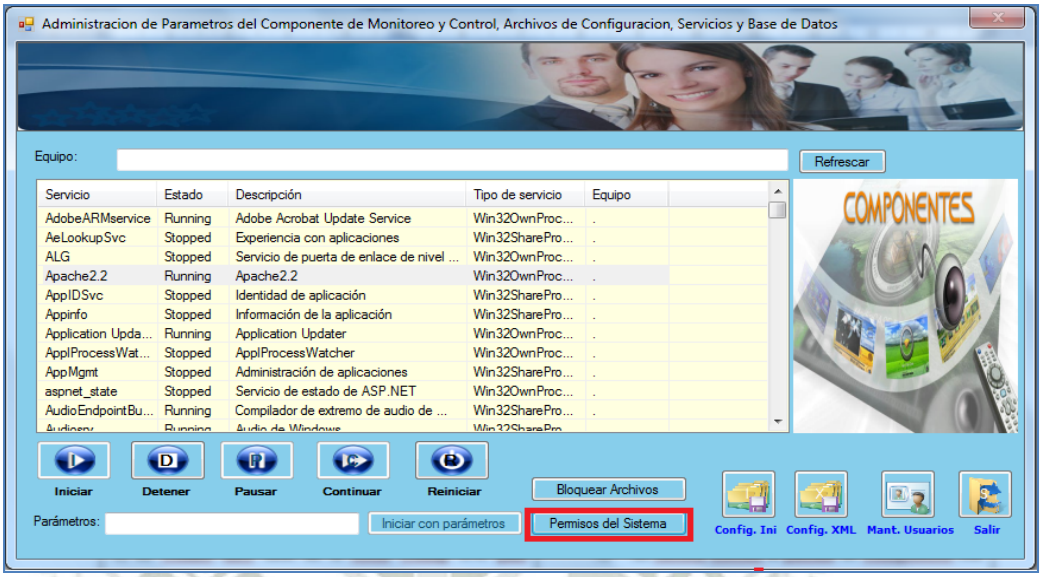


Figura 3.14 Pantalla de formulario principal – Acceso a Formulario de Mantenimiento de Usuarios.

Fuente: Elaboración Propia.

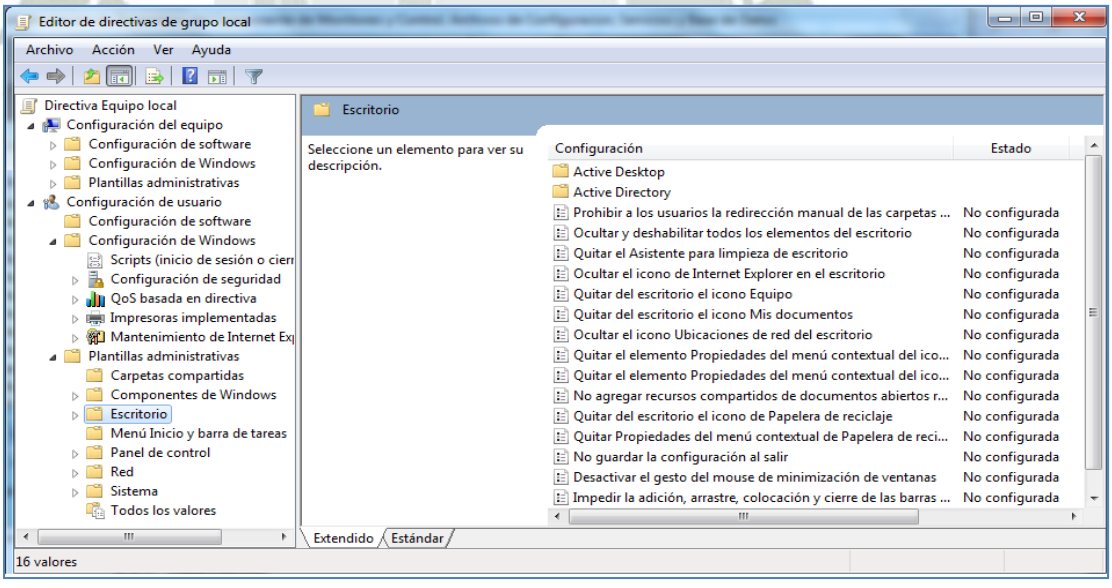


Figura 3.15 Pantalla de Edición de Directivas (GPEDIT) de Windows.
Fuente: Elaboración Propia.

3.4.4.7.7 Loguear Usuario

3.4.4.7.7.1 Descripción

Casos de Uso	**Loguear Usuario.**	**[CDU-08]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-01] – Caso de Uso Principal**	
Resumen	Caso de uso para loguearse como usuario del sistema.	
Tipo de Caso de Uso	Primario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Ninguna.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
1.- El Usuario hace click en la barra de tareas del sistema, en el icono del componente. 3.- El Usuario ingresa su nombre usuario y password y da click en el botón Ingresar.	2.- El Sistema muestra la pantalla de Login de Usuario. 4.- El Sistema valida los datos ingresados y los compara con los de la base de datos de usuario, si los datos son correctos procede a crear el formulario principal y lo muestra al usuario.	
Flujos Alternativos		
Acción: del Actor(es)	**Acción: del Sistema**	
3.1.- El usuario da click al botón Ingresar pero los datos son erróneos.	3.2.- El sistema valida los datos contra la base de datos y muestra una pantalla con el mensaje “Usuario y Password Incorrectos”.	
Post-condiciones	El Usuario se loguea correctamente en la Aplicación.	

Tabla 3.17 Caso de Uso Loguear Usuario.
Fuente: Elaboración Propia.

3.4.4.7.7.2 Pantallas que describen el Caso de Uso

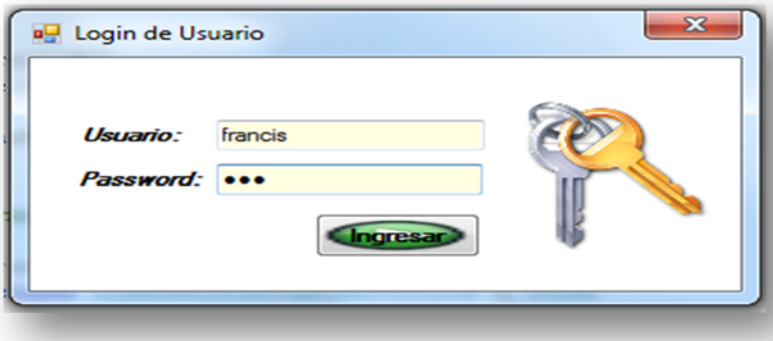


Figura 3.16 Pantalla de Login de Usuario.
Fuente: Elaboración Propia.

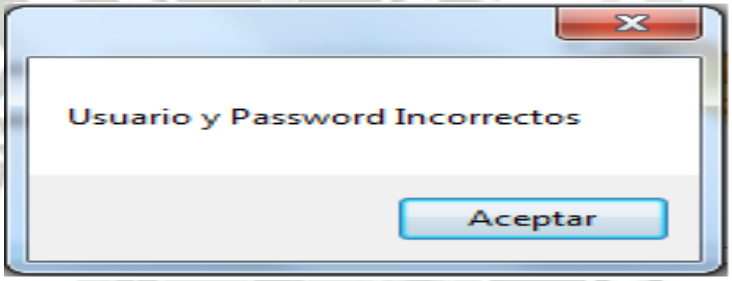


Figura 3.17 Pantalla de Error de Logueo.
Fuente: Elaboración Propia.

3.4.4.7.8 Iniciar/Detener/Pausar/Reiniciar/Refrescar/Continuar Servicio

3.4.4.7.8.1 Descripción

Casos de Uso	**Iniciar/Detener/Pausar/Reiniciar/Refrescar/Continuar Servicio.**	**[CDU-09]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-02] – Administrar Servicios**	
Resumen	Caso de uso para administrar todas las operaciones de un servicio windows.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	

Pre-condiciones	Estar correctamente logueado en el sistema y en la pantalla de administración principal.

Flujo Normal de Eventos	
Acción: del Actor(es)	**Acción: del Sistema**
2.- El usuario da click en la grilla que muestra los servicios y selecciona un servicio. 3.- El usuario da click en cualquiera de las operaciones relacionadas al servicio para que este se pueda Iniciar/Detener/Pausar/Reiniciar/Refrescar/Continuar.	1.- El sistema muestra una grilla con todos los servicios disponibles. 4.- El sistema ejecuta la operación seleccionada por el usuario haciendo uso de la API de Windows32.

Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
4.2 El usuario recibe una pantalla de error del sistema, en caso el servicio no haya podido responder a la petición del punto 3.	4.1 El sistema no puede ejecutar la operación por algún error relacionado con el sistema.

Post-condiciones	El servicio atiende la petición del usuario según la operación requerida.

Tabla 3.18 Caso de Uso Iniciar/Detener/Pausar/Reiniciar/Refrescar/Continuar Servicio

Fuente: Elaboración Propia.

3.4.4.7.8.2 Pantallas que describen el Caso de Uso

3.4.4.7.9 Crear WCF

3.4.4.7.9.1 Descripción

Casos de Uso	**Crear WCF.**	**[CDU-10]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-02] – Administrar Servicios**	
Resumen	Caso de uso que instancia la clase WCF para el Servicio WEB.	
Tipo de Caso de Uso	Secundario.	

Ref. Cruzada	Ninguna.
Pre-condiciones	Iniciar el Servicio de Windows relacionado al componente.
Flujo Normal de Eventos	
Acción: del Actor(es)	**Acción: del Sistema**
*El actor no interviene directamente.	1.- El Sistema hace una instancia de la clase creada para el servicio web (Interfaz del servicio), utilizando las clases del framework para wcf, creando un servidor host, para los clientes(usuarios) que se conecten, esto mediante el uso del framework de la plataforma .net.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
Post-condiciones	Se instancia la clase del servicio web creando un servidor en escucha a la espera de peticiones de clientes.

Tabla 3.19 Caso de Uso Crear WCF.

Fuente: Elaboración Propia.

3.4.4.7.9.2 Pantallas que describen el Caso de Uso

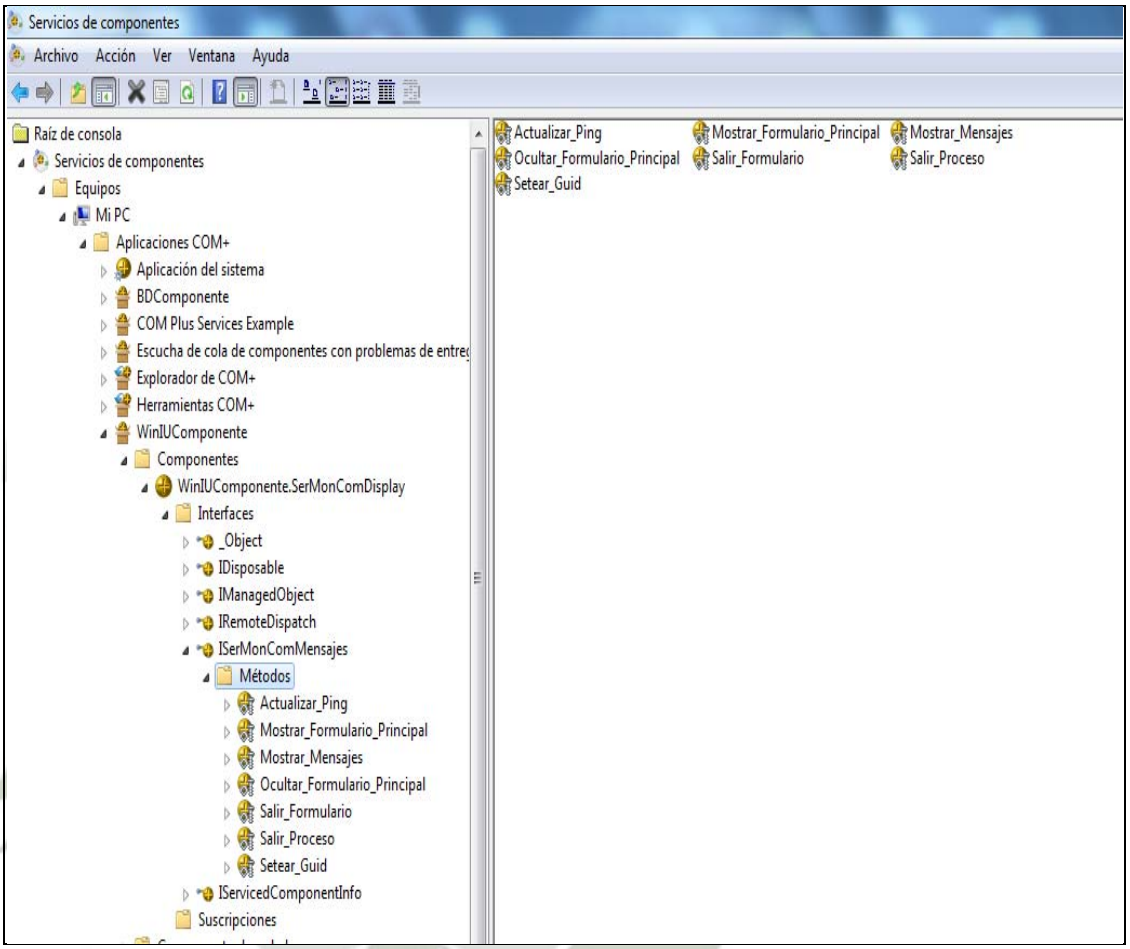


Figura 3.18 Pantalla de Servicios de componentes – Interfaz de Componentes.
Fuente: Elaboración Propia.

3.4.4.7.10 Iniciar Servicios con Parámetros

3.4.4.7.10.1 Descripción

Casos de Uso	**Iniciar Servicios con Parámetros.**	**[CDU-11]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-02] – Administrar Servicios**	
Resumen	Caso de uso para iniciar servicios con parámetros.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y en la pantalla principal de administración.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
1.- El usuario tipea los parámetros para el servicio a iniciar. 2.- El usuario da click en el botón iniciar servicio con parámetros.	3.- El sistema valida los parámetros enviados, y si son correctos ejecuta el servicio.	
Flujos Alternativos		
Acción: del Actor(es)	**Acción: del Sistema**	
3.2 El usuario reingresa los parámetros para el servicio seleccionado en la grilla y da click en el botón iniciar servicio con parámetros.	3.1 Si los parámetros enviados son incorrectos no se inicia el servicio y se manda un mensaje de error.	
Post-condiciones	Servicio iniciado correctamente con parámetros.	

Tabla 3.20 Caso de Uso Crear WCF.

Fuente: Elaboración Propia.

3.4.4.7.10.2 Pantallas que describen el Caso de Uso

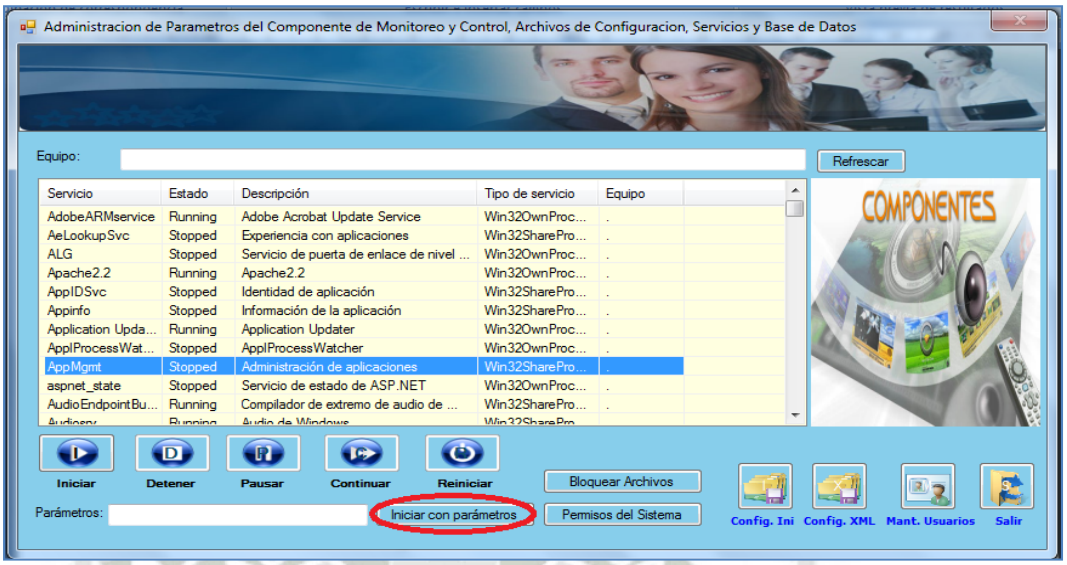


Figura 3.19 Pantalla de formulario principal – Acceso a Inicio de Servicios con parámetros.

Fuente: Elaboración Propia.

3.4.4.7.11 Actualizar Usuario

3.4.4.7.11.1 Descripción

Casos de Uso	**Actualizar Usuario.**	**[CDU-12]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-03] – Administrar Usuarios**	
Resumen	Caso de uso para Actualizar Usuario en la Base de datos del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y estar en la pantalla principal.	
Flujo Normal de Eventos		

Acción: del Actor(es)	Acción: del Sistema
1.- El usuario hace click en el botón mantenimiento de usuarios. 3.- El usuario hace click en el botón actualizar del formulario de mantenimiento de usuarios. 5.- El usuario actualiza los siguientes campos: nombre de usuario, código de usuario, antigua clave de usuario, nueva clave de usuario, retipee nueva clave de usuario, estado de usuario en el formulario de actualización de datos del usuario. 6.- El usuario con las modificaciones de los campos actualiza y guarda los datos del usuario dando click en el botón Actualizar del Formulario de actualización del usuario.	2.- El sistema muestra la pantalla de mantenimiento de usuarios. 4.- El sistema muestra el formulario de actualización de datos del usuario. 7.- El sistema valida los datos y si estos con correctos guarda la información en la base de datos y muestra la pantalla el mensaje "grabación exitosa".
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
6.1.- Si los datos ingresados no son correctos entonces el usuario tiene que reingresar los datos correctamente según los campos que el sistema muestra en rojo con un signo de admiración en rojo. 6.2.- Una vez reingresados los datos el usuario da click en el botón actualizar del formulario de actualización del usuario.	6.3.- Si los datos son correctos el sistema muestra la pantalla con el mensaje "grabación exitosa", en caso contrario volveremos al punto 6.1
Post-condiciones	Los datos actualizados son grabados en la base de datos en la tabla de usuarios del sistema (en nuestro caso la base de datos bd_componente en postgresql)

Tabla 3.21 Caso de Uso Actualizar Usuario.

Fuente: Elaboración Propia.

3.4.4.7.11.2 Pantallas que describen el Caso de Uso

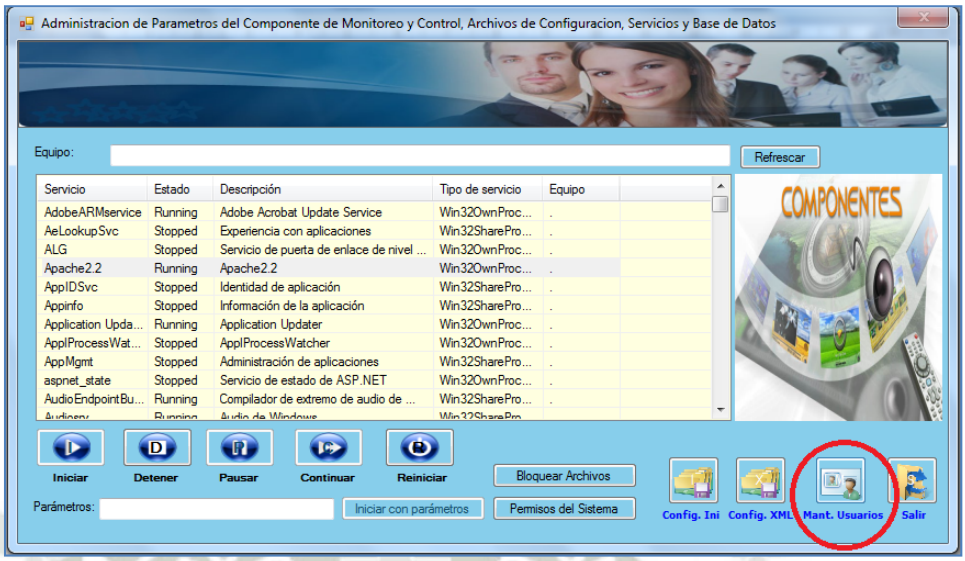


Figura 3.20 Pantalla de Formulario principal – Acceso a Formulario de Mantenimiento de Usuarios.

Fuente: Elaboración Propia.

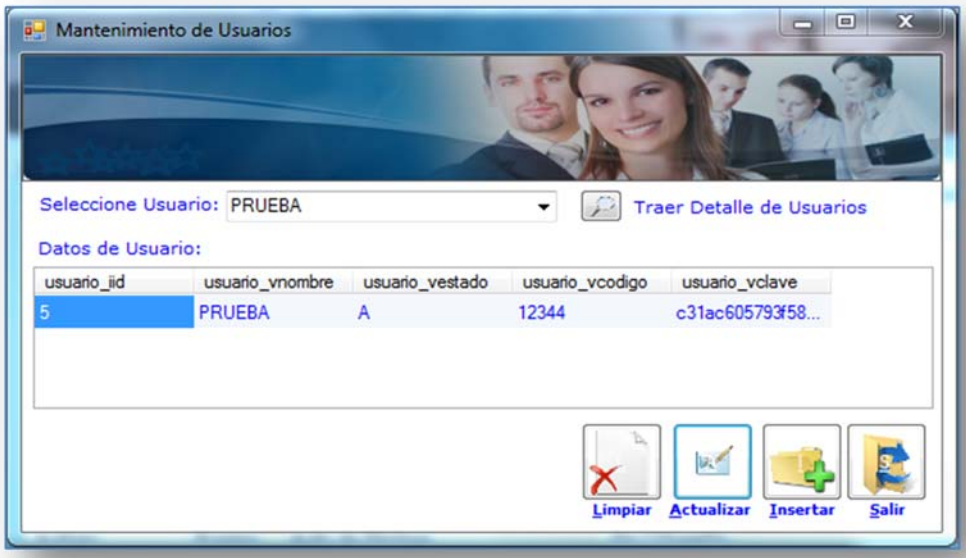


Figura 3.21 Pantalla de Formulario de Mantenimiento de Usuarios.
Fuente: Elaboración Propia.

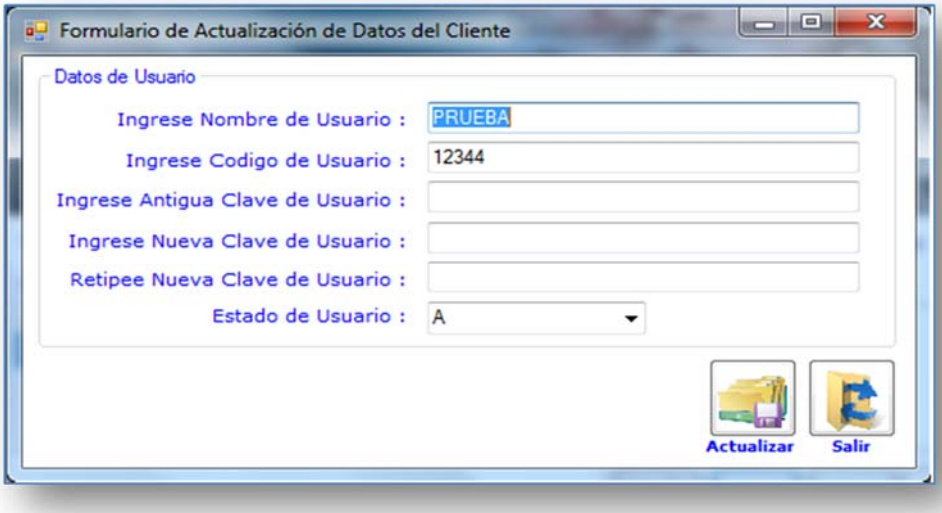


Figura 3.22 Pantalla de Formulario de Actualización de Datos del Cliente.

Fuente: Elaboración Propia.

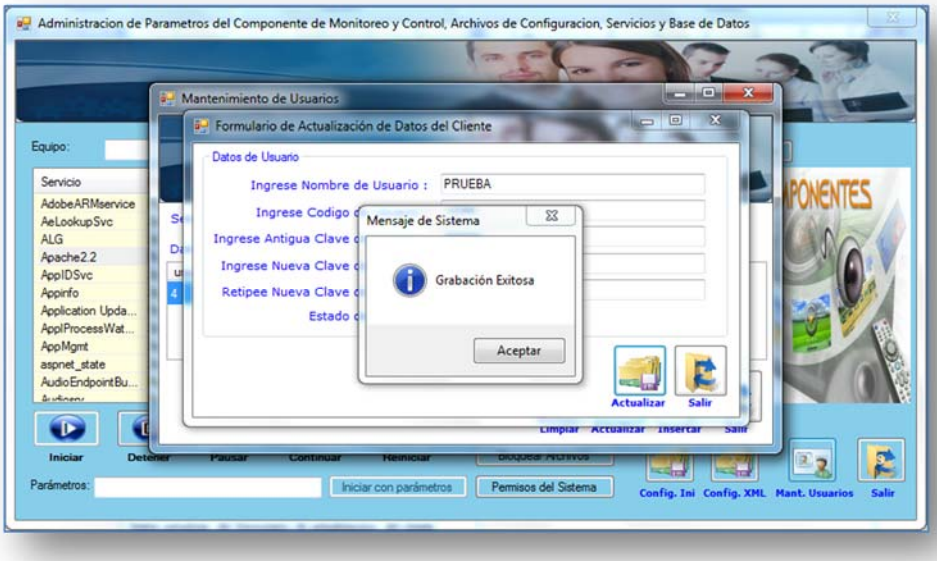


Figura 3.23 Pantalla de Grabación Exitosa.

Fuente: Elaboración Propia.

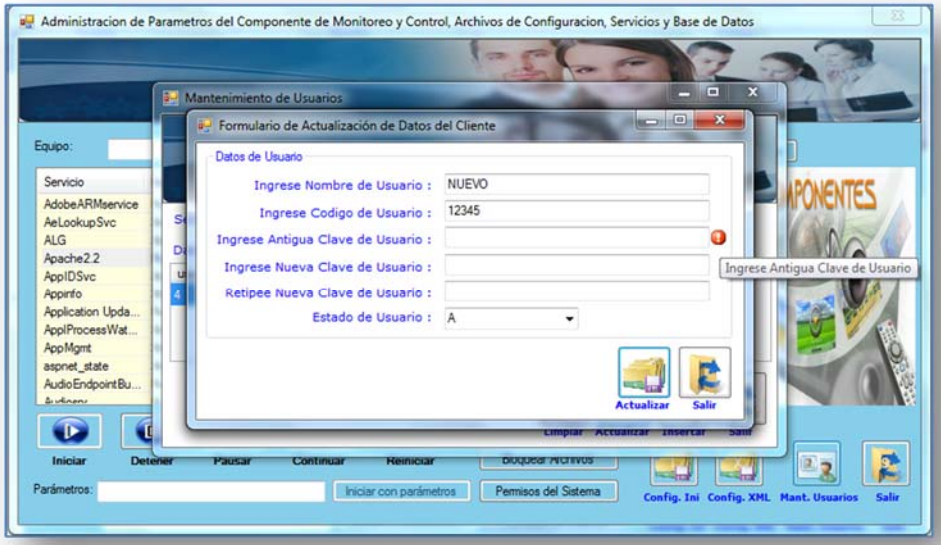


Figura 3.24 Pantalla de Formulario de Actualización de Datos del Cliente (Validación de Datos Incorrectos).
Fuente: Elaboración Propia.

3.4.4.7.12 Insertar Usuario

3.4.4.7.12.1 Descripción

Casos de Uso	**Insertar Usuario.**	**[CDU-13]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-03] – Administrar Usuarios**	
Resumen	Caso de uso para Insertar Usuario en la Base de datos del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema.	
Flujo Normal de Eventos		
Acción: del Actor(es)		**Acción: del Sistema**
1.- El usuario hace click en el botón mantenimiento de usuarios. 3.- El usuario hace click en el botón insertar del formulario		2.- El sistema muestra la pantalla de mantenimiento de usuarios. 4.- El sistema muestra el formulario de

de mantenimiento de usuarios. 5.- El usuario actualiza los siguientes campos: nombre de usuario, código de usuario, clave de usuario, retipee nueva clave de usuario en el formulario de actualización de datos del usuario. 6.- El usuario con las modificaciones de los campos actualiza y guarda los datos del usuario dando click en el botón grabar del Formulario de inserción del usuario.	inserción de datos del usuario. 7.- El sistema valida los datos y si estos con correctos guarda la información en la base de datos y muestra la pantalla el mensaje "grabación exitosa".
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
6.1.- Si los datos ingresados no son correctos entonces el usuario tiene que reingresar los datos correctamente según los campos que el sistema muestra en rojo con un signo de admiración en rojo. 6.2.- Una vez reingresados los datos el usuario da click en el botón grabar del formulario de inserción del usuario.	6.3.- Si los datos son correctos el sistema muestra la pantalla con el mensaje "grabación exitosa", en caso contrario volveremos al punto 6.1
Post-condiciones	Grabación correcta de nuevo usuario en base de datos.

Tabla 3.22 Caso de Uso Insertar Usuario.

Fuente: Elaboración Propia.

3.4.4.7.12.2 Pantallas que describen el Caso de Uso

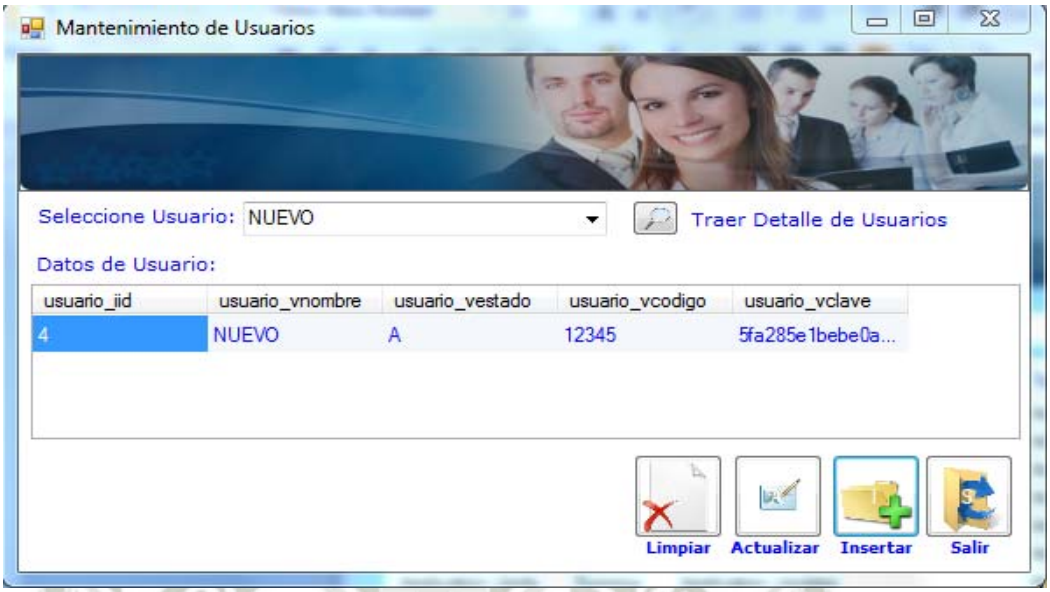


Figura 3.25 Pantalla de Formulario de Mantenimiento de Usuarios – Nuevo Usuario.

Fuente: Elaboración Propia.

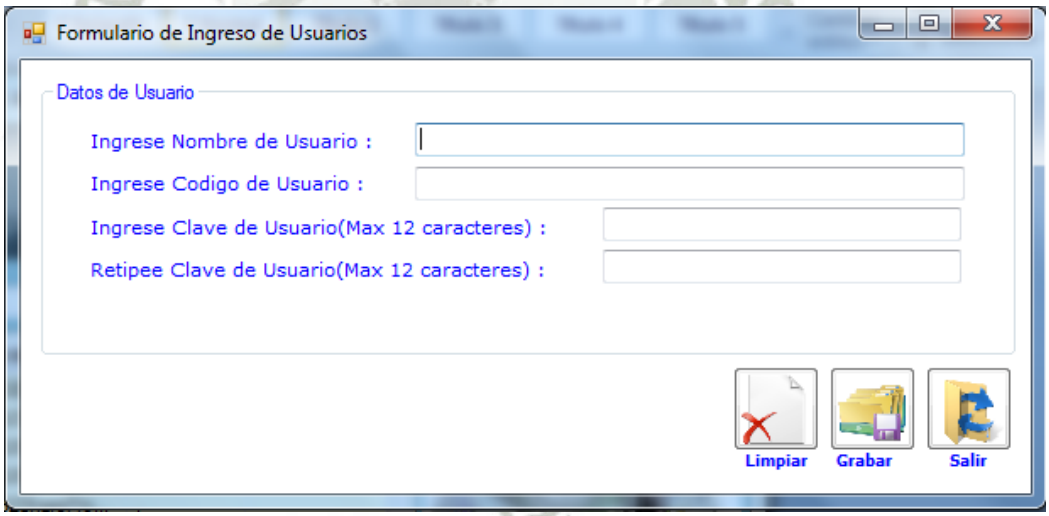


Figura 3.26 Pantalla de Formulario de Ingreso de Usuarios – Nuevo Usuario.

Fuente: Elaboración Propia.

3.4.4.7.13 Buscar Usuario

3.4.4.7.13.1 Descripción

Casos de Uso	**Buscar Usuario.**	**[CDU-14]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-03] – Administrar Usuarios**	
Resumen	Caso de uso para Buscar un usuario del Sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y ver la pantalla principal.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
1.- El Usuario hace click en el mantenimiento de usuarios desde la pantalla principal. 3.- El Usuario selecciona un usuario de la lista y da click en el botón de búsqueda.	2.- El Sistema muestra la pantalla de mantenimiento de usuarios, esto es un combo con la lista de todos los usuarios, un botón de búsqueda y una grilla donde se verán los datos a detalle del usuario que se seleccione, abajo se tendrán otros botones que no intervienen en este caso de uso. 4.- El Sistema buscara en la base de datos el detalle del usuario que se encuentre seleccionado en el combox de usuarios, luego de esto se mostrara seguidamente en la grilla los datos del usuario.	
Flujos Alternativos		
Acción: del Actor(es)	**Acción: del Sistema**	
2.1.- El usuario da click al botón de búsqueda y no se puede conectar a la base de datos.	2.2.- El sistema mandara un mensaje de excepción diciendo que no se puede conectar a la base de datos.	
Post-condiciones	El usuario vera la data del usuario buscado correctamente.	

Tabla 3.23 Caso de Uso Buscar Usuario.

Fuente: Elaboración Propia.

3.4.4.7.13.2 Pantallas que describen el Caso de Uso

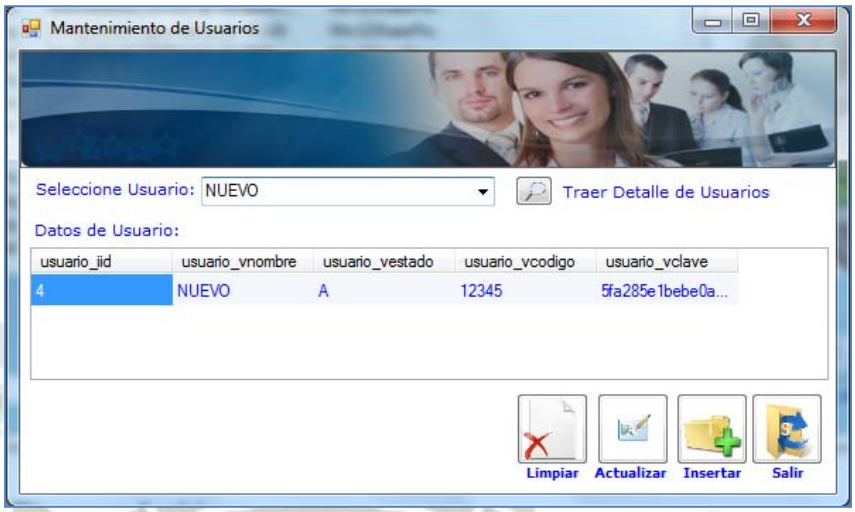


Figura 3.27 Pantalla de Búsqueda de Usuarios del Sistema.
Fuente: Elaboración Propia.

3.4.4.7.14 Cambiar Contraseña

3.4.4.7.14.1 Descripción

Casos de Uso	**Cambiar Contraseña.**	**[CDU-15]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-03] – Administrar Usuarios**	
Resumen	Caso de uso para cambiar la contraseña de un usuario en la base de datos del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema y ver la pantalla principal.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	

1.- El Usuario hace click en el mantenimiento de usuarios desde la pantalla principal. 3.- El Usuario selecciona un usuario de la lista y da click en el botón actualizar. 5.- El Usuario actualizara solo los campos que hacen referencia a la Clave, de las cajas de texto descritas en el punto 4), luego de hacer esto correctamente, dará click al botón Actualizar de la pantalla.	2.- El Sistema muestra la pantalla de mantenimiento de usuarios, esto es un combo con la lista de todos los usuarios, un botón de búsqueda y una grilla donde se verán los datos a detalle del usuario que se seleccione, abajo se tendrá el botón actualizar que servirá para cambiar datos del usuario que se seleccione en el combobox. 4.- Buscará en la base de datos el detalle del usuario que se encuentre seleccionado en el combox de usuarios, luego de esto se mostrara seguidamente la pantalla de mantenimiento de usuarios donde se verán los datos en cinco cajas de texto que son: Ingrese Nombre de Usuario, Ingrese Código de Usuario, Ingrese Antigua Clave de Usuario, Ingrese Nueva Clave de Usuario, Retipee Nueva Clase de Usuario y además un combobox con el estado actual del Usuario->A para activo y I para inactivo. 6.- Validará los datos haciendo la comparación en la base de datos, para la clave antigua, si todo es correcto, se guardara la nueva clave para el usuario.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
5.1.- Si el usuario no tipeo la clave antigua correctamente, o si no escribió dos veces la nueva clave correctamente entonces se validaran los campos erróneos para que el usuario retipee los datos. 5.2.- Si el usuario retipea correctamete los datos y da click al botón actualizar se regresa al punto 6).	5.3.- Si no se pudo conectar a la base de datos el sistema mandara un mensaje de excepción.
Post-condiciones	La contraseña del usuario estará cambiada correctamente y esta será guardada en la base de datos.

Tabla 3.24 Caso de Uso Cambiar Contraseña.

Fuente: Elaboración Propia.

3.4.4.7.14.2 Pantallas que describen el Caso de Uso

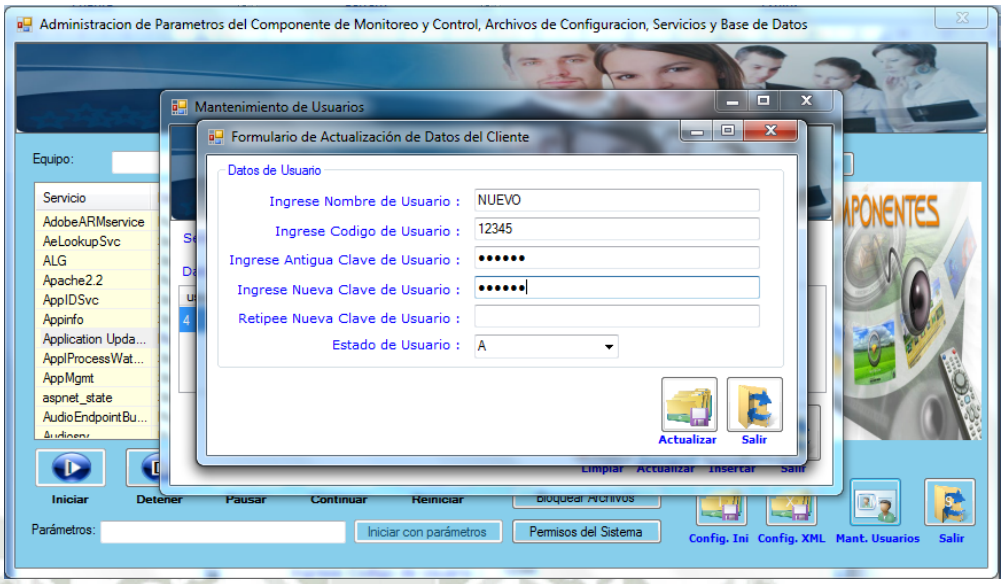


Figura 3.28 Pantalla de Cambio de Contraseña de Usuarios del Sistema.

Fuente: Elaboración Propia.

3.4.4.7.15 Llamar Componente de Base de Datos

3.4.4.7.15.1 Descripción

Casos de Uso	**Llamar Componente de Base de Datos**	**[CDU-16]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-03] – Administrar Usuarios**	
Resumen	Caso de uso que conecta al componente de Base de Datos (BD) del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema.	
Flujo Normal de Eventos		

Acción: del Actor(es)	Acción: del Sistema
1.- El Usuario hace una petición de Bd por medio de cualquier formulario que conecte a la base de datos del sistema.	2.- El sistema se conecta mediante COM+ al componente BDComponente.UsuariosBD y al componente BDComponente.Usuarios los cuales son utilizados para realizar peticiones (selección de datos) y transacciones (guardar o eliminar datos).
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
	2.1.- Si el servidor de componentes no está activado o la base de datos no existe el sistema no se conectara y se mandara una excepción.
Post-condiciones	Se realizara una petición o transacción a base de datos satisfactoriamente.

Tabla 3.25 Caso de Uso Llamar Componente de Base de Datos.

Fuente: Elaboración Propia.

3.4.4.7.15.2 Pantallas que describen el Caso de Uso

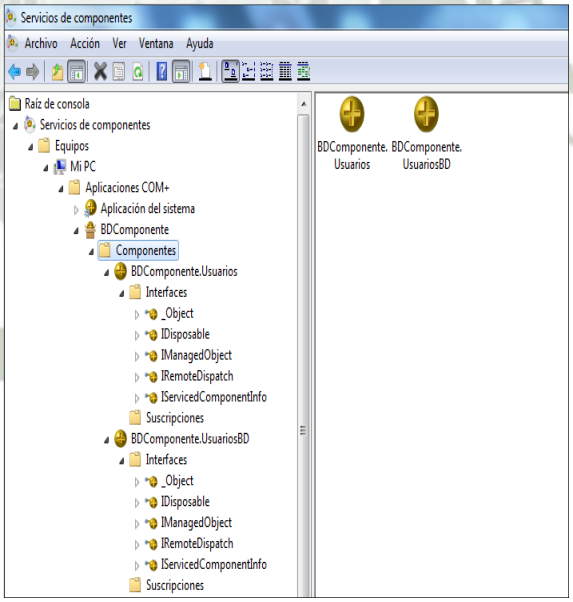


Figura 3.29 Pantalla de Componente BDComponente.UsuariosBD, BDComponente.Usuarios.
Fuente: Elaboración Propia.

3.4.4.7.16 Bloquear /Desbloquear Drag and Drop

3.4.4.7.16.1 Descripción

Casos de Uso	**Bloquear Drag and Drop.**	**[CDU-17]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-04] – Bloquear Archivos**	
Resumen	Caso de uso para bloquear/desbloquear el Drag and Drop que permite arrastrar archivos para copiar del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
2.- El Usuario hace click en el botón bloquear archivos. 4.- El Usuario intenta arrastrar y soltar (drag and drop) un archivo y no podrá ya que se encuentra bloqueado.	1.- El Sistema muestra la pantalla principal. 3.- El Sistema muestra la pantalla de bloqueo de archivos y apenas se muestra bloquea los archivos para que estos no puedan arrastrados y soltados (drag and drop) impidiendo así la copia, llamando a la librería dinámica, user32.dll.	
Flujos Alternativos		
Acción: del Actor(es)	**Acción: del Sistema**	
3.1.- El usuario da click al botón pausa para detener el bloqueo de archivos y permitir arrastrar y soltar (drag and drop) un archivo.	3.2.- El sistema llama a la librería dinámica, user32.dll y detiene el bloqueo de archivos permitiendo copiarlos.	
Post-condiciones	El usuario no puede copiar ningún archivo del sistema haciendo drag and drop.	

Tabla 3.26 Caso de Uso Bloquear Drag and Drop.
Fuente: Elaboración Propia.

3.4.4.7.16.2 Pantallas que describen el Caso de Uso

Figura 3.30 Pantalla de Bloqueo de Archivos Drag and Drop.

Fuente: Elaboración Propia.

3.4.4.7.17 Iniciar/Detener Bloqueo de Archivos

3.4.4.7.17.1 Descripción

Casos de Uso	**Iniciar/Detener Bloqueo.**	**[CDU-18]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-04] – Iniciar/Detener Bloquear Archivos**	
Resumen	Caso de uso para Iniciar/Detener Bloqueo que permite copiar archivos del sistema.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Estar correctamente logueado en el sistema.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
2.- El Usuario hace click en el botón bloquear	1.- El Sistema muestra la pantalla principal.	

archivos. 4.- El Usuario intenta copiar un archivo y no podrá ya que se encuentra bloqueado.	3.- El Sistema muestra la pantalla de bloqueo de archivos y apenas se muestra bloquea los archivos para que estos no puedan ser copiados, llamando a la librería dinámica, user32.dll.
Flujos Alternativos	
Acción: del Actor(es)	**Acción: del Sistema**
3.1.- El usuario da click al botón pausa para detener el bloqueo de archivos y permitir copiar un archivo.	3.2.- El sistema llama a la librería dinámica, user32.dll y detiene el bloqueo de archivos permitiendo copiarlos.
Post-condiciones	El usuario no puede copiar ningún archivo del sistema.

Tabla 3.27 Caso de Uso Iniciar/Detener Bloqueo.

Fuente: Elaboración Propia.

3.4.4.7.18 Llamar Enlace Librería Dinámica/Llamar API Windows

3.4.4.7.18.1 Descripción

Casos de Uso	**Llamar Enlace Librería Dinámica.**	**[CDU-19]**
Actores	Administrador, Sistema.	
Referencia REF	**[CDU-04] – Bloquear Archivos, [CDU-05]– Configuración INI (Base de Datos),[CDU-06]-Configuración XML (Servicio WCF).**	
Resumen	Caso de uso para llamada de Librerías de enlace dinámico de la API de Windows.	
Tipo de Caso de Uso	Secundario.	
Ref. Cruzada	Ninguna.	
Pre-condiciones	Tener instaladas las librerías dinámicas a utilizar.	
Flujo Normal de Eventos		
Acción: del Actor(es)	**Acción: del Sistema**	
	1.- El sistema hace el llamado de las librerías de la API de Windows, describiendo la estructura de cada función.	
Flujos Alternativos		

Acción: del Actor(es)	Acción: del Sistema
	1.1- El sistema manda un mensaje de error cuando la .dll de la API de Windows no se puede cargar.
Post-condiciones	Llamada exitosa a API de Windows.

Tabla 3.28 Caso de Uso Llamar Enlace Librería Dinámica.

Fuente: Elaboración Propia.

3.4.5 Diagramas de Clases

3.4.5.1 Diagrama de Clases de AdministrarServiciosCS

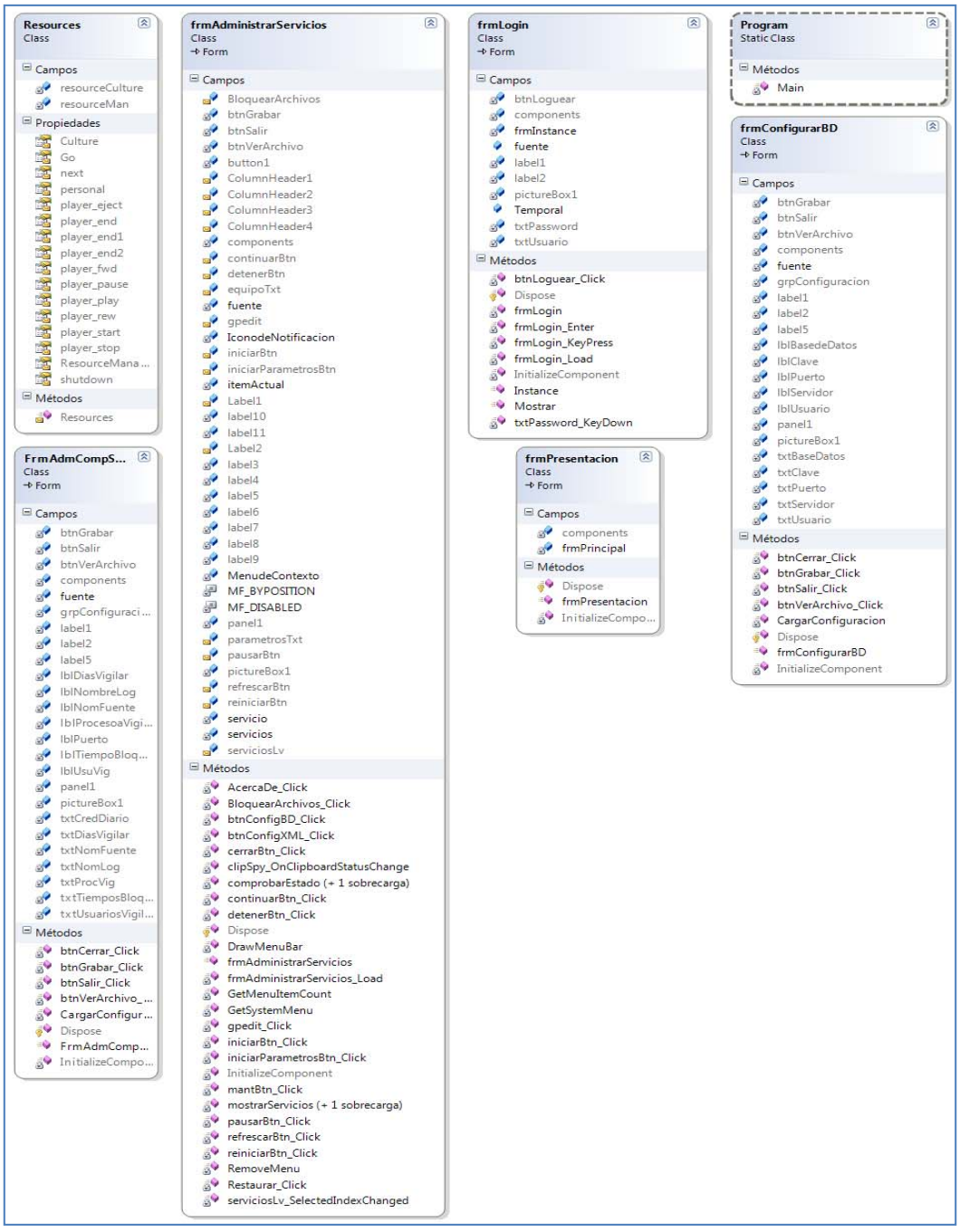


Figura 3.31 Diagrama de Clase AdministrarServiciosCS.

Fuente: Elaboración Propia.

3.4.5.2 Diagrama de Clases de BDCliente

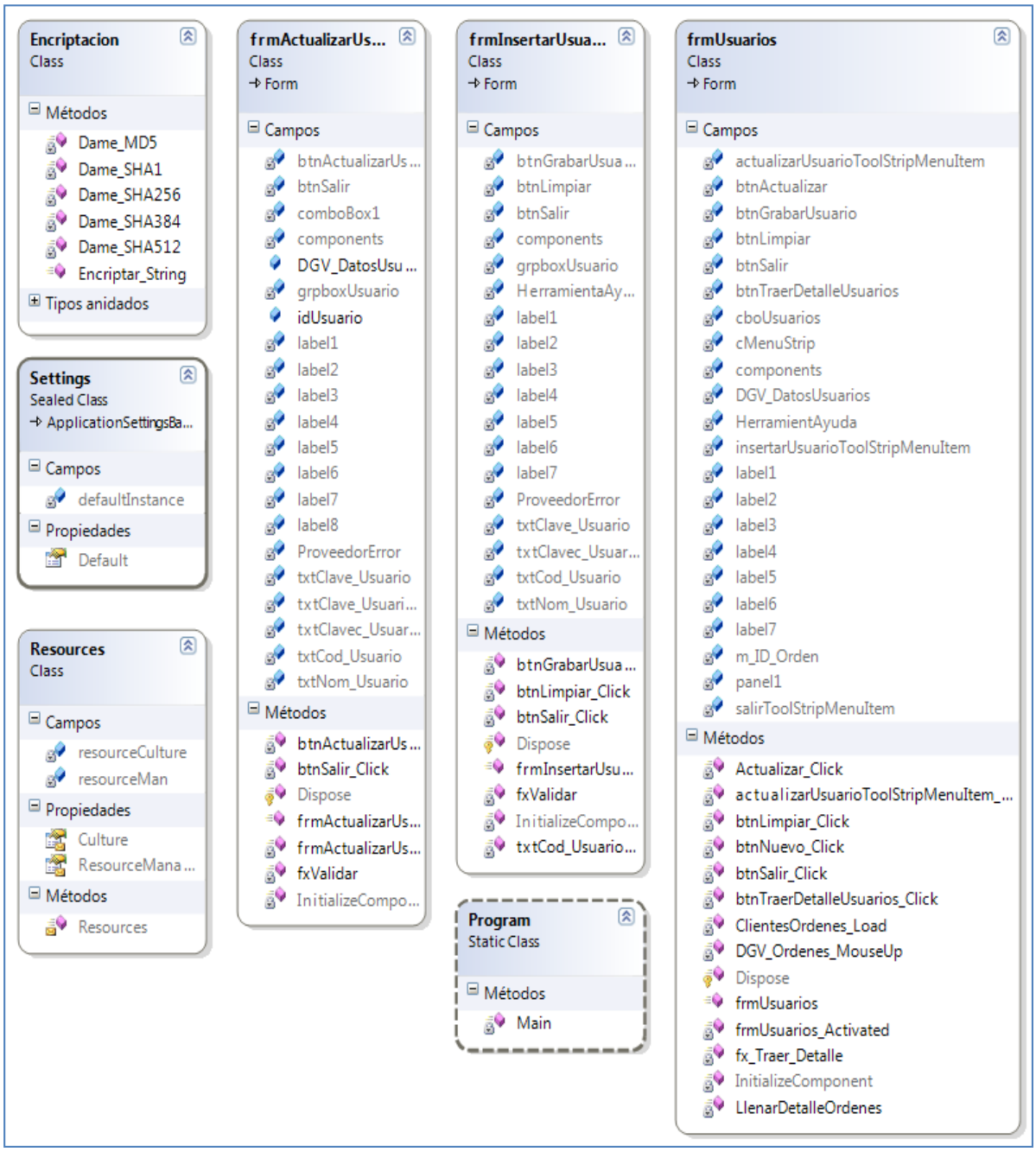


Figura 3.32 Diagrama de Clase BDCliente.

Fuente: Elaboración Propia.

3.4.5.3 Diagrama de Clases de BDComponente

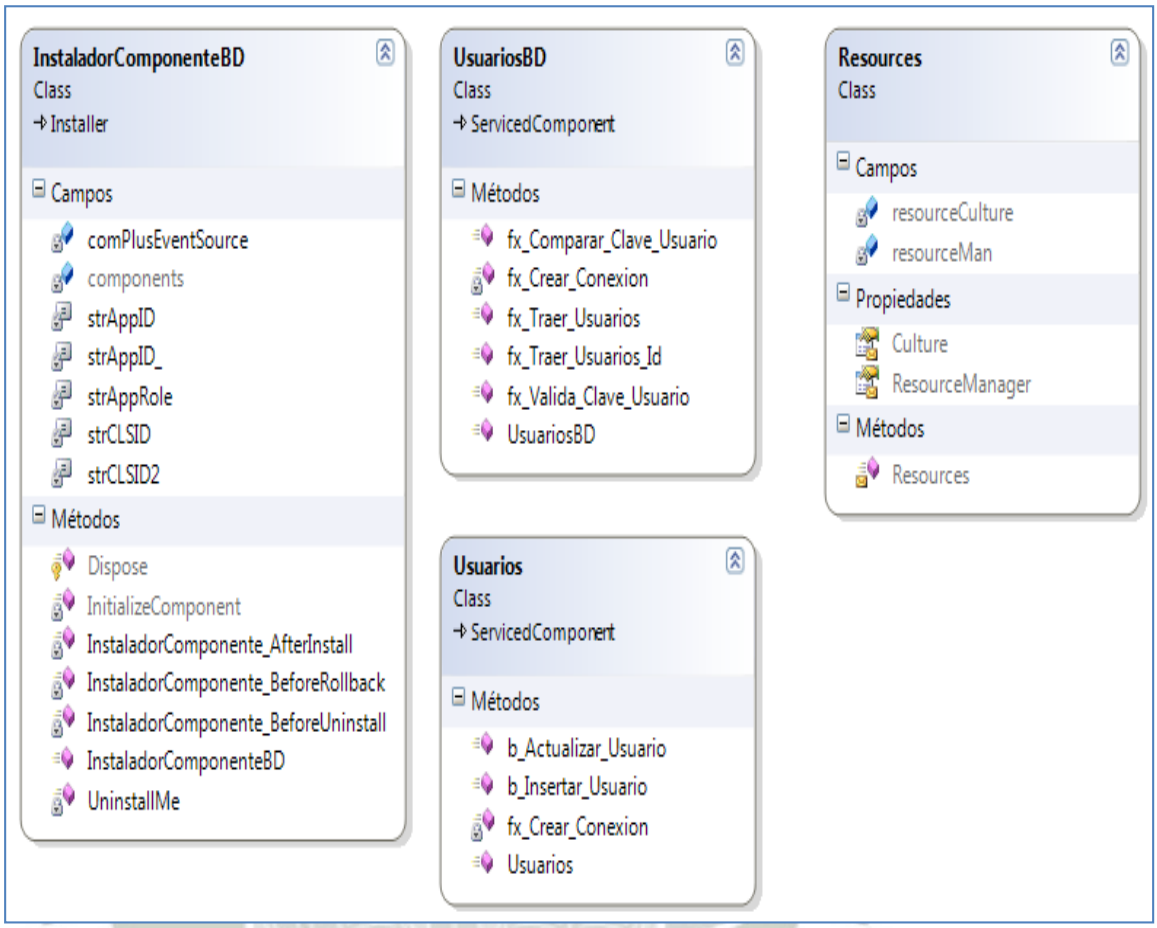


Figura 3.33 Diagrama de Clase BDComponente.

Fuente: Elaboración Propia.

3.4.5.4 Diagrama de Clases de ServicioMonitoreo

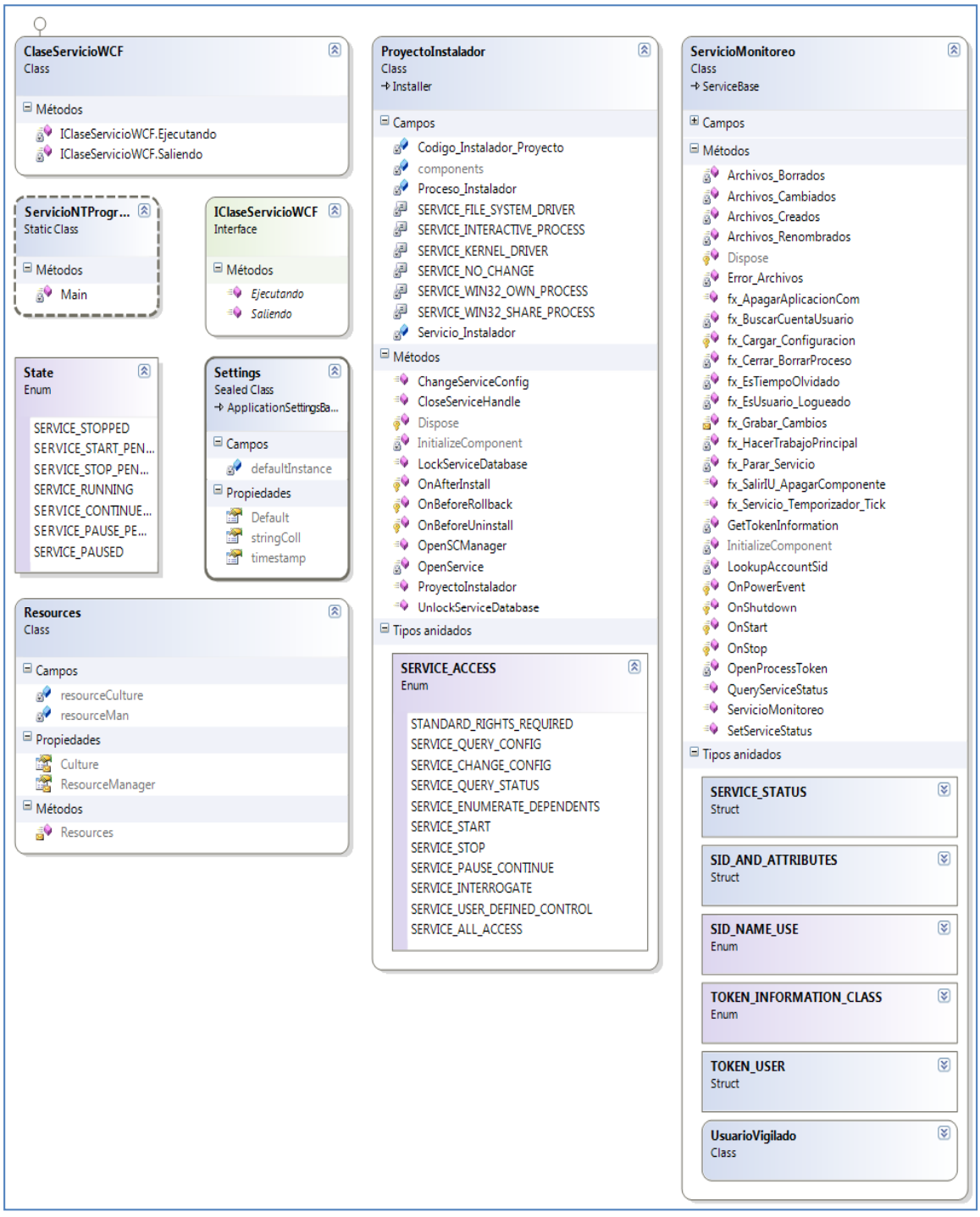


Figura 3.34 Diagrama de Clase Servicio Monitoreo.

Fuente: Elaboración Propia.

3.4.5.5 Diagrama de Clases de WinIUComponente

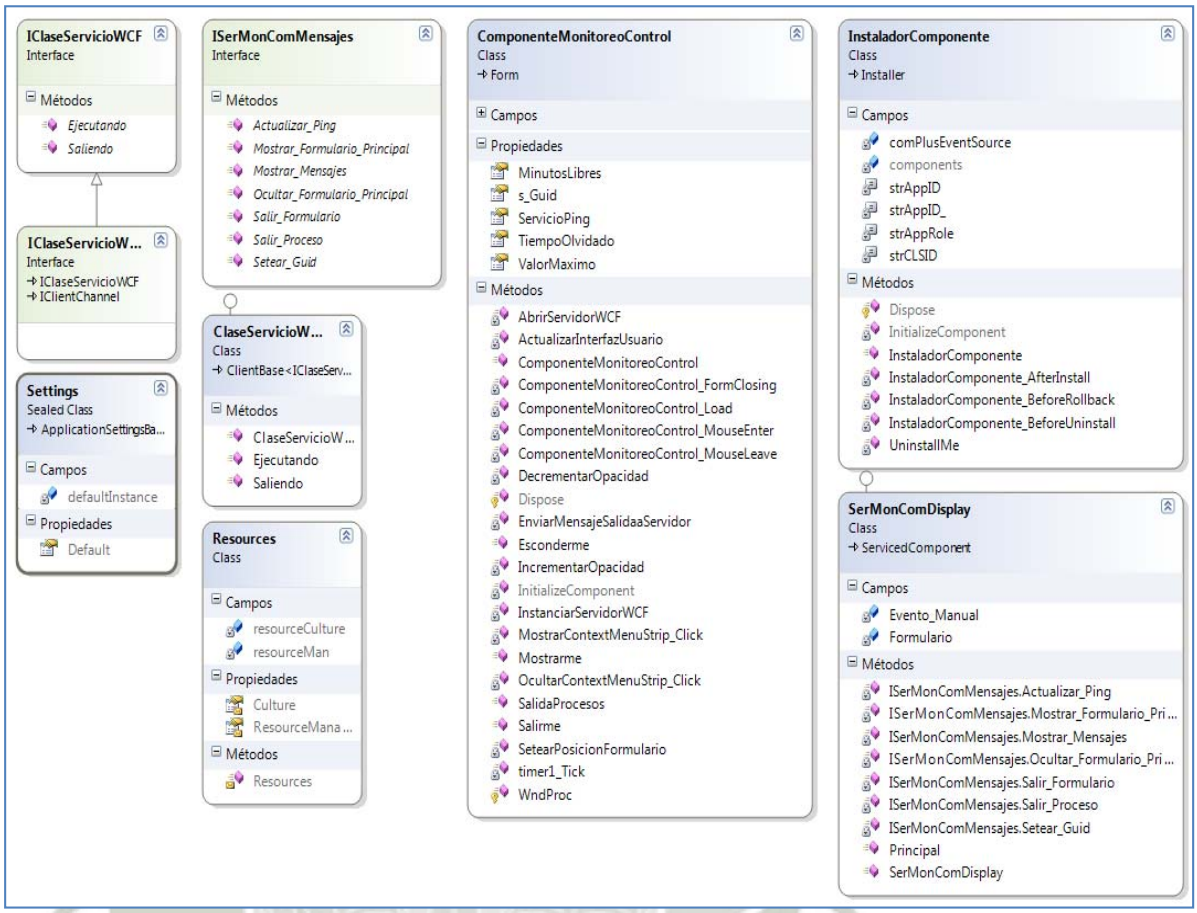


Figura 3.35 Diagrama de Clase Winiucomponente.

Fuente: Elaboración Propia.

3.4.6 Diagramas de Paquetes

3.4.6.1 Paquete Administrarservicioscs

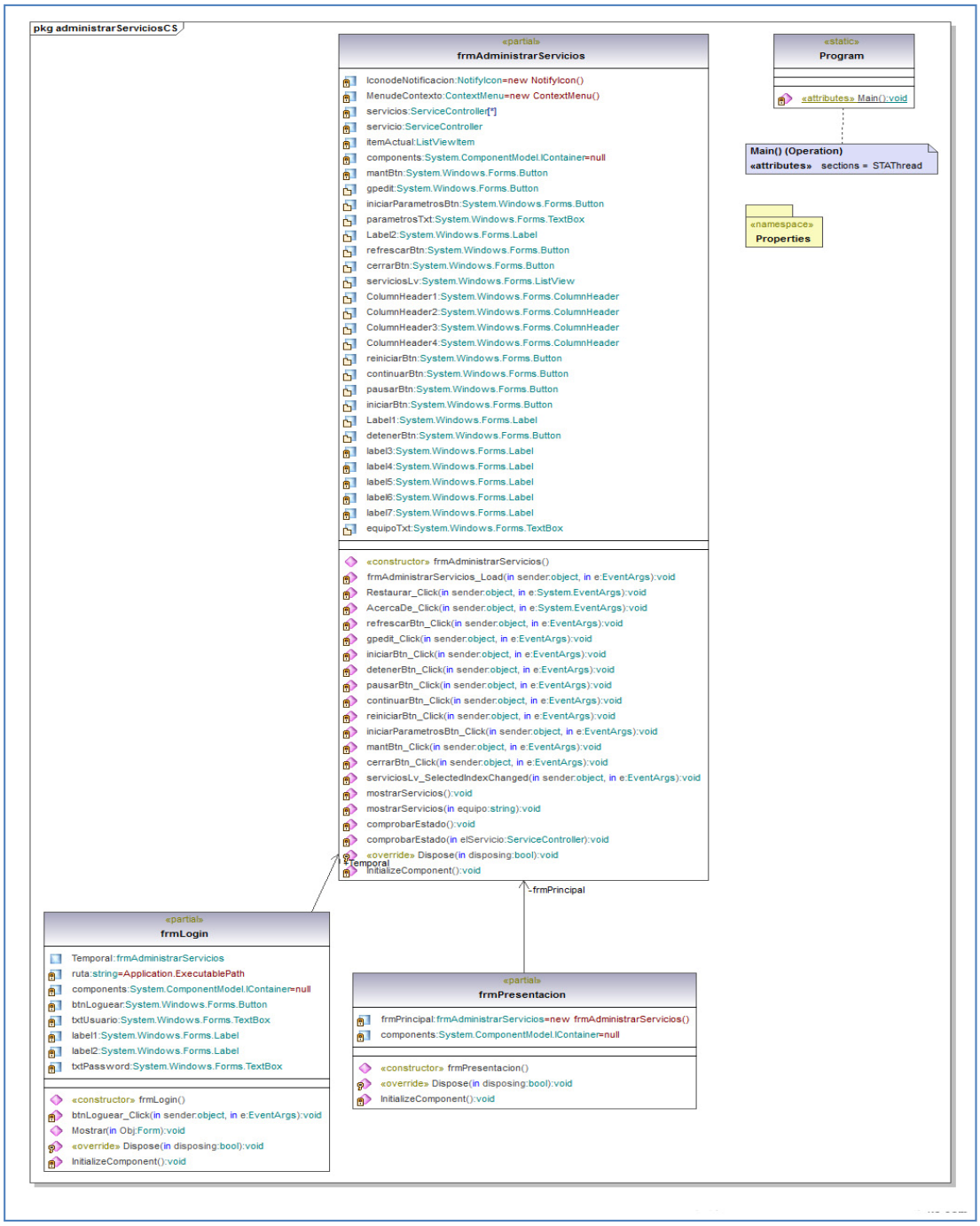


Figura 3.36 Diagrama de Paquete Administrar Servicios.

Fuente: Elaboración Propia.

3.4.6.2 Paquete Bdcliente

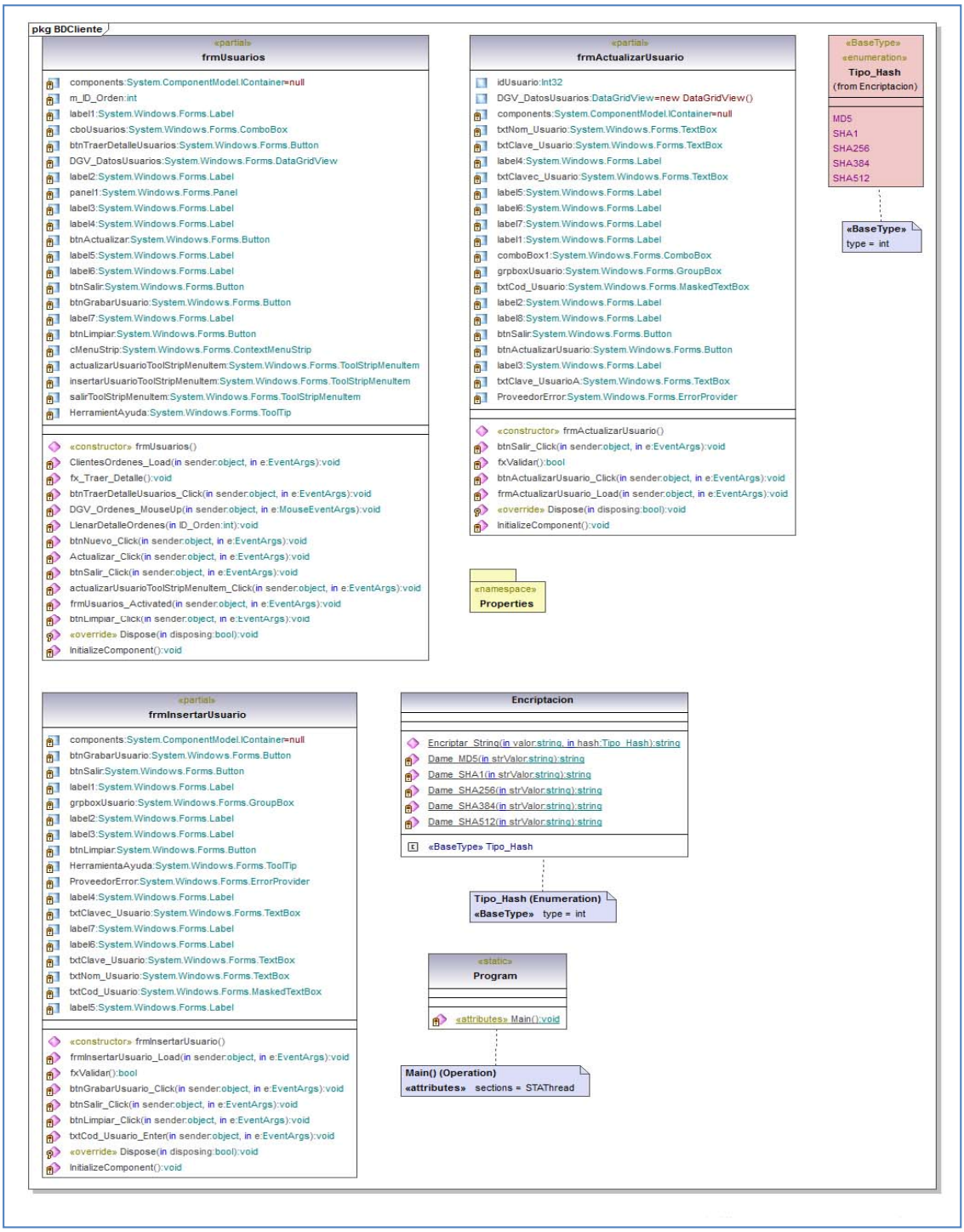


Figura 3.37 Diagrama de Paquete BDCliente.

Fuente: Elaboración Propia.

3.4.6.3 Paquete Bdcomponente

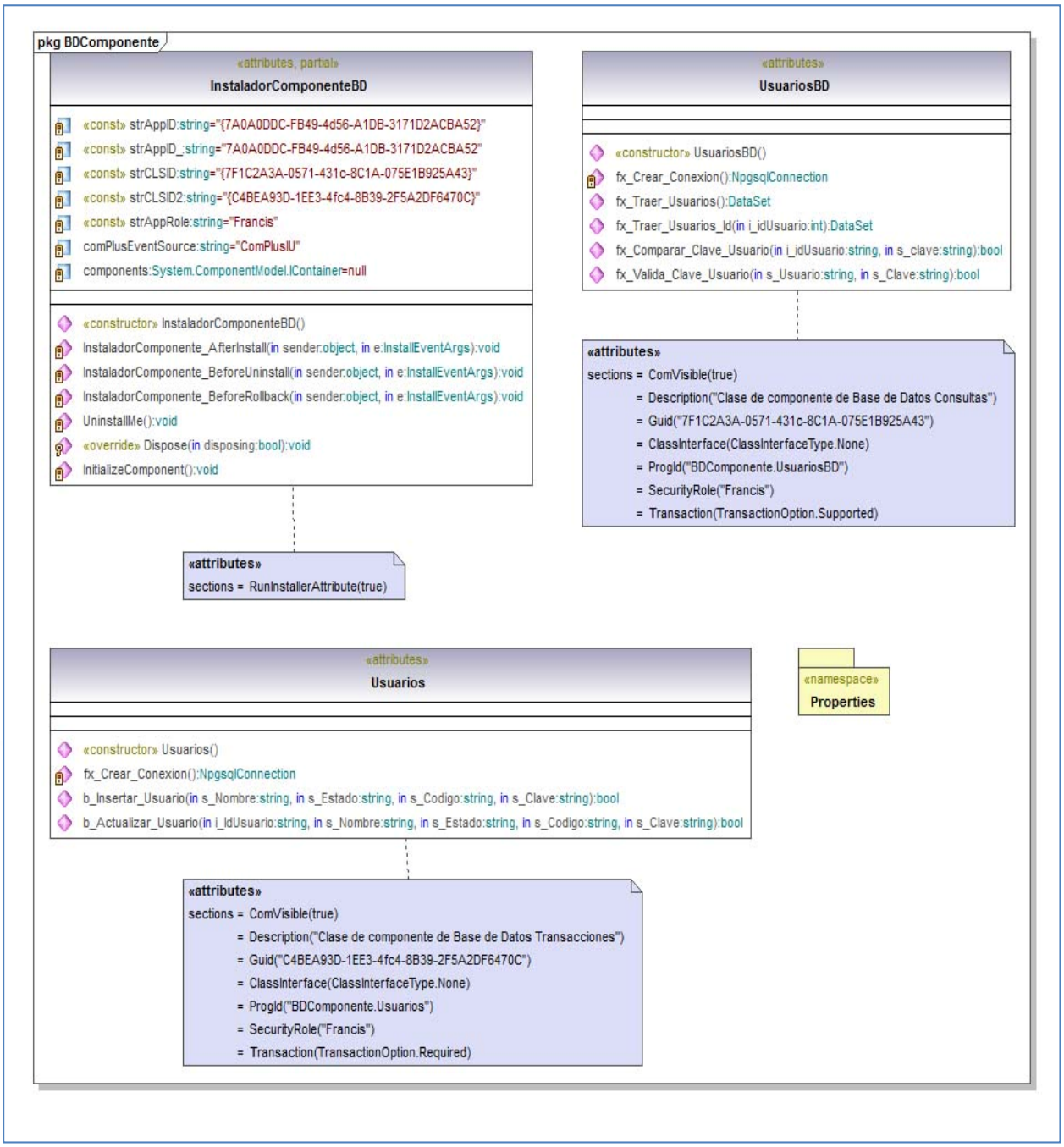


Figura 3.38 - Diagrama de Paquete BDComponente.

Fuente: Elaboración Propia.

3.4.6.4 Paquete Serviciomonitoreo

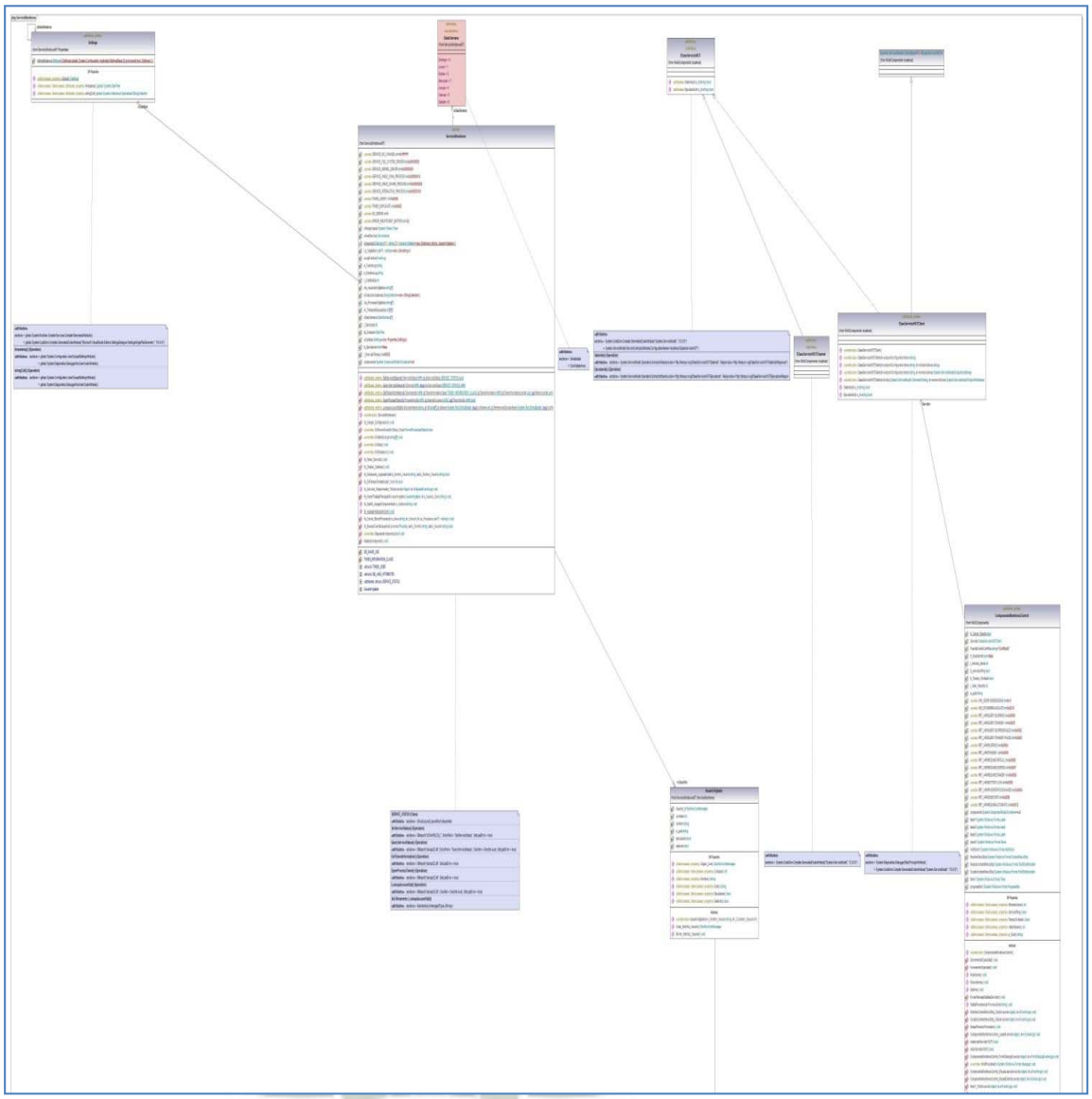


Figura 3.39 - Diagrama de Paquete Servicio Monitoreo – Parte 1.

Fuente: Elaboración Propia.

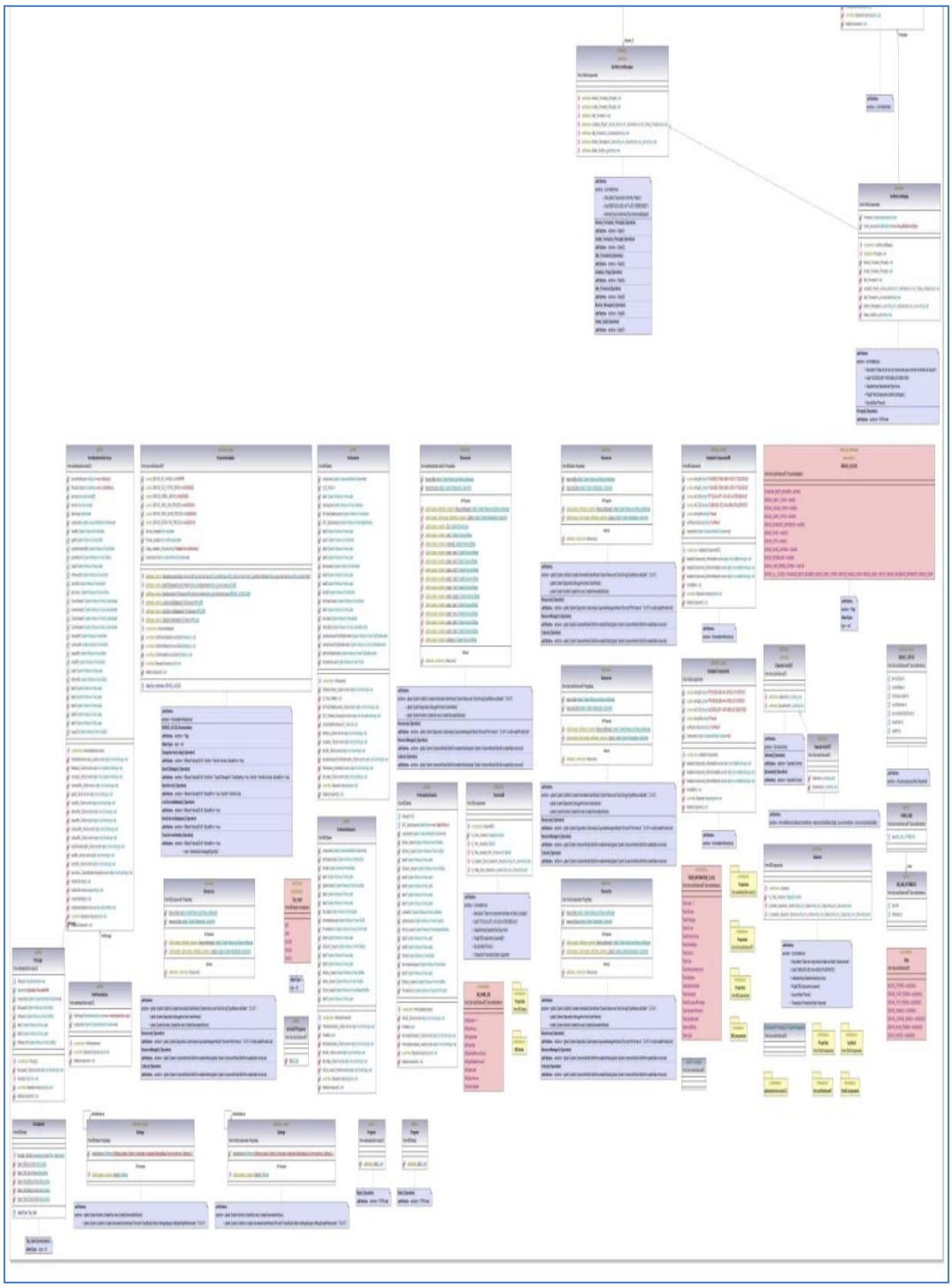


Figura 3.40 - Diagrama de Paquete Servicio Monitoreo – Parte 2.

Fuente: Elaboración Propia.

3.4.6.5 Paquete ServicioWindowsNT

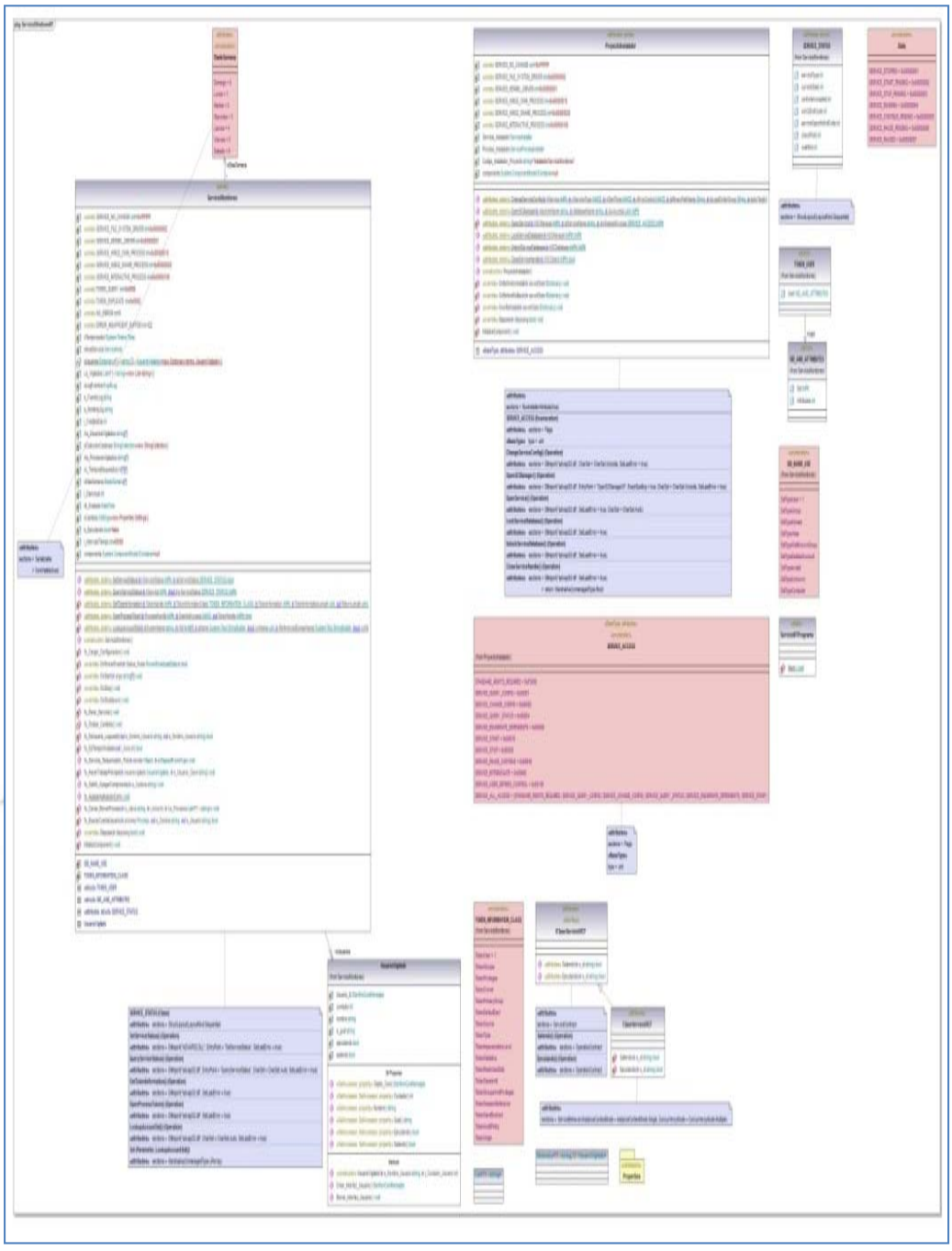


Figura 3.41 - Diagrama de Paquete ServicioWindowsNT.

Fuente: Elaboración Propia.

3.4.6.6 Paquete WinIUComponente

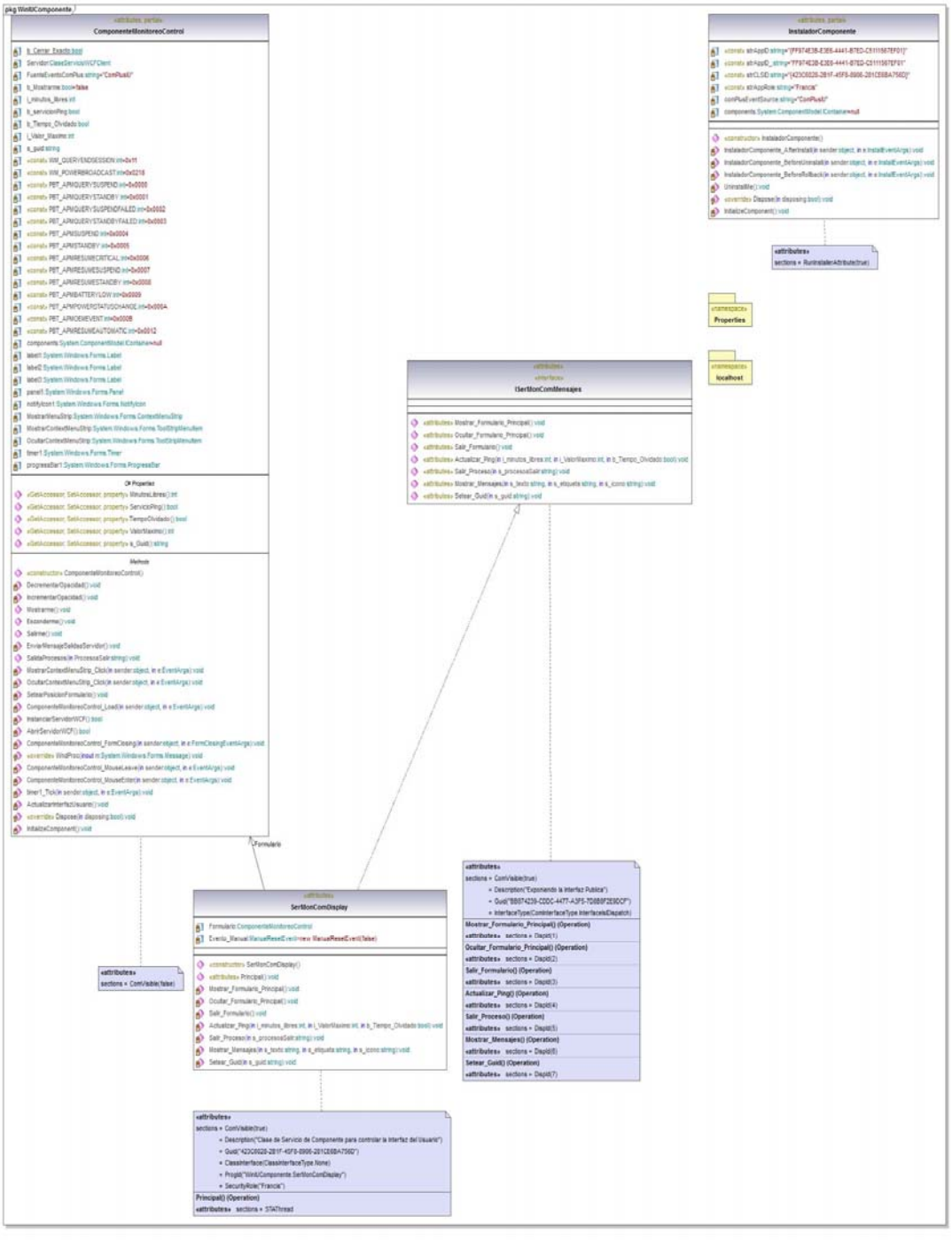


Figura 3.42 - Diagrama de Paquete WinUComponente.

Fuente: Elaboración Propia.

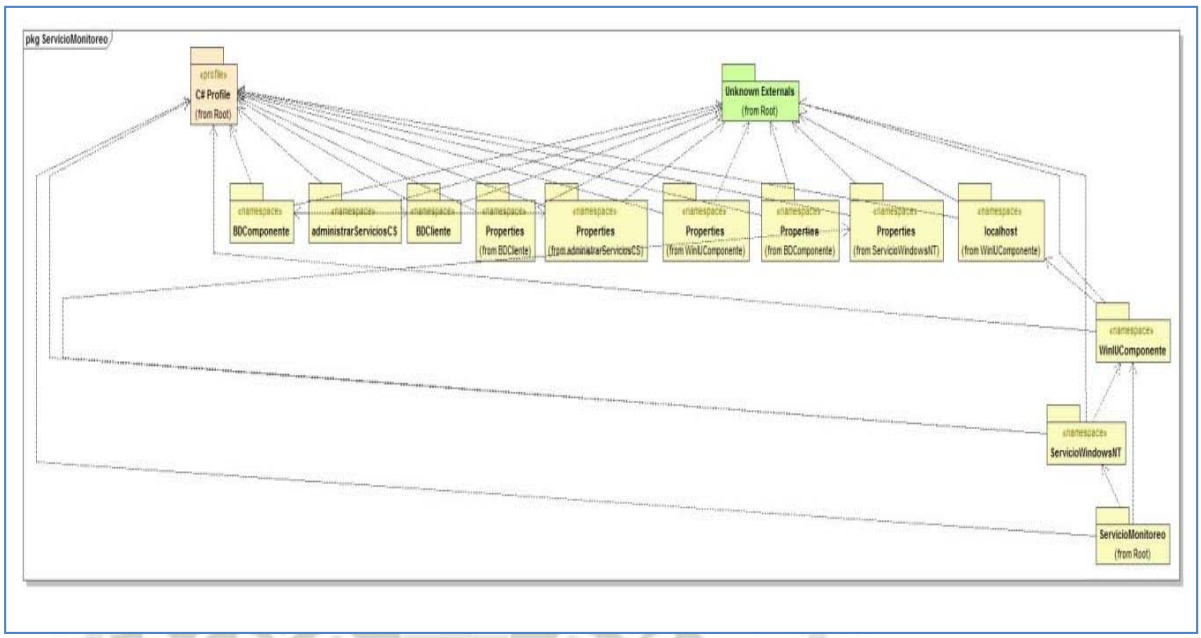


Figura 3.43 - Diagrama de Paquete ServicioMonitoreo.

Fuente: Elaboración Propia.

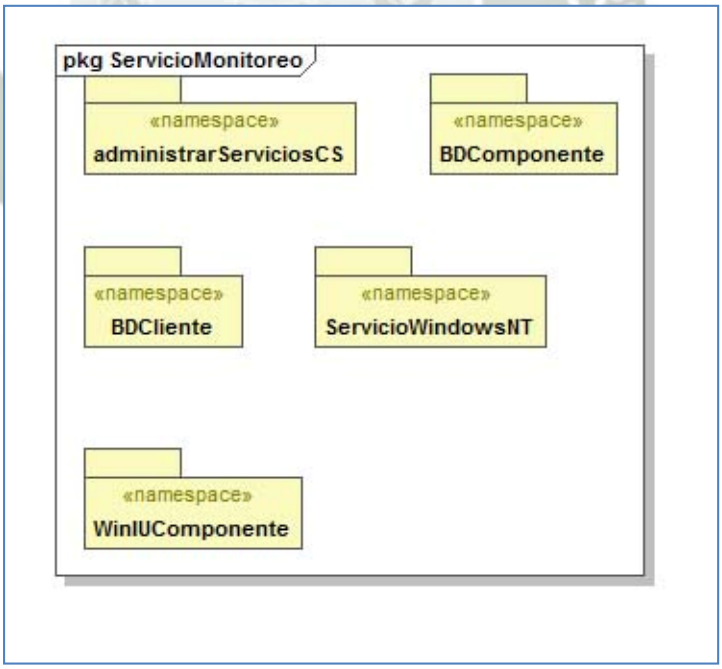


Figura 3.44 - Diagrama de Paquete ServicioMonitoreo.

Fuente: Elaboración Propia.

3.4.7 Diagramas de Secuencia

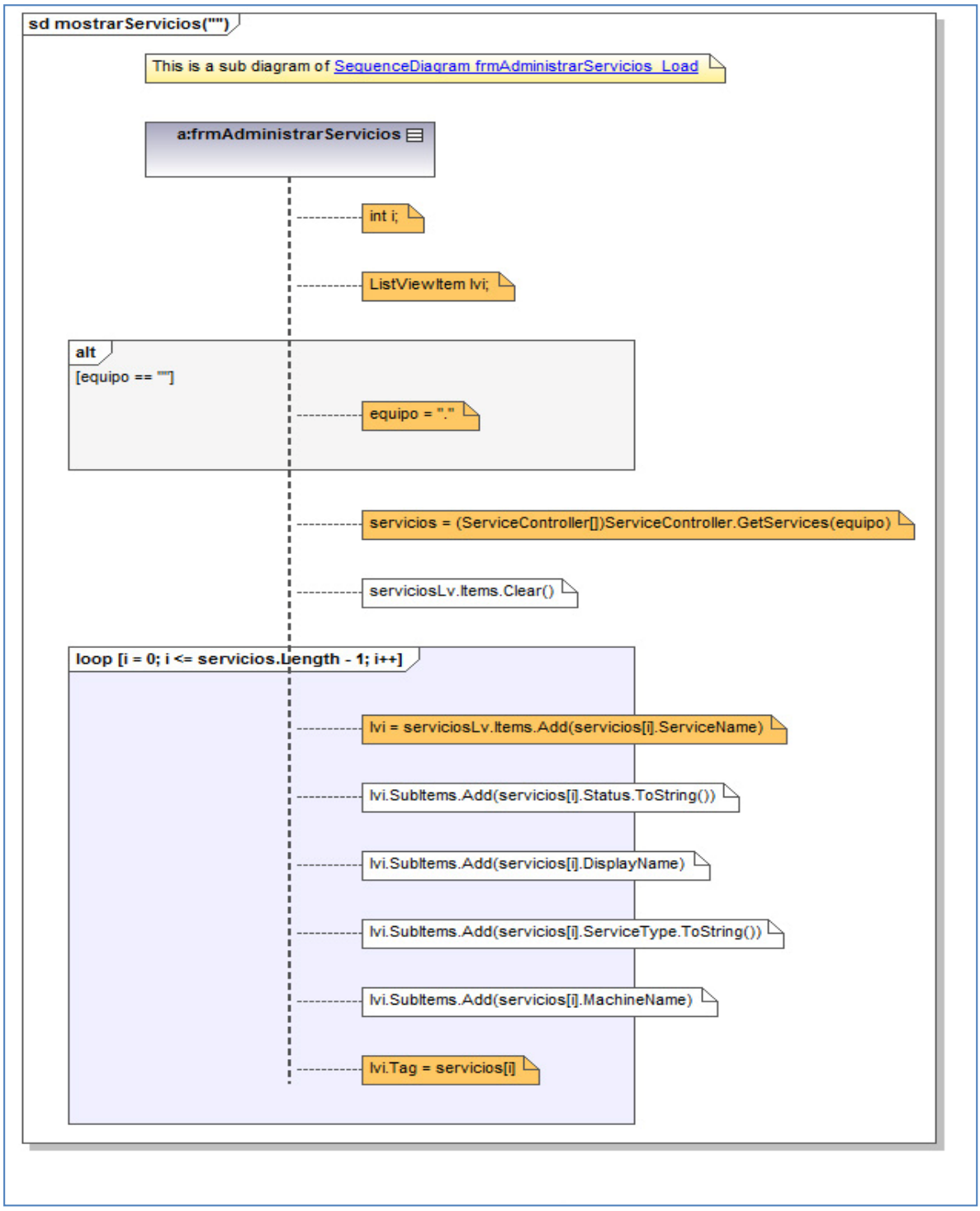


Figura 3.45 – Diagrama de Secuencia MostrarServicios.

Fuente: Elaboración Propia.

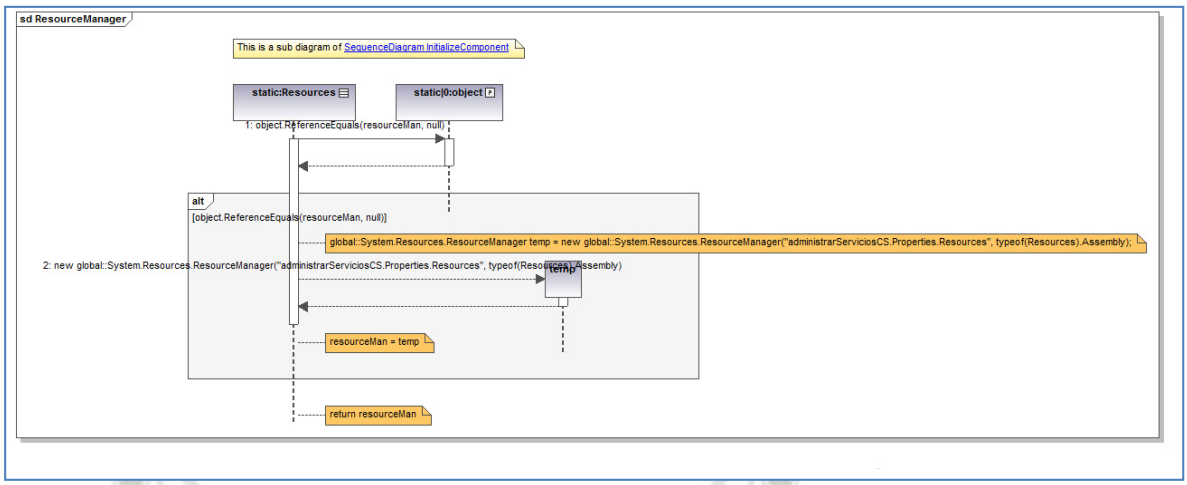


Figura 3.46 – Diagrama de Secuencia Administrador de Recursos.

Fuente: Elaboración Propia.

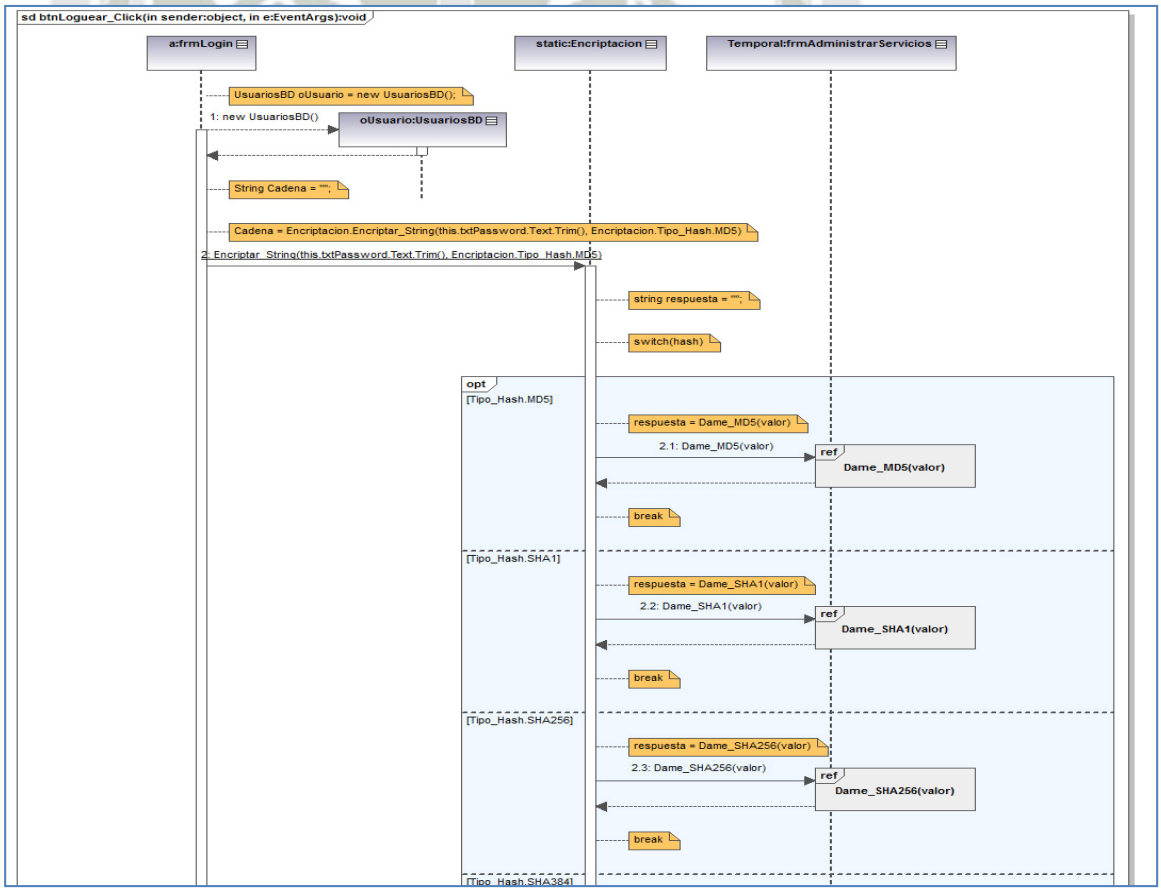


Figura 3.47 – Diagrama de Secuencia Loguear_Click (Parte 1).

Fuente: Elaboración Propia.

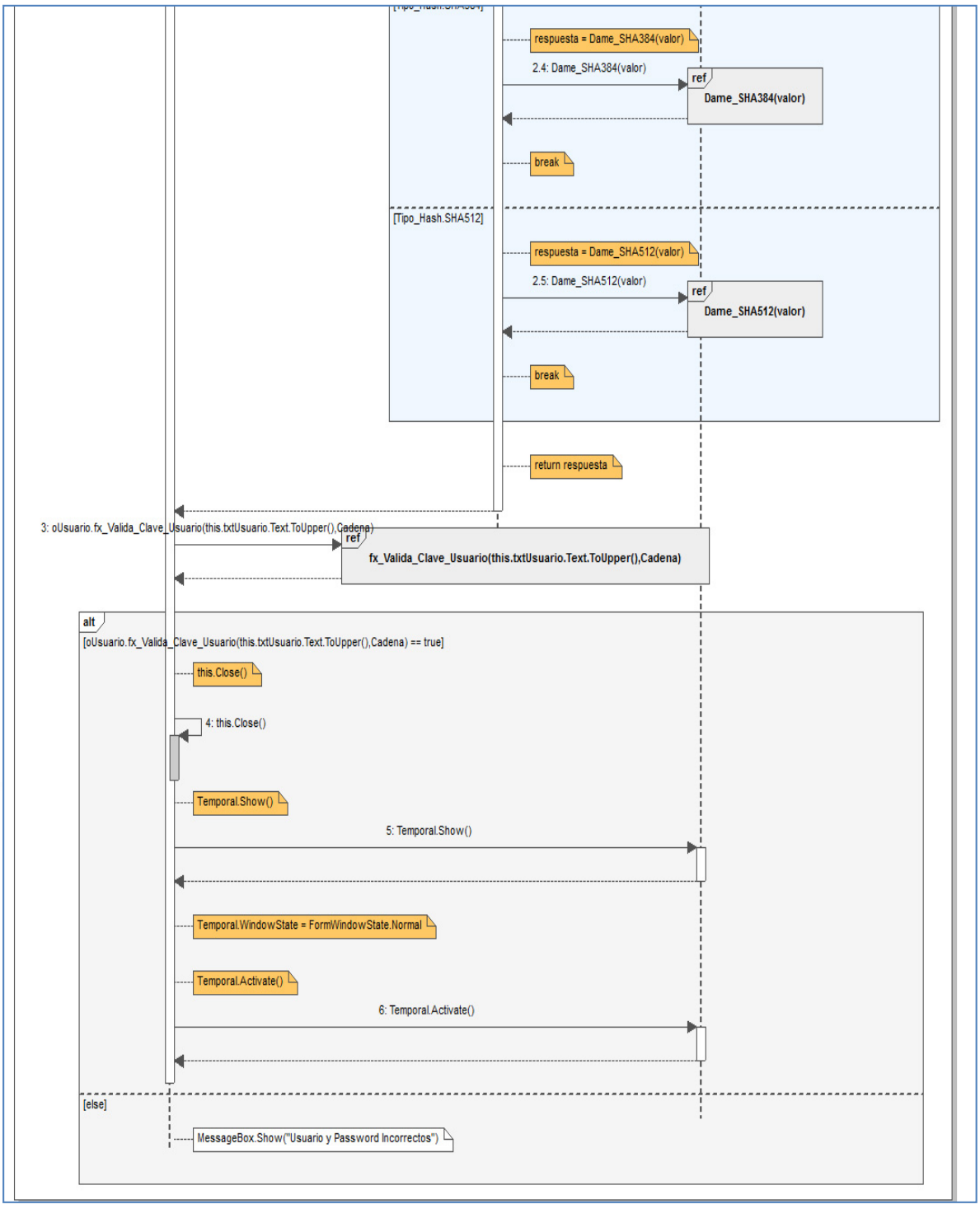


Figura 3.48 – Diagrama de Secuencia Loguear_Click (Parte 2).

Fuente: Elaboración Propia.

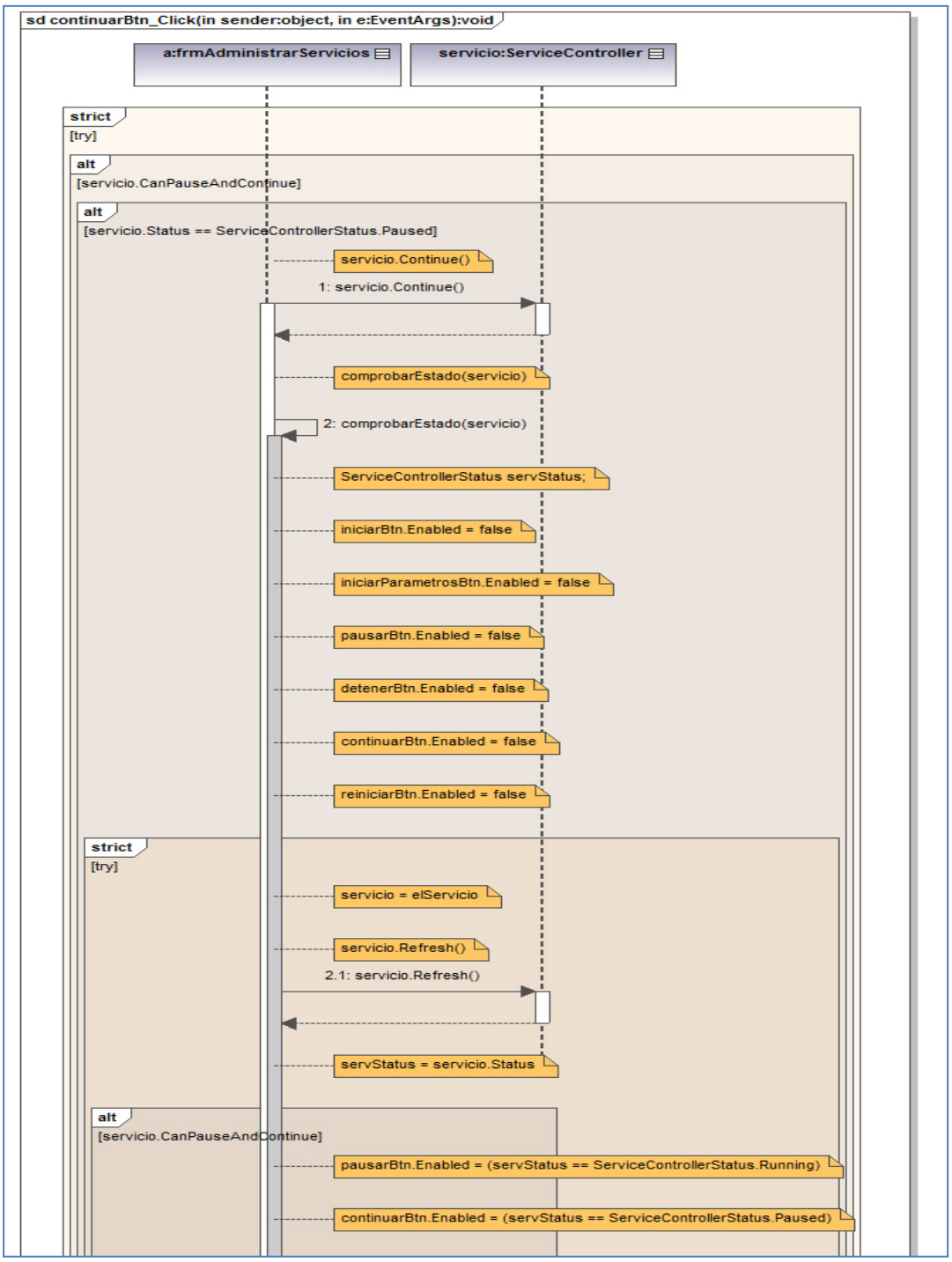


Figura 3.49 – Diagrama de Secuencia Continuar_Click (Parte 1).

Fuente: Elaboración Propia.

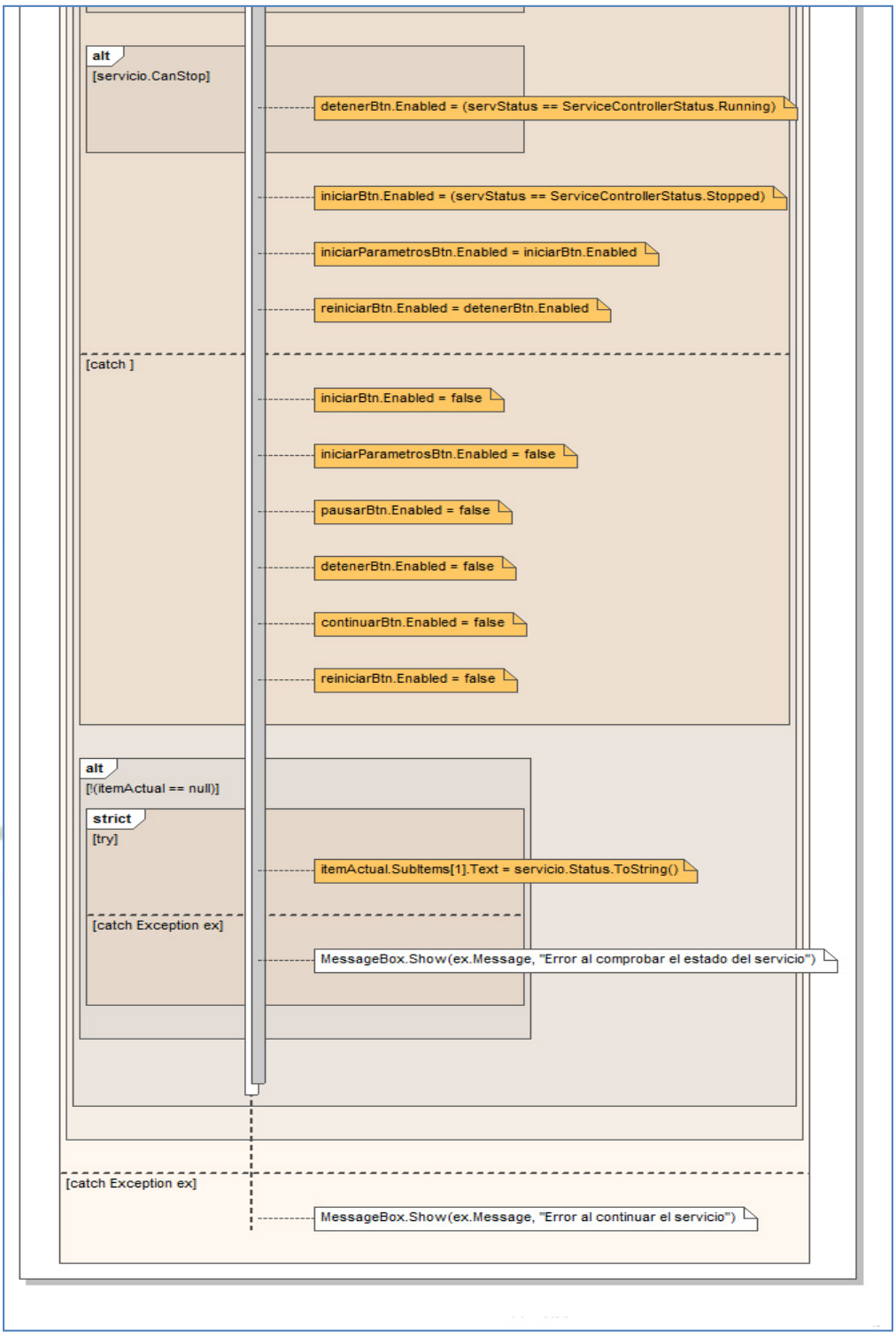


Figura 3.50 – Diagrama de Secuencia Continuar_Click (Parte 2).

Fuente: Elaboración Propia.

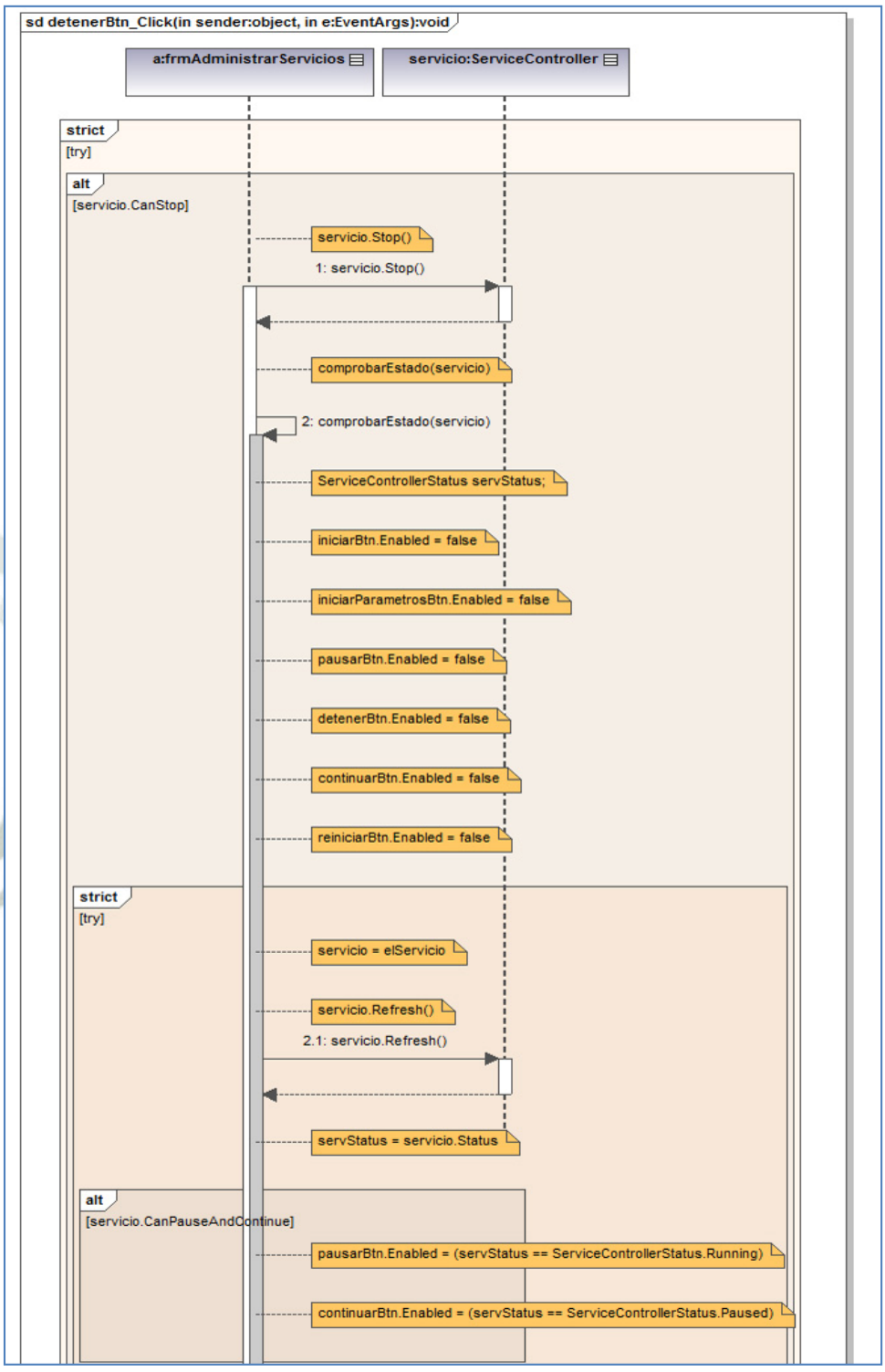


Figura 3.51 – Diagrama de Secuencia Detener_Click (Parte 1).

Fuente: Elaboración Propia.

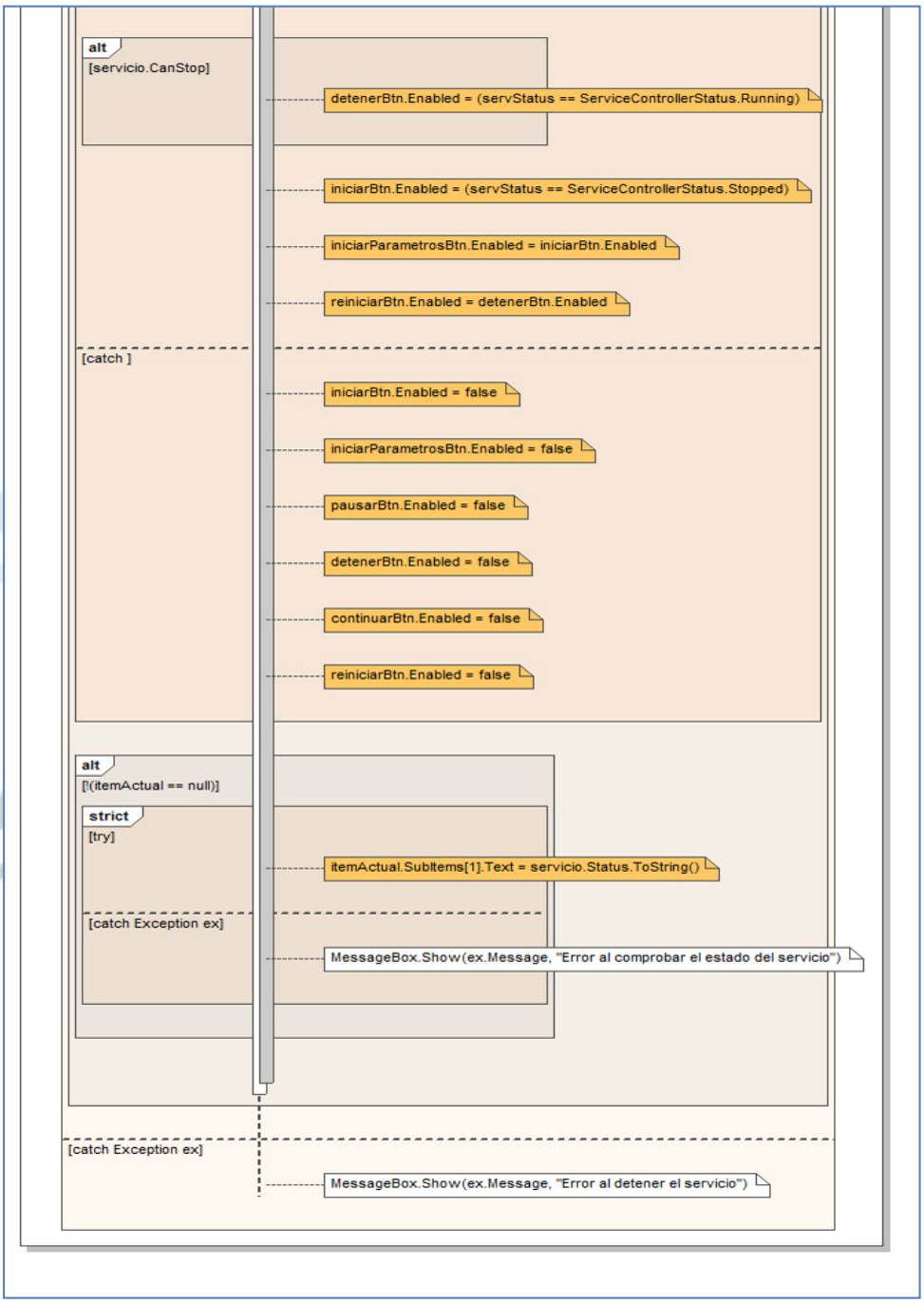


Figura 3.52 – Diagrama de Secuencia Detener_Click (Parte 2).

Fuente: Elaboración Propia.

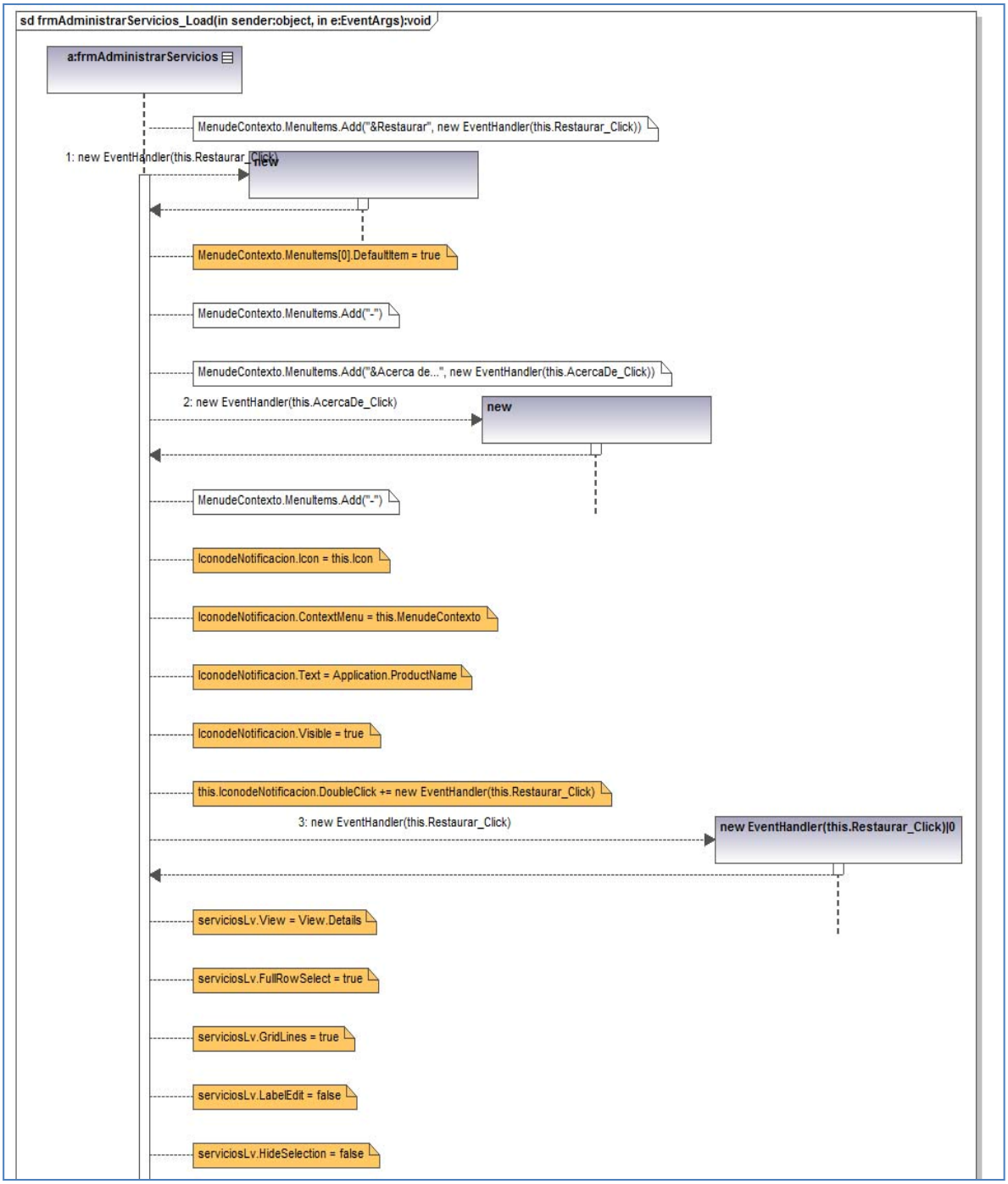


Figura 3.53 – Diagrama de Secuencia FrmAdministrarServicios_Load (Parte 1).

Fuente: Elaboración Propia.

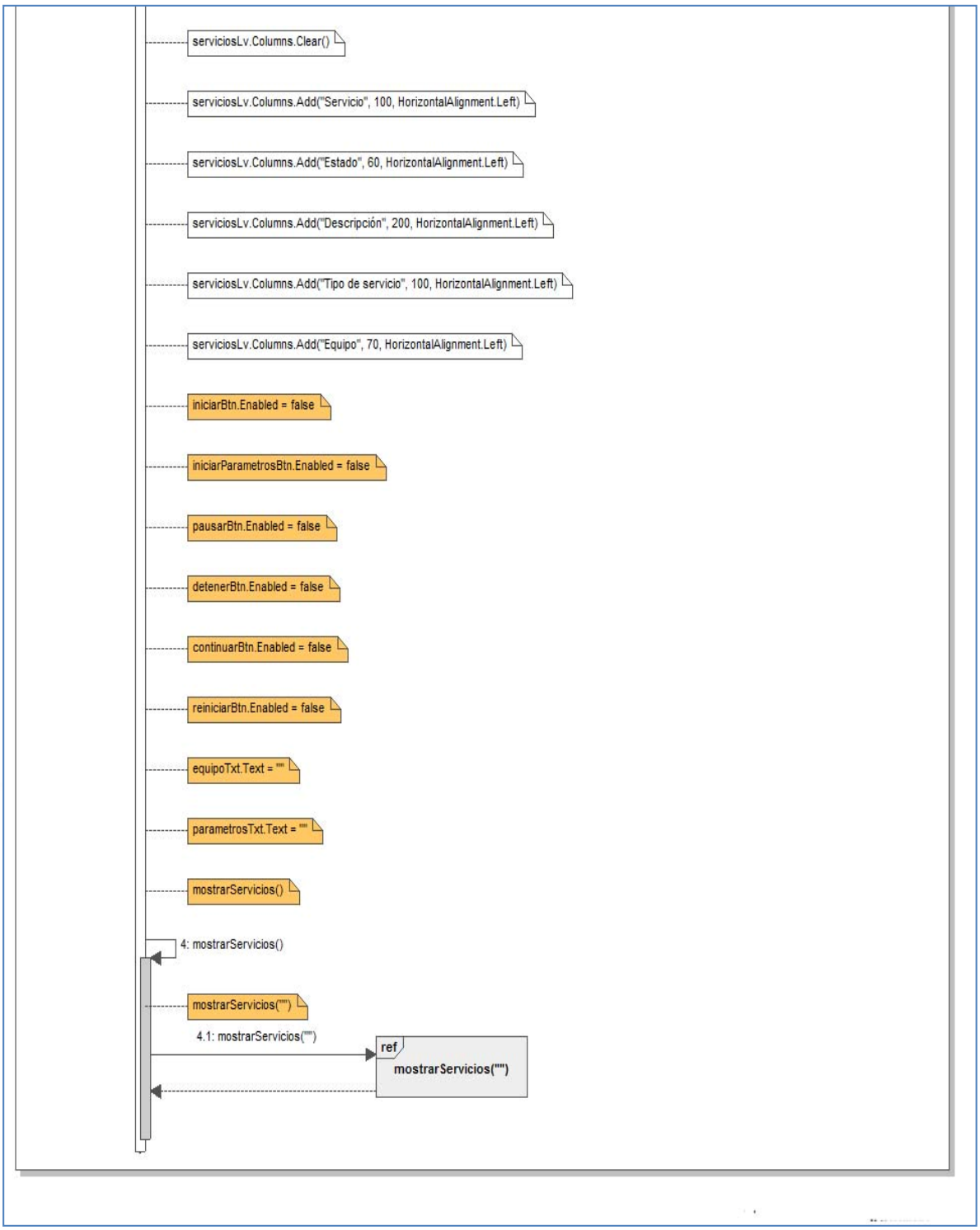


Figura 3.54 – Diagrama de Secuencia FrmAdministrarServicios_Load (Parte 2).

Fuente: Elaboración Propia.

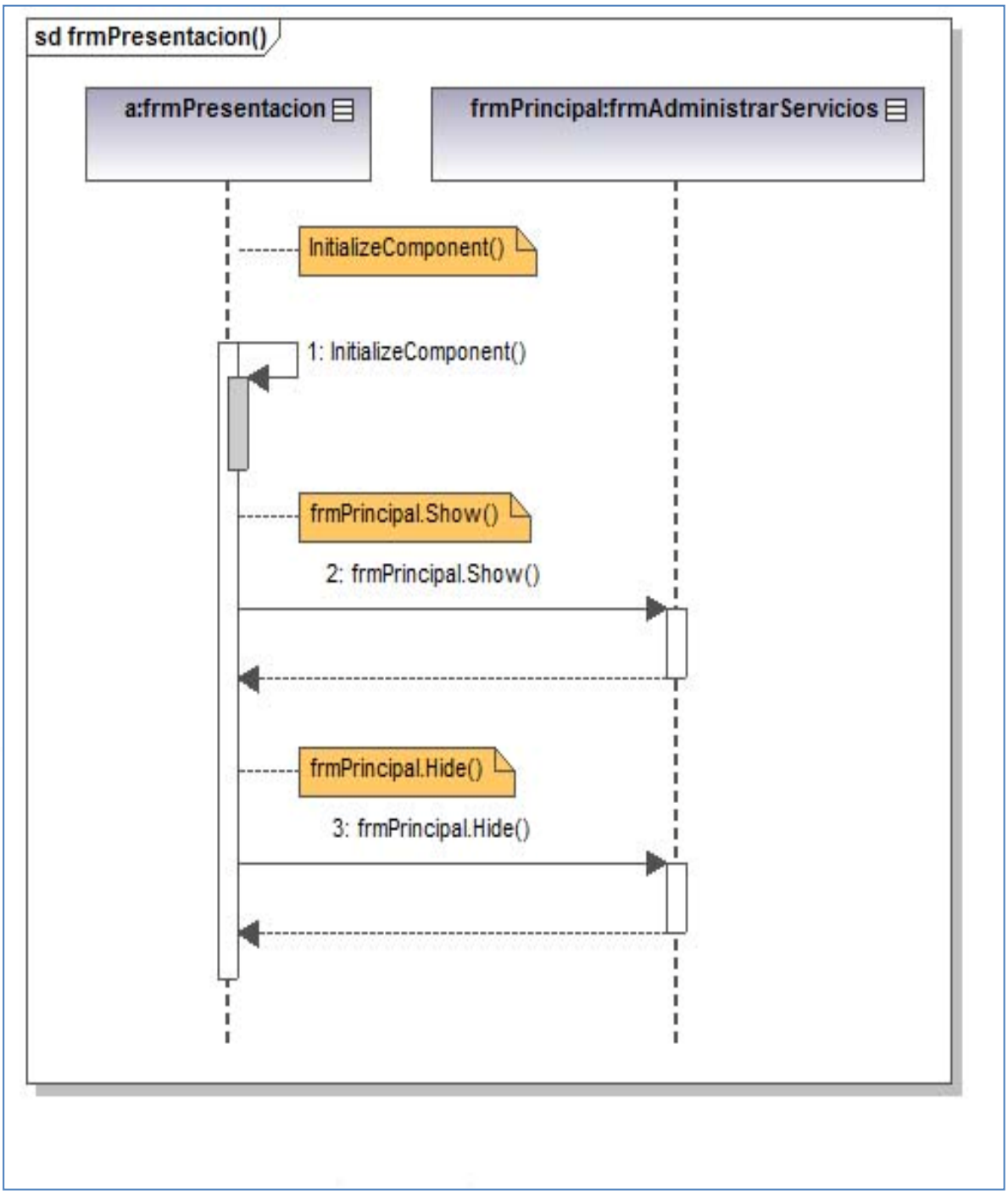


Figura 3.55 – Diagrama de Secuencia FrmPresentacion.

Fuente: Elaboración Propia.

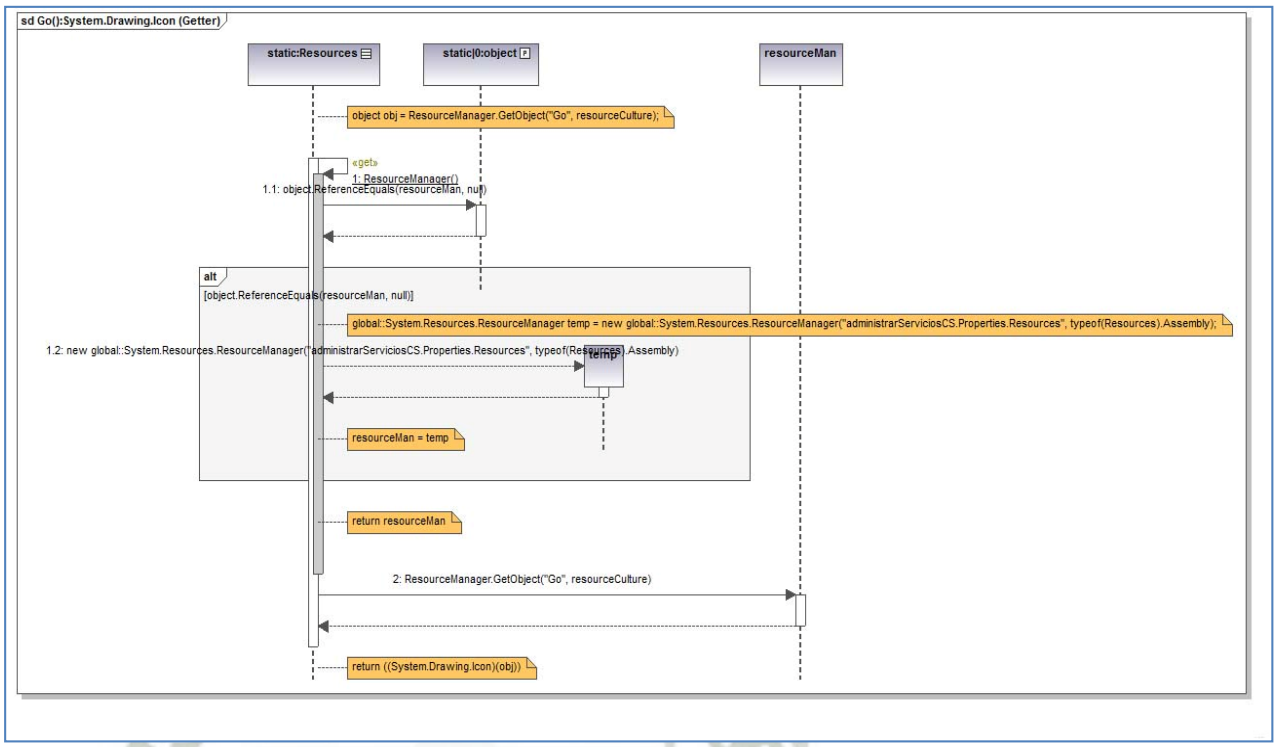


Figura 3.56 – Diagrama de Secuencia Go.

Fuente: Elaboración Propia.

Figura 3.57 – Diagrama de Secuencia gpedit_Click.

Fuente: Elaboración Propia.

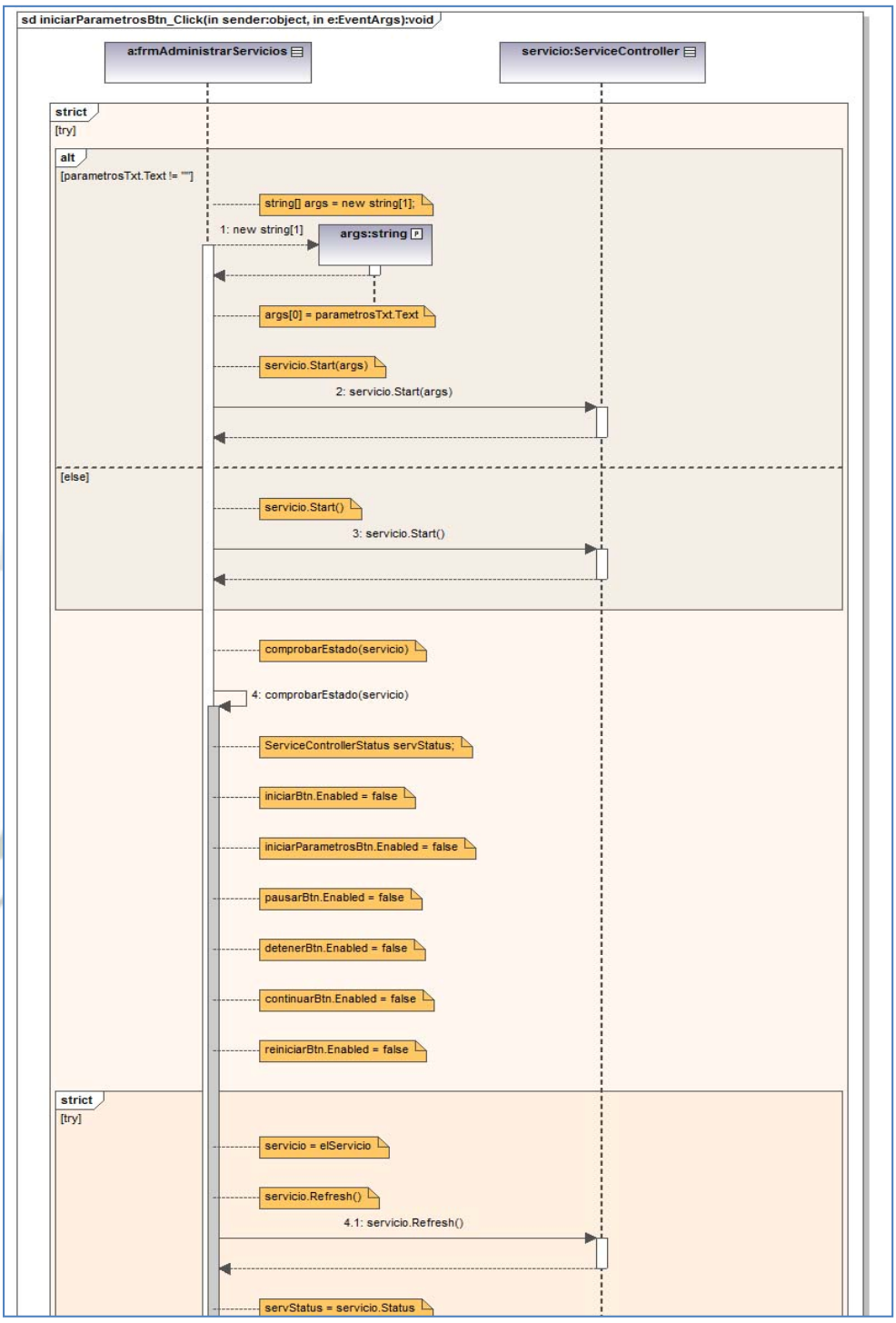


Figura 3.58 – Diagrama de Secuencia iniciarParametrosBtn_Click (Parte 1).

Fuente: Elaboración Propia.

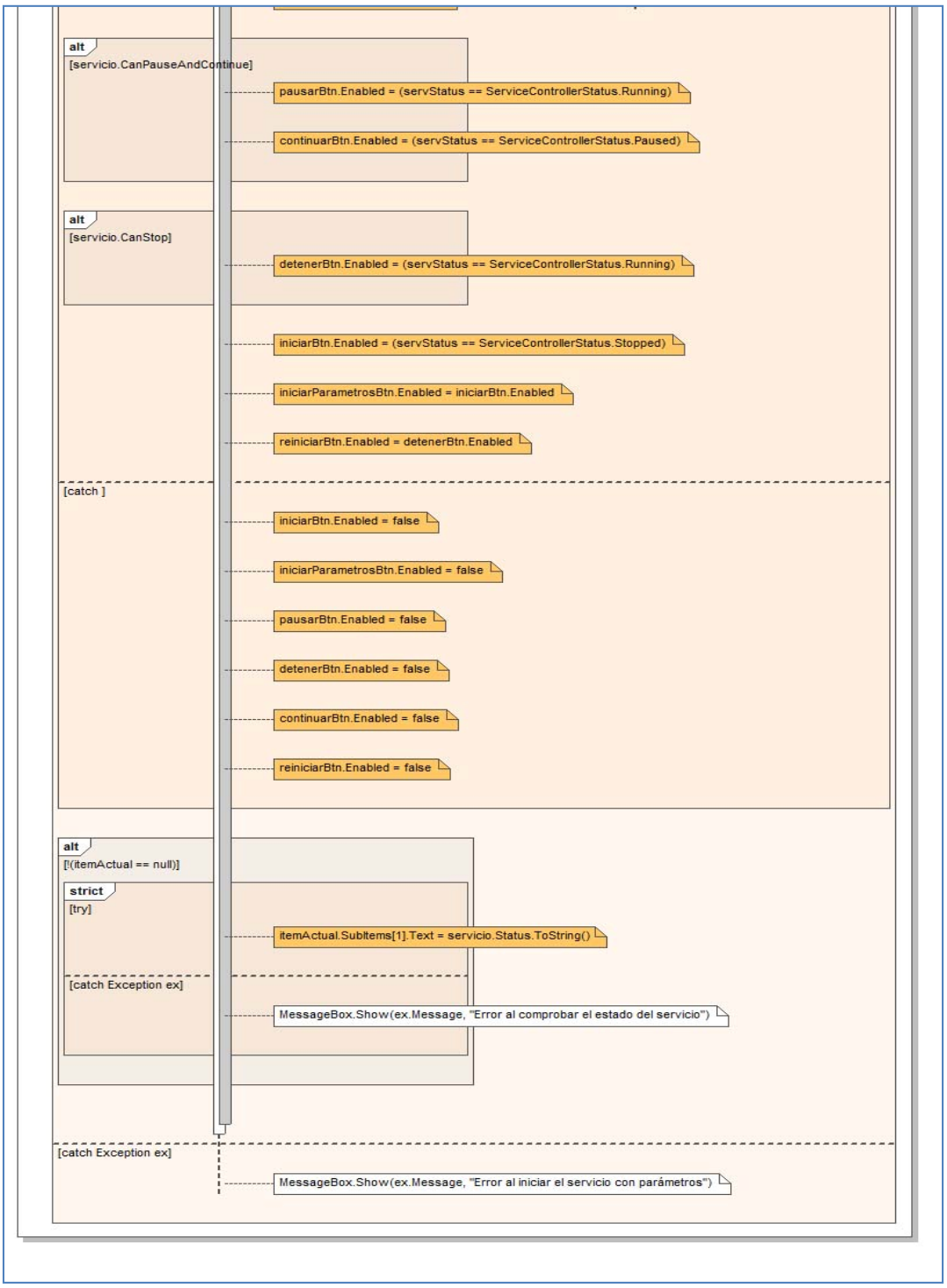


Figura 3.59 – Diagrama de Secuencia iniciarParametrosBtn_Click (Parte 2).

Fuente: Elaboración Propia.

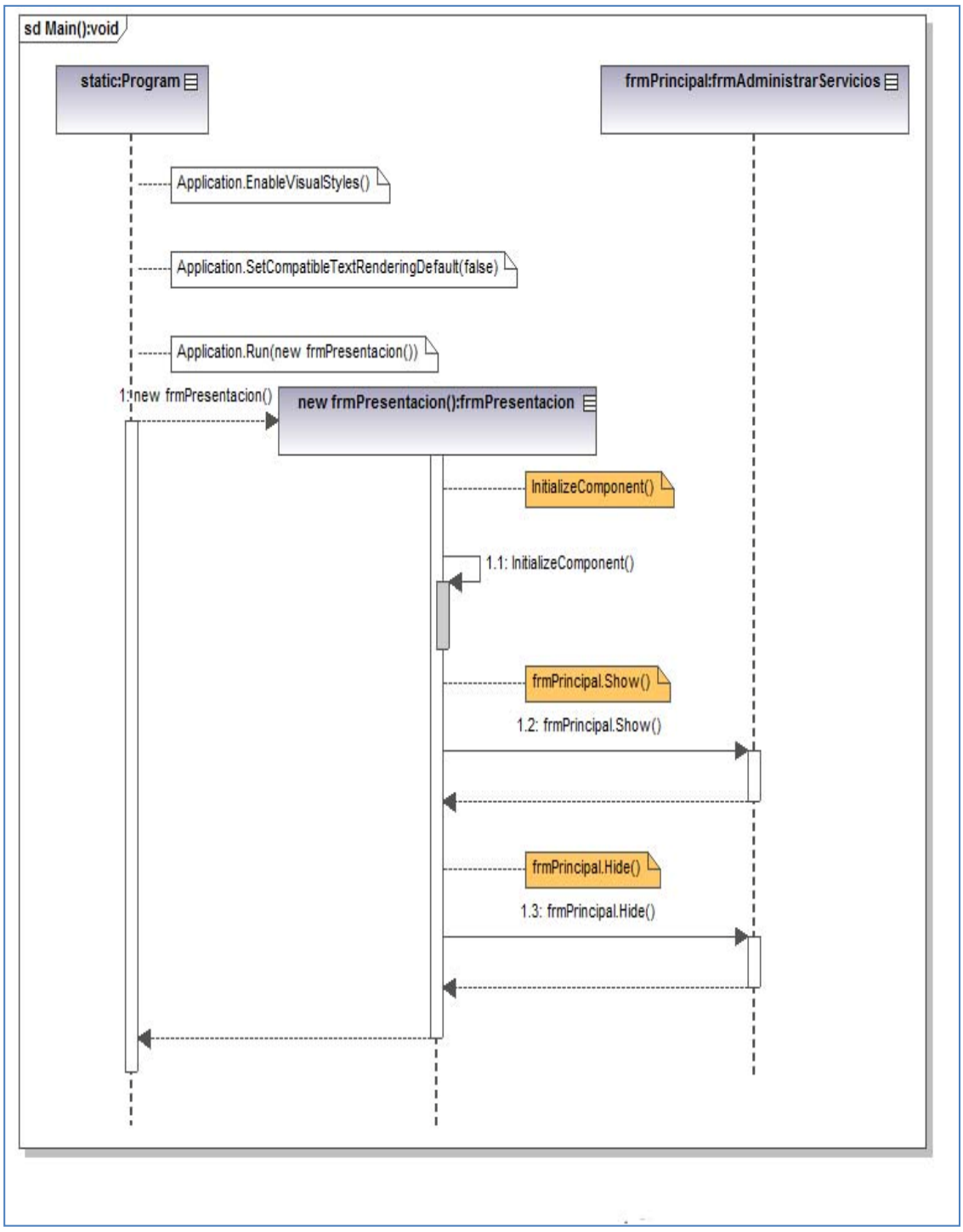


Figura 3.60 – Diagrama de Secuencia Main().

Fuente: Elaboración Propia.

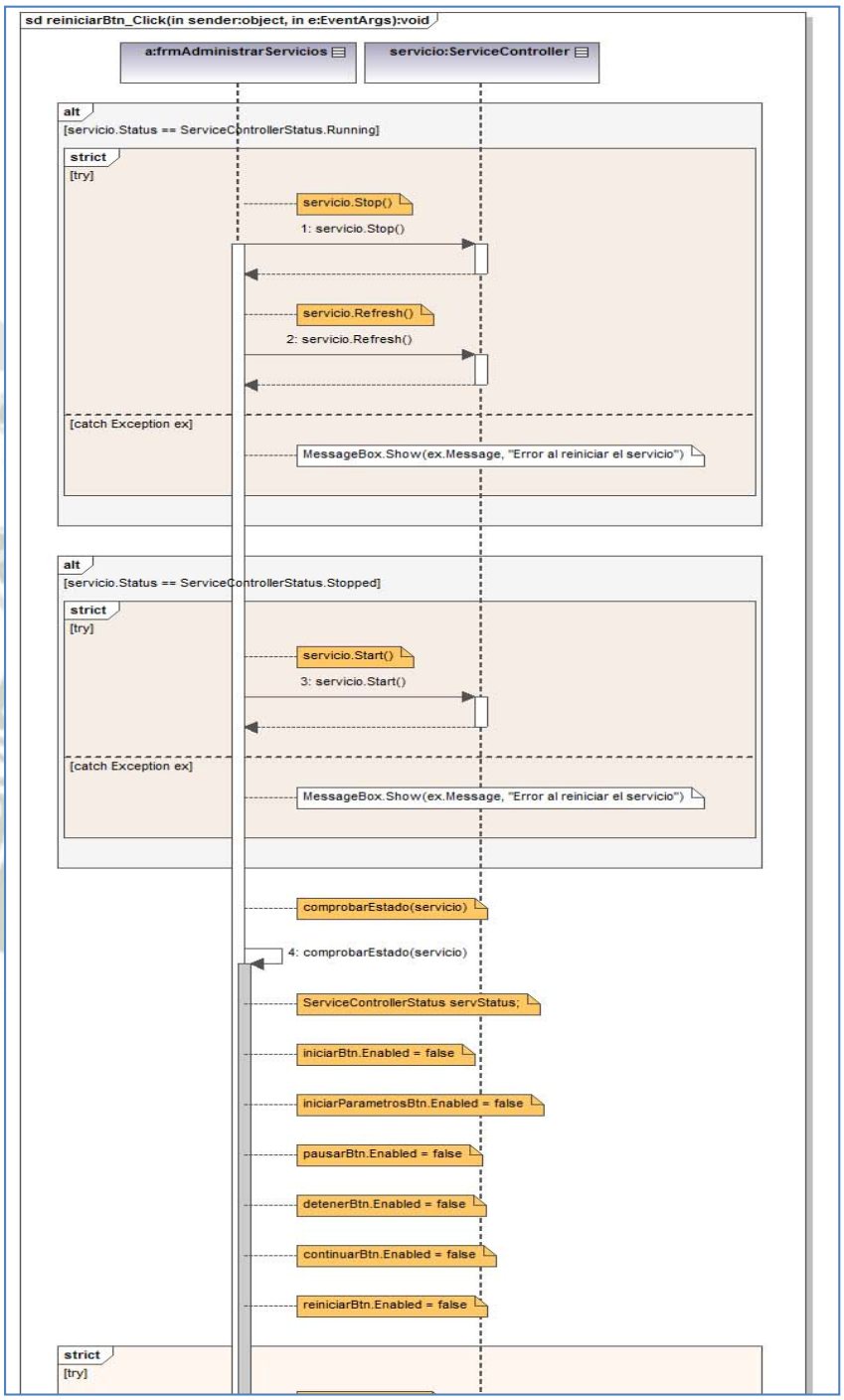


Figura 3.61 – Diagrama de Secuencia Reiniciar_BtnClick (Parte 1).

Fuente: Elaboración Propia.

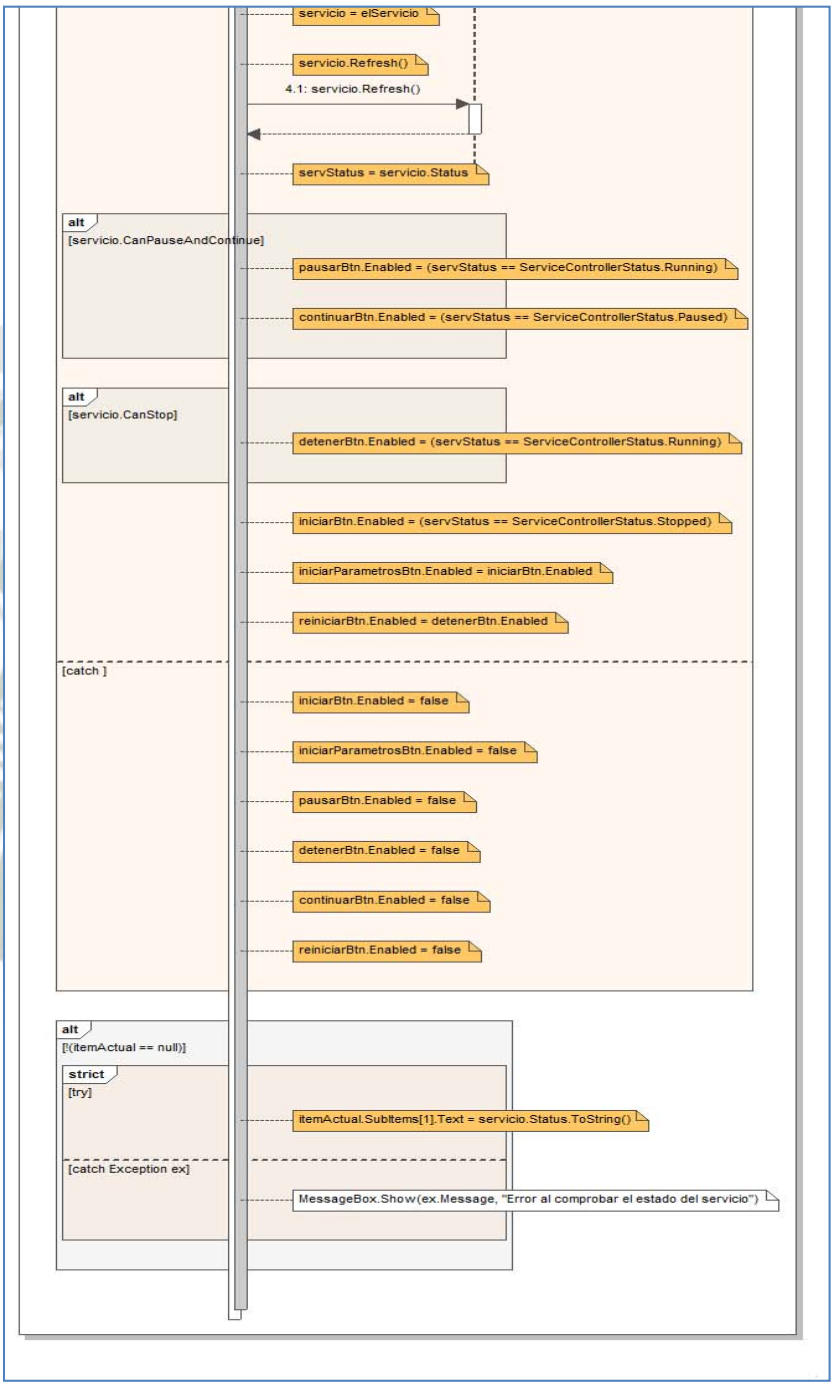


Figura 3.62 – Diagrama de Secuencia Reiniciar_BtnClick (Parte 2).

Fuente: Elaboración Propia.

Figura 3.63 – Diagrama de Secuencia b_Insertar_Usuario.

Fuente: Elaboración Propia.

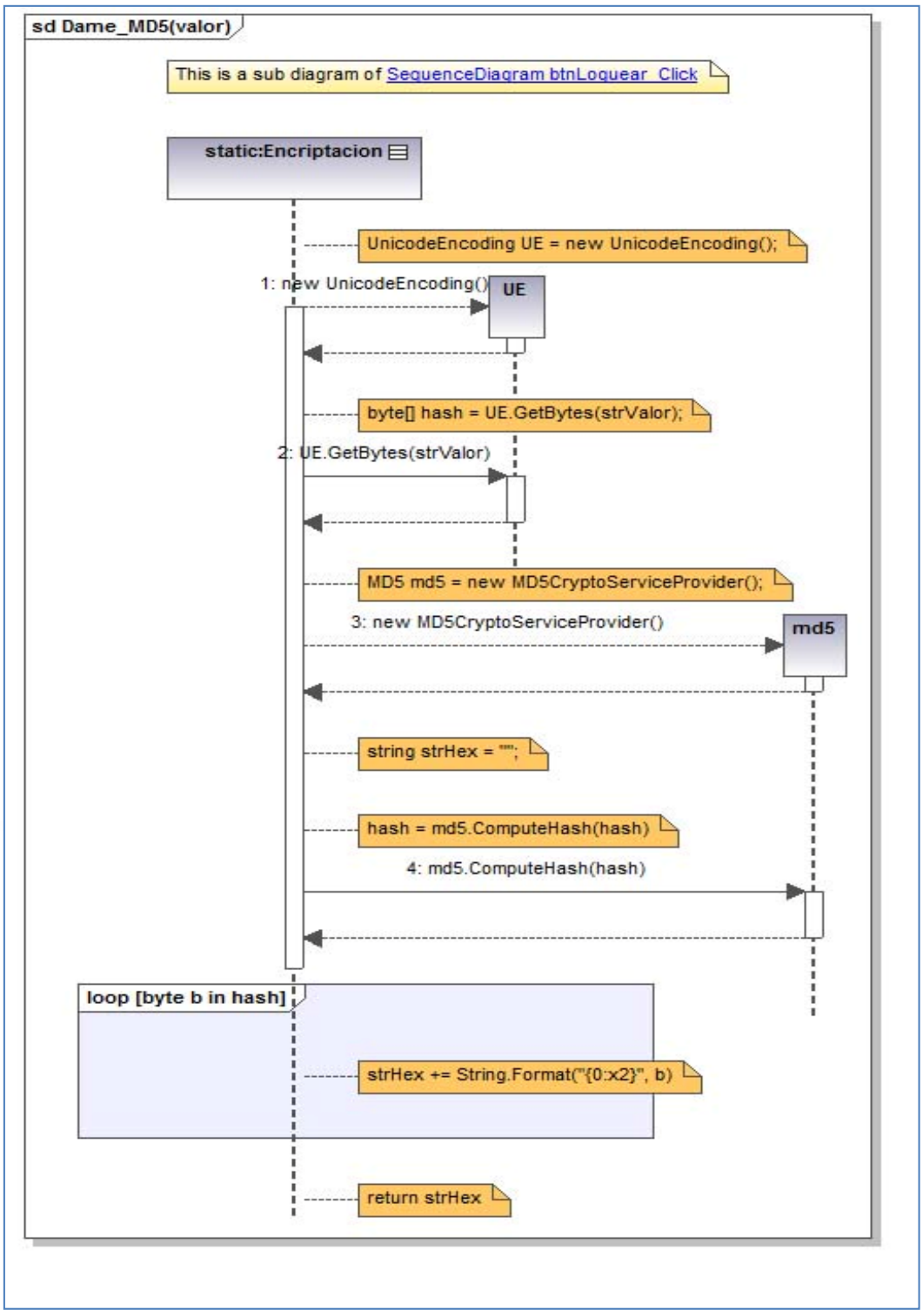


Figura 3.64 – Diagrama de Secuencia Dame_MD5.

Fuente: Elaboración Propia.

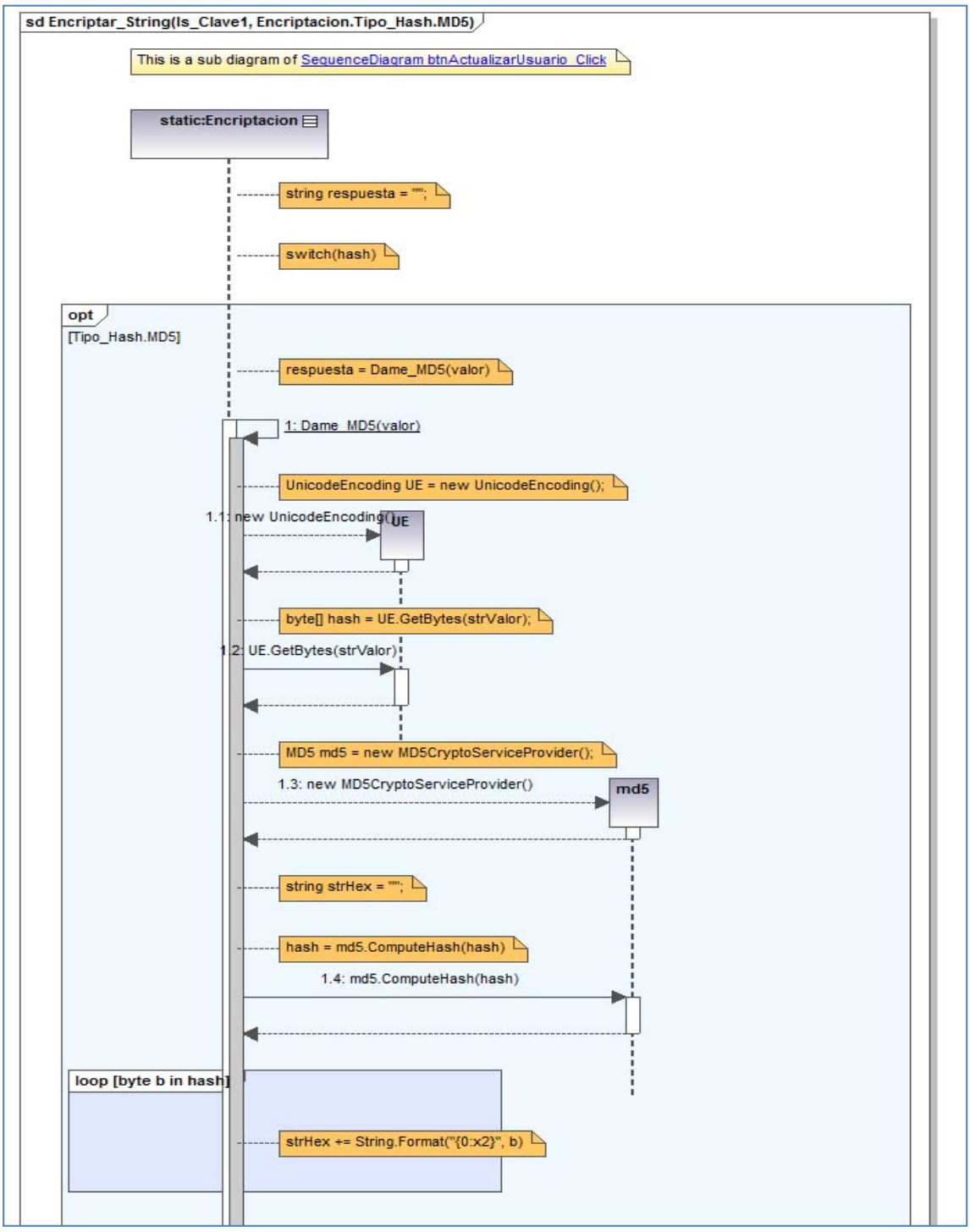


Figura 3.65 – Diagrama de Secuencia Encriptar_String (Parte 1).

Fuente: Elaboración Propia.

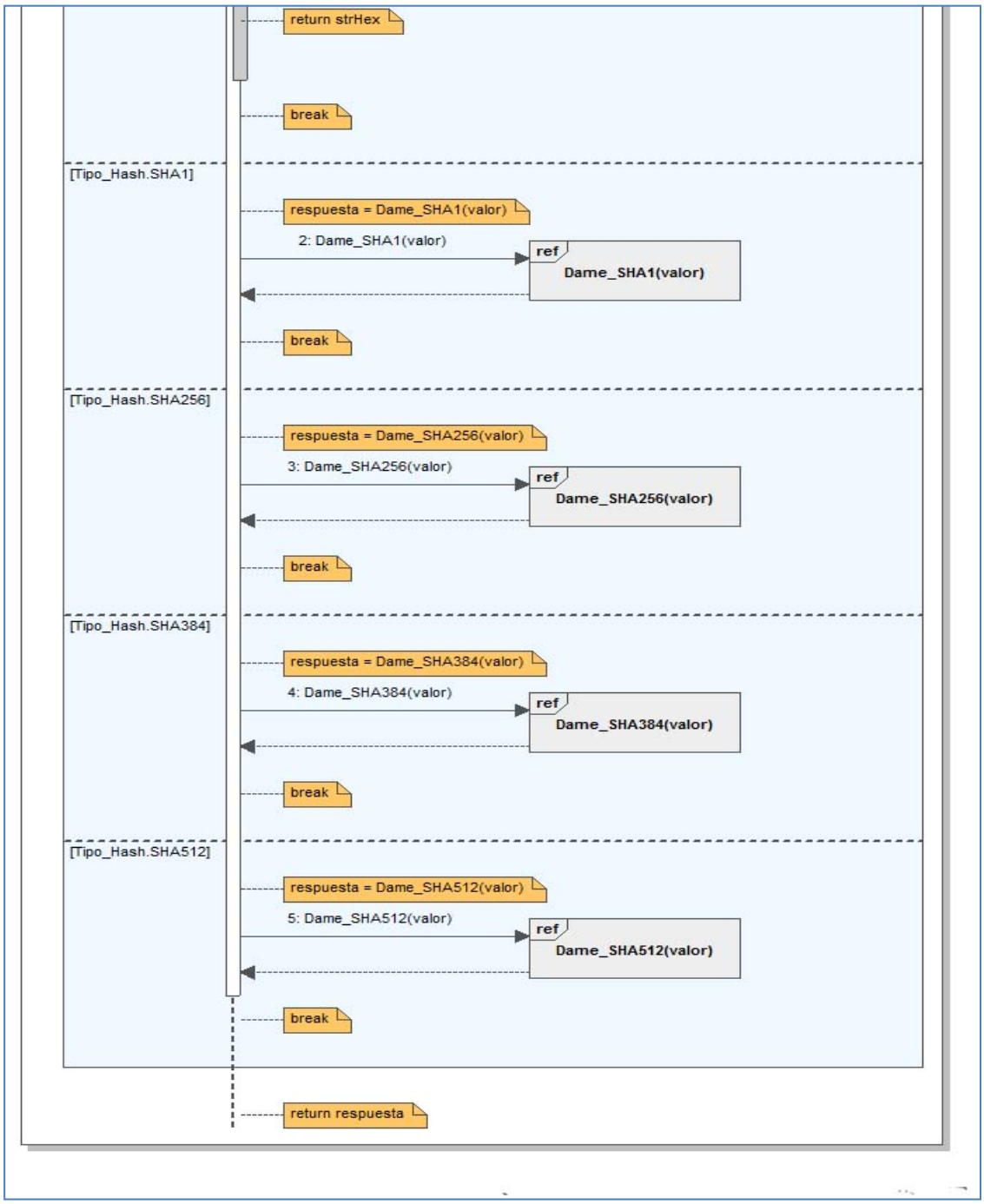


Figura 3.66 – Diagrama de Secuencia Encriptar_String (Parte 2).

Fuente: Elaboración Propia.

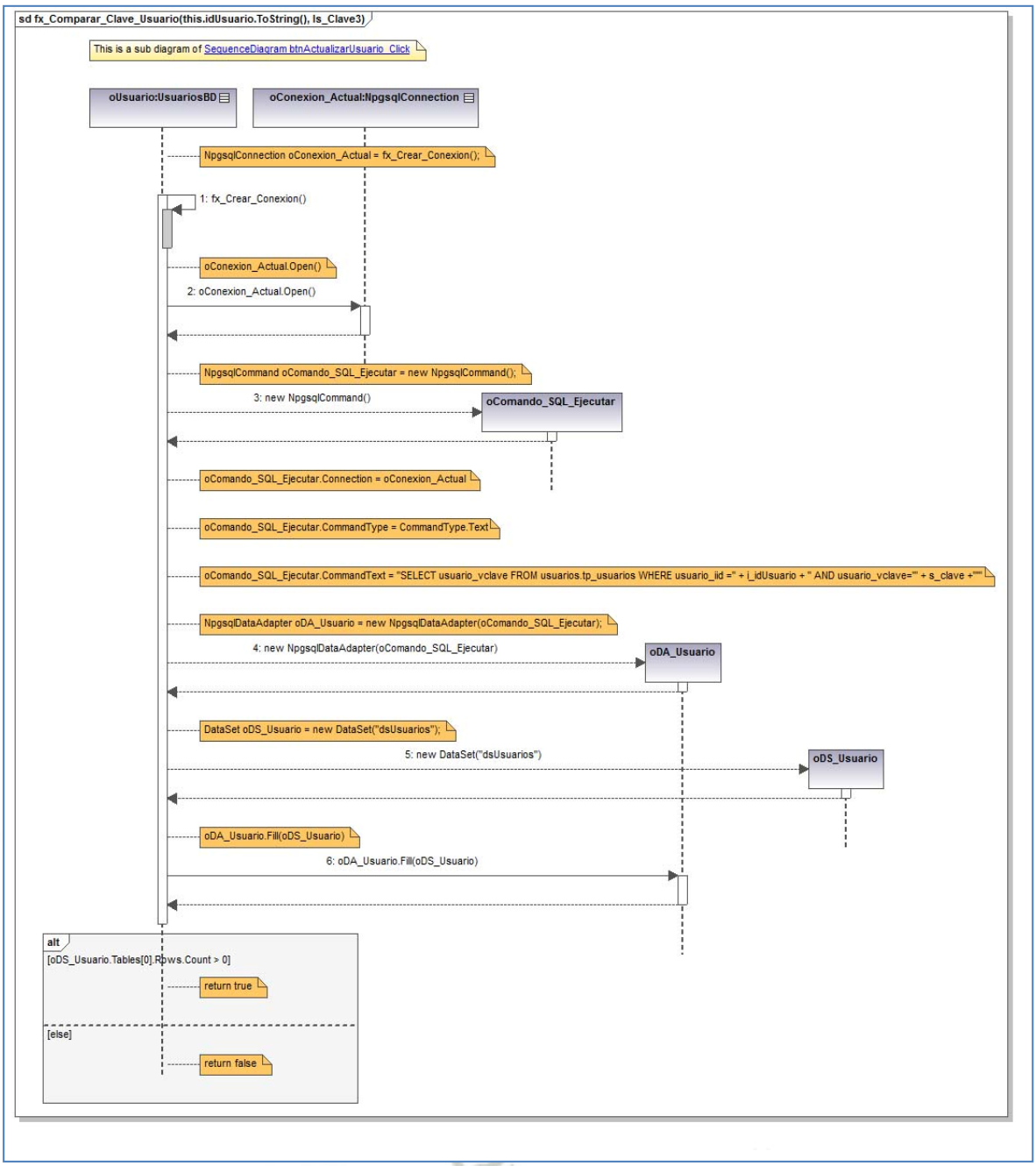


Figura 3.67 – Diagrama de Secuencia fx_Comparar_Clave_Usuario.

Fuente: Elaboración Propia.

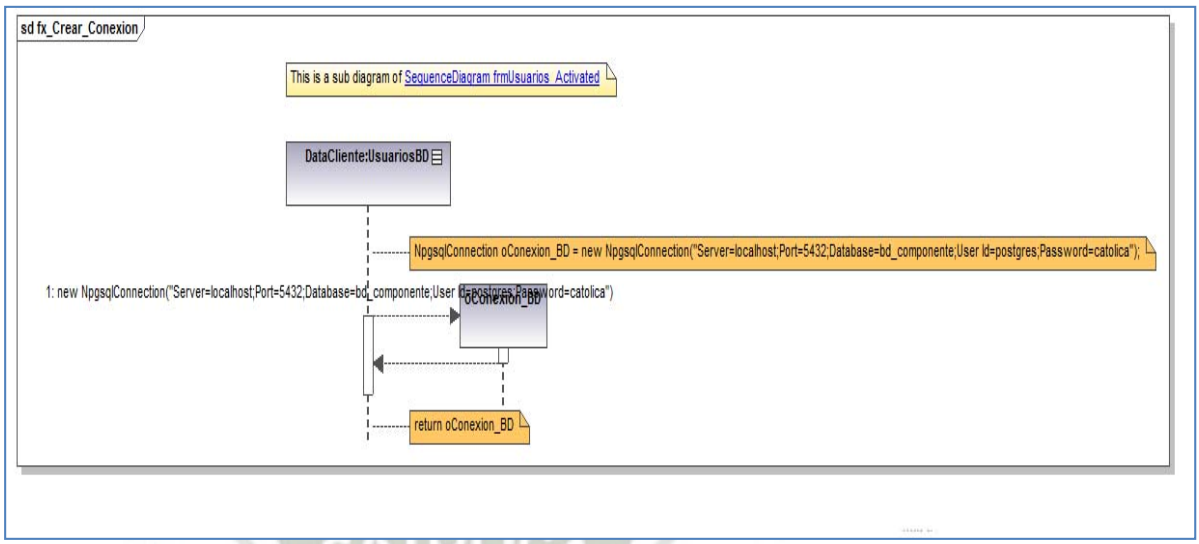


Figura 3.68 – Diagrama de Secuencia fx_Crear_Conexion.

Fuente: Elaboración Propia.

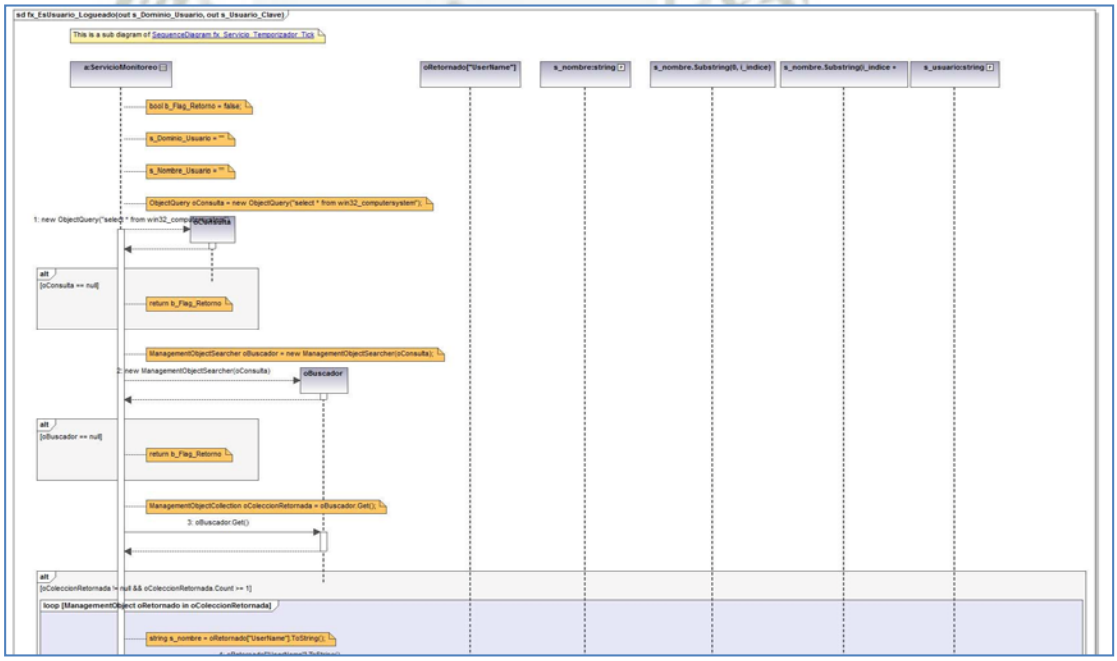


Figura 3.69 – Diagrama de Secuencia fx_EsUsuario_Logueado (Parte 1).

Fuente: Elaboración Propia.

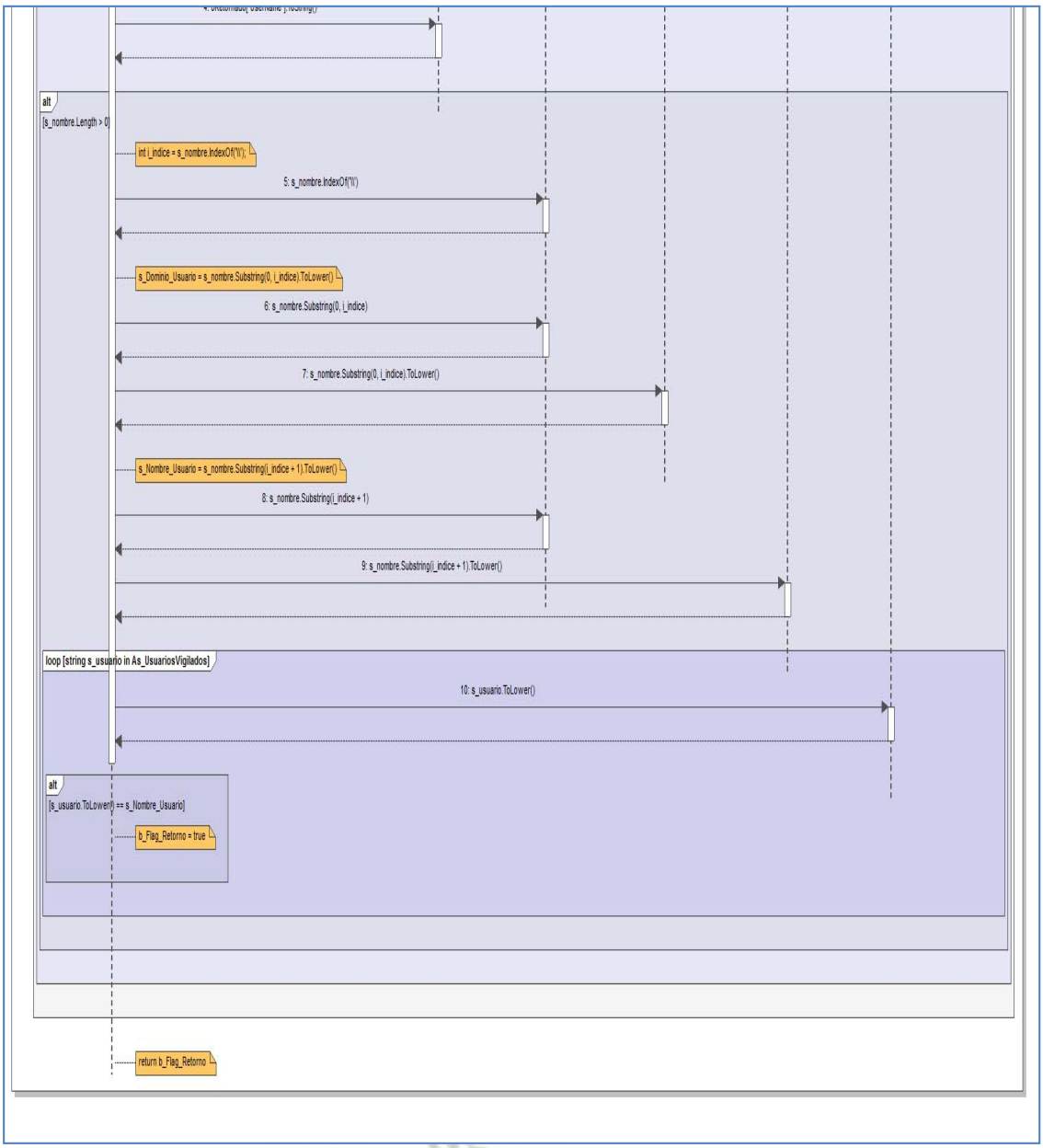


Figura 3.70 – Diagrama de Secuencia fx_EsUsuario_Logueado (Parte 2).

Fuente: Elaboración Propia.

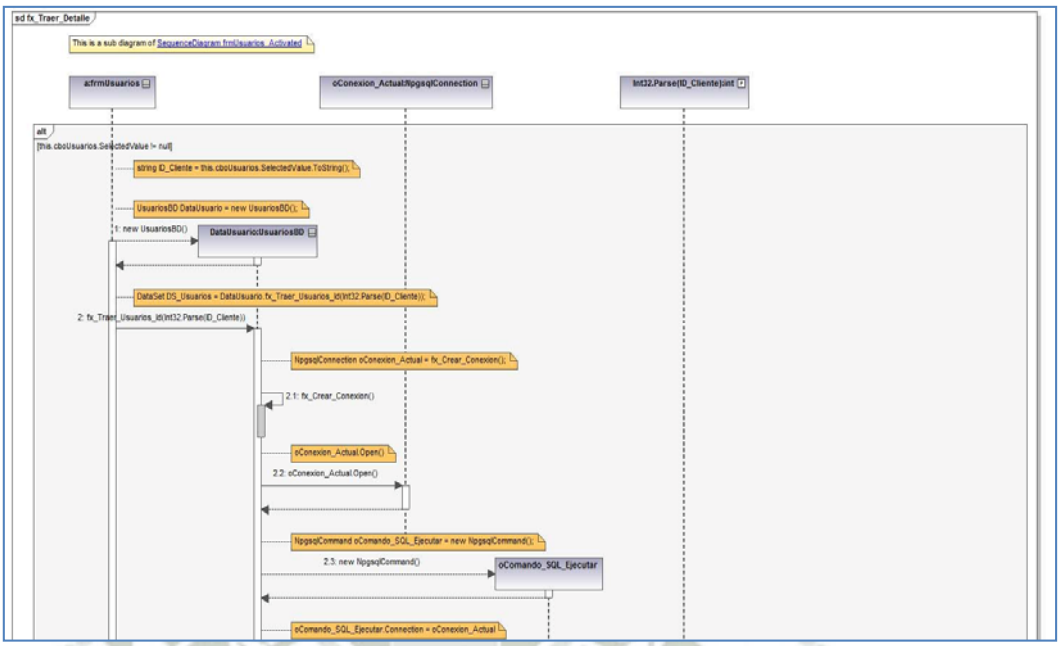


Figura 3.71 – Diagrama de Secuencia fx_Traer_Detalle (Parte 1).

Fuente: Elaboración Propia.

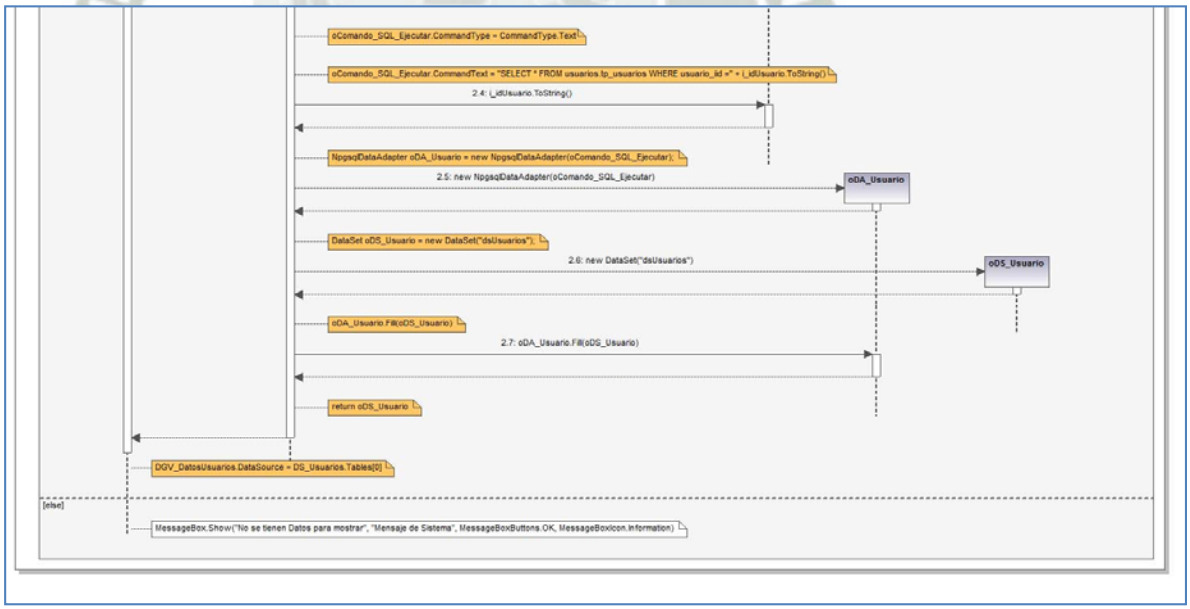


Figura 3.72 – Diagrama de Secuencia fx_Traer_Detalle (Parte 2).

Fuente: Elaboración Propia.

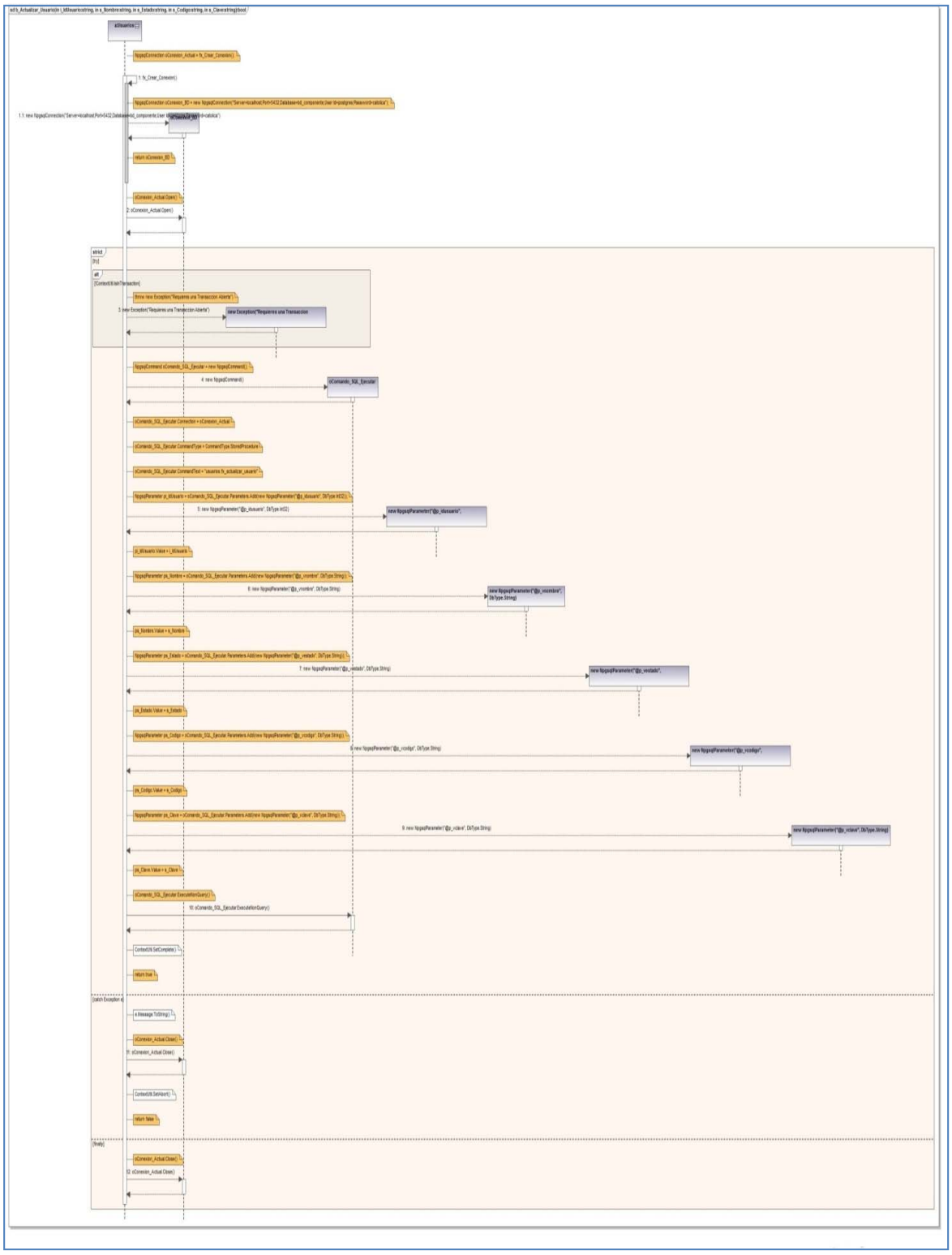


Figura 3.73 – Diagrama de Secuencia b_Actualizar_Usuario.

Fuente: Elaboración Propia.

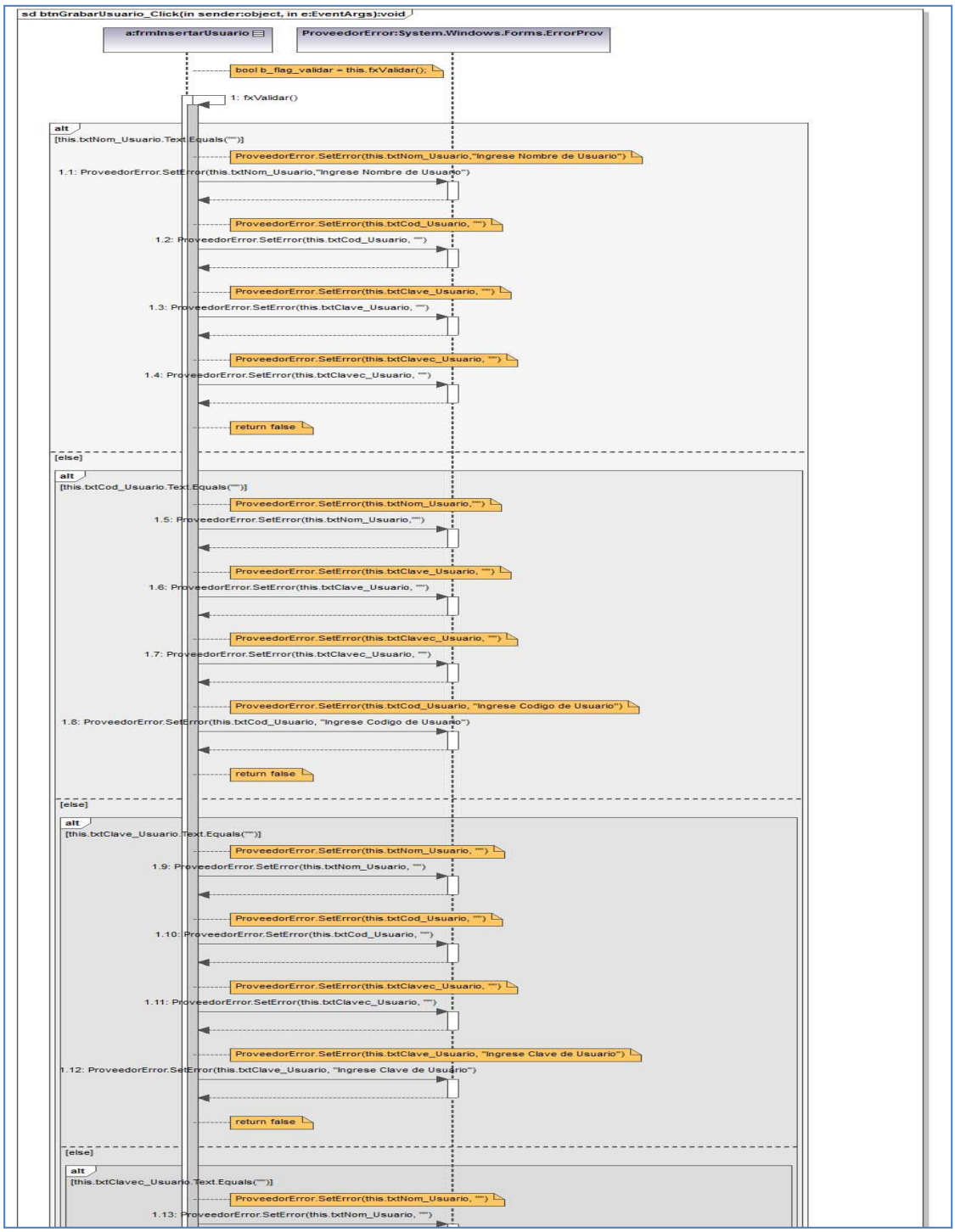


Figura 3.74 – Diagrama de Secuencia btnGrabarUsuario_Click (Parte 1).

Fuente: Elaboración Propia.

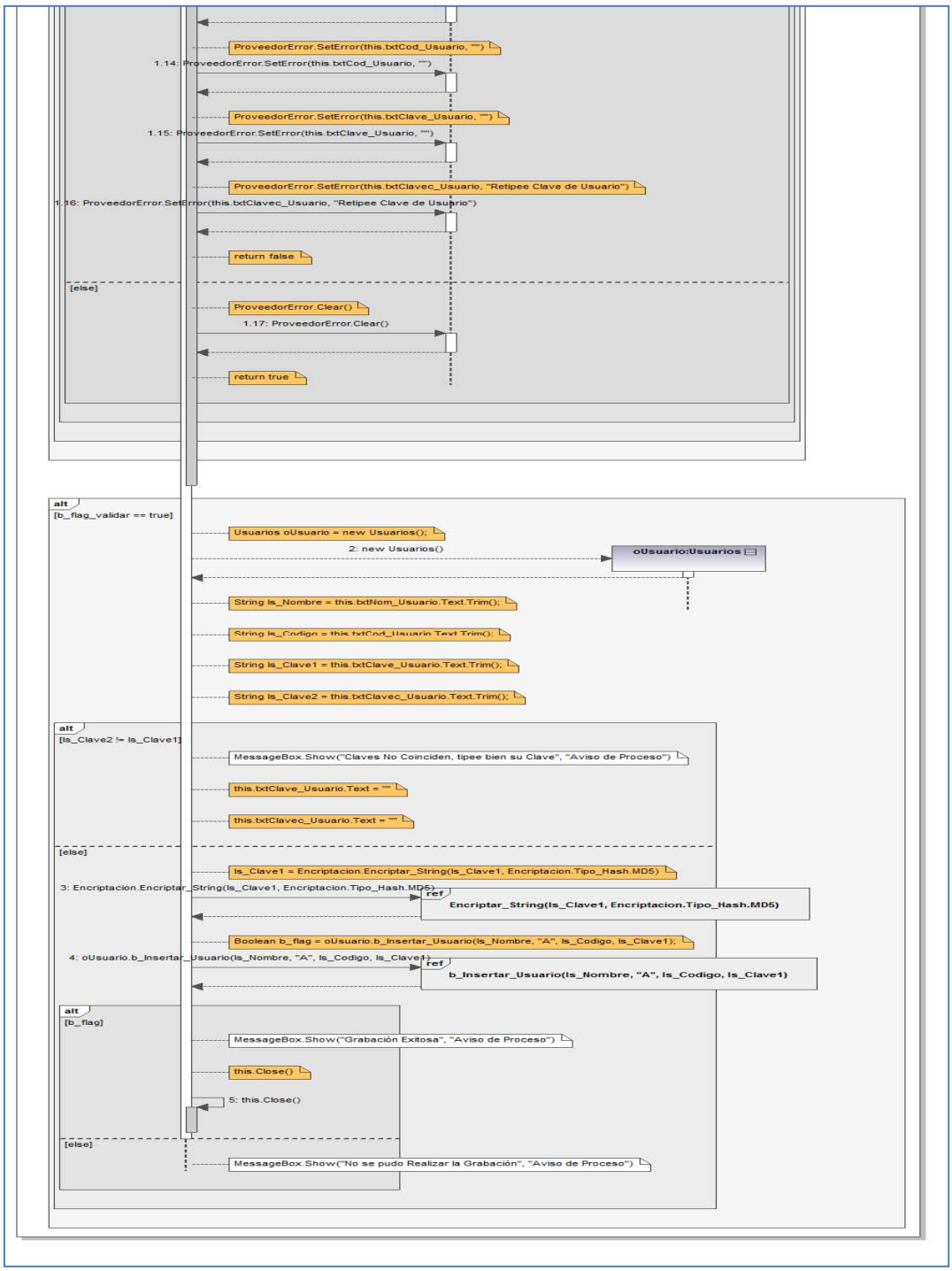


Figura 3.75 – Diagrama de Secuencia btnGrabarUsuario_Click (Parte 2).

Fuente: Elaboración Propia.

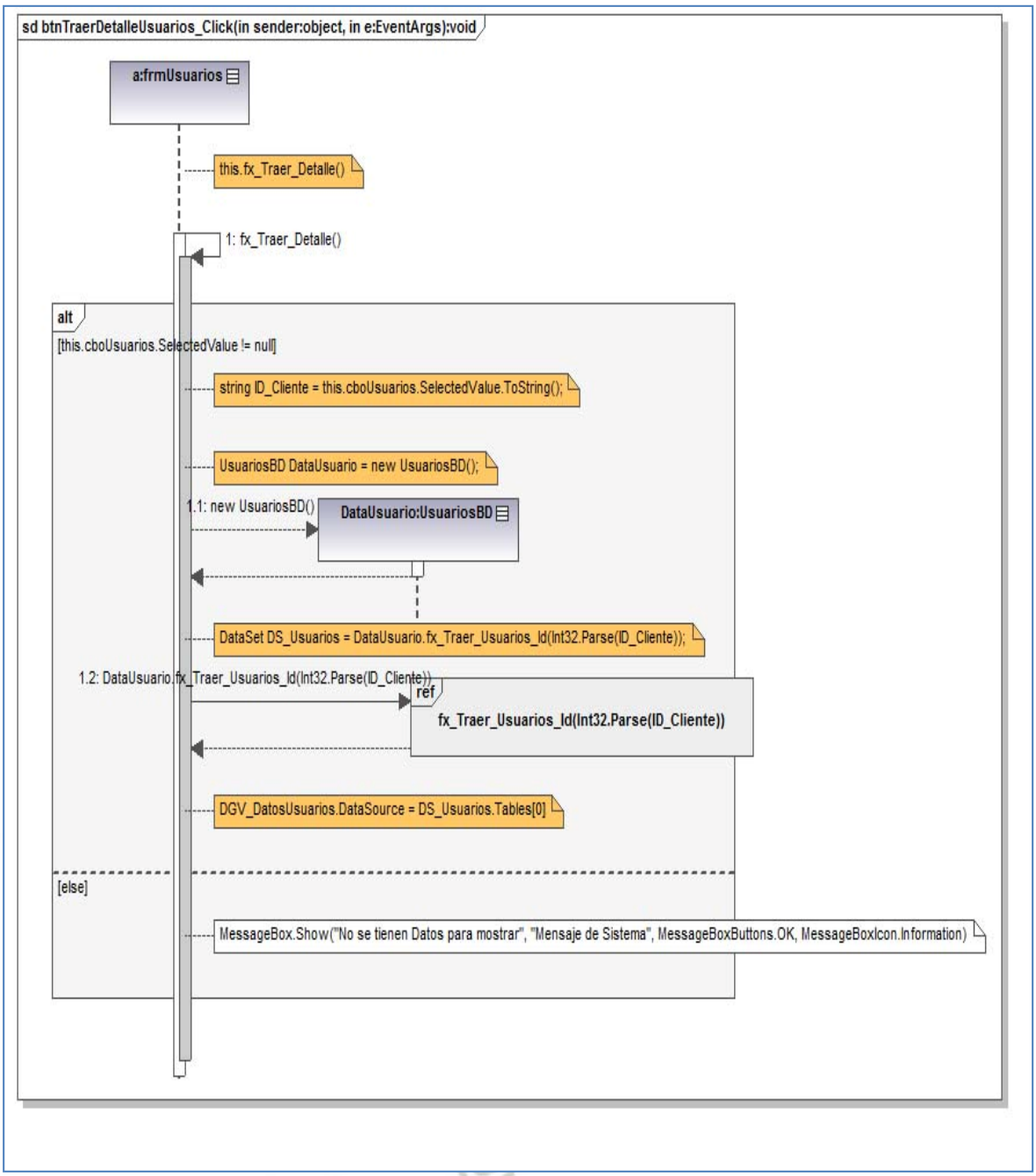


Figura 3.76 – Diagrama de Secuencia btnTraerDetalleUsuarios_Click.

Fuente: Elaboración Propia.

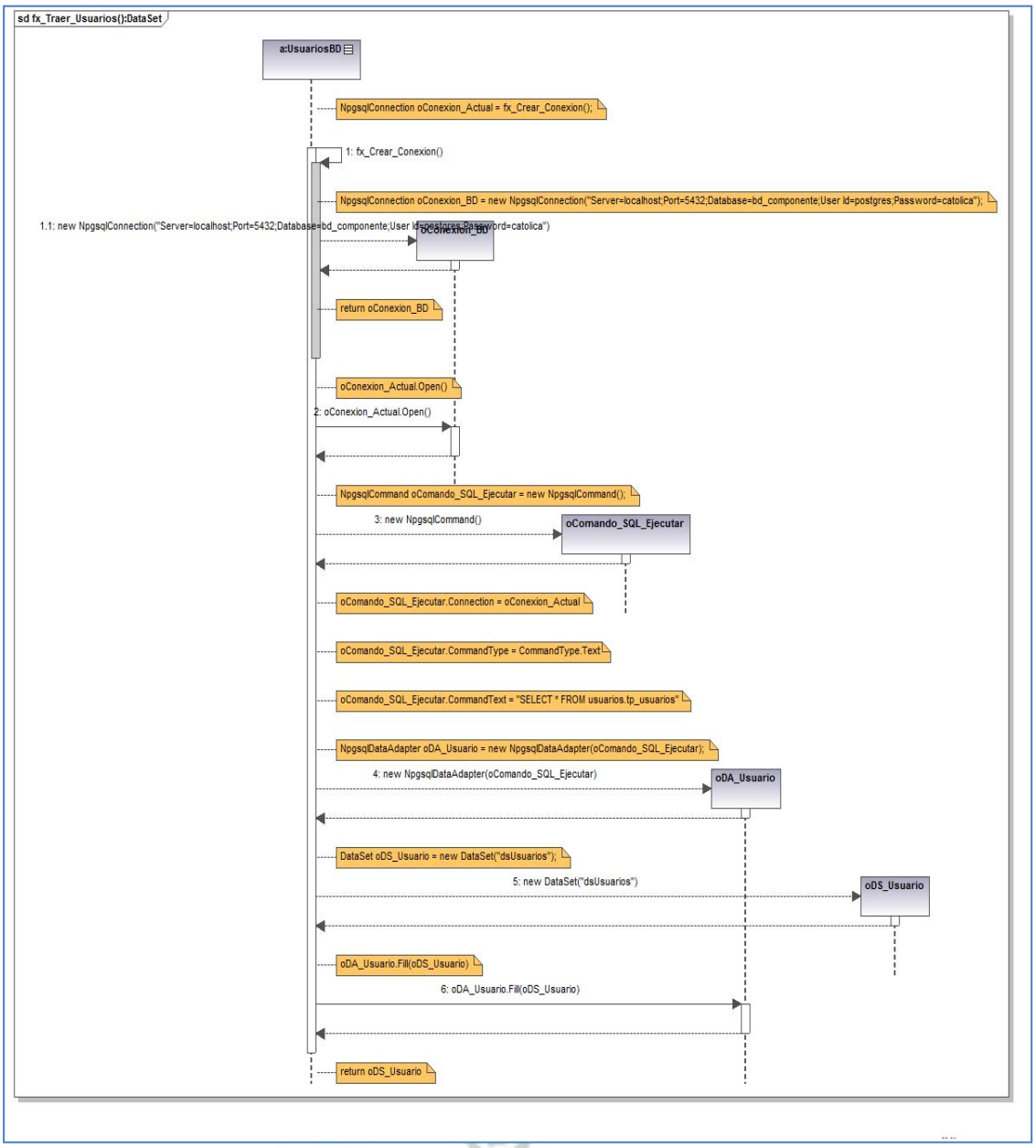


Figura 3.77 – Diagrama de Secuencia fx_Traer_Usuarios.

Fuente: Elaboración Propia.

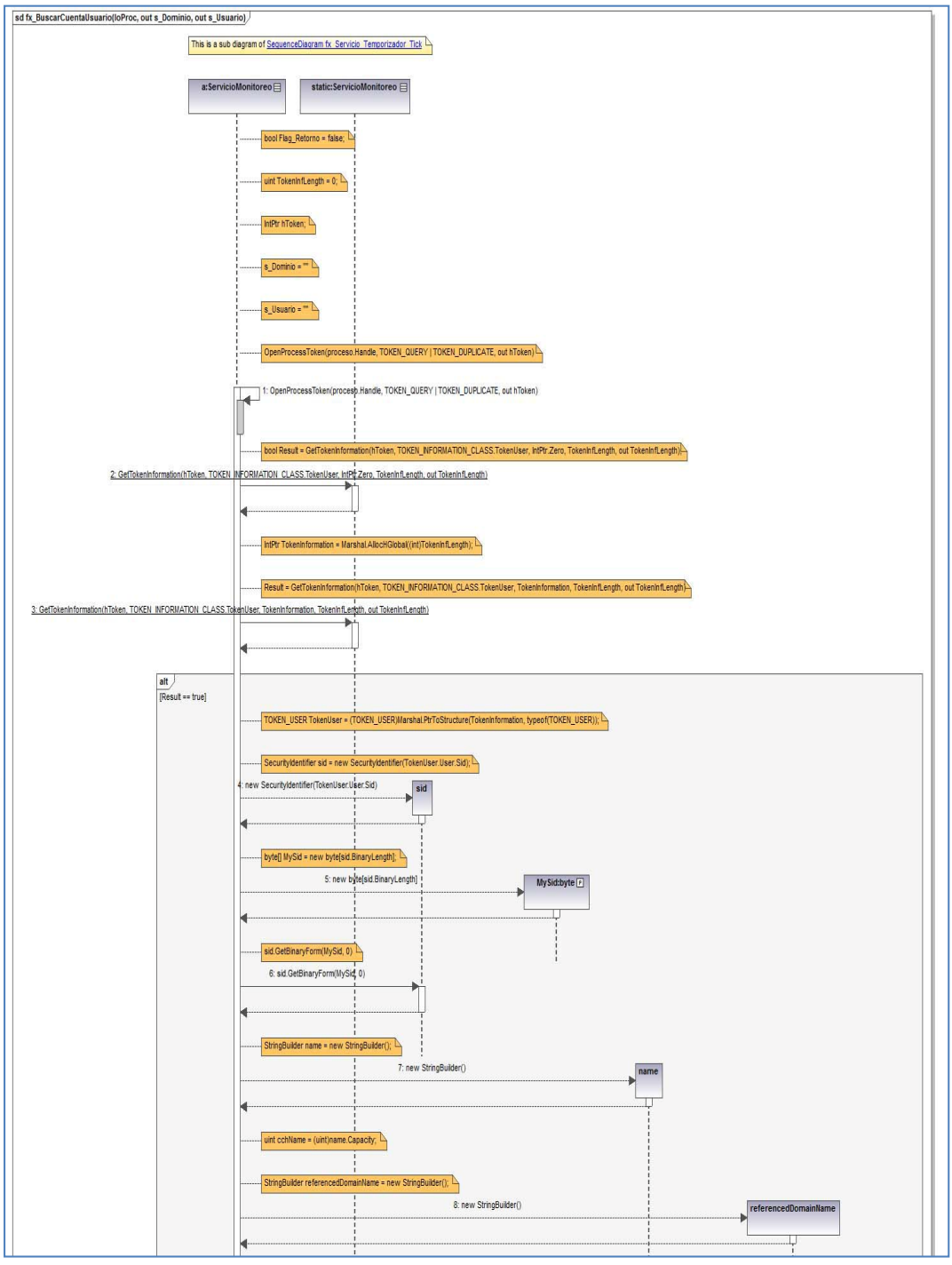


Figura 3.78 – Diagrama de Secuencia fxBuscarCuentaUsuario (Parte 1).

Fuente: Elaboración Propia.

Figura 3.79 – Diagrama de Secuencia fxBuscarCuentaUsuario (Parte 2).

Fuente: Elaboración Propia.

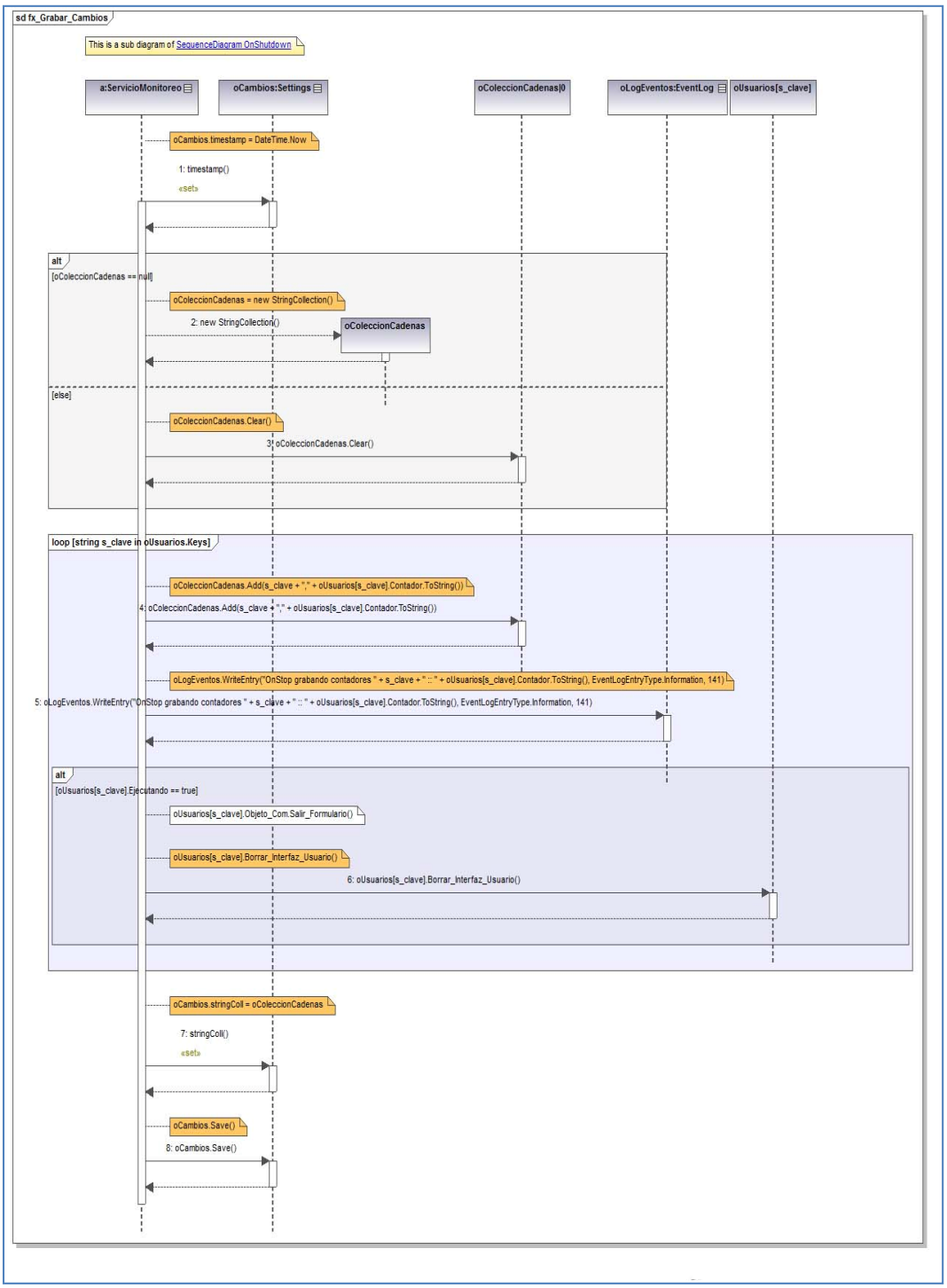


Figura 3.80 – Diagrama de Secuencia fx_Grabar_Cambios.

Fuente: Elaboración Propia.

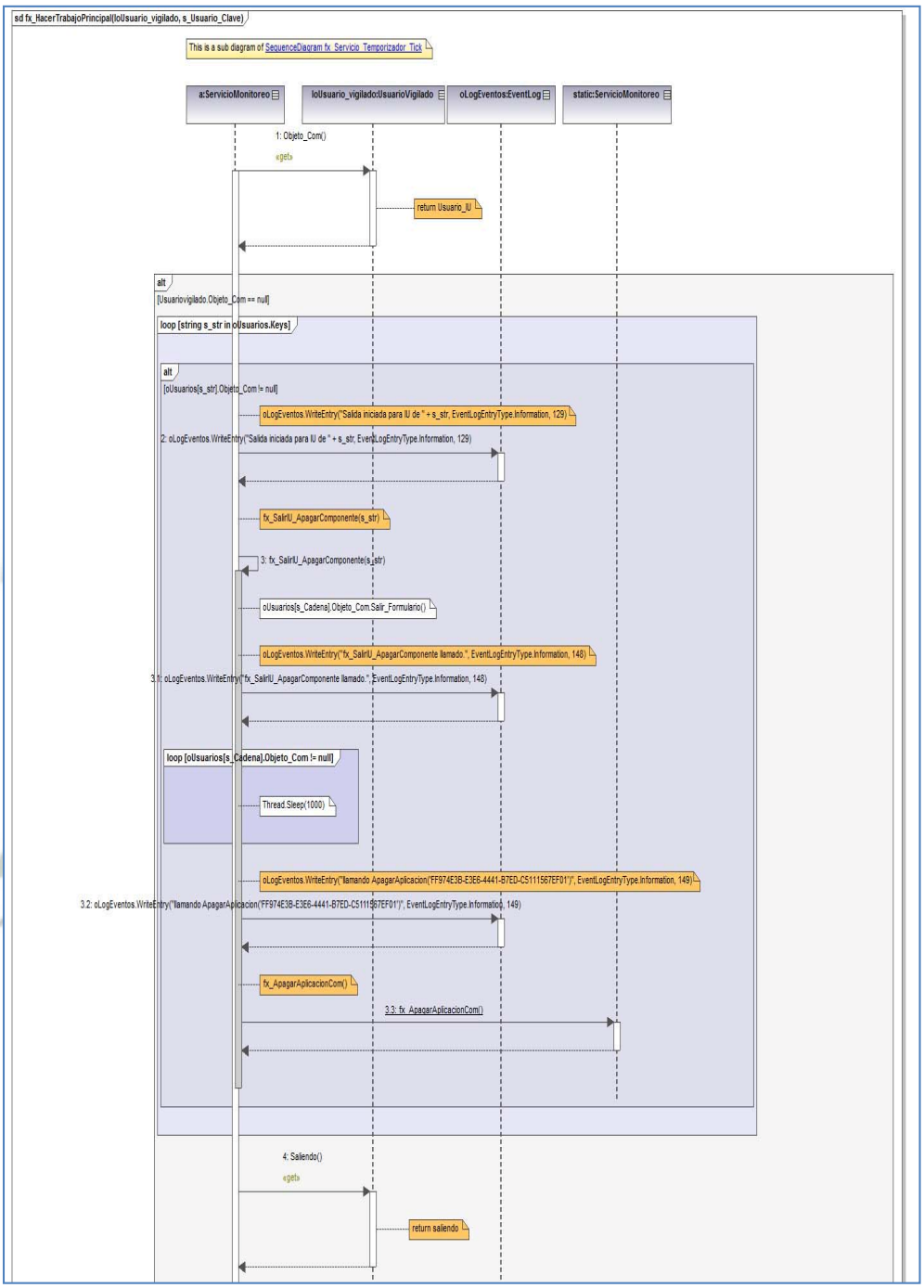


Figura 3.81 – Diagrama de Secuencia fx_HacerTrabajoPrincipal (Parte 1).

Fuente: Elaboración Propia.

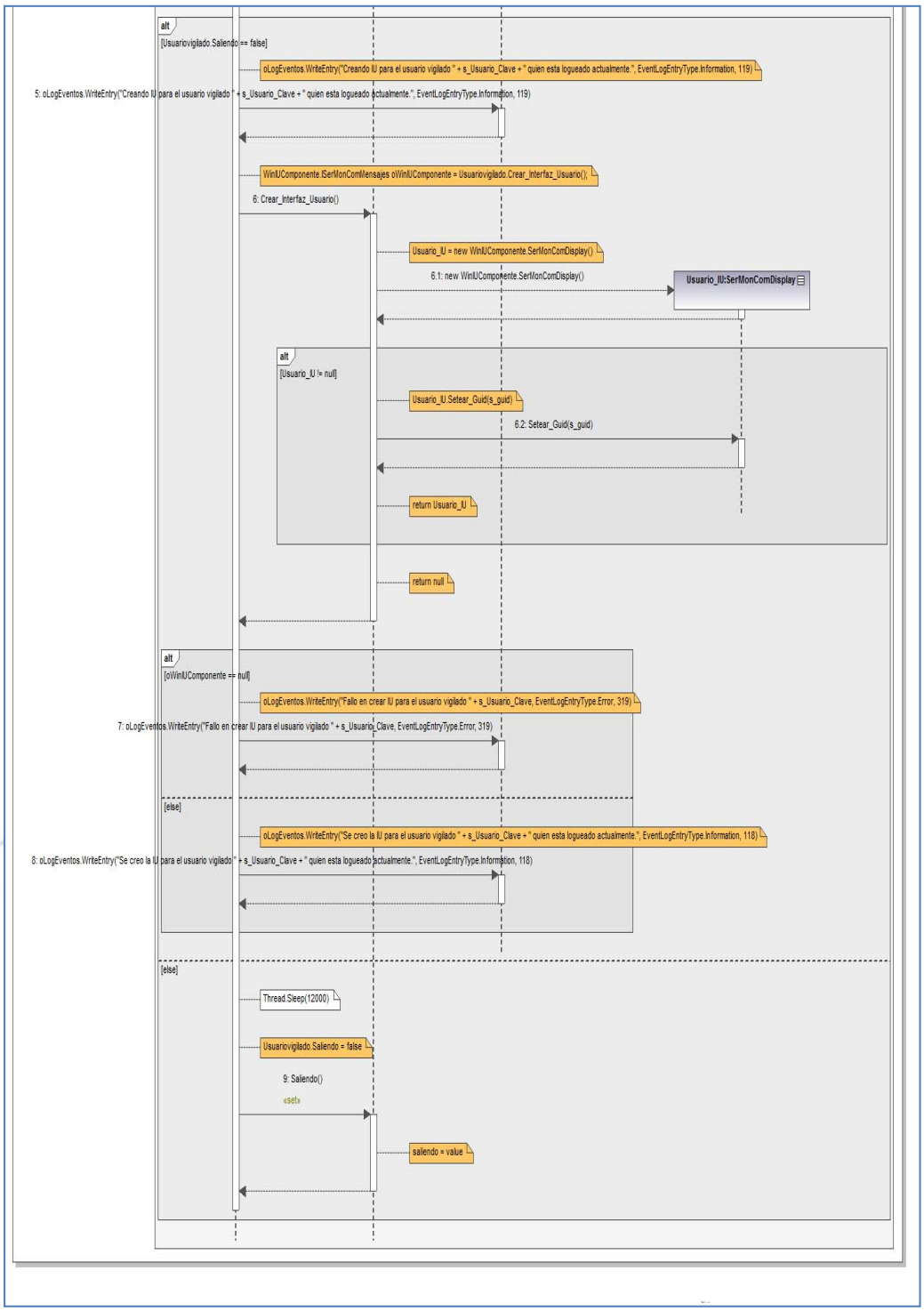


Figura 3.82 – Diagrama de Secuencia fx_HacerTrabajoPrincipal (Parte 2).

Fuente: Elaboración Propia.

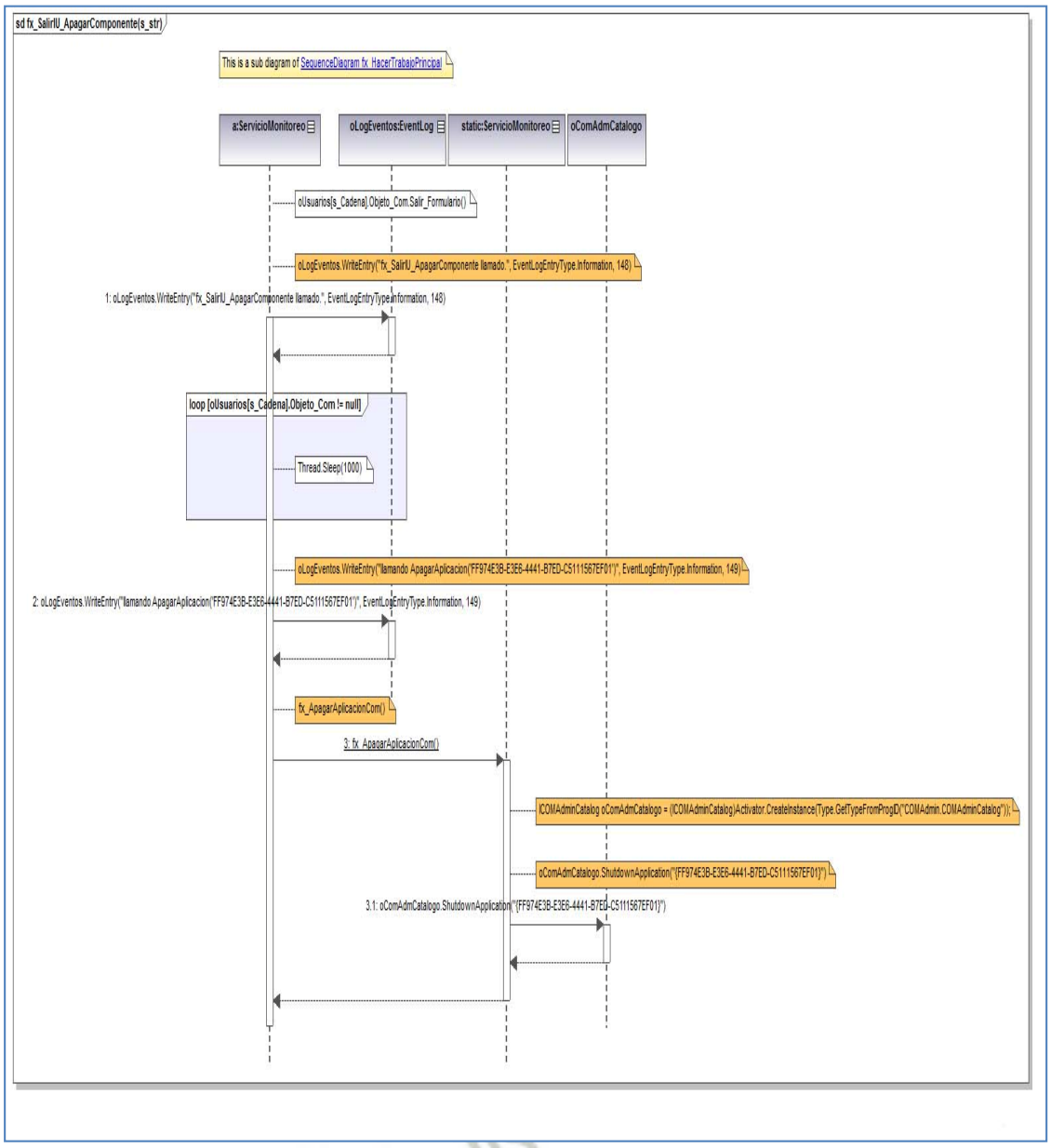


Figura 3.83 – Diagrama de Secuencia fx_SalirIU_ApagarComponente.

Fuente: Elaboración Propia.

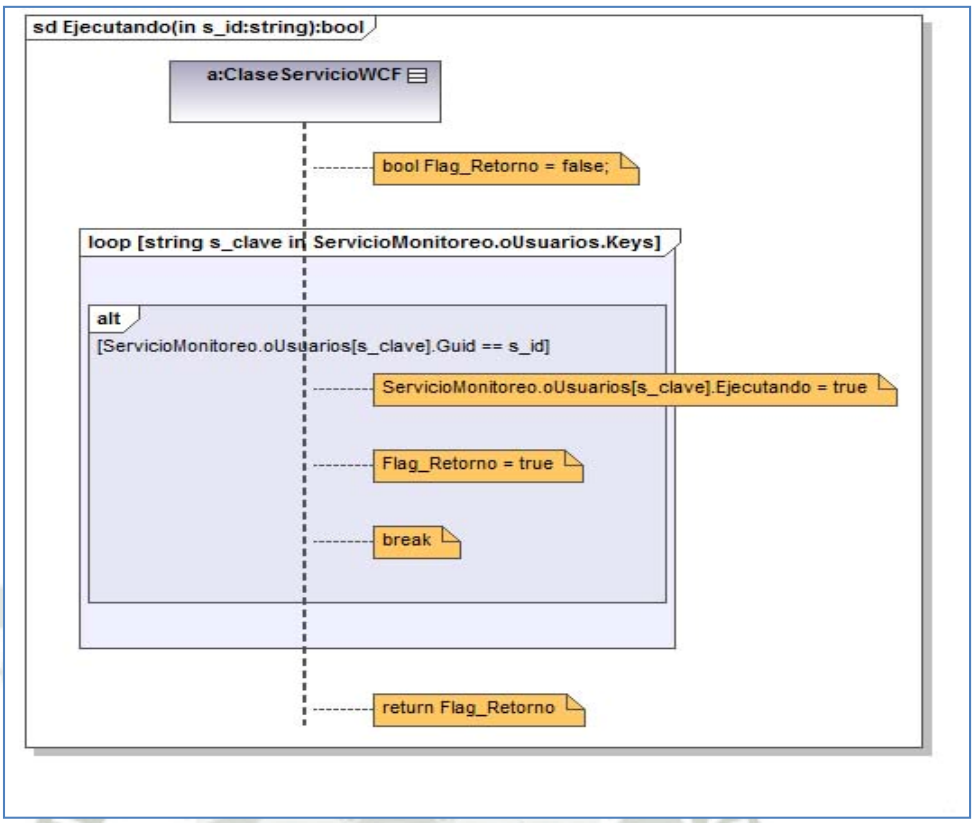


Figura 3.84 – Diagrama de Secuencia Ejecutando.

Fuente: Elaboración Propia.

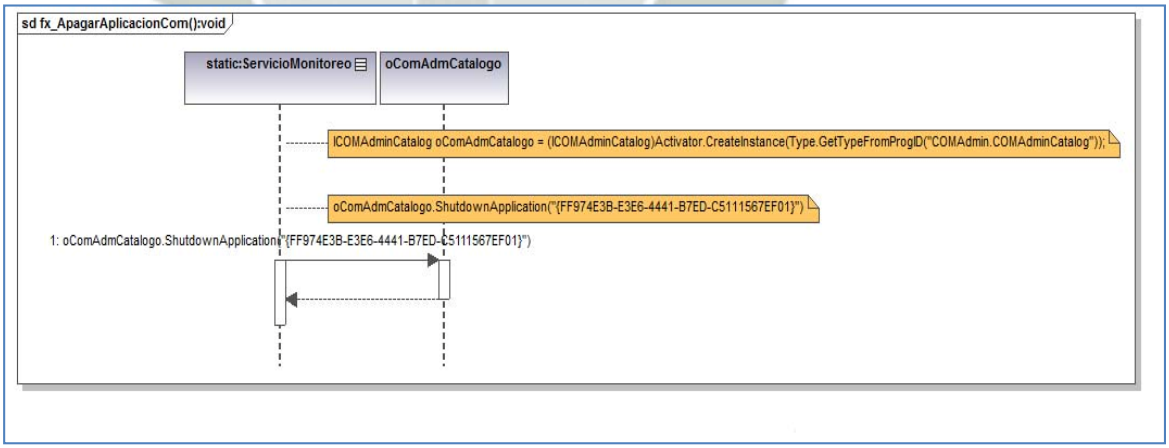


Figura 3.85 – Diagrama de Secuencia ApagarAplicacionCom.

Fuente: Elaboración Propia.

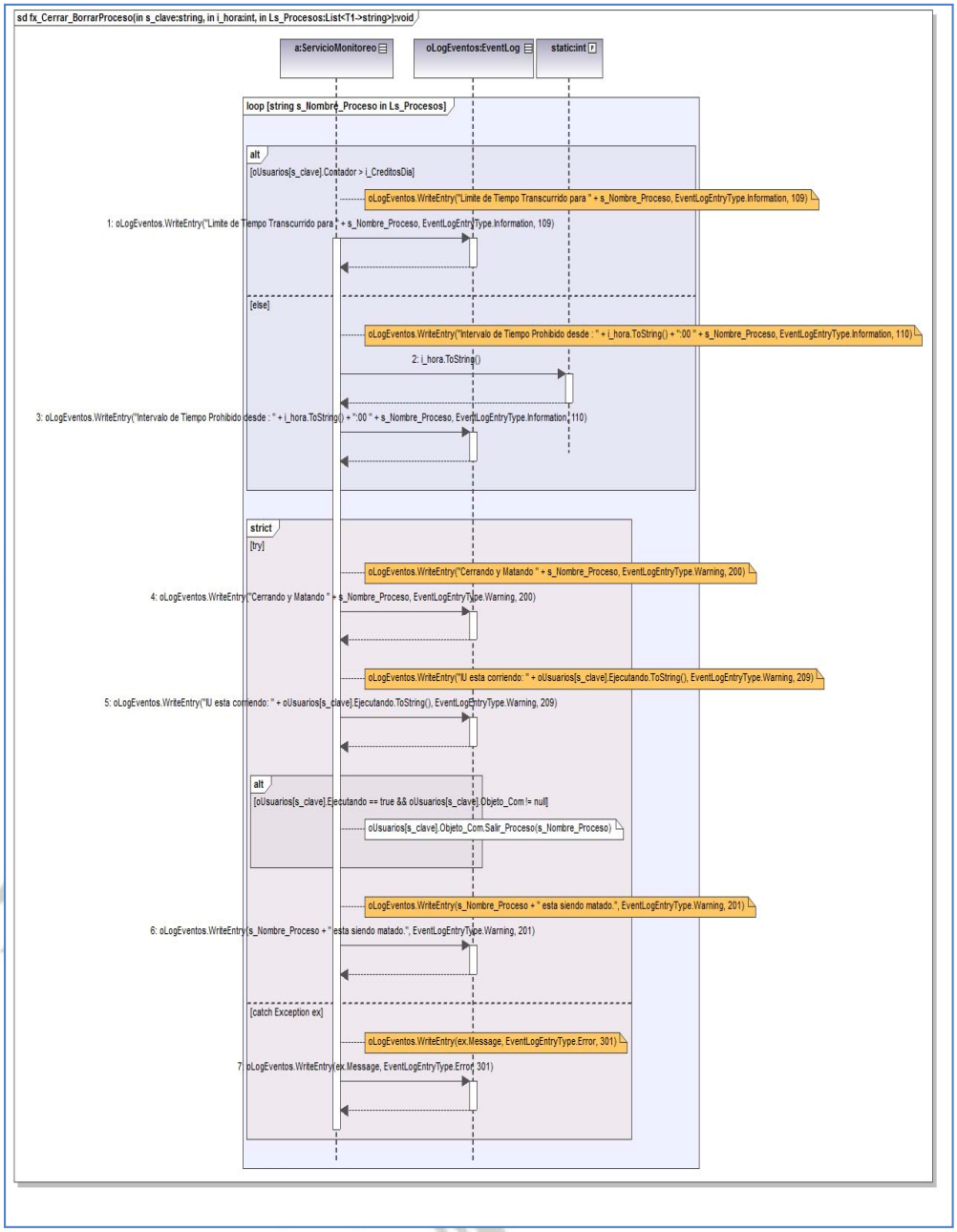


Figura 3.86 – Diagrama de Secuencia fx_Cerrar_BorrarProceso.

Fuente: Elaboración Propia.

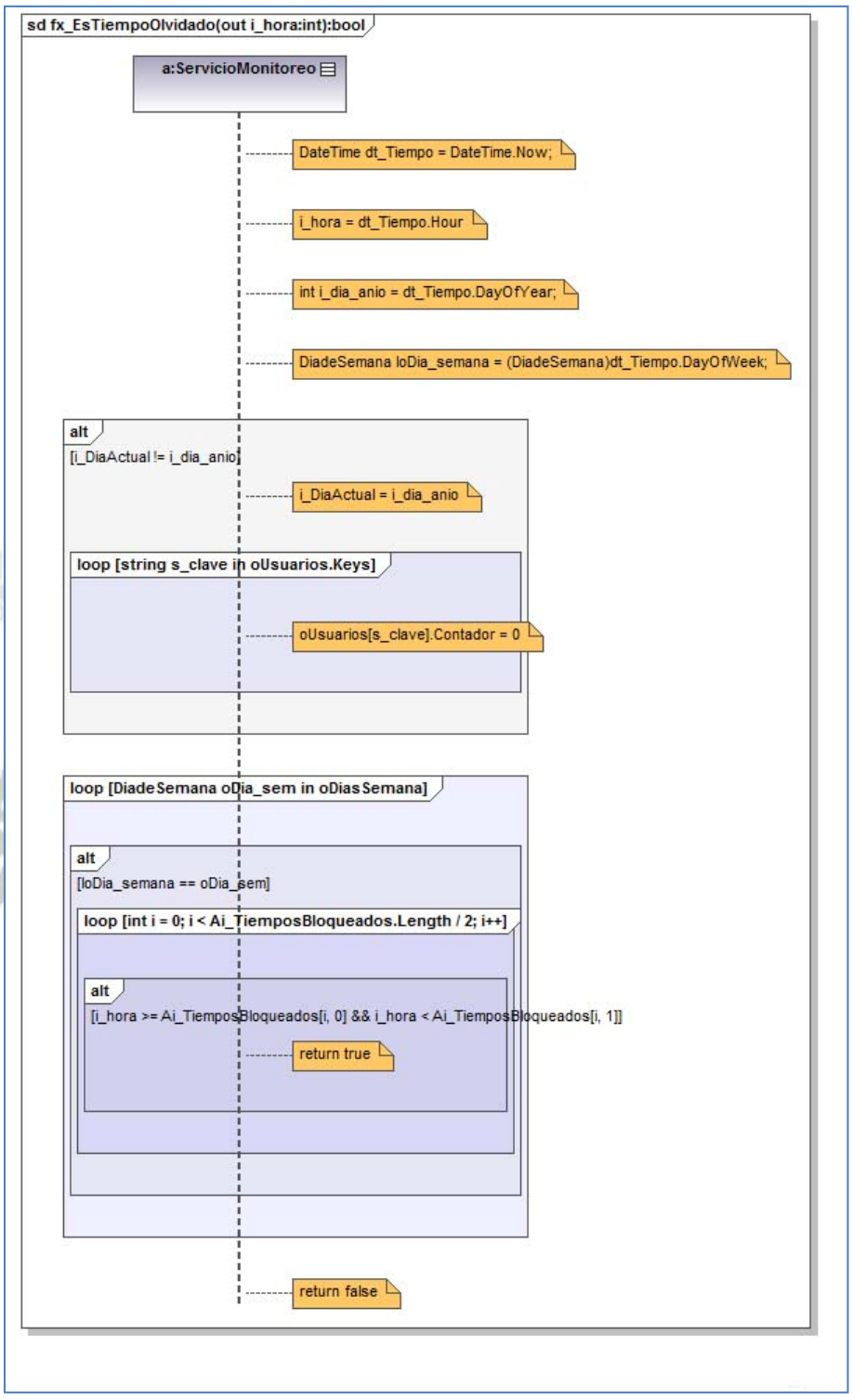


Figura 3.87 – Diagrama de Secuencia fx_EsTiempoOlvidado.

Fuente: Elaboración Propia.

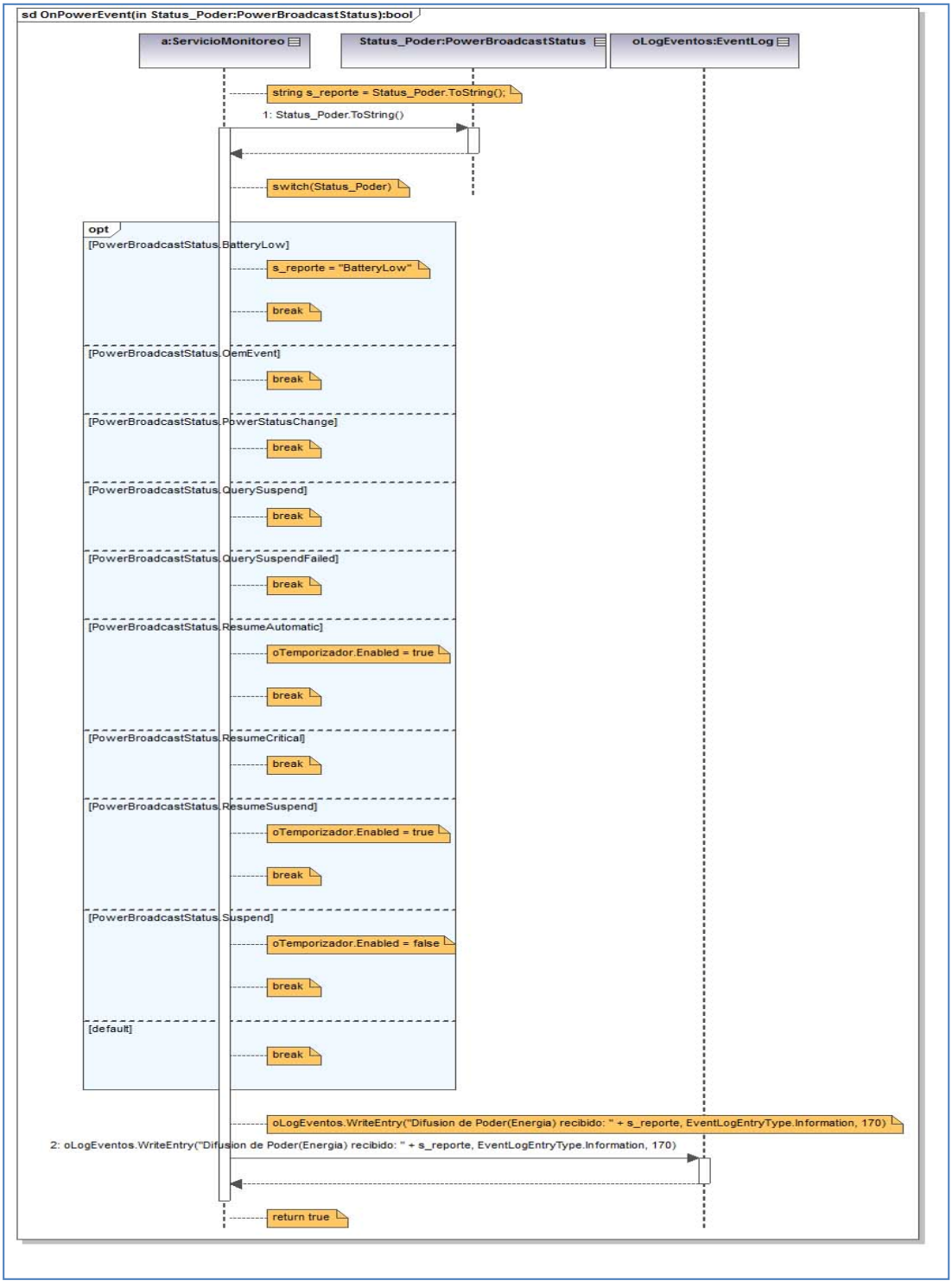


Figura 3.88 – Diagrama de Secuencia OnPowerEvent.

Fuente: Elaboración Propia.

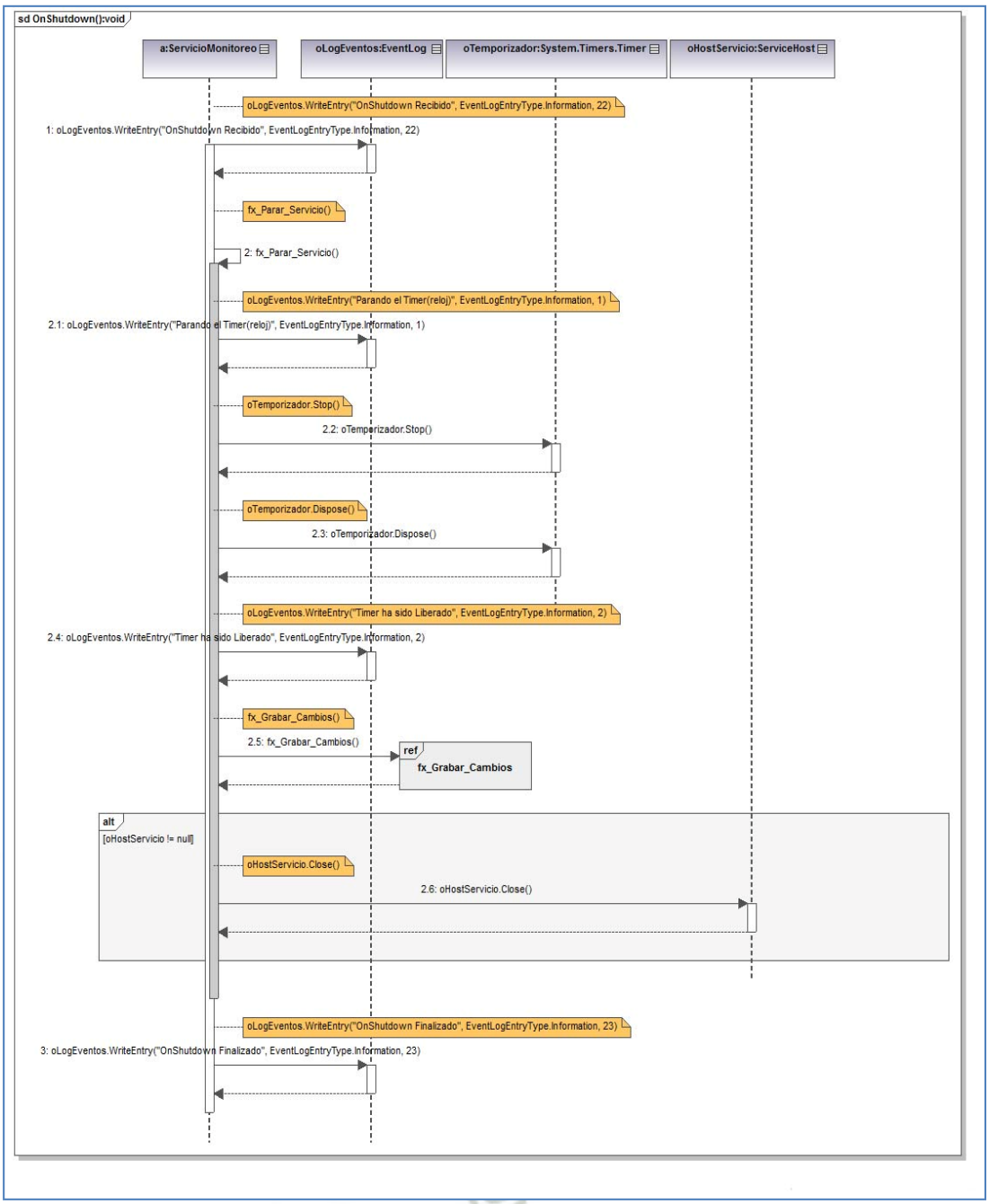


Figura 3.89 – Diagrama de Secuencia OnShutdown.

Fuente: Elaboración Propia.

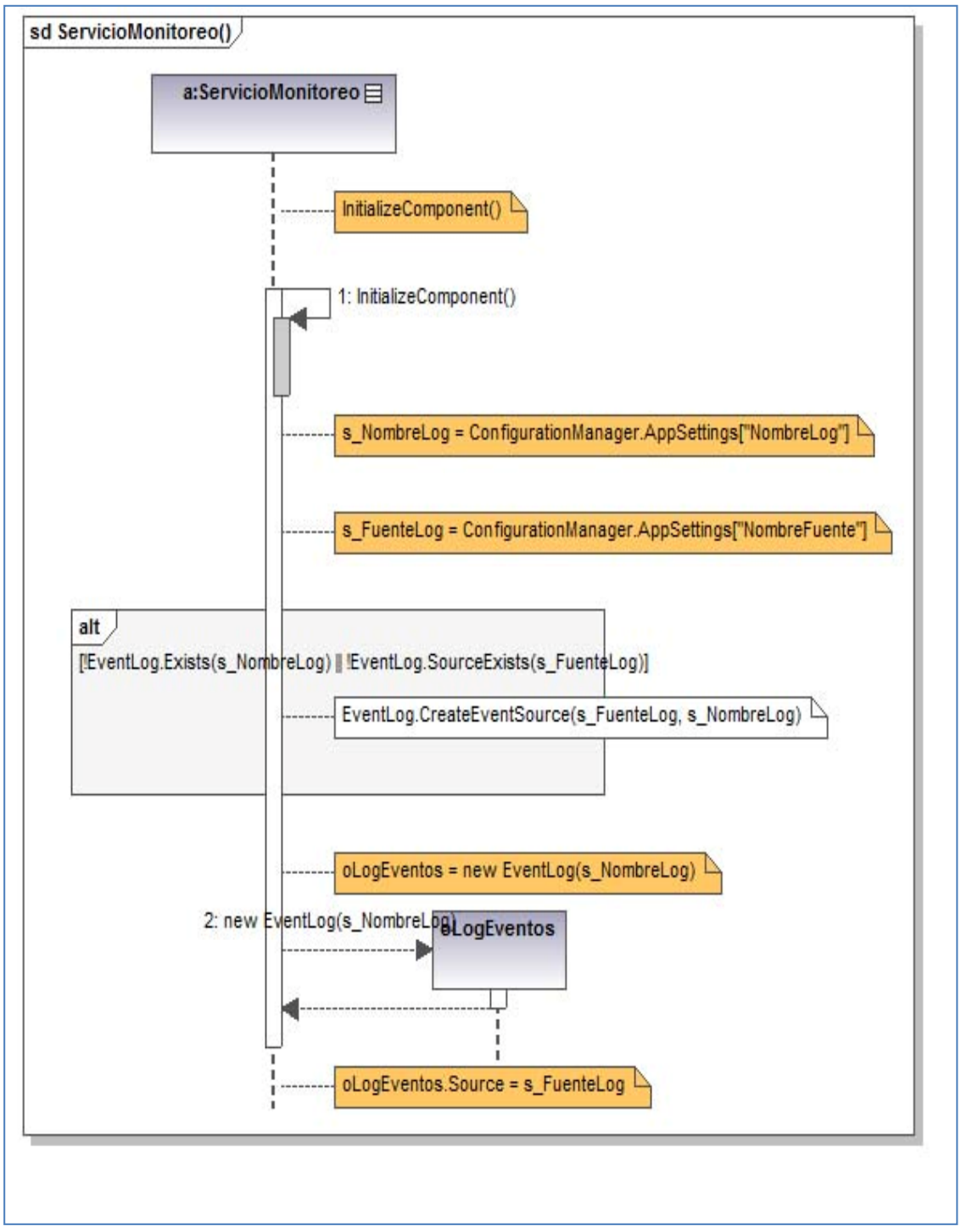


Figura 3.90 – Diagrama de Secuencia ServicioMonitoreo.

Fuente: Elaboración Propia.

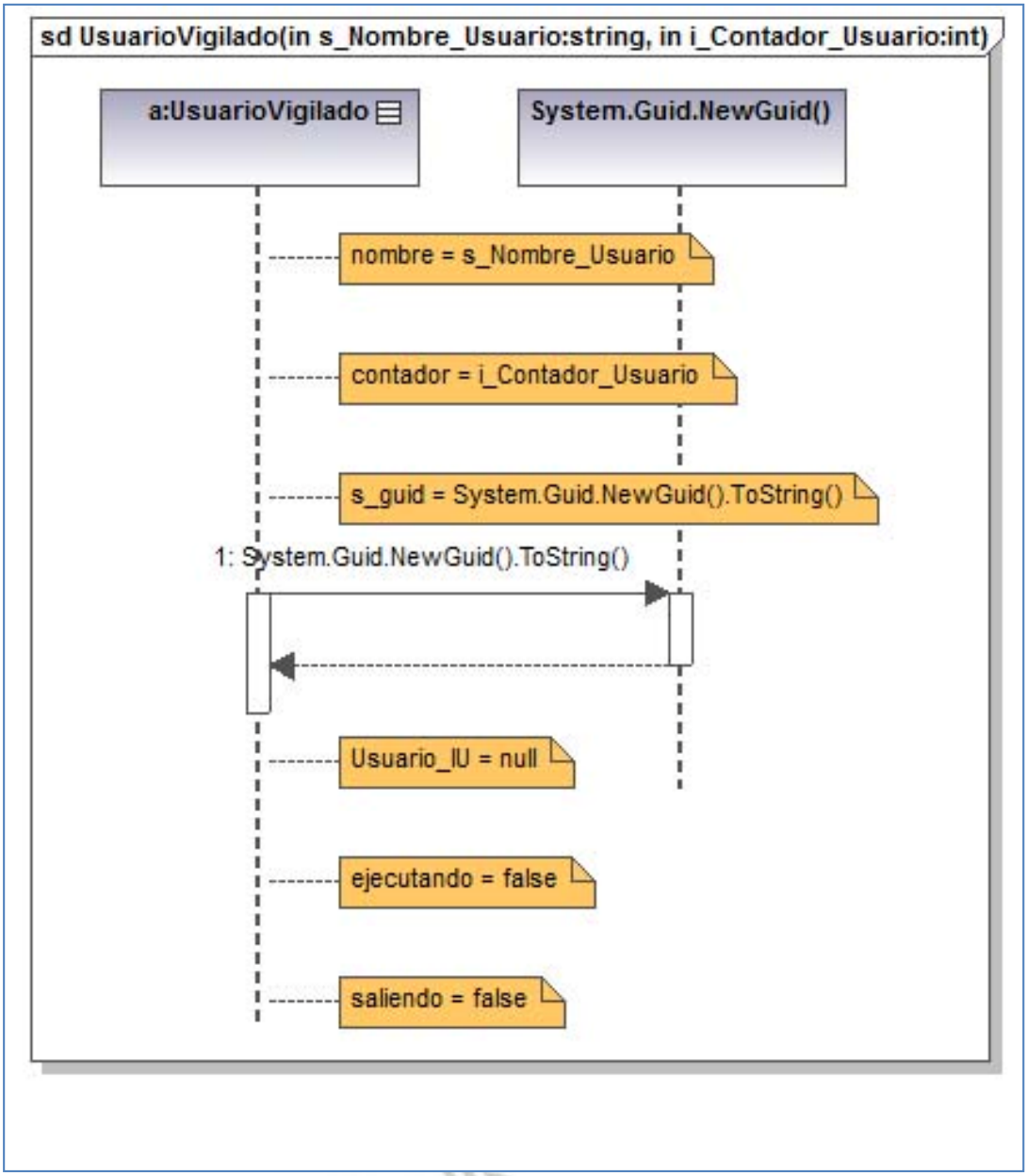


Figura 3.91 – Diagrama de Secuencia UsuarioVigilado.

Fuente: Elaboración Propia.

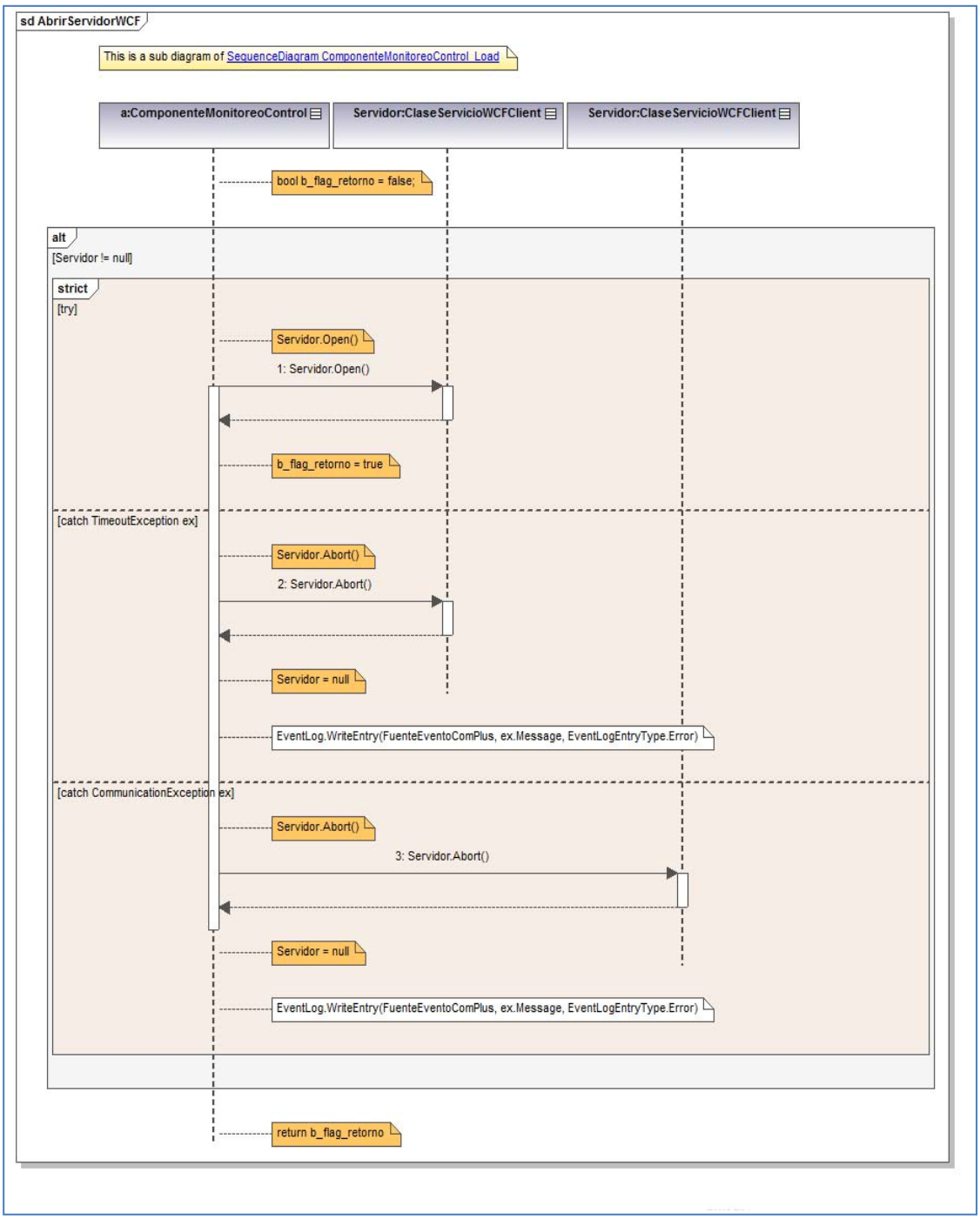


Figura 3.92 – Diagrama de Secuencia AbrirServidorWCF.

Fuente: Elaboración Propia.

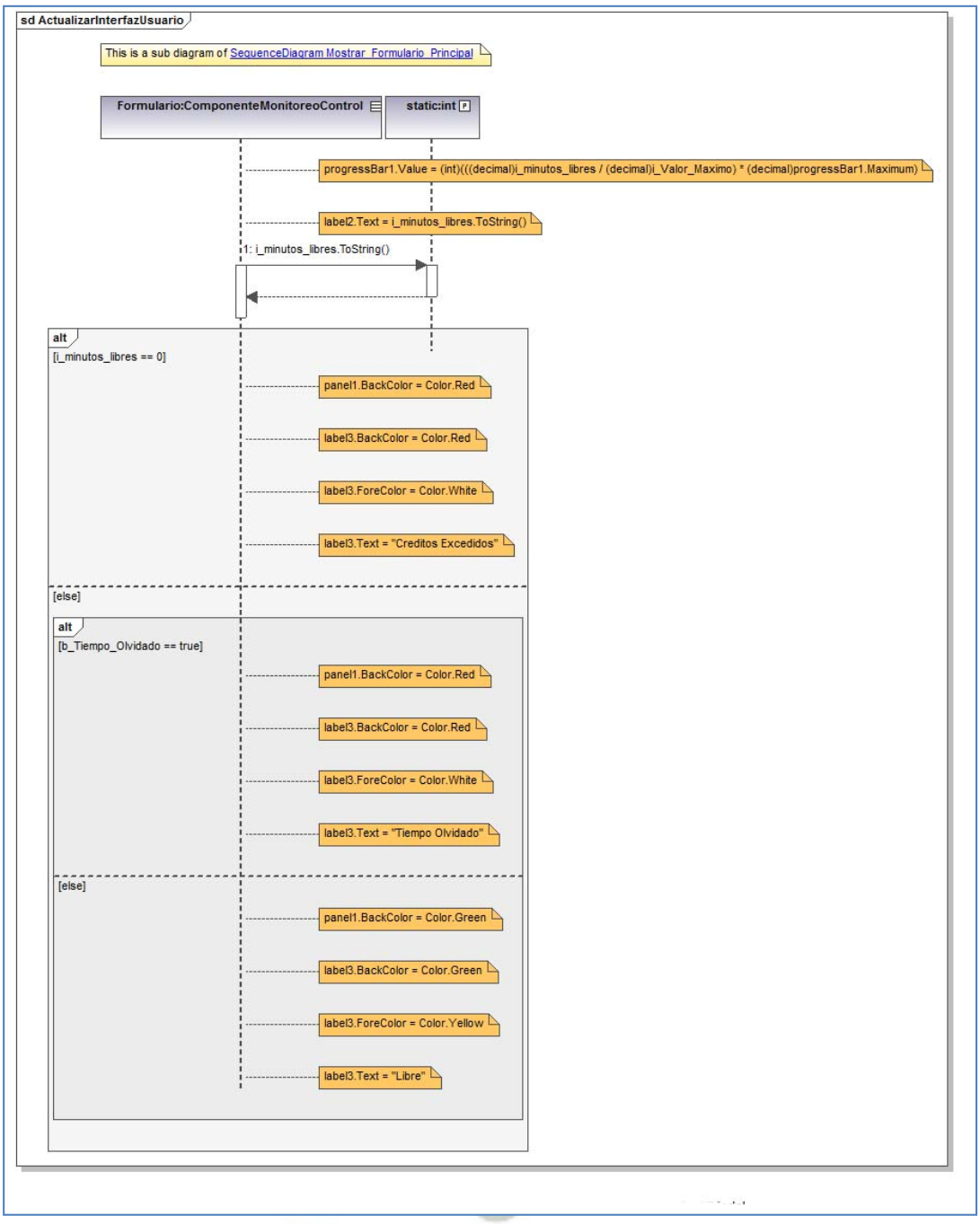


Figura 3.93 – Diagrama de Secuencia ActualizarInterfazUsuario.

Fuente: Elaboración Propia.

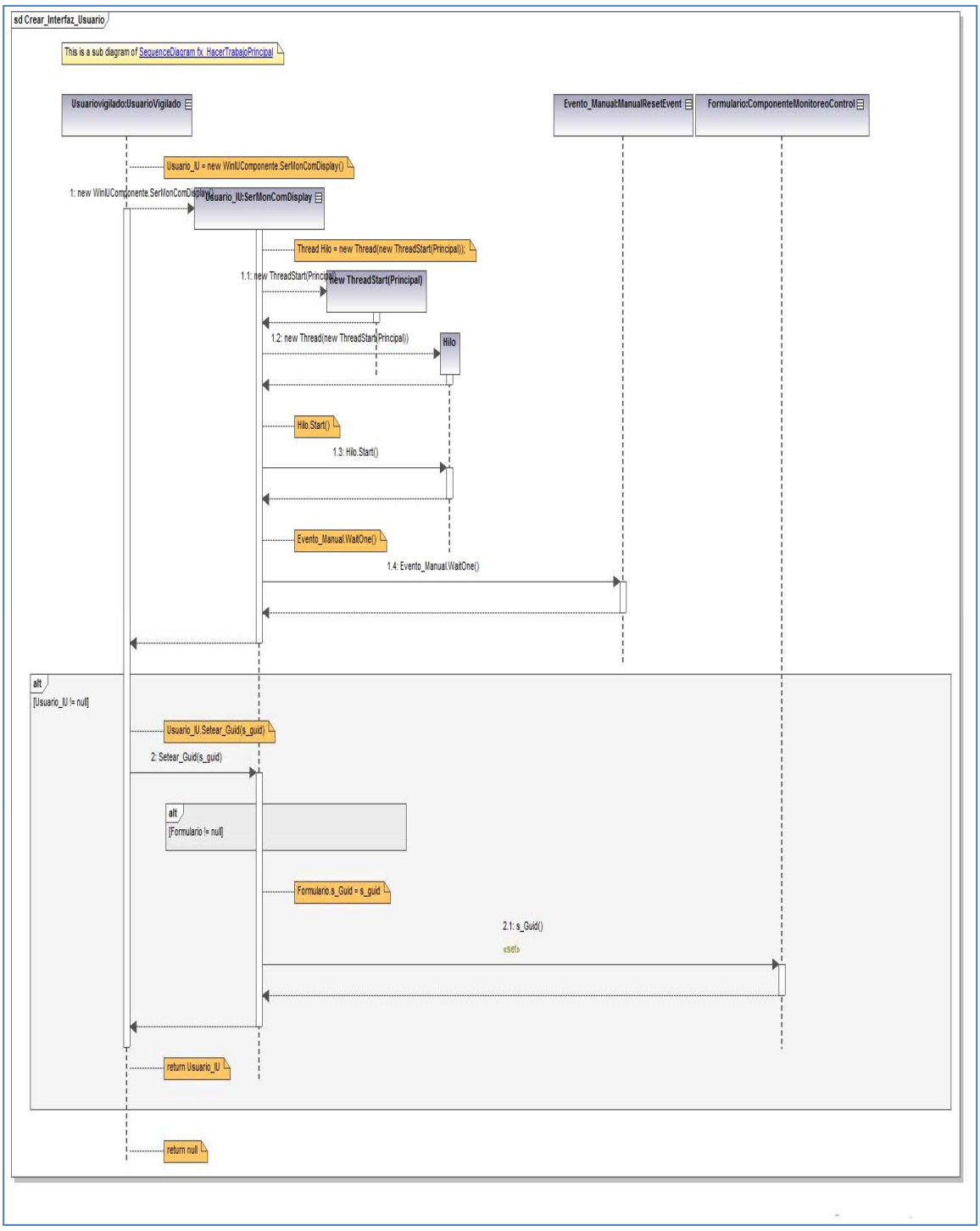


Figura 3.94 – Diagrama de Secuencia Crear_Interfaz_Usuario.

Fuente: Elaboración Propia.

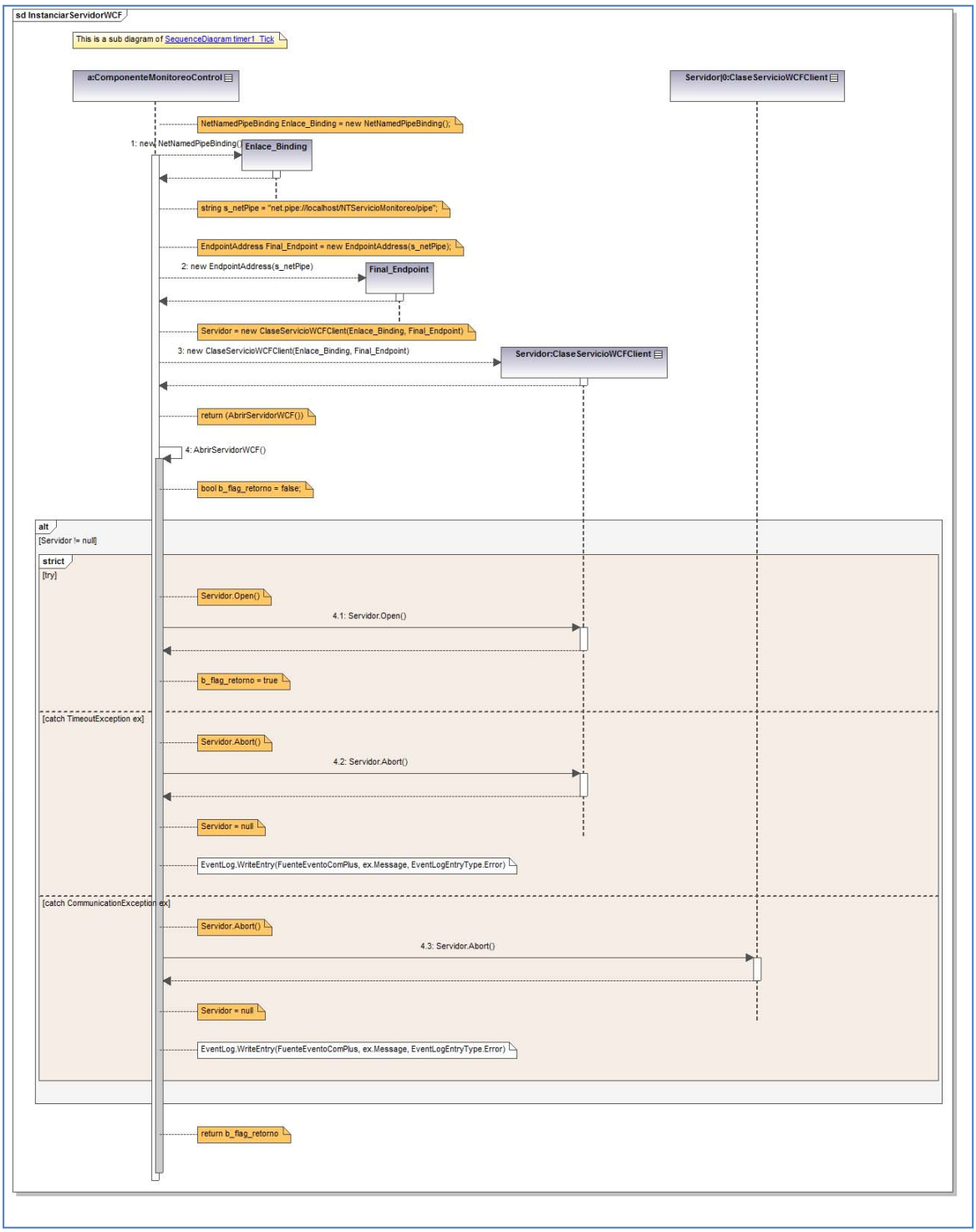


Figura 3.95 – Diagrama de Secuencia InstanciarServidorWCF.

Fuente: Elaboración Propia.

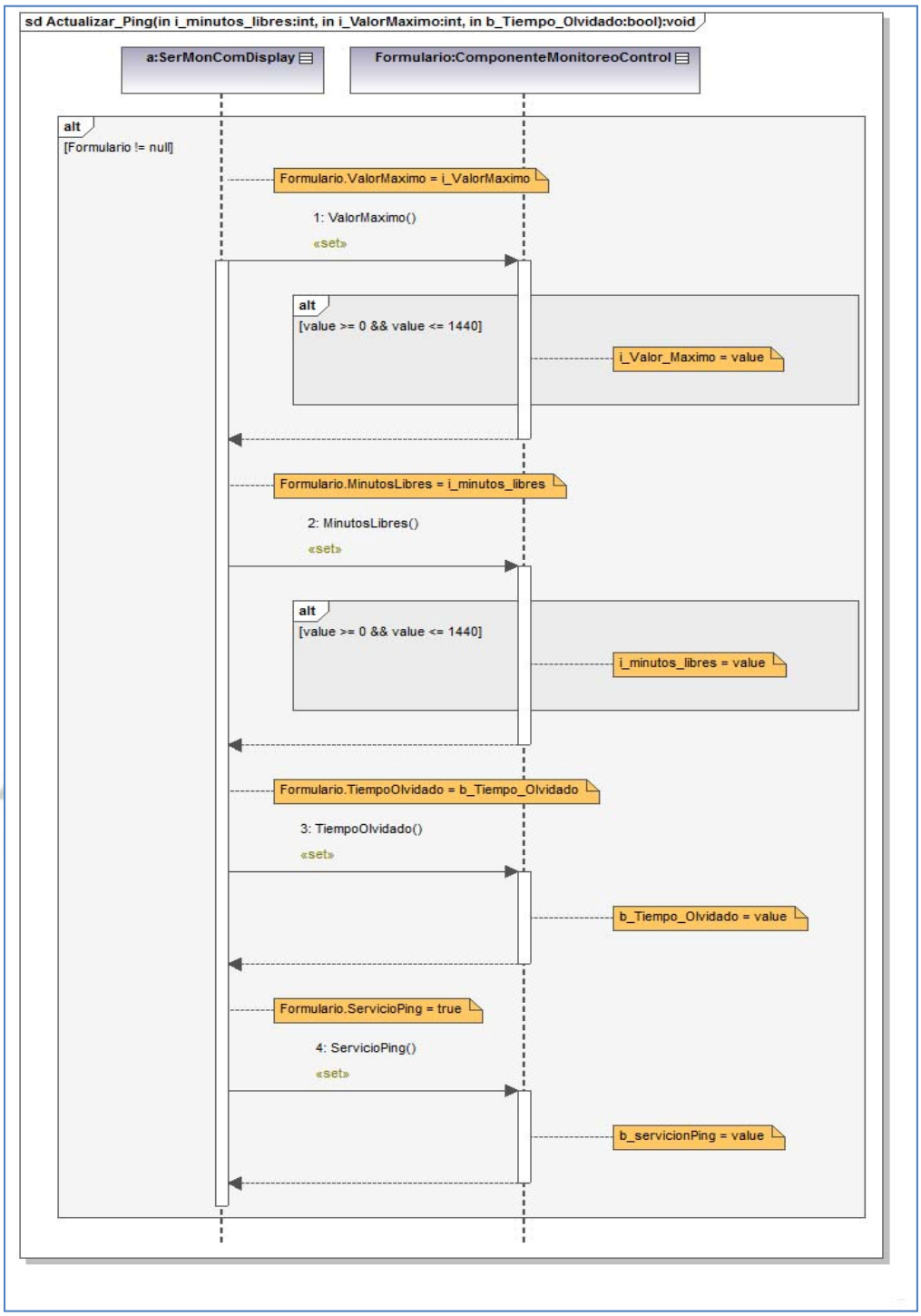


Figura 3.96 – Diagrama de Secuencia Actualizar_Ping.

Fuente: Elaboración Propia.

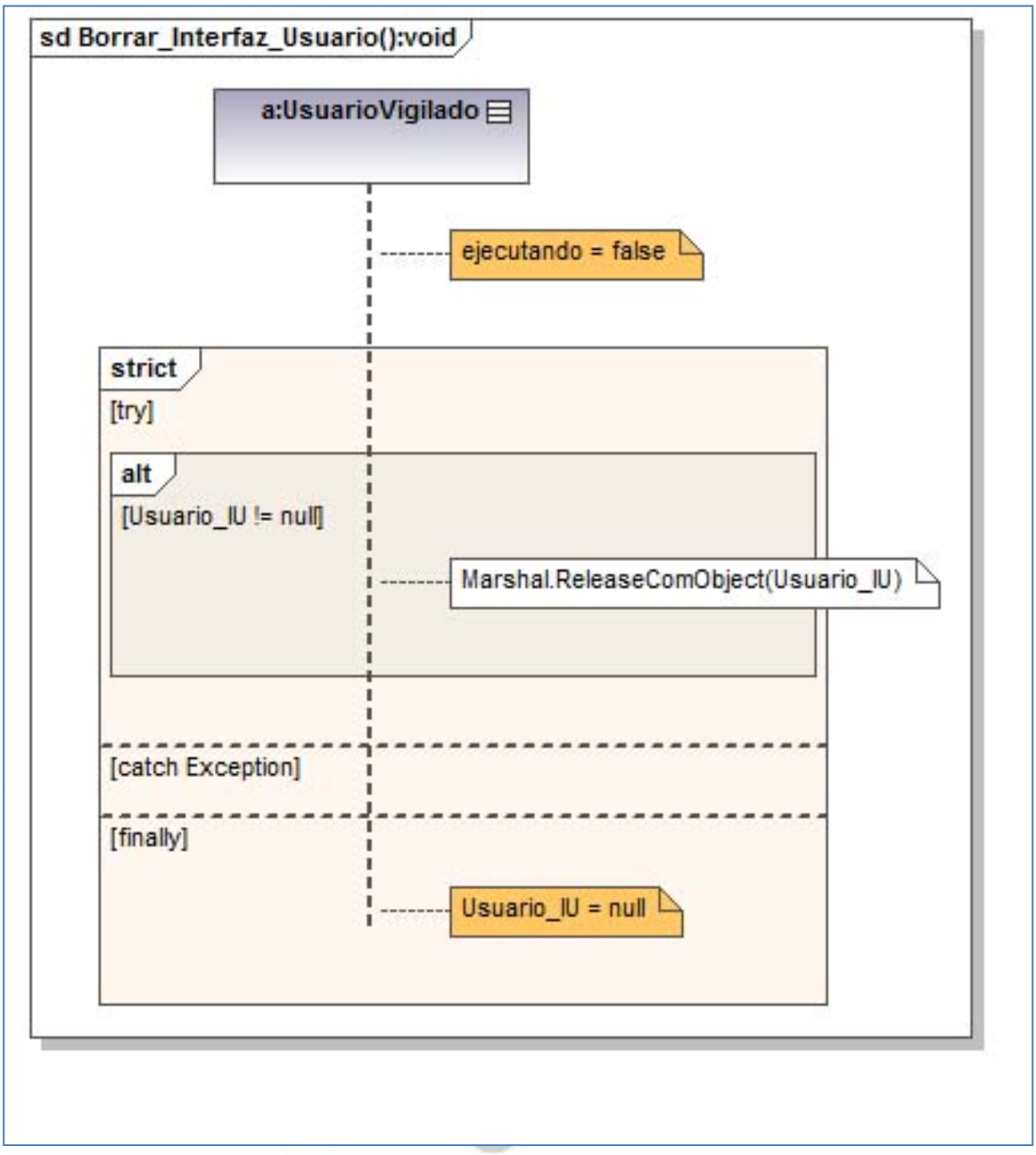


Figura 3.97 – Diagrama de Secuencia Borrar_Interfaz_Usuario.

Fuente: Elaboración Propia.

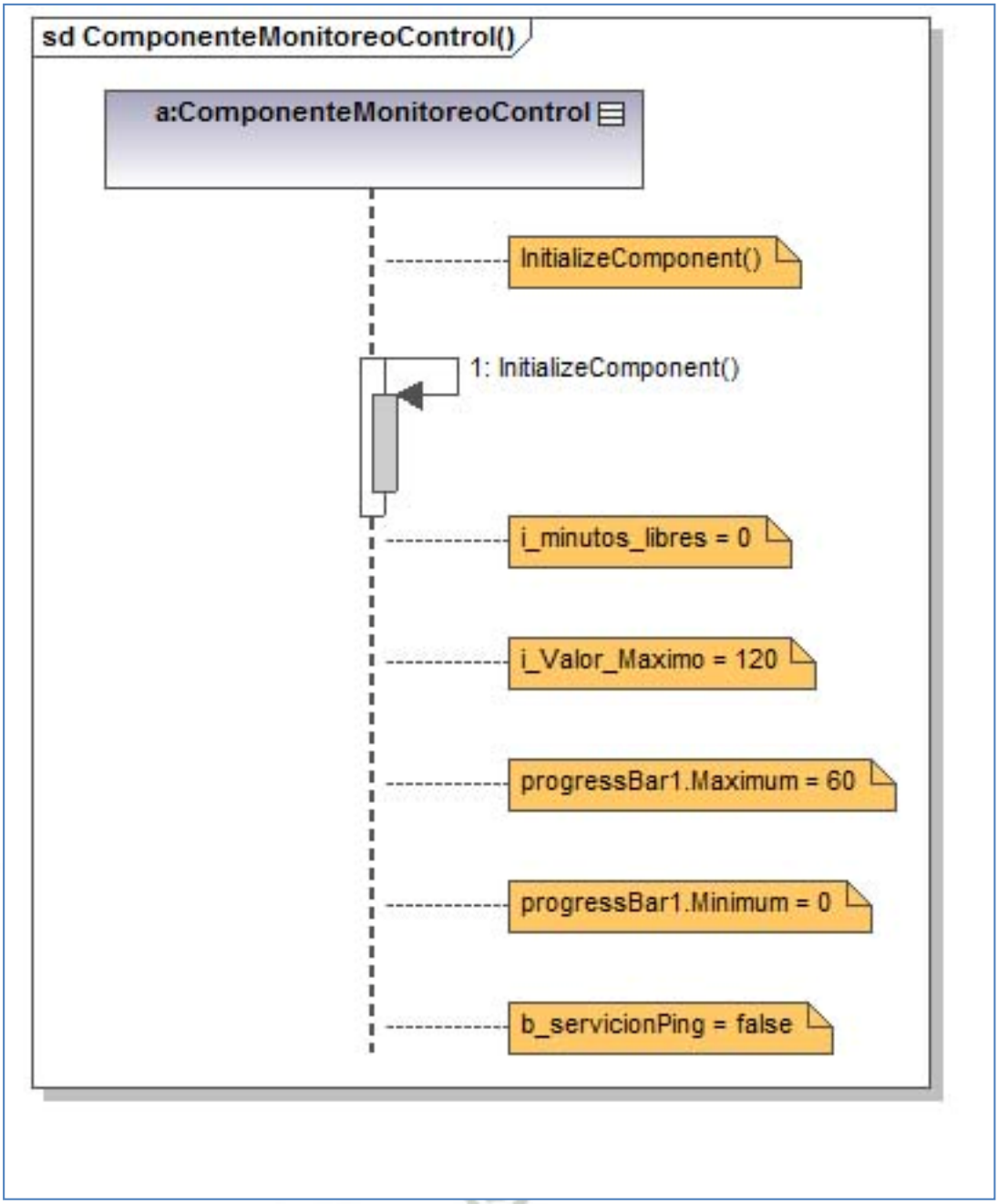


Figura 3.98 – Diagrama de Secuencia ComponenteMonitoreoControl.

Fuente: Elaboración Propia.

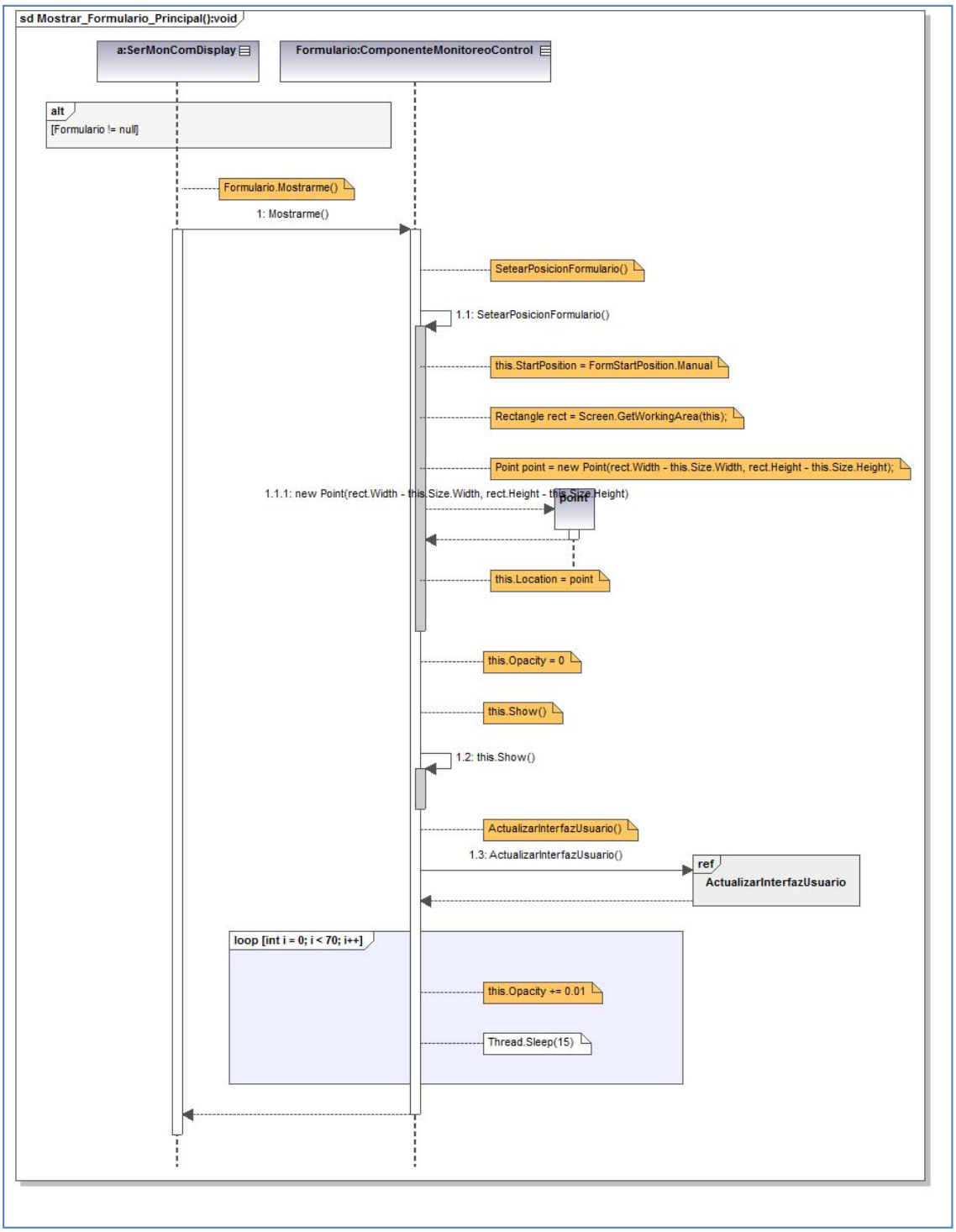


Figura 3.99 – Diagrama de Secuencia Mostrar_Formulario_Principal.

Fuente: Elaboración Propia.

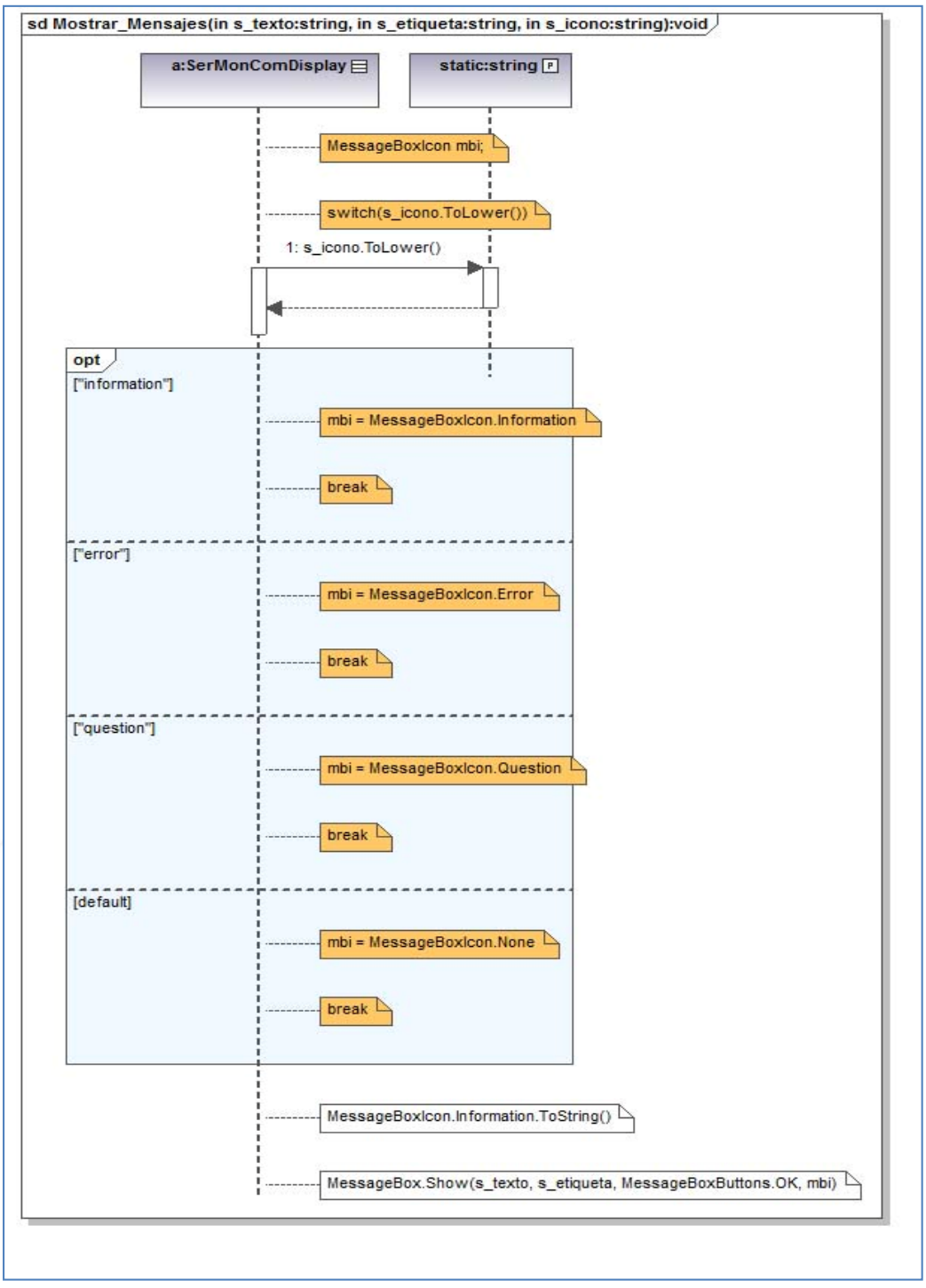


Figura 3.100 – Diagrama de Secuencia Mostrar_Mensajes.

Fuente: Elaboración Propia.

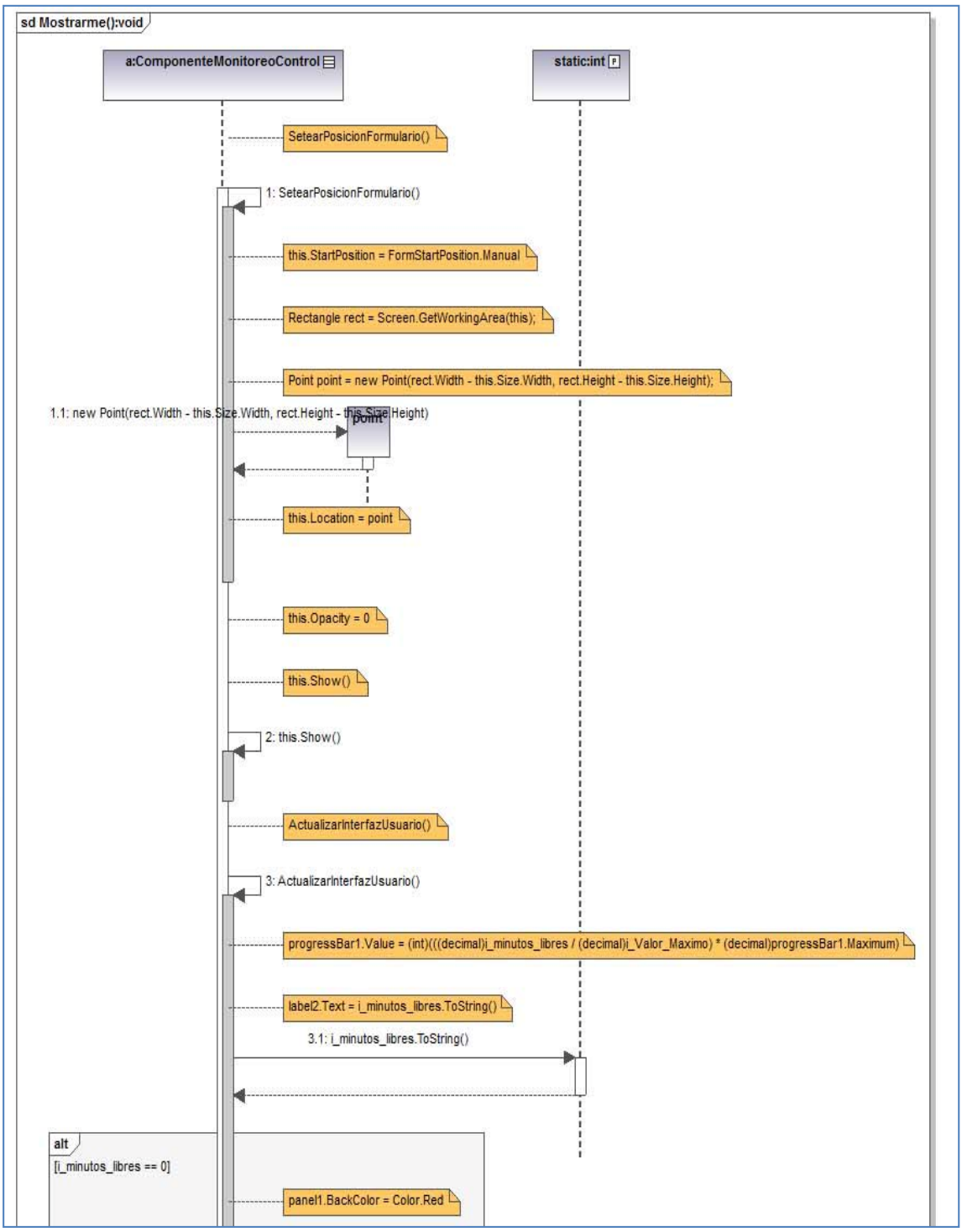


Figura 3.101 – Diagrama de Secuencia Mostrarme (Parte 1).

Fuente: Elaboración Propia.

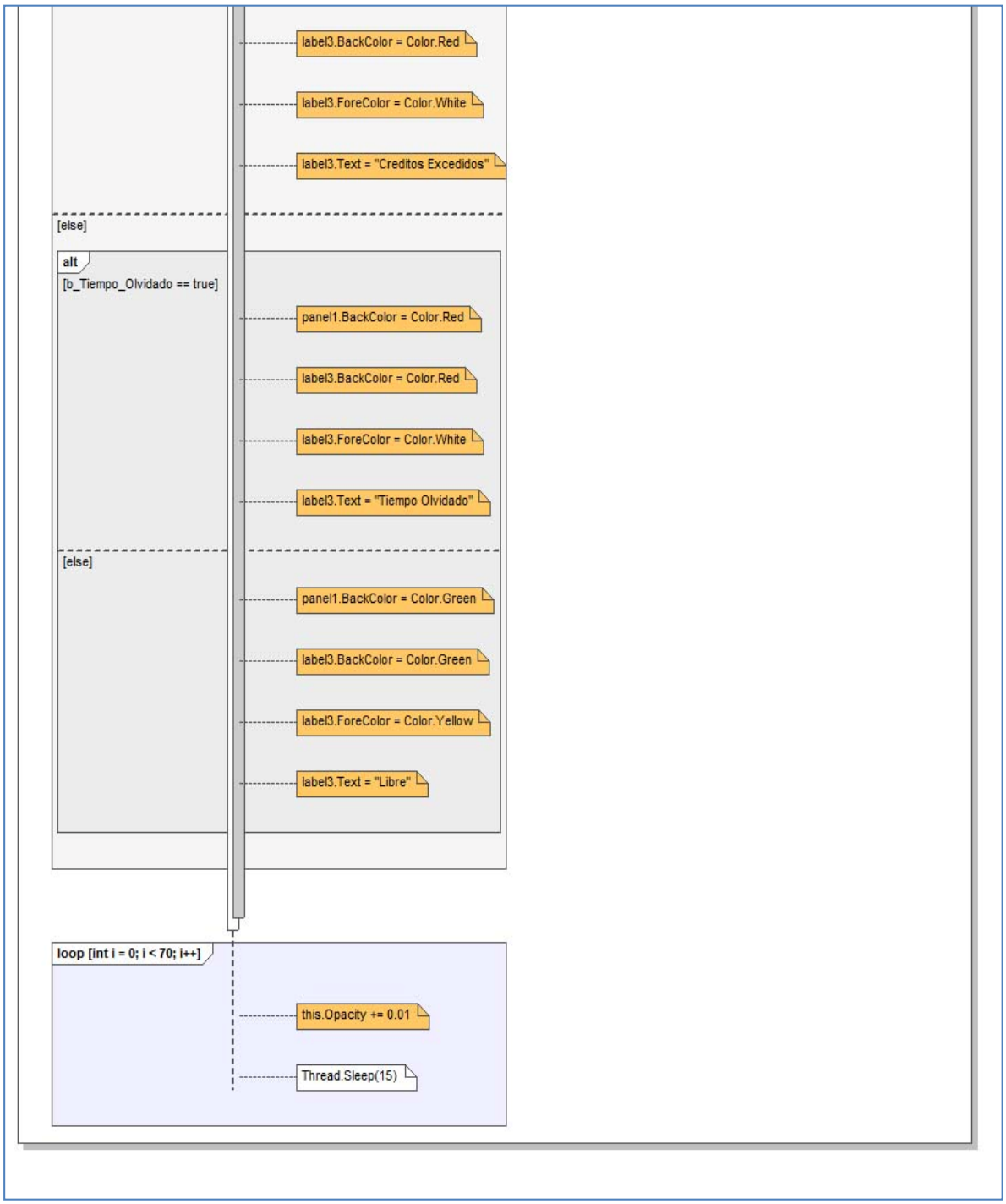


Figura 3.102 – Diagrama de Secuencia Mostrarme (Parte 2).

Fuente: Elaboración Propia.

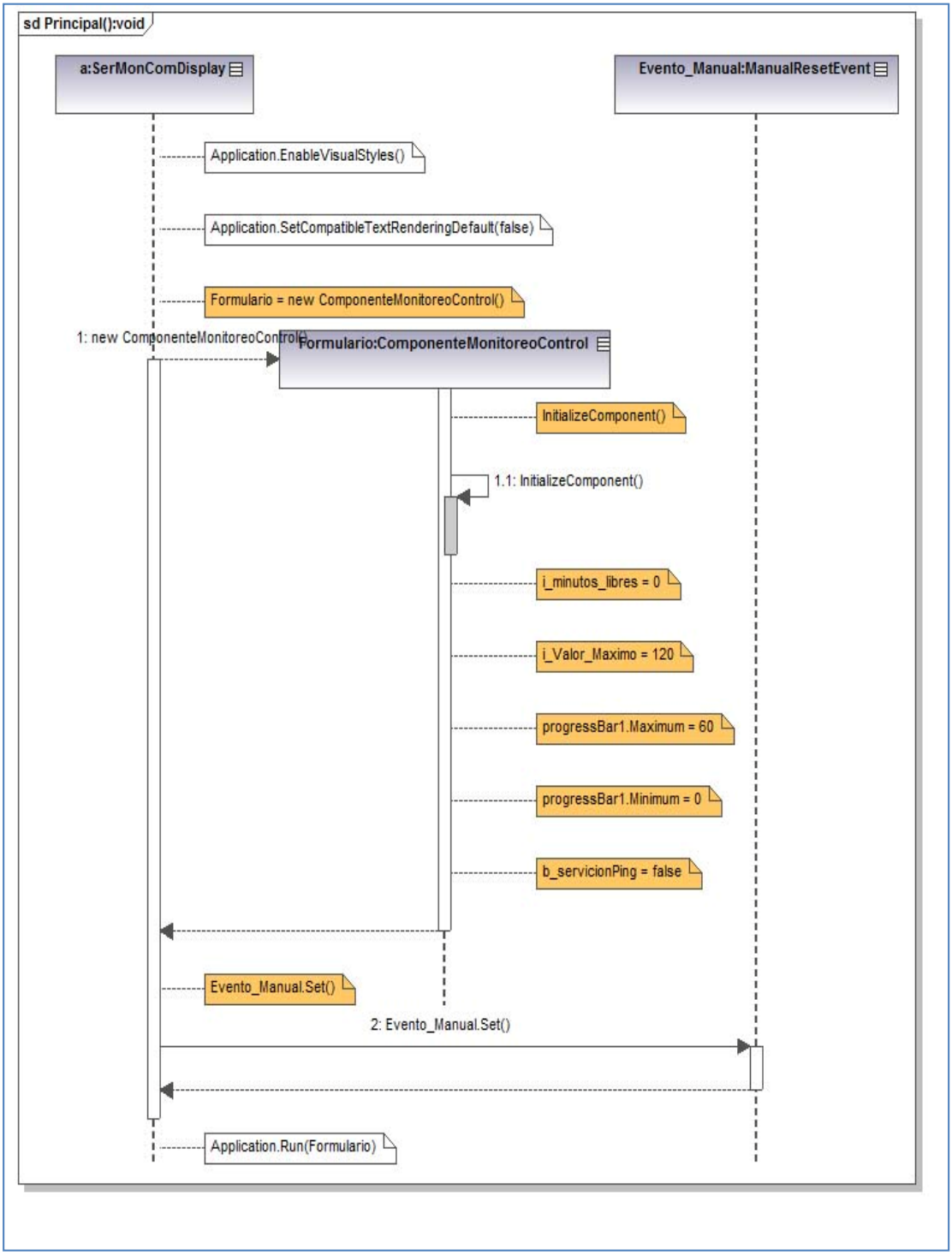


Figura 3.103 – Diagrama de Secuencia Principal.

Fuente: Elaboración Propia.

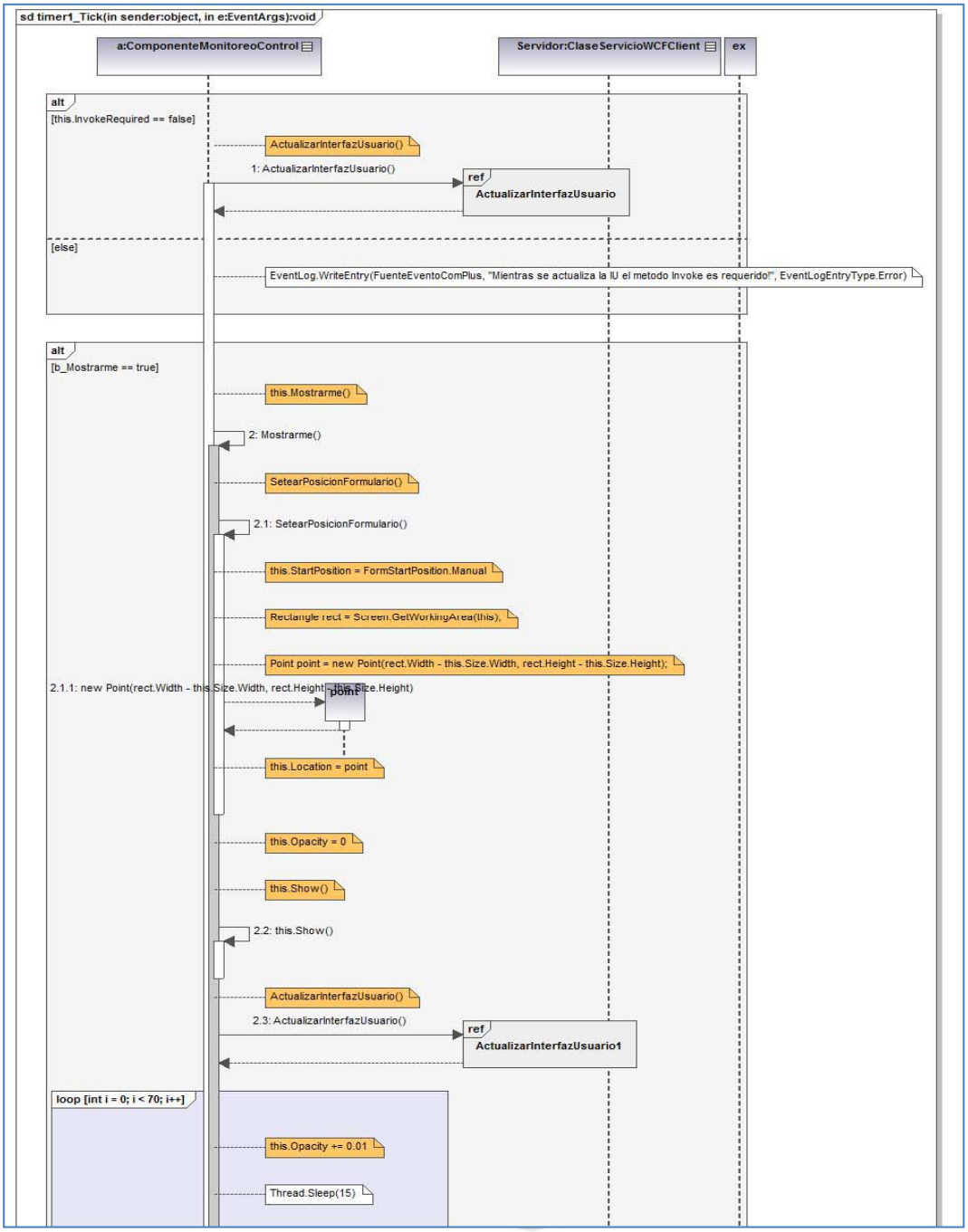


Figura 3.104 – Diagrama de Secuencia Timer1_Tick (Parte 1).

Fuente: Elaboración Propia.

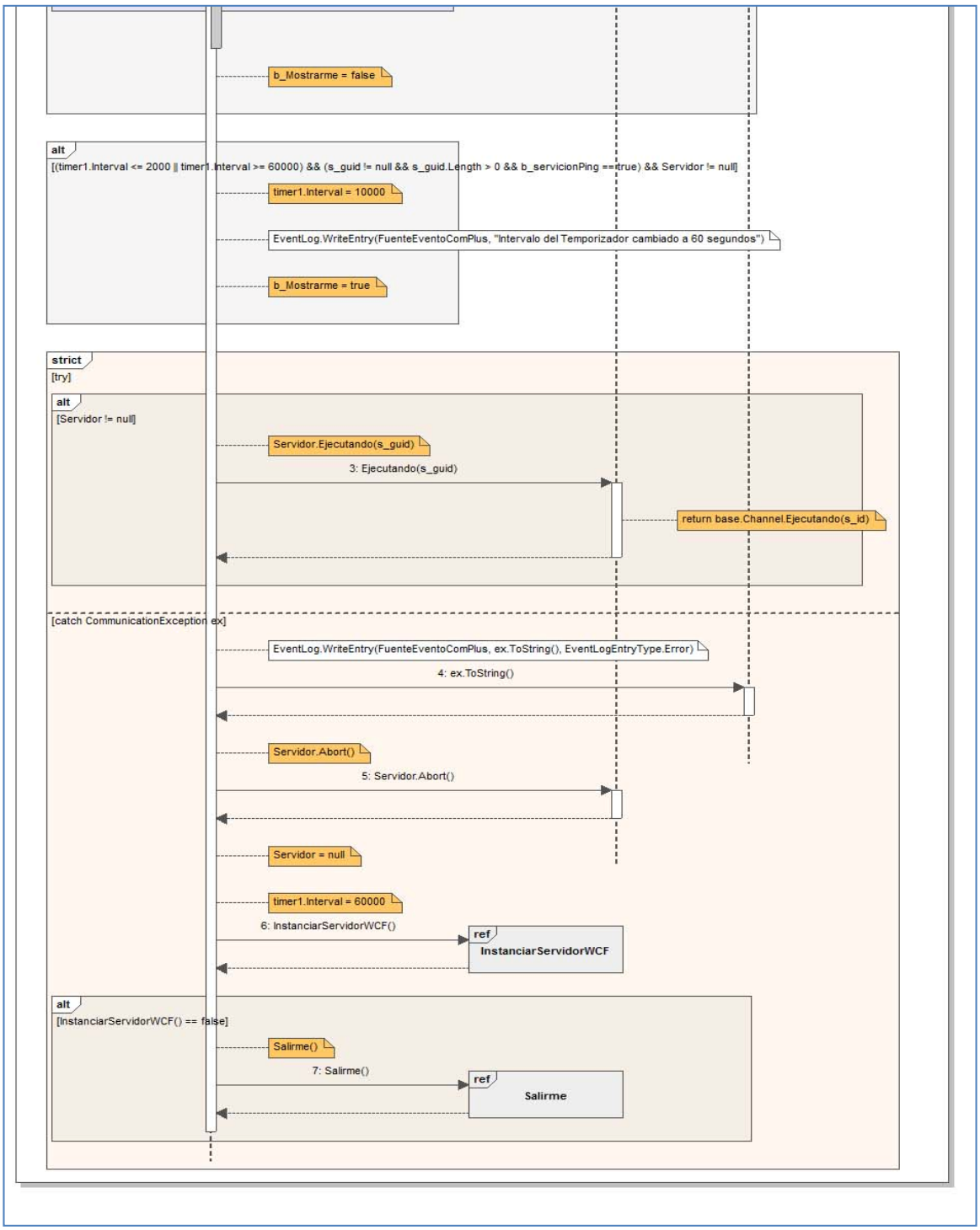


Figura 3.105– Diagrama de Secuencia Timer1_Tick (Parte 2).

Fuente: Elaboración Propia.

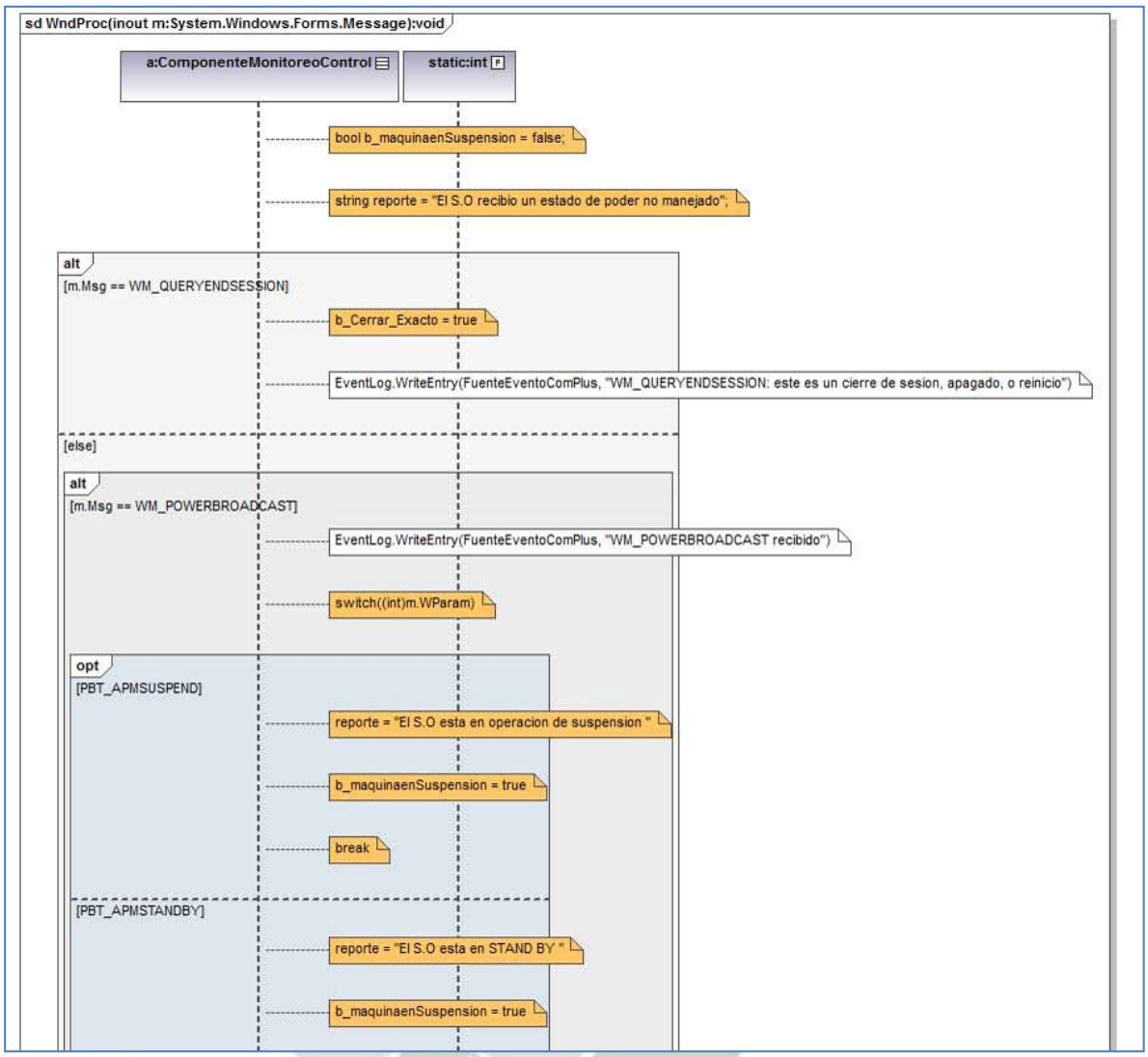


Figura 3.106 – Diagrama de Secuencia WndProc (Parte 1).

Fuente: Elaboración Propia.

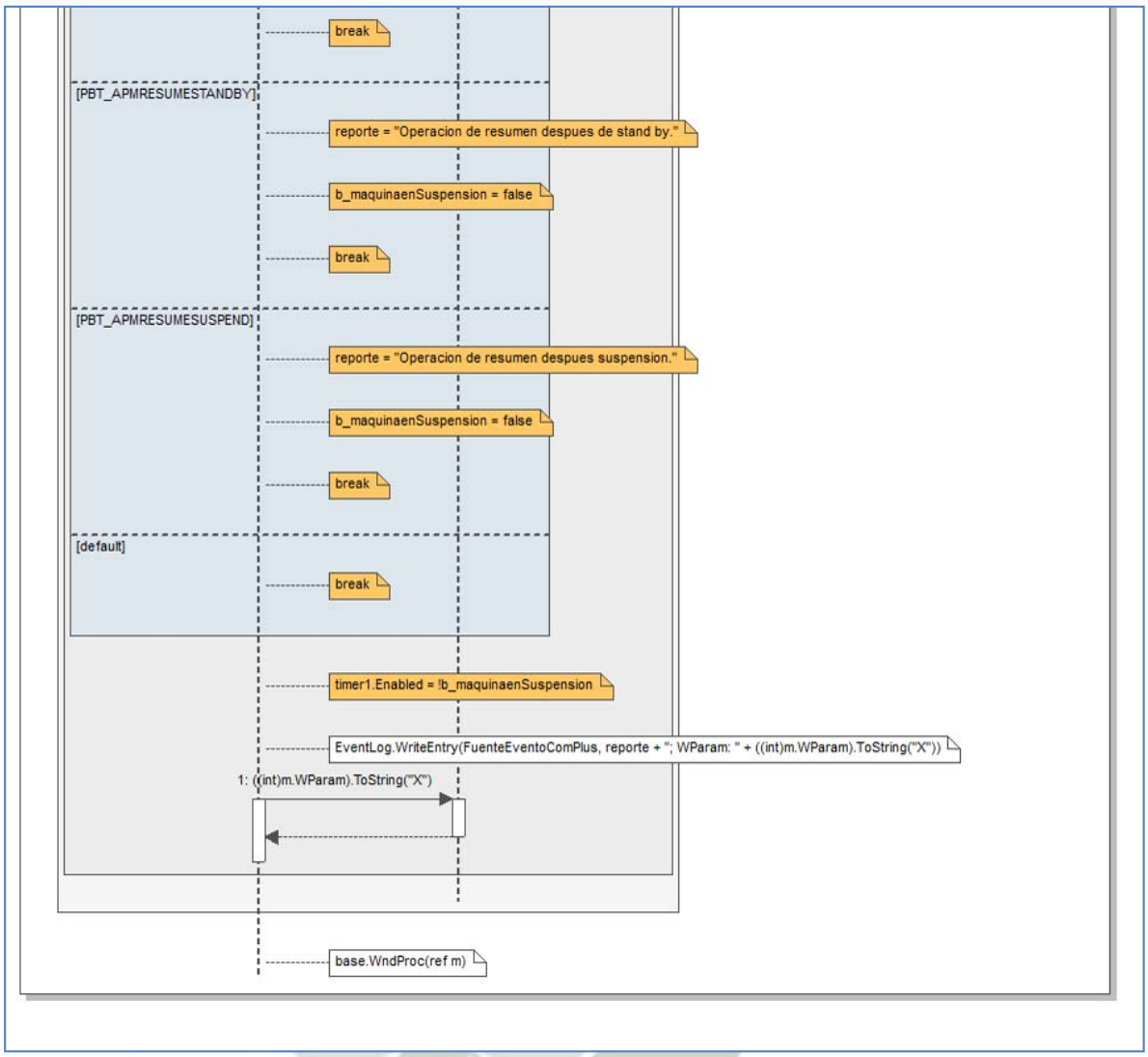


Figura 3.107 – Diagrama de Secuencia WndProc (Parte 2).

Fuente: Elaboración Propia.

3.4.8 Diagrama de Componentes

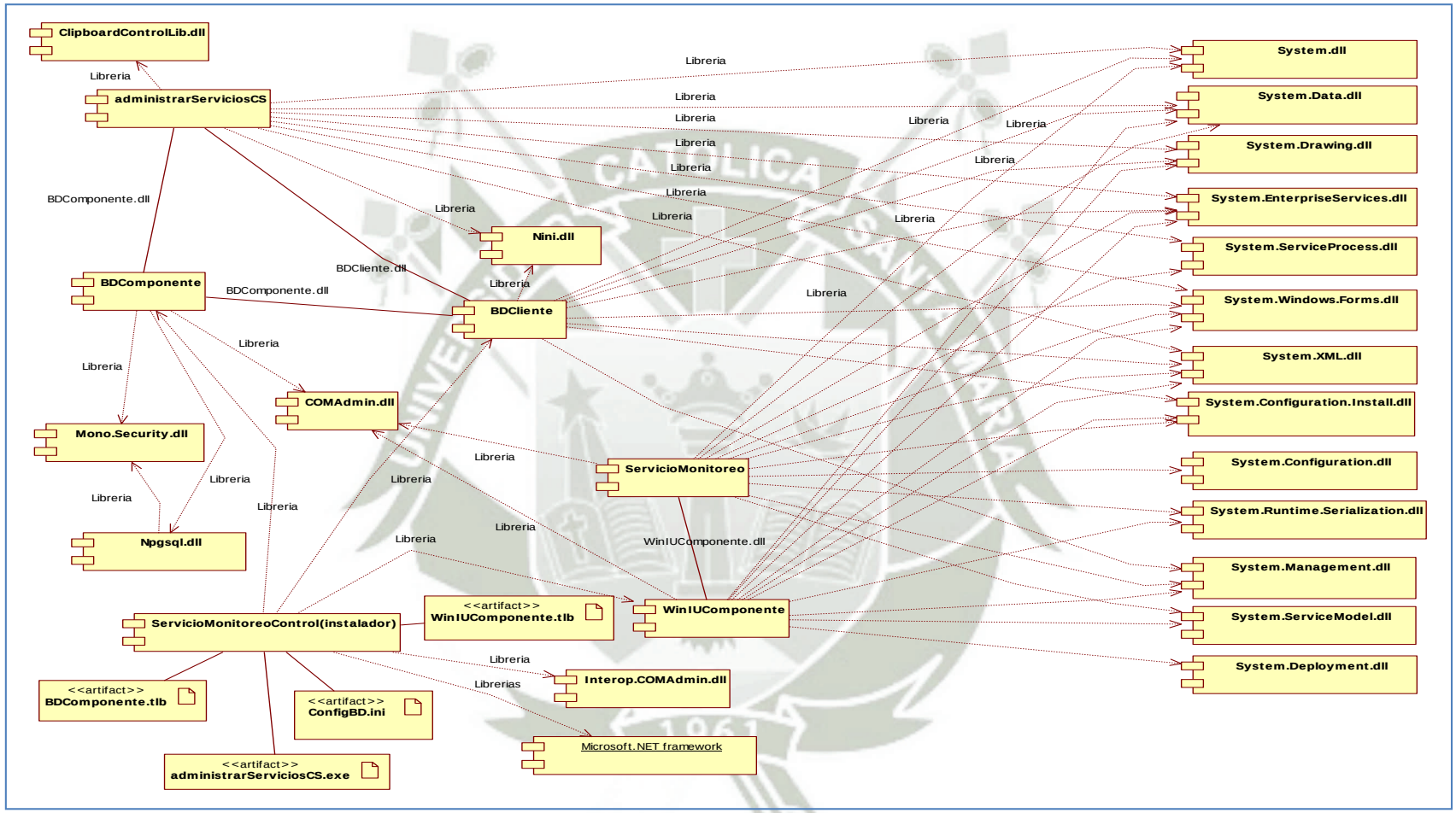


Figura 3.108 – Diagrama de Componentes.

Fuente: Elaboración Propia.

3.4.9 Diagrama de Despliegue

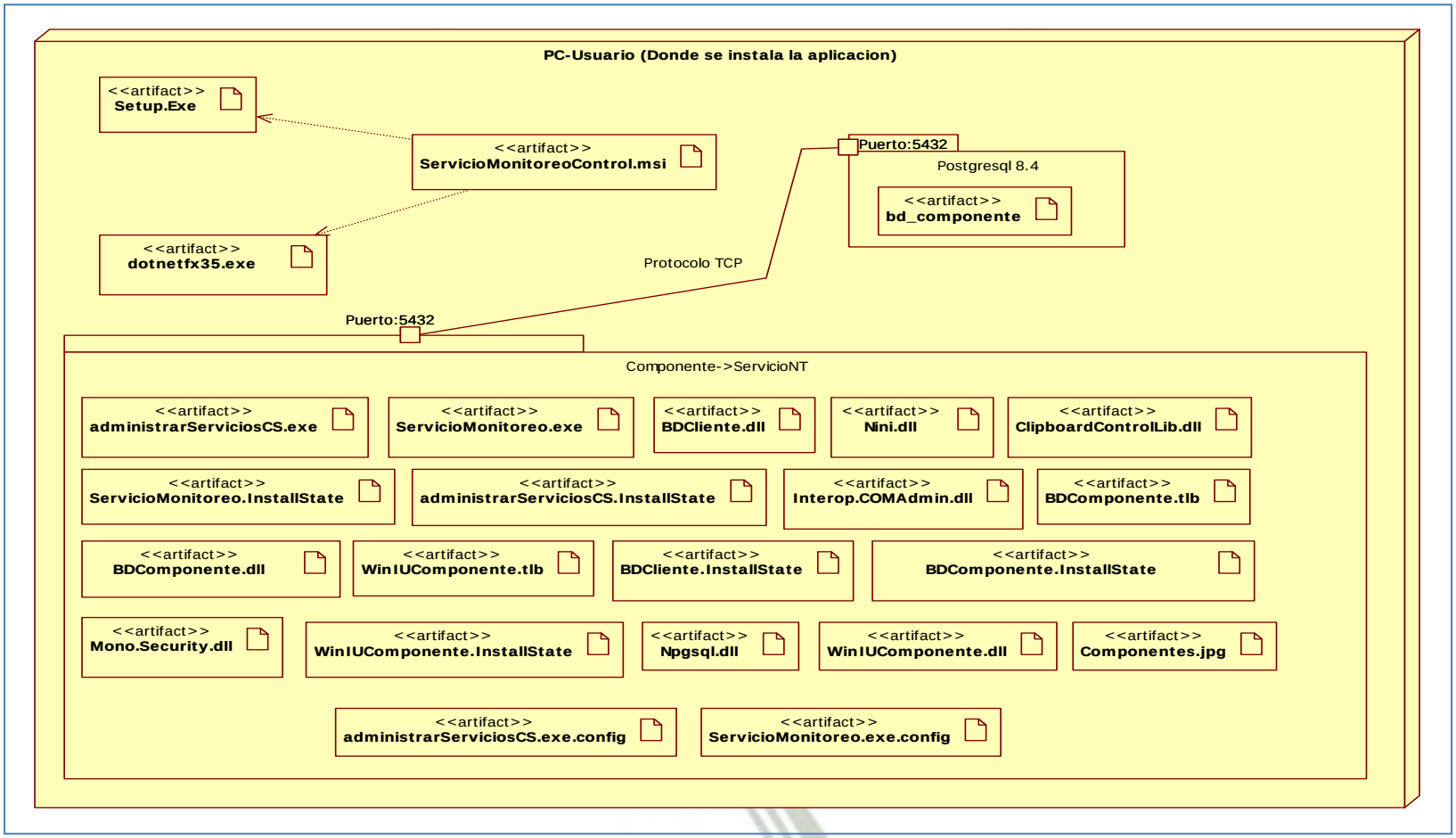


Figura 3.109 – Diagrama de Despliegue.

Fuente: Elaboración Propia.

3.5 Desarrollo y Construcción del Componente

En este punto explicaremos paso a paso cómo se creó el componente de seguridad basándonos en el análisis y diseño del mismo y en su núcleo básico que es COM+ acá se describirá como haremos para que este interactué con el escritorio, el componente deberá funcionar a la perfección con las versiones anteriores de Windows como XP, Vista y Server 2003, así como con Windows 7 que es nuestra plataforma de SO elegido para la propuesta.

3.5.1 Módulos de la Solución Propuesta

La solución que se propone en la presente tesis constara de 6 módulos que son los siguientes:

3.5.1.1 Módulo ServicioMonitoreo

Este módulo tiene como finalidad crear y manejar el servicio de seguridad con WCF que interactuara con una interfaz de Usuario definida en WinIUComponente.

3.5.1.2 Módulo WinIUComponente

Este módulo tiene como finalidad crear y manejar el componente COM+ que interactuara con el servicio de seguridad con WCF.

3.5.1.3 Módulo BDComponente

Este módulo tiene como finalidad crear y manejar el componente para la conexión a la Base de Datos haciendo uso de COM+.

3.5.1.4 Módulo BDCliente

Este módulo tiene como finalidad crear y manejar las interfaces de usuario IU, para interactuar con el componente de Base de Datos.

3.5.1.5 Módulo AdministrarServicioCS

Este módulo tiene como finalidad administrar el servicio de seguridad con WCF haciendo uso de la API de Windows, así como también de administrar los archivos de configuración de la solución propuesta tanto para la configuración del servicio, la conexión de base de datos, el mantenimiento de usuarios y los permisos del sistema operativo. Adicionalmente se tendrá un control para evitar el copiado de archivos a todos los usuarios locales.

3.5.1.6 Módulo ServicioMonitoreoControl

Este módulo tiene como finalidad realizar el despliegue de la solución completa, generando un instalador haciendo uso de las librerías y dependencias de la solución propuesta.

3.5.2 Requisitos de Diseño

Aquí se propone una lista arbitraria de los requisitos que se encontraron como factibles en cuanto a los requisitos de nuestro componente:

- La aplicación está dirigida a usar los Servicios de Windows y debe ser compatible con los sistema operativos Windows que puedan además instalar el .NET Framework 3.0
- La aplicación debe ser configurable por ejemplo, a través de un archivo de configuración de aplicación, como vemos en la Figura 3.110.
- El archivo de configuración contiene por lo menos estos atributos de sólo lectura:

- La lista de usuarios a vigilar, los cuales representan los nombres de los usuarios que están en el equipo (cuentas de usuario del equipo). Véase el atributo "**UsuariosaVigilar**".
- La lista de aplicaciones a vigilar, como son iexplore, msnmsgr, firefox, etc. Véase el atributo "**ProcesosaVigilar**".
- El crédito diario (el límite en cuestión de minutos) para usar los programas mencionados anteriormente, por ejemplo, 150 minutos (2,5 horas). Esto se aplicará a cada usuario vigilado. Véase el atributo "**CreditoDiario**".
- La lista de los días de semana a ver. Véase el atributo "**DiasaVigilar**". Los números que aparecen (1 al 7) son codificadas como 1 que sería lunes, 2 es martes, y así sucesivamente.
- Los Tiempos bloqueados que serian intervalos de tiempo prohibidos (por ejemplo, 16:00-18:00) en el que incluso si la dosis diaria no se supera, los programas vistos no se pueden utilizar. Vea el atributo "**TiemposBloqueados**", esto consiste en pares de cifras. Esto significa que si ponemos (10, 12, 20, 12), se leerá como que de 10 a.m.-12 a.m. y entre las 20:00 y el mediodía, el uso de aplicaciones vigiladas estará prohibido.
- Un nombre de Log para guardar los sucesos registrados, Vea el atributo "**NombreLog**".
- Un nombre de Fuente para el log de eventos. Vea el atributo "**NombreFuente**".
    - En cuanto al Logging de encendido/apagado y de reinicio del equipo no se debe inicializar el contador y sólo se debe seguir de acuerdo a la política

del archivo de configuración de la aplicación. Esto supone también, que la interrupción del servicio ha de conservar los datos que van siendo contados.

- El usuario debe ser notificado de una manera no intrusiva de los minutos todos los días y sobre la prohibición entre lapsos de tiempo. Esto supone una interfaz de usuario (UI) que debe mostrar esto.
- La interfaz de usuario no debe cerrarse y salir de la aplicación, sólo deberá esconderse. El usuario debe tener la posibilidad de reactivar la interfaz de usuario a través de un icono de notificación en la barra de herramientas del sistema.
- La interfaz de usuario debe ejecutarse en el contexto de forma interactiva tan solo con usuarios registrados.
- Cambio rápidos que tienen que contar con el apoyo de Eventos.
- Los Eventos de Poder de Energía (Power) no debe interrumpir la aplicación.

```
…
<configuration>

<appSettings>
    <add key="ProcesosaVigilar" value="iexplore,msnmsgr,firefox"/>
    <add key="CreditoDiario" value="150"/>
    <add key="NombreLog" value="FrancisLog"/>
    <add key="NombreFuente" value="ProgramaMonitoreo"/>
    <add key="DiasaVigilar" value="1,2,3,4,5,6,7"/>
    <add key="TiemposBloqueados" value="15,24"/>
    <add key="UsuariosaVigilar" value="Francis"/>
 </appSettings>

</configuration>
```

Figura 3.110 Archivo de configuración de aplicaciones.

Fuente: Elaboración Propia.

3.5.3 Adecuando nuestra Solución

En primer lugar debemos tener en cuenta que el servicio del que estamos hablando es el de un observador a nivel de procesos que tiene una lista actualizada y que actúa de forma silenciosa en segundo plano (background). Además, el servicio tiene que ser capaz de examinar los tokens de seguridad de los procesos que se están observando. Esto tiene que llevarse a cabo periódicamente en función del filtrado de los usuarios que ejecutan los procesos de acuerdo con la lista de usuarios vigilados.

Según lo estudiado Microsoft sugiere ejecutar los servicios con el usuario que tenga los privilegios mínimos posibles. De acuerdo a esto podemos elegir desde un: **Servicio Local, Servicio de Red, Sistema Local y una Cuenta de Usuario Dedicado**.

Teniendo en cuenta todas las posibilidades de comenzar con el servicio local, es que debemos inclinarnos por la cuenta de SISTEMA LOCAL. Vamos a explicar el por qué; La apertura de tokens (fichas) de los procesos nos dicen que "La seguridad es una operación que necesita privilegios SE_SECURITY_NAME. Este privilegio es, sin embargo concedido por defecto sólo para los administradores y la cuenta del SISTEMA LOCAL.

Investigando un poco más sabemos que desde Windows Vista, tan sólo al iniciar sesión de forma interactiva los usuarios puede interactuar con el escritorio. Y es aquí que viene el verdadero desafío del componente de la propuesta, como se ha planteado nuestro servicio, estará observando en

silencio todos los procesos en ejecución, no puede mostrar ninguna interfaz de usuario y por lo tanto, esta tarea debe ser delegada en el usuario que está actualmente conectado de alguna manera digamos “mágica”. También debemos tener en cuenta, que no queremos ser responsables de mantener las contraseñas o por ejemplo, utilizar la conocida suplantación, entonces acá la tarea obligatoria es proporcionar una interfaz de usuario iniciado por el servicio que se ejecuta, para esto debemos tener en cuenta que el servicio y la interfaz de usuario, obviamente, van a convivir en diferentes sesiones.

Para esto existe una tecnología antigua y bien conocida y que todavía está disponible y es compatible con todas las versiones de Windows. Nos referimos a los servicios empresariales, o los servicios COM+, también conocido como el Servicio de Componentes. Si Utilizamos una aplicación fuera del proceso de COM+ este puede ser configurado para ejecutarse en el contexto de seguridad del usuario conectado en forma interactiva como se muestra en la figura 3.111.

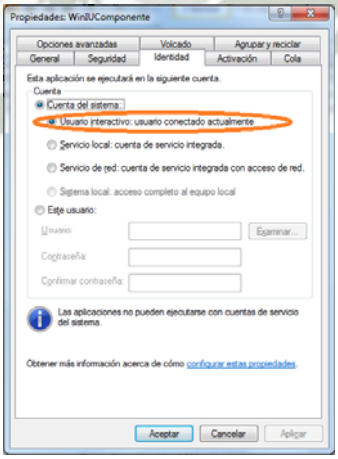


Figura 3.111 la aplicación COM+ de identidad.
Fuente: Elaboración Propia.

Esto parece una solución muy eficaz, ya que cada vez que el servicio se inicie se creará una instancia nueva de interfaz del usuario, automáticamente se tomará el usuario actual que haya iniciado sesión interactivamente en la identidad del usuario (ID). Técnicamente, esto significa que el sistema iniciará un proceso suplente nombrado como **Dllhost.exe** que se ejecutara en el contexto de seguridad de ese usuario. Este proceso de interfaz de usuario no tendrá permitido salir mediante el usuario interactivo, también preferentemente se tendrá que proporcionar un icono de notificación en la barra de herramientas del sistema. El proceso de nuestro servicio puede controlar fácilmente el tiempo de vida del proceso de la interfaz de usuario, a partir de que al recibir OnStart() inicio de eventos del servicio, o el cierre de la recepción de eventos OnStop(), podrá enviar periódicamente actualizaciones, mientras que el servicio este levantado y funcionando.

Luego nos quedaría otro problema por resolver que seria, el de cómo controlar eventos de log-off/log-on de un usuario?, pongamos como ejemplo que un usuario interactivo termina la sesión y un usuario (que, de acuerdo con la lista proporcionada, es también un usuario a vigilar) inicia una sesión. Cerrar la sesión en la interfaz de usuario será la salida si nuestro servicio lo quiere o no. Por lo tanto, el servicio tiene que estar al tanto de la interfaz de usuario al salir. Hay un sistema de ventanas que se conoce como WM_QUERYENDSESSION, este evento se provoca cada vez que el usuario cierra la sesión. Lamentablemente, este evento está destinado a aplicaciones de Windows y los servicios de Windows por defecto no son conscientes de los eventos del sistema de Windows. Por lo tanto, parece una buena idea que la interfaz de usuario se ejecute como un componente COM+ que debe tener un proceso de sustitución con

la capacidad de hablar de nuevo a nuestro Servicio de NT por lo menos con el fin de notificar al Servicio de NT acerca de salir y correr los estados. Por este motivo, se decidió utilizar Windows Communication Foundation (WCF). El Servicio de NT puede hospedar un servicio WCF que proporciona un canal de canalización con nombre para la interfaz de usuario.

Teniendo en cuenta que el diseño de la aplicación consiste de dos piezas (El Servicio NT y la interfaz de usuario sustituto (COM+) estos parecieron ser un problema más, cuando se realizaron pruebas de cambio rápido de sesión de usuario. Imagínese que el primer usuario registrado no cierre la sesión, en lugar de eso hace clic en "Cambiar de usuario". Esto preservará el escritorio del primer usuario y creara una nueva sesión para el segundo. En este momento, ya se está ejecutando la aplicación COM+ por lo que la interfaz de usuario no experimentará ninguna notificación de fin de sesión y seguirá funcionando. En estos términos, incluso si el servicio de NT crea una nueva interfaz de usuario COM+, esto se materializará en el contexto del primer usuario y el segundo usuario no recibirá ninguna interfaz de usuario por lo que no se mostrara nada. Esto se debe al hecho de que, el proceso de COM+ sustituto que se creó con el primer usuario no se acabará y seguirá corriendo, con la identidad del primer usuario. Esto tiene que ser terminado de una manera ligera y el servicio de NT tiene que cerrar la aplicación COM+ cuando se detecte que se logueó un usuario nuevo.

3.5.4 Dentro de la interfaz de usuario

Echemos un vistazo más de cerca en el COM+ fuera del proceso de ser una interfaz de usuario. Más allá de las cuestiones de diseño y desarrollo, también tenemos que considerar la fase de despliegue. A veces es una buena solución y casi suficiente para

el uso de despliegue usar el comando xcopy, esto es confiar en el sistema operativo para hacer todas las cosas detrás de las escenas con el fin de registrarse y empezar el pedido hacia el Servicio de Componentes. Aunque esto suele ser una buena práctica para los componentes In-Proc, en nuestro caso y según el escenario que tenemos esto resulta en una estrategia desesperada y que nos producirá un AccessDeniedException que será lanzado al crear la instancia de la interfaz de usuario. Por lo tanto para remediar esto, tenemos que decirle a la aplicación COM+ que es exactamente instancia con permisos de confianza.

3.5.5 Proceso de Diseño de Componentes - Fuera de COM+

Iniciaremos con una plantilla de proyecto de Aplicación de Windows. Ya que nos permite diseñar la interfaz de usuario rápidamente como vemos en la (Figura 3.112). En este momento no tenemos que preocuparnos demasiado por si el componente se ejecutará como un proceso del servidor (aplicación de biblioteca). Esto se decidirá en el momento del despliegue. La interfaz de usuario a diseñar es simple y nos mostrara los minutos restantes y una barra de progreso ajustada. La interfaz de usuario también le dirá al usuario si existe un plazo prohibido (fondo rojo) o si puede usar las aplicaciones normalmente (fondo verde). Estos ajustes se pueden configurar en el archivo de configuración del servicio de NT como se describió anteriormente.

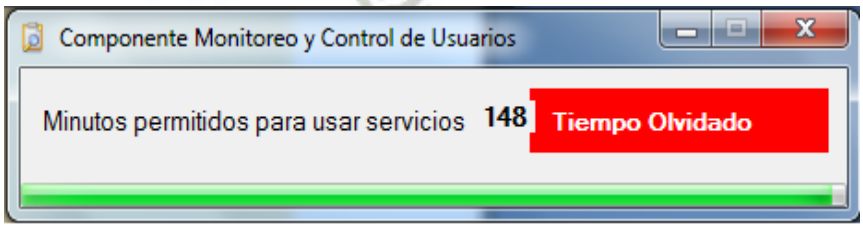


Figura 3.112 Interfaz de usuario de la Aplicación hecha en Windows Forms. Fuente: Elaboración Propia.

Cambiaremos el nombre por defecto Form1 de nuestra clase a algo más factible como ProgramaComponente.cs, para esto debemos cambiar el nombre de archivo del formulario de ProgramaComponente.cs.cs y luego ir a Program.cs (donde está la función Application.Run()) y debemos cambiarle el nombre a ProgramaComponente.cs, a continuación, abrimos este archivo y creamos una interfaz donde se registraran todas las firmas de los métodos del Servicio de NT que podremos utilizar para controlar la interfaz de usuario, véase la Figura 3.113.

```
…
    [ComVisible(true)]
    [Description("Exponiendo la Interfaz Publica")]
    [Guid("BB874239-CDDC-4477-A3F5-7D8B8F2E9DCF")]
    [InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]
    public interface ISerMonComMensajes
    {
         [DispId(1)]
        void Mostrar_Formulario_Principal();
        [DispId(2)]
        void Ocultar_Formulario_Principal();
        [DispId(3)]
        void Salir_Formulario();
         [DispId(4)]
         void Actualizar_Ping(int i_minutos_libres, int
         i_ValorMaximo, bool b_Tiempo_Olvidado);
        [DispId(5)]
        void Salir_Proceso(string s_procesoaSalir);
        [DispId(6)]
        void Mostrar_Mensajes(string s_texto, string s_etiqueta,
         string s_icono);
        [DispId(7)]
        void Setear_Guid(string s_guid);}
…
```

Figura 3.113 Interfaz de Componentes.

Fuente: Elaboración Propia.

Explicaremos brevemente el propósito de estos métodos.

El primero marcado con el identificador **DispId(1)** ayuda simplemente a que aparezca el formulario en el escritorio. Esta será ubicarla en la esquina inferior derecha del escritorio.

El segundo método **DispId(2)** hará lo contrario, y ocultara el formulario.

El tercer método **DispId(3)** permitirá que la interfaz de usuario pueda eliminarse a sí misma para luego salir.

El cuarto método y el más importante es el identificador **DispId(4)**, el cual será utilizado para enviar periódicamente actualizaciones de la interfaz de usuario sobre los límites de tiempo disponible.

El quinto método **DispId(5)** le pedirá a la interfaz de usuario que salga del proceso de prohibición. Esta es una buena solución, porque el poderoso servicio de NT sólo permitirá a la interfaz de usuario saber cuándo es el momento para salir del proceso de prohibición. El Servicio de NT no va a matar todos los procesos, en su lugar la tarea sucia se delega precisamente en el contexto de seguridad de los usuarios, que iniciaron la aplicación.

El sexto método **DispId (6)** mostrará una DialogBox regular.

El séptimo método **DispId(7)** le enviará una identidad única a la interfaz de usuario con el fin de identificarlo.

El siguiente paso es implementar este tipo de interfaz. Por esta razón, abrimos el archivo ProgramaComponente.cs y crearemos una nueva clase que se deriva de la clase del framework ServicedComponent e implementa la interfaz ISerMonCom Mensajes previamente definidos (Véase la Figura 3.114), que junto a las firmas de interfaz implementado, no es un constructor, ahora definimos un método public void Principal con un atributo [STAThread]. El constructor de la clase SerMonCom Display() iniciara un subproceso que creará una instancia del formulario.

El Application.Run (myForm) no se devolverá a todos hasta que la vida formulario comience (que es toda la vida de la sesión). Por lo tanto, no podemos insertar ese

pedazo de código en el constructor, eliminaremos la parte del código de Visual Studio genera que comienza con algo como el programa de clase estática. Nuestro componente no va a usarlo en absoluto. Una clase estática estaría en contradicción con el diseño considerado, lo que requiere instancias exactamente distinguidas por todos los usuarios.

```
…
    [ComVisible(true)]
    [Description("Clase de Servicio de Componente para controlar la
    Interfaz del Usuario")]
    [Guid("423C6028-2B1F-45F8-8906-281CE6BA756D")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("WinIUComponente.SerMonComDisplay")]
    [SecurityRole("Francis")]
    public class SerMonComDisplay : ServicedComponent,
    ISerMonComMensajes
    {

        ComponenteMonitoreoControl Formulario;

        ManualResetEvent Evento_Manual = new
         ManualResetEvent(false);

        public SerMonComDisplay()
        {
            Thread Hilo = new Thread(new ThreadStart(Principal));
            Hilo.Start();
            Evento_Manual.WaitOne();
        }

        [STAThread]
        public void Principal()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Formulario = new ComponenteMonitoreoControl();
             Evento_Manual.Set(); // Señala, que la instancia del
              formulario existe y puede ser usado;
            Application.Run(Formulario);
        }

        void ISerMonComMensajes.Mostrar_Formulario_Principal()
        {
            if (Formulario != null) { }
                Formulario.Mostrarme();
         }        }
         #endregion
    }
…
```

Figura 3.114 COM Clase visible implementa la interfaz.

Fuente: Elaboración Propia.

La implementación de la clase ProgramaComponente.cs sólo contiene un código. El trabajo principal del formulario es para actualizar periódicamente la interfaz de usuario mediante un System.Windows.Forms.Timer. El Servicio de NT envía datos actualizados sobre los tiempos transcurridos y prohibidos según las franjas horarias para la interfaz de usuario a través de Actualizar_Ping(). En esta etapa no nos importa hablar de nuevo a nuestro servidor.

Para terminar la tarea, tenemos que ir al archivo AssemblyInfo.cs e insertar unas cuantas líneas más de código declarativo. En él se abordarán las características esenciales de COM+ (Véase la Figura 3.115) como el nombre de la aplicación, la aplicación COM+ GUID, los atributos de control de acceso y una función de seguridad que se define como "Francis" (este es un nombre arbitrario). El atributo SecurityRole hará cumplir un nuevo papel para la aplicación COM + durante la implementación (Véase la Figura 3.116). Debe tenerse en cuenta que es muy recomendable generar los GUID con la utilidad **guidgen.exe** viene como parte de la instalación del IDE de Visual Studio. Si se desea también se puede cambiar el tipo de la salida del proyecto de "Aplicación para Windows" a "Biblioteca de clases". Esto va a cambiar la extensión del ensamblado producido a .dll que sería en nuestro caso lo más apropiado. Ya que no tiene mucho sentido tener un Exe. El cual no se puede iniciar como un ejecutable real.

```
…
[assembly: AssemblyVersion("1.0.1.0")]
[assembly: AssemblyFileVersion("1.0.1.0")]
[assembly: ApplicationName("WinIUComponente")]
[assembly: ApplicationActivation(ActivationOption.Server)]
[assembly: ApplicationID("FF974E3B-E3E6-4441-B7ED-C5111567EF01")]
[assembly: ApplicationAccessControl(Authentication =
AuthenticationOption.Packet,ImpersonationLevel =
ImpersonationLevelOption.Identify,
```

```
AccessChecksLevel = AccessChecksLevelOption.ApplicationComponent)]
[assembly: SecurityRole("Francis", Description = "Puede Acceder a
Todos los Componentes", SetEveryoneAccess = true)]
...
```

Figura 3.115 AssemblyInfo.cs de la interfaz de usuario.

Fuente: Elaboración Propia.

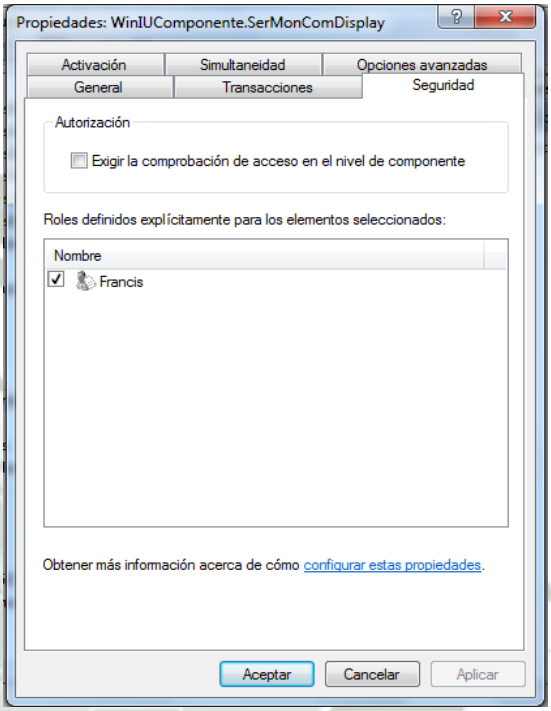


Figura 3.116 Servidor de aplicación de la ficha de seguridad.

Fuente: Elaboración Propia.

Para crear el componente de la base de datos seguimos los mismos pasos, mencionados en el párrafo anterior, quedando configurados como sigue:

```
 /// <summary>
 /// Clase para Conexion a Base de Datos Relacional de Postgresql
 /// Esta Clase maneja todo lo Referente a Consulta de Datos(SOLO
TRAE
 DATOS)
 /// </summary>
   [ComVisible(true)]
   [Description("Clase de componente de Base de Datos Consultas")]
   [Guid("7F1C2A3A-0571-431c-8C1A-075E1B925A43")]
```

```
   [ClassInterface(ClassInterfaceType.None)]
   [ProgId("BDComponente.UsuariosBD")]
   [SecurityRole("Francis")]
   [Transaction(TransactionOption.Supported)]

public class UsuariosBD: ServicedComponent

    {

        /// <summary>
        /// Constructor de la Clase
        /// </summary>
        public UsuariosBD() { }
…
```

Figura 3.117 ComBDRelacional.cs – configuración de componente para UsuariosBD.

Fuente: Elaboración Propia.

```
    /// <summary>
    /// Clase para Transacciones en Base de Datos.
    /// </summary>
    [ComVisible(true)]
    [Description("Clase de componente de Base de Datos
Transacciones")]
    [Guid("C4BEA93D-1EE3-4fc4-8B39-2F5A2DF6470C")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("BDComponente.Usuarios")]
    [SecurityRole("Francis")]
    [Transaction(TransactionOption.Required)]
    public class Usuarios : ServicedComponent
    {
        /// <summary>
        /// Constructor de la Clase
        /// </summary>
       public Usuarios() { }
```

Figura 3.118 ComBDRelacional.cs – configuración de componente para Usuarios.

Fuente: Elaboración Propia.

3.5.6 Despliegue de componentes COM +

Básicamente hay tres maneras de desplegar un Componente de Servicios (ServicedComponent). De acuerdo a lo que elijamos, el ensamblado debe tener un nombre, para lo cual nos vemos obligados a utilizar una firma (por ejemplo, un archivo de claves que contiene los pares de claves pública y privada).

1. Usando el comando RegSvcs.exe.
2. Automáticamente utilizando las capacidades del NET Framework usando su registro automático.
3. Programáticamente, utilizando la clase System.EnterpriseServices. RegistrationHelper, sin embargo, a veces se necesitan ajustes adicionales para la aplicación desplegada.

En general los pasos a seguir son:

- Cargar y registrar un ensamblado.
- Generar, registrar e instalar una biblioteca de tipos en una aplicación COM+ 1.0 según una aplicación especifica.
- Configurar los servicios que se han añadido a su programación de clases.

El registro de un conjunto requiere de un ambiente de plena confianza y para la instalación se necesitan privilegios elevados.

Vamos a hablar de lo que el registro no puede hacer frente a los problemas. El registro No puede agregar cuentas a la función creada (recuerde como agregamos la cuenta Francis Figura 3.116).

Iniciaremos la utilidad **dcomcnfg.exe** para mostrar de forma interactiva como configurar los Servicios de componentes. Haremos clic en la carpeta Usuarios (Figura 3.119) por debajo de Francis daremos seleccionar "Nuevo usuario" y, posteriormente, añadiremos el usuario particular o grupo de seguridad para este Rol. La Adición de las cuentas a la función sin embargo no es suficiente. La solicitud podría contener más de un componente que tiene que volver a configurar con el fin de permitir el acceso a los usuarios de la función. Por

esta razón, la casilla de verificación (check box) en la ficha de seguridad que se muestra en la figura 3.116 debe estar marcada, que por defecto después de la inscripción sigue estando sin marcar.

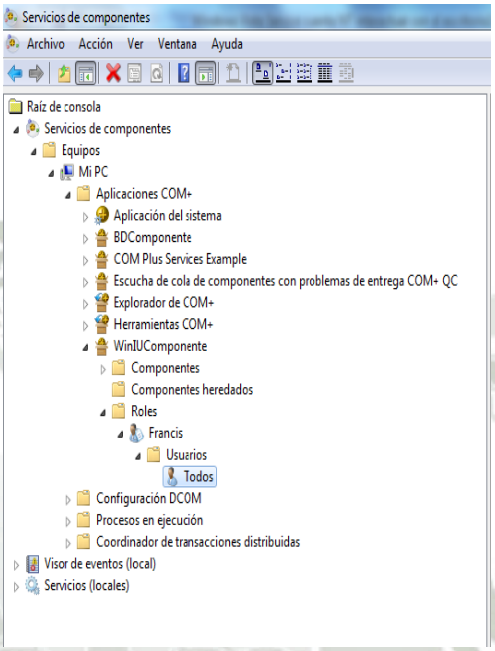


Figura 3.119 Funciones de los Servicios de componentes.

Fuente: Elaboración Propia.

Para hacer todas estas cosas, tenemos la Type Library Admin de la COM+ 1.0. Vamos a ir a nuestro proyecto de aplicación para Windows y haremos clic derecho en Referencias, seleccionaremos el panel de COM y buscáremos la aplicación Type Library Admin COM+ 1.0. Esto agregará COMAdmin a nuestras referencias. Antes de desplegar COMAdmin, hay que añadir un instalador para el proyecto actual.

Haremos clic derecho sobre el proyecto en el Explorador de soluciones y elegiremos la opción "Add Item/Nueva..." y seleccionaremos "La Clase del

instalador" de la lista de items disponibles. Cambiemos el nombre del nuevo archivo, por ejemplo, a InstaladorComponente.cs, Abramos el nuevo archivo y cambiemos el nombre a por ejemplo, la clase por defecto a InstaladorComponente que hereda de la clase Installer.

Tenemos que cambiar a la vista de diseño y en la pestaña de propiedades, haremos clic en Eventos. Desde aquí podemos ver todos los controladores de eventos disponibles para el instalador y que se pueden implementar en el mismo. Necesitamos por lo menos dos de ellos AfterInstall y BeforeUninstall. En la Figura 3.120 es el AfterInstall el cual tenemos que leer de la siguiente manera:

1) Primero usamos RegistrationHelper para que el componente sea registrado en una forma perfectamente previsible y normal.

2) El método InstallAssembly es nulo y no será una excepción si algo sale mal.

3) Luego se usa System.Activator para que una instancia de la COMAdmin se cree (véase variable CAC).

4) El COMAdmin devuelve la colección de todas las aplicaciones (ver variable CACC).

5) En la colección de aplicaciones, tenemos que encontrar nuestra propia aplicación (ver strAppID) la cual se acaba de registrar un par de pasos atrás, y también la colección de funciones que pertenecen a esta solicitud (véase variable caccRoles).

6) Seguidamente se busca del rol Francis que es un rol particular en la colección de Roles.

7) En el rol que se ha encontrado, es necesario identificar los UsersInRole (véase variable caccUsers) por debajo del Rol Francis (ver Figura 3.119)

8) Ahora el NT AUTHORITY\SYSTEM LOCAL o simplemente cuenta "SYSTEM" se añade a los Usuarios. Recuerde que esta cuenta se utiliza para ejecutar el servicio de NT. Los cambios se guardarán. Si va a ejecutar algunas pruebas de aplicación (por ejemplo, una consola) para esta aplicación COM+, se tendrá que añadir también su cuenta de usuario interactivo para el Rol.

9) Ahora el componente identificado por el GUID de clase-ID (ver variable cacoComponent) se encuentra en la colección de todos los componentes con el fin de permitir el acceso a los usuarios en función del rol principal (Francis)

10) Finalmente los cambios se guardan.

Nota: El Usuario Francis es el equivalente a un usuario promedio (con privilegios mínimos)

```
private void InstaladorComponente_AfterInstall(object sender,
InstallEventArgs e)
{
if (!EventLog.SourceExists(comPlusEventSource))
{
EventLog.CreateEventSource(comPlusEventSource, "FrancisLog");
}
string assembly = GetType().Assembly.Location;
string applicatioName = null;
string typelibName = null;
egistrationHelper regHelper = new RegistrationHelper();
try
{
regHelper.InstallAssembly(assembly,
 ref applicatioName,
 ref typelibName,

InstallationFlags.FindOrCreateTargetApplication);

//System.Diagnostics.Debugger.Break();
```

```
ICOMAdminCatalog cac =
(ICOMAdminCatalog)Activator.CreateInstance(Type.GetTypeFromProgID("C
OMAdmin.COMAdminCatalog"));
COMAdminCatalogCollection cacc =
(COMAdminCatalogCollection)cac.GetCollection("Applications");

COMAdminCatalogCollection caccRoles =
(COMAdminCatalogCollection)cacc.GetCollection("Roles", strAppID);
caccRoles.Populate();

// Encontrar Rol
COMAdminCatalogObject cacoRole = null;
bool roleFound = false;
foreach (COMAdminCatalogObject objectRole in caccRoles)
{
if (objectRole.Key.ToString().ToUpper() == strAppRole.ToUpper())
{
roleFound = true;
cacoRole = objectRole;
break;
}
}

// Role Encontrado?
if (roleFound == true && cacoRole != null)
{
// Asignamos cuentas al rol asignado
COMAdminCatalogCollection caccUsers =
(COMAdminCatalogCollection)caccRoles.GetCollection("UsersInRole",
cacoRole.Key);
caccUsers.Populate();
COMAdminCatalogObject cacoUser = null;
cacoUser = (COMAdminCatalogObject)caccUsers.Add();
cacoUser.set_Value("User", "SYSTEM");
caccUsers.SaveChanges();

// reconfiguramos el componente para para que los usuarios accedan
al rol
                    //
*****************************************************************
COMAdminCatalogCollection caccComponents =
(COMAdminCatalogCollection)cacc.GetCollection("Components",
strAppID);
caccComponents.Populate();

bool componenetFound = false;
foreach (COMAdminCatalogObject cacoComponent in caccComponents)
{
if (cacoComponent.Key.ToString().ToUpper() == strCLSID)
{
componenetFound = true;
break;
}
}
if (componenetFound == true)
{COMAdminCatalogCollection caccRolesForComponent =
(COMAdminCatalogCollection)caccComponents.GetCollection("RolesForCom
ponent", strCLSID);
COMAdminCatalogObject cacoRoleForComponent =
(COMAdminCatalogObject)caccRolesForComponent.Add();
```

```
cacoRoleForComponent.set_Value("Name", cacoRole.Name);
caccRolesForComponent.SaveChanges();}}
}
catch (Exception)
{throw ;
}
```

Figura 3.120 Instalación de los componentes con servicio.

Fuente: Elaboración Propia.

3.5.7 El servicio de Windows NT

La otra mitad de la solución es el servicio de NT. Tenemos que ir a la solución que en este momento sólo contiene un solo proyecto de aplicación de Windows, y agregar un nuevo proyecto, debemos usar en nuestra selección la plantilla de servicio de Windows. Ahora tenemos que cambiar el nombre de los archivos por ejemplo, a ServicioMonitoreo.cs que por defecto contiene los métodos OnStop () y OnStart () como controladores de eventos. Básicamente lo que tenemos que hacer es pensar sobre lo que el servicio está destinado a hacer. En este caso particular, el negocio principal del servicio es ser el vigilante de la aplicación y examinar periódicamente todos los procesos en ejecución además de contar el tiempo transcurrido de los procesos vigilados que se ejecutan en un contexto del usuario que es vigilado. La palabra "periódicamente" se subraya. Por esta razón, tenemos que empezar un hilo dedicado, lo que hará una pausa y se reanudara continuamente durante toda la vida útil del Servicio de NT. Al detener el servicio se reduce, por tanto, la tarea de detener un hilo periódicamente activado.

3.5.8 Estructura de Servicio NT

Hay dos posibilidades para cumplir con esta tarea:

El primero es el enfoque clásico a partir de un hilo dedicado que puede hacer el trabajo y después de terminar una carrera completa debe conciliar el sueño y luego debe despertar de acuerdo con hora predefinida que será la hora de dormir (Figura 3.121).

El segundo método utiliza un ejemplo de servidor basado en temporizador la System.Windows.Forms.Timer que periódicamente reinicia un proceso de trabajo (Figura 3.122).

Se ha decidido utilizar el enfoque basado en temporizador, el cual tiene la ventaja de ser más preciso en cuento al tiempo transcurrido. Si programamos el temporizador para disparar una vez por minuto, esto se hará exactamente cada 60 segundos de tiempo transcurrido. Utilizando el enfoque de los Workers-Thread en que la frecuencia de los despertares no son exactos, y se definen como la suma del tiempo en que duerme el hilo y la del tiempo de ejecución del hilo en el tiempo. La desventaja de este enfoque basado en temporizador es el peligro que estos hilos se superponen. Esto puede suceder si por alguna razón el temporizador en ejecución aún no está terminado, mientras que el próximo se está empezando. Esto tiene que evitarse con la creación de una variable booleana StillExecuting que se establecerá en true, mientras que el hilo se está ejecutando (Figura 3.122). Se encontró que este tipo de situaciones ocurren durante el arranque del ordenador y cuando el registro se hace por primera vez.

Este es el lugar donde nuestra interfaz de usuario basado en Servicios de Componentes se creará como una instancia junto con los métodos de interfaz, que se llamarán. En los ordenadores portátiles lentos a veces puede tomar unos minutos establecer la comunicación entre el servicio NT y el componente de interfaz de usuario.

```
...
protected override void OnStart(string[] args)
{
InitializeOnStart();

if ((workerThread == null) ||
((workerThread.ThreadState & (System.Threading.ThreadState.Unstarted
|
 System.Threading.ThreadState.Stopped)) != 0))
 {
serviceLog.WriteEntry("Inicializando el Hilo Trabajador.",
EventLogEntryType.Information, 10);
workerThread = new Thread(new ThreadStart(ServiceWorkerMethod));
goLoop = true;
workerThread.Start();
}
if (workerThread != null)
{
serviceLog.WriteEntry("Estado del Hilo = " +
workerThread.ThreadState.ToString(),
EventLogEntryType.Information, 11);
}
}

protected override void OnStop()
{
this.RequestAdditionalTime(5000);
if ((workerThread != null) && (workerThread.IsAlive))
{
 serviceLog.WriteEntry("Parando el Servicio del Hilo Trabajador.",
            EventLogEntryType.Information, 2);
goLoop = false;
Thread.Sleep(5000);
}
if (workerThread != null)
    {
        workerThread.Abort();
        judiLog.WriteEntry("OnStop Estado del Hilo = " +
            workerThread.ThreadState.ToString(),
            EventLogEntryType.Information, 1);
    }
    SaveSettings();
    this.ExitCode = 0;
}
public void ServiceWorkerMethod()
{
```

```
    serviceLog.WriteEntry("El servicio del hilo trabajador ha sido
satisfactoriamente inicializado.");
    try
    {
        do
        {
            Thread.Sleep(timeInterval);
                    ....
        }
        while (goLoop);
    }
    catch (ThreadAbortException)
    {

        serviceLog.WriteEntry("Hilo_Trabajador fue abortado mientras
lo deteniamos!",
            EventLogEntryType.Information, 3);
    }
    serviceLog.WriteEntry("Saliendo del Hilo Trabajador como el
        Stop-Event esta señalando.", EventLogEntryType.Information,
4);
}
```

Figura 3.121 Hilos-Trabajadores impulsados por el esqueleto de un Servicio NT.

Fuente: Elaboración Propia.

```
...
protected override void OnStart(string[] args)
{
    InitializeOnStart();

    serviceLog.WriteEntry("Inicializando el Hilo Trabajador.",
        EventLogEntryType.Information, 10);
    theWorkerTimer = new System.Timers.Timer();
    theWorkerTimer.Elapsed += new
ElapsedEventHandler(ServiceTimerTick);
    theWorkerTimer.Interval = timeInterval;
    theWorkerTimer.Enabled = true;
    GC.KeepAlive(theWorkerTimer);
    serviceLog.WriteEntry("Inicializando el Servicio del Hilo
Trabajador.",
        EventLogEntryType.Information, 12);
}

protected override void OnStop()
{
    this.RequestAdditionalTime(5000);
    serviceLog.WriteEntry("OnStop parando el Temporizador",
        EventLogEntryType.Information, 2);
    theWorkerTimer.Stop();
    theWorkerTimer.Dispose();
    serviceLog.WriteEntry("OnStop El Temporizador ha sido
Eliminado",
```

```
        EventLogEntryType.Information, 1);
    SaveSettings();
    this.ExitCode = 0;
}

public void ServiceTimerTick(Object sender, ElapsedEventArgs e)
{
    if (stillRunning == true)
    {
        serviceLog.WriteEntry("Una Instancia Previa del
ServiceTimerTick no ha finalizado aun!",
EventLogEntryType.Information, 113);
        return;
    }

    try
    {
            ....
    }
    catch (ThreadAbortException)
    {
        serviceLog.WriteEntry(""Hilo_Trabajador fue abortado
mientras lo deteniamos!",, EventLogEntryType.Information,3);
     }
     finally
     {
         stillRunning = false;
     }
}
```

Figura 3.122 Timer impulsado por el esqueleto de servicio NT.

Fuente: Elaboración Propia.

Obviamente, el siguiente paso mientras se procede con el proyecto de servicio de Windows es agregar una referencia de proyecto a la aplicación de Windows en términos de consumo de la interfaz declarada en nuestro ensamblado (WinIUComponente).

Además, vamos a crear una clase adicional UsuarioVigilado, que representa a un usuario vigilado como se muestra en la figura 3.123. Debemos tener en cuenta los métodos () Crear_Interfaz_Usuario() y Borrar_Interfaz_Usuario().

El constructor toma el nombre del usuario vigilado y el contador actual de minutos transcurridos. Para mayor brevedad la lista de todas las propiedades se descarta.

```
        /// <remarks></remarks>
        public class UsuarioVigilado
        {
         /// <summary>
         /// Inicializa una nueva instancia de la clase <see
         cref="UsuarioVigilado"/>.
         /// </summary>
         /// <param name="s_Nombre_Usuario">El Nombre de
Usuario.</param>
         /// <param name="i_Contador_Usuario">El Contador de
Usuario.</param>
         /// <remarks></remarks>
         public UsuarioVigilado(string s_Nombre_Usuario, int
i_Contador_Usuario)
            {
                nombre = s_Nombre_Usuario;
                contador = i_Contador_Usuario;
                s_guid = System.Guid.NewGuid().ToString();
                Usuario_IU = null;
                ejecutando = false;
                saliendo = false;
            }

            #region Propiedades

            /// <summary>
            /// Interfaz de Usuario del Componente
            /// </summary>
            private WinIUComponente.ISerMonComMensajes Usuario_IU;

            /// <summary>
            /// Trae el Objeto_COM.
            /// </summary>
            /// <remarks></remarks>
            public WinIUComponente.ISerMonComMensajes Objeto_Com
            {
                get { return Usuario_IU; }
            }

            /// <summary>
            /// Contador
            /// </summary>
            private int contador;
            /// <summary>
            /// Trae o setea el contador.
            /// </summary>
            /// <value>Variable Contador.</value>
            /// <remarks></remarks>
            public int Contador
            {
                get { return contador; }
                set { contador = value; }
```

```
            }

            /// <summary>
            /// Nombre del Usuario
            /// </summary>
            private string nombre;
            /// <summary>
            /// Trae el Nombre del Usuario.
            /// </summary>
            /// <remarks></remarks>
            public string Nombre
            {
                get { return nombre; }
            }

            /// <summary>
            /// GUID(s_guid)
            /// </summary>
            private string s_guid;
            /// <summary>
            /// Trae o setea la GUID.
            /// </summary>
            /// <value>El GUID de la aplicacion.</value>
            /// <remarks></remarks>
            public string Guid
            {
                get { return s_guid; }
                set { s_guid = value; }
            }

            /// <summary>
            /// Variable que nos dice si se esta ejecutando el Objeto
            /// </summary>
            private bool ejecutando;
            /// <summary>
            /// Trae o Setea el valor indicando cuando se esta
ejecutando la clase <see
            cref="UsuarioVigilado"/> .
            /// </summary>
            /// <value><c>Verdadero</c> si esta ejecutando; en otro
caso,
            <c>Falso</c>.</value>
            /// <remarks></remarks>
            public bool Ejecutando
            {
                get { return ejecutando; }
                set { ejecutando = value; }
            }

            /// <summary>
            /// Variable que nos dice si se esta Saliendo del Objeto
            /// </summary>
            private bool saliendo;
            /// <summary>
            /// Trae o Setea un valor indicando cuando se esta
saliendo de la clase <see
             cref="UsuarioVigilado"/>.
            /// </summary>
            /// <value><c>Verdadero</c> si esta saliendo; en otro
caso,
```

```
            <c>Falso</c>.</value>
            /// <remarks></remarks>
            public bool Saliendo
            {
                get { return saliendo; }
                set { saliendo = value; }
            }

            #endregion
            /// <summary>
            /// Funcion para Crear la Interfaz del Usuario.
            /// </summary>
            /// <returns></returns>
            /// <remarks></remarks>
            public WinIUComponente.ISerMonComMensajes
            Crear_Interfaz_Usuario()
            {
                //Asignamos el Componente a la Interfaz del Usuario
                Usuario_IU = new WinIUComponente.SerMonComDisplay();

                // Esto instanciara el formulario en estado escondido
                if (Usuario_IU != null)
                {
                    // Seteamos el GUID del Usuario
                    Usuario_IU.Setear_Guid(s_guid);
                    return Usuario_IU;
                }
                return null;
            }
            /// <summary>
            /// Funcion para Borrar La Interfaz del Usuario.
            /// </summary>
            /// <remarks></remarks>
            public void Borrar_Interfaz_Usuario()
            {
                ejecutando = false;
                try
                {
                    if (Usuario_IU != null)
                        Marshal.ReleaseComObject(Usuario_IU);
                }
                catch (Exception)
                {
 // solo procedemos, porque estamos saliendo de todos modos;}
 finally
{ //Seteamos la IU a nulo
  Usuario_IU = null;}}}
```

Figura 3.123 La clase UsuarioVigilado.

Fuente: Elaboración Propia.

1. El servicio tratara en primer lugar detectar quién está conectado y también si alguien está saliendo (cerrando sesión), identificando al usuario como un usuario vigilado. Para esto vamos a utilizar WMI (Windows Message

Interface) a través de las clases del espacio de nombres System.Management haciendo una consulta a Win32_ComputerSystem como vemos en la (Figura 3.124). El método bool fx_EsUsuario_Logueado devuelve true si el usuario se registra en la lista de usuarios vigilados y, además, devuelve el nombre de dominio y el nombre de usuario de la sesión actual.

2. Seguidamente el servicio verá si el usuario registrado identificado ya tiene una interfaz de usuario existente y además si esta instancia de interfaz sigue ejecutándose. Si esto no se aplica, se dará ira al paso N°3 o de lo contrario se irá al paso N°5.

3. El servicio hace la verificación de todos los casos de IU (Interfaz de Usuario) creados para todos los usuarios vigilados. Si el servicio detecta una interfaz de usuario ejecutándose pero que no pertenece a la sesión iniciada actual en la interacción con el usuario, el usuario actual es reconocido como un nuevo usuario. Todos los demás casos de interfaz de usuario ejecutándose tienen que ser cerrados. Esta estrategia soluciona la limitación de los servicios COM + en cuanto a cambios rápidos de usuario (Cierre de sesión o cambio de usuario interactivo). Pongamos que por ejemplo, U1 es el usuario actual y que este no cierre la sesión actual y ahora se registra un nuevo usuario U2, Nuestro COM+ fuera de proceso de sustitutos seguirá funcionando durante un par de minutos en el contexto del usuario original U1. Cualquier intento de crear una nueva instancia de WinUIComponente.SerMonComDisplay(), terminará con una nueva instancia en la sesión del usuario U1. Por lo tanto, el Servicio de

NT tiene que asegurarse de que, la interfaz de usuario U1 se cierra y la aplicación COM+ también se cierra con el fin de manejar el usuario U2.

4. Ahora el servicio puede empezar a crear una interfaz de usuario haciendo un nuevo llamado a UsuarioVigilado.Crear_Interfaz_Usuario() como vemos en la Figura 3.123.

5. Ahora en el servicio NT vamos a crear una lista de procesos en ejecución y filtrar los usuarios vigilados para acceder a los tokens de seguridad de los procesos en lista a ser vigilados. Esta es una operación privilegiada por lo que el servicio de NT tiene que funcionar como Sistema Local. El código se compone esencialmente de puras llamadas a métodos P/Invoke. Para ver un ejemplo podemos buscar el método privado fx_BuscarCuentaUsuario (Process proceso, out string s_Dominio, out string s_Usuario) en el archivo ServicioMonitoreo.cs, que devolverá true si la operación tiene éxito y devuelve también el nombre de usuario actual y el dominio que ya ha recibido el token de seguridad de los procesos vigilados.

6. Finalmente el Servicio de NT tiene que enviar una actualización de estado a la instancia de la interfaz de usuario. Este mensaje contendrá datos sobre el contador actual y el tiempo prohibido por períodos de tiempo. La actualización de estado se ejecuta a través de ISerMonCom Mensajes.Actualizar_Ping() como se vio en la Figura 3.113.

```
  private bool fx_EsUsuario_Logueado(out string s_Dominio_Usuario,
out string s_Nombre_Usuario)
        {
            //Variable boleana que retorna la funcion
            bool b_Flag_Retorno = false;

            //Variables para Usuario de Dominio y Nombre de Usuario
            s_Dominio_Usuario = "";
            s_Nombre_Usuario = "";

            //Consulta a un Objeto de administracion
            ObjectQuery oConsulta = new ObjectQuery("select * from
win32_computersystem");

            //Verificamos si la consulta es nula
            if (oConsulta == null)
                return b_Flag_Retorno;

            //Ejecuto la Consulta
            ManagementObjectSearcher oBuscador = new
ManagementObjectSearcher(oConsulta);
            if (oBuscador == null)
                return b_Flag_Retorno;

            //Traigo el Resultado en un Administrador de Colecciones
de Objetos
            ManagementObjectCollection oColeccionRetornada =
oBuscador.Get();
            if (oColeccionRetornada != null &&
oColeccionRetornada.Count >= 1)
            {
                //Recorremos los objetos retornados en
oColeccionRetornada
                foreach (ManagementObject oRetornado in
oColeccionRetornada)
                {
                    //Capturamos la cadena UserName para Cada
Usuario
                    string s_nombre =
oRetornado["UserName"].ToString();
                    //Si la cadena es mayor a cero en su
Longitud(Length)
                    if (s_nombre.Length > 0)
                    {

                        //Capturamos el indice para el caracter "\"
                        int i_indice = s_nombre.IndexOf('\\');
                        //Traemos el Dominio del Usuario
                        s_Dominio_Usuario = s_nombre.Substring(0,
i_indice).ToLower();
                        //Traemos el Nombre del Usuario
                        s_Nombre_Usuario =
s_nombre.Substring(i_indice + 1).ToLower();

                        //Buscamos cuando el usuario esta logueado y
esta en la lista de usuarios vigilados
                        foreach (string s_usuario in
As_UsuariosVigilados)
```

```
                        {
                            //Si al menos 1 usuario esta en la lista
de usuarios vigilados retornamos verdadero
                            if (s_usuario.ToLower() ==
s_Nombre_Usuario)
                            {
                                b_Flag_Retorno = true;
                                //return false; // !! SOLO PARA
TESTEO
                            }
                        }
                    }
                }
            }
            //Retornamos Verdadero si tenemos algun usuario a
vigilar y si no retornamos falso
            return b_Flag_Retorno;
        }
```

Figura 3.124 Uso de consulta de WMI para detectar usuarios conectados.

Fuente: Elaboración Propia.

Para completar este punto se debe añadir un archivo de configuración para el proyecto de servicio de Windows (Agregar nuevo elemento/Aplicación de archivos de configuración). Los contenidos que podemos configurar se definieron como vimos en la figura 3.110.

Además se tiene la necesidad de guardar los datos de la cuenta del usuario, mientras que el servicio se está cerrando. Estos datos se pueden leer en el momento de iniciar el servicio y se inicializa con los valores iniciales. El archivo de configuración creado anteriormente sólo es apropiado para datos de sólo lectura. Para preservar la dinámica de cambiar la configuración, se deben abrir las propiedades de servicio de Windows del proyecto (botón derecho del ratón en el proyecto) y ir al panel de configuración. Hacemos clic en la muestra "El proyecto no contiene un archivo de configuración por defecto..." se creará un vínculo con una nueva entrada Settings.settings debajo de las Propiedades del Proyecto (ver el explorador de soluciones) y también tendremos

abierto una configuración de la tabla de la vista. La tabla contiene, nada más que una sola línea en blanco. Debemos cambiar el nombre de la configuración por defecto en la columna Nombre con algo como (timestamp) “marca de tiempo" y elegimos el tipo de datos System.DateTime. Dejamos el valor vacío. Esto preservará la fecha exacta y la hora del evento al detener el servicio. A continuación añadimos una línea más de ajustes nombrado como StringColl y seleccionamos el tipo System.Collections.Specialized. StringCollection. Esta propiedad es con la intención de preservar los contadores (un punto de guardado por cada usuario vigilado).

3.5.9 Implementación de Servicio NT

Es posible implementar el servicio de NT a través de la utilidad **installutil.exe** que está disponible en el archivo de Paquete del NET Framework. Sin embargo, con el fin de hacerlo, hay que añadir un instalador en el proyecto de Servicio de Windows, como lo hicimos en el Servidor de Componentes (ServicedComponent) de Aplicaciones de Windows del proyecto. La diferencia es que en este caso, el instalador será consumido por el **Installutil.exe** y en el caso anterior era consumido por la utilidad **Regsvcs.exe**.

Para esto tenemos que hacer doble clic en el explorador de solución en el archivo ServicioMonitoreo.cs que activará la vista del diseñador. A continuación, vayamos en el diseñador de vista (que aparece en color gris y que contiene un texto que comienza así: ". Para agregar componentes a la clase....". Damos clic en el botón derecho del ratón en este campo y seleccionamos" Agregar instalador "Esto inmediatamente agregar un ProyectoInstalador.cs de nuestro proyecto y se activará esta clase en el modo de visualización de diseño. Lo que debemos tener en cuenta

acerca de estos casos es el hecho de que la clase ServicePocessInstaller representa el servicio de NT del proceso en sí mismo y tiene que ser único. La clase ServiceInstaller representa el servicio actualmente en desarrollo (no todo el proceso). El proceso de servicio de NT puede contener más de un servicio alojado en un solo proceso. Así que en teoría podría haber más de una instancia de clase ServiceInstaller en nuestro proyecto, que sin embargo no es el caso en este proyecto.

Echando un vistazo al constructor del ProjectInstaller (Figura 3.125). Debemos tener en cuenta que este se establece explícitamente, que el proceso tiene que funcionar en el contexto del sistema local. El proceso del servicio está configurado para iniciarse automáticamente. El nombre del servicio de visualización también se encuentra junto con la descripción que se verá en la consola de administración de servicios abiertos.

```
public ProyectoInstalador()
        {
            InitializeComponent();
            // Los servicios correran sobre la cuenta del Sistma.
            Proceso_Instalador.Account = ServiceAccount.LocalSystem;
            // Los servicios son iniciados Automaticamente
            Servicio_Instalador.StartType = ServiceStartMode.Automatic;
            Servicio_Instalador.ServiceName = "COM+ Francis - Servicio de
Monitoreo y Control";
            Servicio_Instalador.Description = "Este Servicio estara
intentando vigilar aplicaciones de acuerdo a los detalles de
configuracion";
        }
```

Figura 3.125 NT Service ProjectInstaller.

Fuente: Elaboración Propia.

Ahora vamos a manejar los eventos generados durante la instalación del servicio. En esencia se podría hacer de tres maneras diferentes. Por ejemplo usando el ServiceInstaller, o usando el ServiceProcessInstaller y también con ProjectInstaller. El ProjectInstaller hereda de la clase System.Configuration. Install.Installer y el mejor método para nuestro proyecto es utilizar esta clase. Podemos cambiar a la vista de

código del ProjectInstaller y reemplazar por lo menos los métodos OnAfterInstall() y OnBeforeUninstall(). El método protected override void OnBeforeUninstall (...) se lleva a cabo con una sola línea de código que llama al método (...) base.OnBeforeUninstall. Esto sería todo lo que tendríamos que definir para preparar nuestro instalador del proyecto.

Algo más difícil es la aplicación del método OnAfterInstall() (Figura 3.126). Lo primero que debemos tener en cuenta es la llamada del método base.OnAfterInstall(...) y poner al lado un mensaje escrito en la Solicitud de registro de eventos. Las cosas más complejas que siguen a esta última línea del código consiste en una serie de llamadas P/Invoke en el advapi32.dll. Lamentablemente, en el espacio de nombres System.Configuration.Install no hay soporte todavía para cambiar la configuración predeterminada del servicio de configuración. Acá nace la pregunta ¿Por qué necesitamos este cambio? Mientras que el inicio del servicio, especialmente durante el arranque del ordenador, no hay manera de determinar por defecto el orden de los servicios al empezar. Esto podría resultar en una mala experiencia ya que ServicioMonitoreo podría comenzar antes que algunos servicios de vital importancia (por ejemplo, antes del sistema COM+ o de los servicios RPC). Con el fin de prevenir malos resultados por el orden incorrecto al iniciar el servicio, tenemos que definir por ejemplo, la dependencia de servicio en el Sistema de eventos COM+ (ver EventSystem). El mismo EventSystem depende del servicio RPC. En cuanto a cómo hacer esta fijación de la dependencia de servicio, se deben realizar las siguientes operaciones:

1. Abrimos la base de datos del Administrador de control de servicios de Base de datos.
2. Bloqueamos el SCM (Service Control Manager) para el momento en que se puso la dependencia, se ha descrito anteriormente en el mismo.
3. Abrimos el servicio particular, utilizando el nombre del servicio.
4. Cambiamos la configuración del servicio (dejando todos los detalles de Configuración de otros cambios).
5. Desbloqueamos el SCM y dejamos de manejar el servicio de Base de datos.

```
protected override void OnAfterInstall(IDictionary
savedState)
        {
            base.OnAfterInstall(savedState);

// Adiciona pasos a ser hechos luego de que la instalacion
esta terminada.
 EventLog.WriteEntry(Codigo_Instalador_Proyecto,
"OnAfterInstall llamado");
            IntPtr databaseHandle = OpenSCManager(null, null,
                (uint)(SERVICE_ACCESS.SERVICE_QUERY_CONFIG |
                SERVICE_ACCESS.SERVICE_CHANGE_CONFIG |
                SERVICE_ACCESS.SERVICE_QUERY_STATUS |
                SERVICE_ACCESS.STANDARD_RIGHTS_REQUIRED |

SERVICE_ACCESS.SERVICE_ENUMERATE_DEPENDENTS));
            if (databaseHandle == IntPtr.Zero)
                throw new
System.Runtime.InteropServices.ExternalException("Abriendo
Servicio de Manejo de Error");
            IntPtr serviceDbLock =
LockServiceDatabase(databaseHandle);
            if (IntPtr.Zero == serviceDbLock)
            {
                int nError = Marshal.GetLastWin32Error();
                Win32Exception win32Exception = new
Win32Exception(nError);
                throw new
System.Runtime.InteropServices.ExternalException(
                    "Fallo al bloquear el Control de
Administracion de Servicios: " + win32Exception.Message);
            }
            IntPtr serviceHandle =
OpenService(databaseHandle, Servicio_Instalador.ServiceName,
```

```
                SERVICE_ACCESS.SERVICE_QUERY_CONFIG |
SERVICE_ACCESS.SERVICE_CHANGE_CONFIG);
            if (serviceHandle == IntPtr.Zero)
                throw new
System.Runtime.InteropServices.ExternalException("Abriendo
Servicio de Error");
            if (!ChangeServiceConfig(serviceHandle,
        SERVICE_NO_CHANGE, //SERVICE_WIN32_OWN_PROCESS |
SERVICE_INTERACTIVE_PROCESS,
        SERVICE_NO_CHANGE,
        SERVICE_NO_CHANGE,
        null,
        null,
        IntPtr.Zero,
        "EventSystem", // "SENS", Evento de Notificacion del
Sistema
        null,
        null,
        null))
       {
           int nError = Marshal.GetLastWin32Error();
           Win32Exception win32Exception = new
Win32Exception(nError);
           throw new
System.Runtime.InteropServices.ExternalException(
           "Podria no cambiar a proceso interactivo : " +
win32Exception.Message);
            }

            EventLog.WriteEntry(Codigo_Instalador_Proyecto,
                "OnAfterInstall satisfactorio se hicieron
cambios en la dependencia del servicio");

            UnlockServiceDatabase(serviceDbLock);
            CloseServiceHandle(serviceHandle);
        }
    }
```

Figura 3.126 La alteración de la configuración del servicio en la base de datos de SCM.

Fuente: Elaboración Propia.

En este punto se puede construir (compilar) la solución y desplegarla de forma manual tanto en el componente de COM+ (a través de la utilidad Regsvcs.exe) y también el servicio de NT (a través de installutil.exe).

3.5.10 Comunicación de la IU con el Servicio NT(Network Terminal)

Como se mencionó anteriormente, la capacidad de la interfaz de usuario para comunicarse con el servicio de NT, sin que se plantee, es especialmente útil por dos razones:

• Periódicamente se debe notificar al Servicio NT si la interfaz de usuario sigue ejecutándose. Como el componente de interfaz de usuario ha implementado un contador de tiempo, cada vez que se ha producido el evento Tick, la interfaz de usuario puede enviar de nuevo un mensaje "Hola, estoy ejecutándome y estoy funcionando bien".

• Mientras que el usuario cierra la sesión, la interfaz de usuario se ejecuta en el contexto del usuario conectado en forma interactiva, y luego se cerrará el sistema. El servicio de NT, por supuesto, cuenta con la instrumentación para revisar periódicamente a los usuarios conectados y también para reaccionar debidamente en caso de cualquier cambio, sin embargo, parece ser más adecuado tener un acoplamiento inmediato a los eventos de Salida de la interfaz de usuario.

• En tercer lugar (algo que aún no está implementado y se puede hacer a futuro), es utilizar la interfaz de usuario del usuario con sesión iniciada para que este pueda solicitar asignación adicional o tiempo extendido para utilizar los procesos prohibidos, mientras que el límite diario de este ya expirado. Esto podría ser manejado por ejemplo por un clic, en un botón que diga "Solicitar más Tiempo" en la parte superior de la interfaz de usuario y escribiríamos el nombre de usuario y la contraseña del usuario dedicado que tenga el papel de un supervisor.

3.5.11 Servicios de Hosting WCF en el Servicio de NT

Ahora volviendo al proyecto de servicio de Windows (ServicioMonitoreo). Los puertos a utilizar por los Servicios WCF se realizarán través de dos referencias de biblioteca adicionales, que son **System.ServiceModel.dll** y **System.Runtime. Serialization.dll**. A continuación debemos agregar al proyecto simplemente un "nuevo elemento/servicio WCF". Si añadimos un nuevo servicio de WCF o simplemente una clase, hay que terminar con una interfaz pública (por ejemplo, IWCFClaseServicio) atribuyendo el uso de la clase ServiceContractAttribute. La interfaz cuenta con dos firmas de los métodos declarados como bool Salir (string s_id) y bool Ejecutando (string s_id). Debemos tener en cuenta que los métodos están recibiendo un parámetro de cadena, que es la identidad de la interfaz de usuario (este se genera y envía por el Servicio de NT para la interfaz de usuario). Además tendrá una clase pública (por ejemplo, ClaseServicioWCF) que implementa esta interfaz (Figura 3.127). También debemos tener en cuenta, que las firmas de los métodos declarados se atribuyen a la clase OperationContractAttribute.

```
    [ServiceContract]
    public interface IClaseServicioWCF
    {
        /// <summary>
         /// Operación Saliendo que indica si se esta saliendo del
Servicio.
        /// </summary>
        /// <param name="s_id">String s_id.</param>
        /// <returns></returns>
        /// <remarks></remarks>
        [OperationContract]
        bool Saliendo(string s_id);

        /// <summary>
        /// Ejecutando el s_id especificado.
        /// </summary>
        /// <param name="s_id">String s_id.</param>
        /// <returns></returns>
        /// <remarks></remarks>
        [OperationContract]
```

```
        bool Ejecutando(string s_id);
    }

    /// <summary>
    /// Especificando el Comportamiento de control interno del Servicio
    /// </summary>
    /// <remarks></remarks>
    [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single,
                     ConcurrencyMode = ConcurrencyMode.Multiple)]
    public class ClaseServicioWCF : IClaseServicioWCF
    {
        /// <summary>
        /// Saliendo del s_id especifado.
        /// </summary>
        /// <param name="s_id">String s_id.</param>
        /// <returns></returns>
        /// <remarks></remarks>
        bool IClaseServicioWCF.Saliendo(string s_id)
        {
            bool Flag_Retorno = false;
            foreach (string s_clave in
ServicioMonitoreo.oUsuarios.Keys)
            {
                if (ServicioMonitoreo.oUsuarios[s_clave].Guid == s_id)
                {
                  ServicioMonitoreo.oUsuarios[s_clave].Saliendo = true;

ServicioMonitoreo.oUsuarios[s_clave].Borrar_Interfaz_Usuario();
                    Flag_Retorno = true;
                    break;
                }
            }
            return Flag_Retorno;
        }

        /// <summary>
        /// Ejecutando el s_id especificado.
        /// </summary>
        /// <param name="s_id">String s_id.</param>
        /// <returns></returns>
        /// <remarks></remarks>
        bool IClaseServicioWCF.Ejecutando(string s_id)
        {
            bool Flag_Retorno = false;
            foreach (string s_clave in
ServicioMonitoreo.oUsuarios.Keys)
            {
                if (ServicioMonitoreo.oUsuarios[s_clave].Guid == s_id)
                {
                ServicioMonitoreo.oUsuarios[s_clave].Ejecutando = true;
                Flag_Retorno = true;
                    break;
                }           }
            return Flag_Retorno;        }}
```

Figura 3.127 Detalles del contrato de servicio.

Fuente: Elaboración Propia.

Ahora debemos completar el Contrato que va a ser utilizado por el componente de interfaz de usuario, por lo cual tenemos que realizar dos pasos más, El primero es ampliar el archivo de configuración con las cosas relacionadas con el WCF (Figura 3.128) y el segundo es para abrir el host de servicio WCF al iniciar el servicio de NT (Figura 3.129) y el cierre de todo al detener el servicio de NT.

Debemos tener en cuenta en el archivo de configuración el atributo baseAddress dentro del elemento <baseAddress> de nuestro archivo de configuración XML. Esto indica al host de servicio WCF que va a escuchar a través de Http a las solicitudes de servicio de metadatos. Este puerto http no representa el canal de trabajo destinados a la interfaz de usuario para hablar con el Servicio de NT. Esto es más bien para que el servicio en tiempo de diseño atienda las solicitudes de WSDL (Contrato de Servicio WCF) y después de que el diseño se haya completado, recién será seguro desprenderse de ella.

Observemos el elemento <endpoint>, que define el ABC de la WCF: Direccionamiento, Enlace y Contrato. El enlace utiliza canalizaciones con nombre entre los procesos. El atributo de dirección también refleja el uso de canalizaciones con nombre, sin embargo, son perfectamente seguros para cambiar el nombre del valor de la dirección, dejando la primera parte "net.pipe: / /localhost/" sin cambios. El Valor exacto del atributo de contrato tiene que coincidir con la definición de la interfaz (Figura 3.127).

```
<system.serviceModel>
   <services>
          <service name="ServicioWindowsNT.ClaseServicioWCF"
behaviorConfiguration="NTBehavior">
    <host>
<baseAddresses>
```

```
  <add baseAddress="http://localhost:9002/NTServicioMonitoreo/"/>
                    </baseAddresses>
   </host>
<endpoint name="ServicioWindowsNT"

address="net.pipe://localhost/NTServicioMonitoreo/pipe"
          binding="netNamedPipeBinding"
          bindingConfiguration="NTPipeConfiguration"
          contract="ServicioWindowsNT.IClaseServicioWCF"/>
         </service>
    </services>
    <bindings>
       <netNamedPipeBinding>
          <binding name="NTPipeConfiguration"></binding>
       </netNamedPipeBinding>
    </bindings>
    <behaviors>
     <serviceBehaviors>
       <behavior name="NTBehavior">
         <serviceMetadata httpGetEnabled="true" />
       </behavior>
     </serviceBehaviors>
    </behaviors>
</system.serviceModel>
```

Figura 3.128 Configuración de WCF para hospedar el Servicio de NT.

Fuente: Elaboración Propia.

```
/// <summary>
/// Metodo que define el Inicio de un Servicio de Windows
/// </summary>
/// <param name="args">Datos pasados por el comando Iniciar.</param>
/// <remarks></remarks>
 protected override void OnStart(string[] args)
 {
     // Cargamos Configuracion
     fx_Cargar_Configuracion();

     // Iniciamos el Host del Servicio
     // ************************************************
    oHostServicio = new
ServiceHost(typeof(ServicioWindowsNT.ClaseServicioWCF));

//Escribo un mensaje para el Log de Eventos indicando que el Host del
Servicio se ha iniciado
    oLogEventos.WriteEntry("Creado el Servicio Host del tipo
ServicioWindowsNT.ClaseServicioWCF.", EventLogEntryType.Information,
13);

 //Si el Objeto Servicio Host no es nulo entonces abro el host del
servicio
    if (oHostServicio != null)
        oHostServicio.Open();
//Escribo un mensaje para el Log de Eventos indicando que el Host del
Servicio se ha iniciado y ya esta abierto
    oLogEventos.WriteEntry("Abierto el Servicio Host del tipo
ServicioWindowsNT.ClaseServicioWCF.", EventLogEntryType.Information,
14);
// Inicializamos un Timer(Reloj) separado que hara el trabajo actual.
    // ************************************************
```

```
    oLogEventos.WriteEntry("Iniciando el Hilo que trabajara con el
Servicio.", EventLogEntryType.Information, 10);

    //Inicializo el Timer para el Servicio
            oTemporizador = new System.Timers.Timer();
            //Defino la funcion que llamara el Timer segun intervalos
            oTemporizador.Elapsed += new
            ElapsedEventHandler(fx_Servicio_Temporizador_Tick);
            //Defino el intervalo con el que trabajara el Timer
            oTemporizador.Interval = i_IntervaloTiempo;
            //Inicializo el Timer(se empieza a ejecutar)
            oTemporizador.Enabled = true;
            //Me limpia la memoria de objetos del timer que ya no se
usen
            GC.KeepAlive(oTemporizador);

//Escribo un mensaje para el Log de Eventos indicando que el Hilo ha
sido iniciado
            oLogEventos.WriteEntry("Iniciado el Hilo con el que
trabajara el Servicio.", EventLogEntryType.Information, 12);
            if (args.Length > 0)
            {
              oVisor_Archivos.Path = args[0].ToString();
              oLogEventos.WriteEntry(
              String.Format("Iniciando el Visor de Archivos. " +
   "Tenemos {0} argumentos. El path a Visualizar sera '{1}'",
              args.Length, oVisor_Archivos.Path));  }
        }

        protected override void OnStop()
        {
         //Requiriendo tiempo adicional para Parar
         this.RequestAdditionalTime(10000);
         //Escribo un mensaje para el Log de Eventos indicando que el
Evento OnStop ha sido Recibido
         oLogEventos.WriteEntry("OnStop Recibido",
EventLogEntryType.Information, 20);
         //Paro el Servicio
         fx_Parar_Servicio();
         //Escribo un mensaje para el Log de Eventos indicando que el
Evento OnStop ha sido Finalizado
         oLogEventos.WriteEntry("OnStop Finalizado",
EventLogEntryType.Information, 21);
         //Ponemos el Codigo de Salida en 0 = Salida Exitosa
         this.ExitCode = 0;
}
```

Figura 3.129 Archivo de Configuración Extendido con Hospedaje de WCF OnStart() y OnStop().

Fuente: Elaboracióm propia.

3.5.12 Consumo del Servicio de WCF en la Interfaz de Usuario

Ahora, el servicio de NT está listo para ser iniciado. Para obtener los metadatos de WCF para el componente de interfaz de usuario, el servicio de NT tiene que estar levantado y corriendo además de estar escuchando en la configuración del

puerto Http 9002. Por esta razón, tenemos que instalar e iniciar el servicio de NT con la utilidad **installutil.exe** como se mencionó anteriormente.

Si se tuvo éxito al instalar e iniciar el servicio de NT, debemos abrir nuestro navegador predeterminado en este caso Google Chrome y escribir en la lista de direcciones, **"http://localhost:9002/ NTServicioMonitoreo/"**. Esto está de acuerdo con el archivo de configuración discutido en el capítulo anterior (Figura 3.128). Si se tiene la respuesta correcta (Figura 3.130), y el host de servicio WCF está ejecutándose podemos proceder. Debemos ir al lado del proyecto de aplicación de Windows. En el Explorador de soluciones derecho hacemos clic en referencias del proyecto y seleccionamos "Agregar referencia de servicio...". No hay que confundir esto con "Agregar referencia..." y también debemos dejar intacta la referencia a " Añadir Web..." alimentaremos el cuadro de diálogo que aparece con la dirección que es muy parecida a la del Explorador de Internet. Esto agregará una referencia de servicio por defecto con el nombre como **"localhost"** a su proyecto. También debemos tener en cuenta el hecho de que se creará un nuevo archivo de configuración que contiene detalles de la referencia de servicio de configuración.

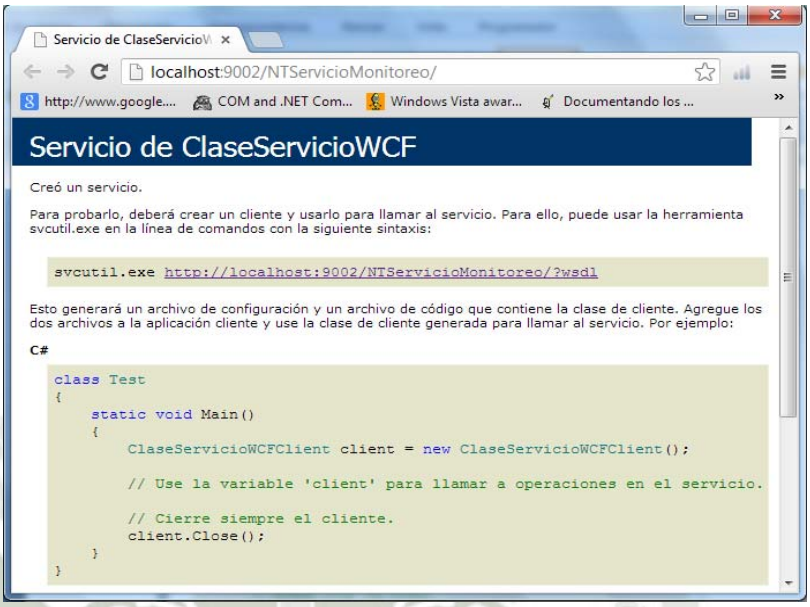


Figura 3.130 El consumo de WCF Metadatos en Internet Explorer.

Fuente: Elaboración Propia.

Veamos la Figura 3.131 en donde se muestra un fragmento de código después de descartar el archivo de configuración de la aplicación. Sólo tenemos que quitarla del proyecto. En el controlador de eventos FormLoad debemos tener un NetNamedPipeBinding y las clases de EndpointAddress que crearán una instancia. Lo que sigue es una instancia del WCFServiceClassClient con las instancias de enlace y la dirección como parámetros de entrada. El Uso de la configuración de la aplicación creada originalmente pondría simplemente una instancia de esta clase a través del constructor predeterminado sin parámetros de enlace y de dirección. El .NET Framework buscará la información necesaria directamente en el archivo de configuración.

```
/// <summary>
/// Cargado lo primero que hara es colocar la posicion de  las Ventanas
y inmediatamente
/// esperara los updates en orden(mensajes-ping) desde el Servidor NT.
Ademas
```

```
/// la comunicacion con el WCF es inicializada y finalmente el
temporizador es iniciado
/// </summary>
/// <param name="sender">La fuente del Evento.</param>
/// <param name="e">La <see
cref="System.Configuration.Install.InstallEventArgs"/> instancia
conteniendo la data del evento.</param>
/// <remarks></remarks>
private void ComponenteMonitoreoControl_Load(object sender, EventArgs
e)
       {
            SetearPosicionFormulario();
            this.notifyIcon1.Visible = true;
            this.Opacity = 0;

            b_Cerrar_Exacto = false;
//timer1.Interval = 60000; //Para que cambie cada minuto produccion
            timer1.Interval = 2000; //Cada Segundos para simulacion
            if (InstanciarServidorWCF() == true)
                timer1.Start();
        }
/// <summary>
/// El setup casteado dentro del codigo para reparar(basado en archivo
de configuracion removido)
/// </summary>
/// <returns></returns>
/// <remarks></remarks>
private bool InstanciarServidorWCF()
{
NetNamedPipeBinding Enlace_Binding = new NetNamedPipeBinding();
string s_netPipe = "net.pipe://localhost/NTServicioMonitoreo/pipe";
EndpointAddress Final_Endpoint = new EndpointAddress(s_netPipe);
Servidor = new ClaseServicioWCFClient(Enlace_Binding, Final_Endpoint);
 return (AbrirServidorWCF());
}
/// <summary>
/// Este metodo intentara abrir el Servidor WCF. El metodo es llamado
/// desde el manejador de evento Load(Cargar) del Formulario y tambien
despues de que
/// Communication-Exceptions es lanzado (a menudo recobrandose desde el
estado Suspend(Suspendido))
/// </summary>
/// <returns></returns>
/// <remarks></remarks>
private bool AbrirServidorWCF()
{
   bool b_flag_retorno = false;
   if (Servidor != null)
   {
    try
    {
      Servidor.Open();
      b_flag_retorno = true;
    }|
catch (TimeoutException ex)
{
    Servidor.Abort();
    Servidor = null;
    EventLog.WriteEntry(FuenteEventoComPlus, ex.Message,
EventLogEntryType.Error);
```

```
}
catch (CommunicationException ex)
{
    Servidor.Abort();
    Servidor = null;
EventLog.WriteEntry(FuenteEventoComPlus, ex.Message,
EventLogEntryType.Error);
 }
 }
   return b_flag_retorno;
 }
```

Figura 3.131 Hacer el saludo inicial con el servidor de WCF.

Fuente: Elaboración Propia.

Ahora solo debemos asegurarnos de lo siguiente:

• El temporizador llama a Servirdor.Ejecutando() con el fin de notificar al Servicio de NT que la interfaz de usuario sigue ejecutándose.

• Salir de la aplicación de la interfaz de usuario debe llamar al método Servidor.Salir() en el controlador de eventos FormClosing.

El controlador de eventos FormClosing es un poco complejo. Esto en términos de requisitos, no se debe permitir que el usuario cierre y salga de la aplicación. La aplicación sólo puede salir si el servicio de NT se lo pide al usuario al cerrar la sesión. El primer requisito es fácil de manejar, ya que el servidor tiene que llamar explícitamente al método Salir_Formulario() como se vio en la Figura 3.113, que puede establecer un indicador booleano como b_Cerrar_Exacto en true. Pero acá nace otra pregunta, ¿Cómo atrapar a un usuario que está cerrando su sesión?

Para hacer frente a los eventos del sistema, debemos reemplazar el método System.Windows.Form.Control.WndProc () que es una manera de conectar los eventos del Sistema de Windows. Después debemos consultar el mensaje WM_QUERYENDSESSION que ocurre exactamente cuando el usuario decide cerrar la sesión.

```
private const int WM_QUERYENDSESSION = 0x11;

protected override void WndProc(ref System.Windows.Forms.Message m)
{
  bool b_maquinaenSuspension = false;
  string reporte = "El S.O recibio un estado de poder no manejado";
  if (m.Msg == WM_QUERYENDSESSION)
  {
     b_Cerrar_Exacto = true;
     EventLog.WriteEntry(FuenteEventoComPlus, "WM_QUERYENDSESSION: este
     es un cierre de sesion, apagado, o reinicio");
  }

  //Si este es un WM_QUERYENDSESSION, el evento de cerrado debera ser
  lanzado en el evento base WndProc.
  base.WndProc(ref m);
}

private void EnviarMensajeSalidaaServidor()
{
    if (Servidor != null)
    {
     try
     {
      bool b = Servidor.Saliendo(s_guid);
      EventLog.WriteEntry(FuenteEventoComPlus, "Servidor.Exiting()
      called; result: " + b.ToString());
      Servidor.Close();
      EventLog.WriteEntry(FuenteEventoComPlus, "Servidor.Close()
      called");
      Servidor = null;
     }
     catch (CommunicationException ex)
     {
      EventLog.WriteEntry(FuenteEventoComPlus, ex.ToString(),
      EventLogEntryType.Error);
      Servidor.Abort();
      Servidor = null;
     }
 }
}

private void ComponenteMonitoreoControl_FormClosing(object sender,
FormClosingEventArgs e)
        { if (b_Cerrar_Exacto == true)
              EnviarMensajeSalidaaServidor();
            else
                this.Hide();
EventLog.WriteEntry(FuenteEventoComPlus, "Form1_FormClosing: !" +
b_Cerrar_Exacto.ToString());
          e.Cancel = !b_Cerrar_Exacto;}
```

Figura 3.132 Conexión de los eventos del sistema en Windows Forms y Cerrar el formulario.

Fuente: Elaboración Propia.

3.5.13 Despliegue de la Aplicación

Lo último que tenemos que hacer es terminar todos los ensamblados producidos en un paquete de instalación. Tenemos que ir a la solución y agregar un tercer proyecto que sea del tipo "Otros tipos de proyectos/Instalación e implementación/ Configuración del proyecto". Con esto se creará un paquete **MSI** que se implementará con privilegios elevados. Debemos utilizar el archivo **setup.exe** que se debe aplicar con la opción "Ejecutar como Administrador".

• Instale la aplicación utilizando el valor por defecto "sólo para mí"(Just for me) como opción en su cuenta de administrador (que es probablemente la cuenta del supervisor).

• Luego debemos ir a la carpeta %ProgramFiles% y en la ruta "...\ Francis\Componente\ServicioNT" debemos cambiar la configuración por defecto (ServicioMonitoreo.exe.config - ver Figura 3.110) de acuerdo a nuestras políticas de uso.

• Después de cambiar la configuración, iniciamos el servicio manualmente. A partir de acá el servicio se iniciará automáticamente en cada arranque.

• Cerramos la sesión. Debemos tener en cuenta que no hay absolutamente ninguna necesidad de dejar correr una cuenta administrativa mientras se navega en la Web - esto es aún más cierto cuando se ejecuta Windows.

CAPITULO IV: PLAN DE PRUEBAS Y EVALUACIÓN DE EXPERTOS

4.1 Plan de Pruebas

El plan de pruebas de software se elabora con el fin de especificar qué elementos o componentes se van a probar para que el grupo de trabajo pueda realizar el proceso de Validación y Verificación de los requerimientos Funcionales y no Funcionales de la herramienta. Además, a través del plan de pruebas se puede continuar con la trazabilidad de los requerimientos, con lo cual el grupo de trabajo, identifica el porcentaje de avance que se ha logrado hasta cierto momento.

Al desarrollar el plan de pruebas, se puede obtener información sobre los errores, defectos o fallas que tiene el prototipo, así se realizan las correcciones pertinentes, según el caso y se asegura la calidad del producto que se está entregando al cliente. El plan de pruebas se aplica sobre el producto, es decir, el código fuente. Las pruebas a implementar son básicas, esto incluye las pruebas unitarias y de integración que son vitales para la validación del producto.

4.1.1 Aproximación

En esta sección se exponen los tipos de pruebas a utilizar para la herramienta (el componente de seguridad).

4.1.1.1 Pruebas Unitarias

Pruebas por cada unidad, en este caso una unidad es equivalente a un requerimiento. El requerimiento es aprobado y aprobado si este cumple con lo que está escrito en la especificación de requerimientos.

NOMBRE	PRUEBAS UNITARIAS	IDENTIFICADOR	UT01
Actividades	Análisis de requerimientos del sistema		
Tiempo estimado	45 – 60 minutos por unidad		
Métodos o herramientas	Plataforma .NET		
Entregables	Lista de chequeo sobre el cumplimiento del requerimiento, ¿realiza lo que el requerimiento describe?		

Tabla 4.1 - Pruebas Unitarias – Fuente: Elaboración Propia.

4.1.1.2 Pruebas de Frontera

Pruebas frontera, son las que toman en cuenta valores límite, para verificar el comportamiento de la herramienta en esos casos.

NOMBRE	PRUEBAS DE FRONTERA	IDENTIFICADOR	LT01
Actividades	Se realizarán las distintas pruebas con los valores límites y mínimos que debe recibir el programa.		
Tiempo Estimado	15 minutos por prueba		
Métodos o Herramientas	Plataforma .NET		

Tabla 4.2 - Pruebas de Frontera – Fuente: Elaboración Propia.

En cuanto a los entregables de esta prueba se llenara la siguiente tabla:

NOMBRE		IDENTIFICADOR	T01
Valor máximo		**Valor mínimo**	
Resultado esperado			
Resultados obtenidos			
Estado		Funciona:	No funciona:
Comentarios			

Tabla 4.3 – Entregable de la Prueba - Fuente: Elaboración Propia.

4.1.1.3 Pruebas de Integración

Las pruebas de integración, como su nombre lo indica, son pruebas hechas a un conjunto de requerimientos.

NOMBRE	PRUEBAS DE INTEGRACIÓN	IDENTIFICADOR	IT01
Actividades	Validación de requerimientos		
Tiempo estimado	15 minutos por pruebas		
Métodos o herramientas	Plataforma .NET		
Entregables	Informe generado en donde se indica si se tiene un correcto funcionamiento o no.		

Tabla 4.4 - Pruebas de Integración - Fuente: Elaboración Propia.

ID	GRUPO DE REQUERIMIENTOS	RESULTADOS DE LA PRUEBA
	Grupo de requerimientos que están relacionados dentro del grafo de dependencias	Resultado de la prueba.

Tabla 4.5 – Resultados Prueba de Integración - Fuente: Elaboración Propia.

4.1.1.4 Pruebas de Sistema

Las pruebas de sistema son pruebas realizadas a la herramienta como un conjunto, que casos de uso cumple a cabalidad, con rutas de éxito y fallo, que han sido definidas en el documento de Casos de Uso.

NOMBRE	Pruebas sistemas	IDENTIFICADOR	ST01
Actividades	Se realizaran pruebas funcionales y No funcionales		
Tiempo estimado	15 minutos por prueba		
Métodos o herramientas	Plataforma .NET		
Entregables	Informe generado por el responsable de esta prueba el cual informara si se tiene un correcto funcionamiento o no.		

Tabla 4.6 – Pruebas de Sistema – Fuente: Elaboración Propia.

ID	CASO DE USO	RESULTADOS DE LA PRUEBA
	Grupo de requerimientos que están relacionados dentro del grafo de dependencias	Resultado de la prueba.

Tabla 4.7 – Resultados Pruebas de Sistema – Fuente: Elaboración Propia.

4.1.2 Proceso de Pruebas

En esta sección se presentan los casos de pruebas generales para usarlos con la herramienta. Cada cuadro está asociado a un Caso de Uso, desde ahí se desglosa en los diferentes módulos involucrados para el funcionamiento y se evalúa el resultado obtenido. En las siguientes tablas, se muestran los casos de pruebas a realizar:

NOMBRE	**Administrar Servicios**	PRUEBAS	**P1**
PROPÓSITO	Verificar si al administrar cualquier servicio, se actualiza el estado de los mismos de acuerdo a la operación solicitada.		
PRERREQUISITOS	• Haber solicitado administrar un servicio de Windows. • Tener requerimientos en la administración de un servicio.		
UBICACIÓN	Pantalla Principal de Administración de Servicios		
ENTRADA	• Búsqueda de Servicios Disponibles.		
ORÁCULO	El historial de cambios ha sido actualizado		
PASOS	1. Buscar en la Pantalla Principal el servicio a administrar. 2. Clic en “operación” a realizar sobre el servicio. 3. Clic en “Aceptar”.		
Módulos Asociados	• Administración de Servicios		

TABLA 4.1: Caso de Prueba 1.

Fuente: Elaboración Propia.

NOMBRE	**Administración de Usuarios**	PRUEBAS	**P2**
PROPÓSITO	Verificar que si es posible crear, eliminar y modificar relaciones entre un par de requerimientos en cuanto a la administración de usuarios.		
PRERREQUISITOS	• Existen requerimientos en administración de usuarios.		
UBICACIÓN	Base de datos PostgreSql, pantalla de Administración de Usuarios, pantalla de Mantenimiento de usuarios (la de creación y modificación y inactivación.		
ENTRADA	• Seleccionar un requerimiento (Creación, Modificación o Inactivación) • Selección de la acción que desea hacer o Crear Usuario o Modificar Usuario o Inactivar Usuario		
ORÁCULO	• (PARA CREAR) La base de datos se encuentra actualizada con el nuevo usuario. • (PARA MODIFICAR) La base de datos se encuentra actualizada con los cambios del usuario. • (PARA INACTIVAR) La base de datos se encuentra actualizada con la inactivación de usuarios.		

PASOS	**PARA CREAR** 1. Visitar la pantalla de Mantenimiento de Usuarios. 2. Seleccionar una operación (Se selecciono Nuevo). 3. Clic en Botón “Nuevo”. 4. Registrar los datos solicitados para el nuevo usuario. 5. Clic en el Botón Guardar. **PARA MODIFICAR** 1. Visitar la pantalla de Mantenimiento de Usuarios. 2. Seleccionar una operación (Se selecciono Modificar). 3. Clic en Botón “Modificar”. 4. Modificar los datos solicitados para el usuario. 5. Clic en el Botón Actualizar. **PARA INACTIVAR** 1. Visitar la pantalla de Mantenimiento de Usuarios. 2. Seleccionar una operación (Se selecciono Modificar Estado). 3. Clic en combo de Estado (Se selecciona el estado A o I). 4. Clic en el Botón Actualizar.
Módulos Asociados	• Mantenimiento de Usuarios (BDComponente,BDCliente)

TABLA 4.2: Caso de Prueba 2.

Fuente: Elaboración Propia.

NOMBRE	**PRUEBAS FRONTERA**	PRUEBAS	**P3**
PROPÓSITO	Verificar que el comportamiento del sistema cuando se ingresan valores extremos.		
PRERREQUISITOS	• Proyecto creado • Atributos seleccionados.		
UBICACIÓN	Base de datos Postgresql		
ENTRADA	Valores de atributos del requerimiento		
ORÁCULO	Mensajes sobre el éxito o no del ingreso de los valores		
PASOS	1. En cada campo, donde el usuario tenga la oportunidad de ingresar información, verificar sus límites. a. Si son Alfanuméricos: Verificar el tamaño máximo. b. Si son Numéricos: Verificar los límites min y máximo. 2. Registrarlos en el archivo de “Reporte de Pruebas”.		
Módulos Asociados	TODOS		

TABLA 4.3: Caso de Prueba 2.

Fuente: Elaboración Propia.

NOMBRE	**METODOS GENERALES**	PRUEBAS	**P4**
PROPÓSITO	Verificar que el comportamiento de cada uno de los métodos generales de la lógica del negocio.		
PRERREQUISITOS	Ninguno		
UBICACIÓN	Base de datos PostgreSql		
ENTRADA	Información requerida en cada uno de los métodos		
ORÁCULO	Verificación de los datos en la base de datos o en las excepciones capturadas		
PASOS	1. Construir una clase para probar cada uno de los métodos 2. Registrarlos en el archivo de "Reporte de Pruebas".		
Módulos Asociados	TODOS		

TABLA 4.4: Metodos Generales.

Fuente: Elaboración Propia.

4.2 Marco de la evaluación de expertos

"La principal ventaja de la evaluación por criterios es su bajo coste, en realidad este tipo de evaluación puede tener el coste que se desee. Un número mínimo de tres evaluadores permite realizar una evaluación por criterios. Los costes son por tanto mucho menores que cualquier otro método de evaluación".

Con el propósito de evaluar la propuesta de Componente de Seguridad se elaboró un video demostrativo en el que se muestra el modo de funcionamiento de la propuesta. Finalmente se escogió a cinco expertos a quienes se les aplicó un cuestionario de siete preguntas.

4.3 Perfil de los expertos

- Estudios superiores en disciplinas relacionadas al análisis, diseño y desarrollo de sistemas informáticos.
- Experiencia superior a cuatro años en el análisis, diseño e implementación de componentes de software informáticos.
- Conocimientos en tecnologías orientadas a los servicios web.

4.4 Resultados de la evaluación a expertos

Pregunta Nº 1:

Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

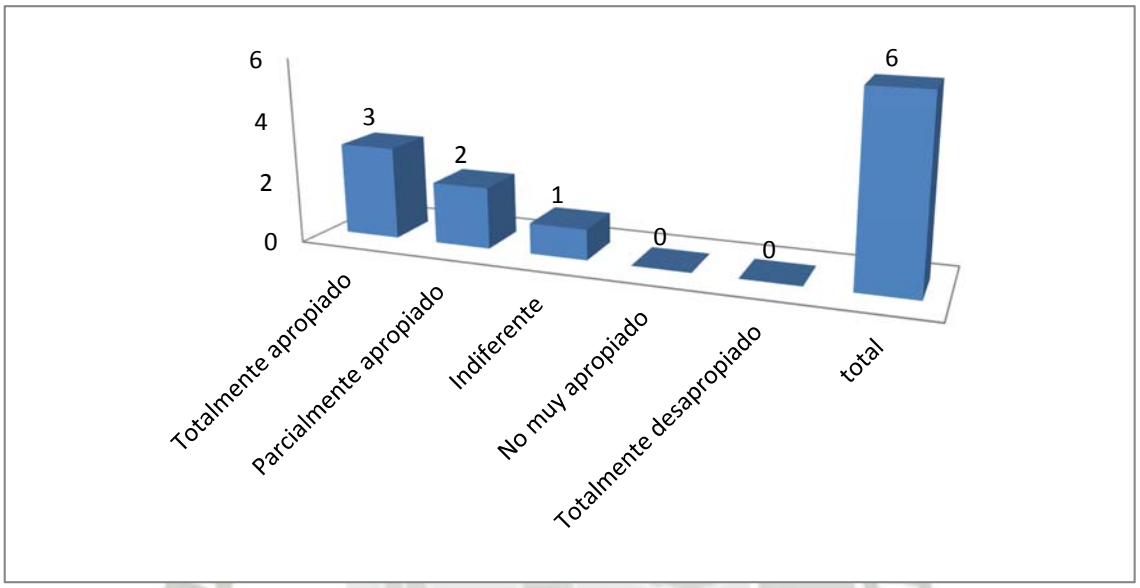


Gráfico Nº 4.1. Resultado Pregunta Nº 1.

Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

Fuente (Elaboración Propia).

A las 5 de 6 participantes les pareció buena la idea de contar con un componente en un tiempo de implementación adecuado. Al centralizar con el control del equipo se corre el riesgo de que una tengamos pendientes dentro de la realización de manejo realizado por el cierre de servicios dentro del equipo. La ventaja de tener el control del equipo informático se centraliza en tener un equipo libre de realizar tareas bajo la supervisión del componente.

Pregunta Nº 2: ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

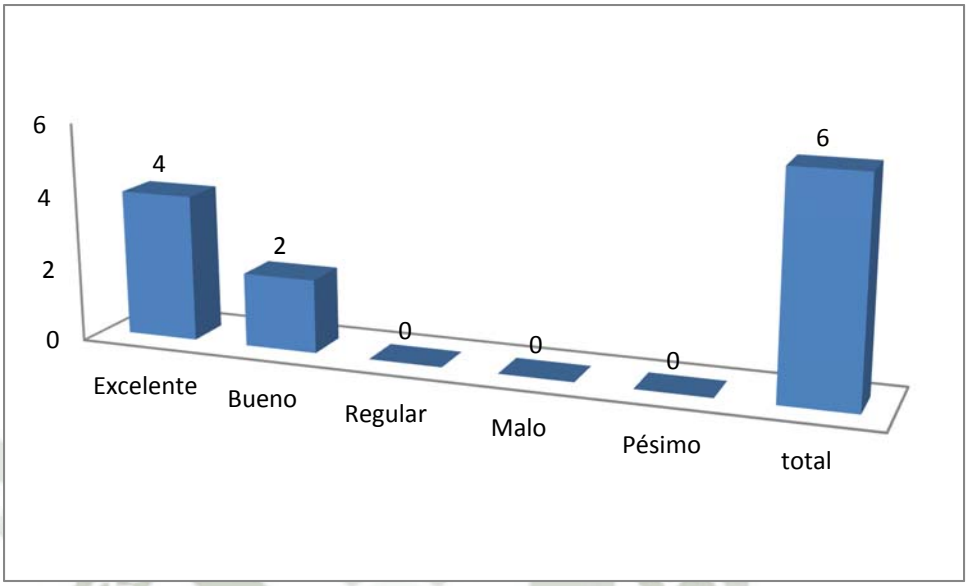


Gráfico Nº 4.2. Resultado Pregunta Nº 2.

¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

Fuente (Elaboración Propia).

Cuatro de seis participantes opinaron que es excelente el funcionamiento del Componente bajo la plataforma WCF en un tiempo de respuesta adecuado, el uso de la plataforma WCF se da para el desarrollo de aplicaciones para definir un la comunicación mediante mensajes, esto es conveniente dado que el componente de seguridad informática es planteado bajo la modalidad de mantener al usuario conectado con la seguridad de su equipo informático, lo que hace posible su comprensión por parte de los usuarios; no depende de la plataforma que se utilice y pueden crearse con un plataformas distintas, además con plataforma WCF se logra que el usuario defina el comportamiento del *Componente*.

Pregunta Nº 3: ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

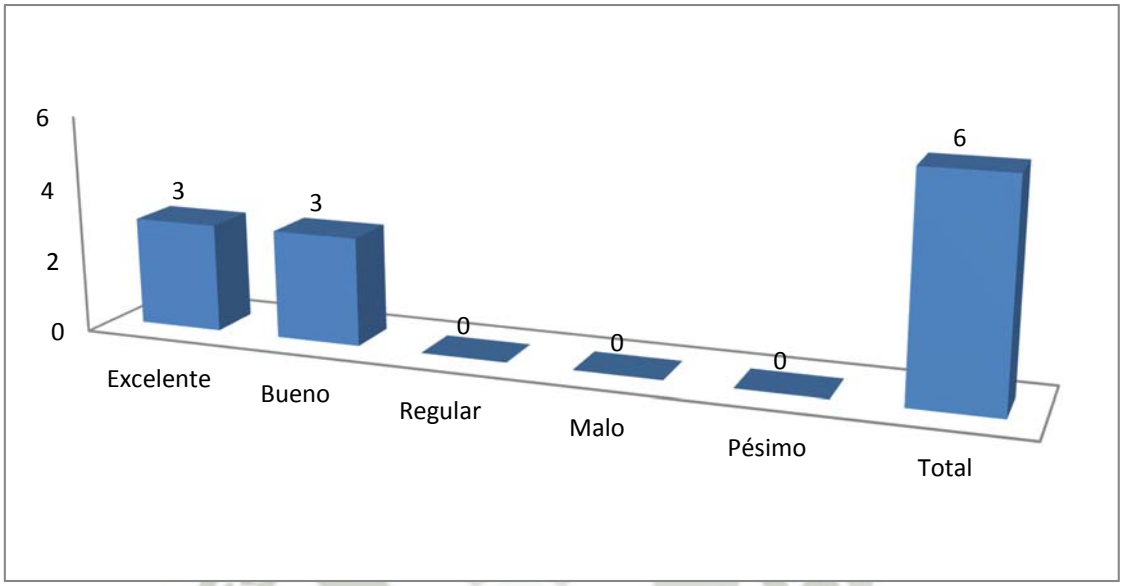


Gráfico Nº 4.3. Resultado Pregunta Nº 3.

¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?.

Fuente (Elaboración Propia).

Tres participantes calificaron como excelente y tres como bueno los dos modos de funcionamiento del Componente, lo que indica que se pueden tener utilidad para realizar mejoras, tal como se indican en las recomendaciones y para la finalización del componente bajo plataforma WCF.

Pregunta Nº 4: ¿Está usted de acuerdo con la siguiente afirmación: "La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows"?

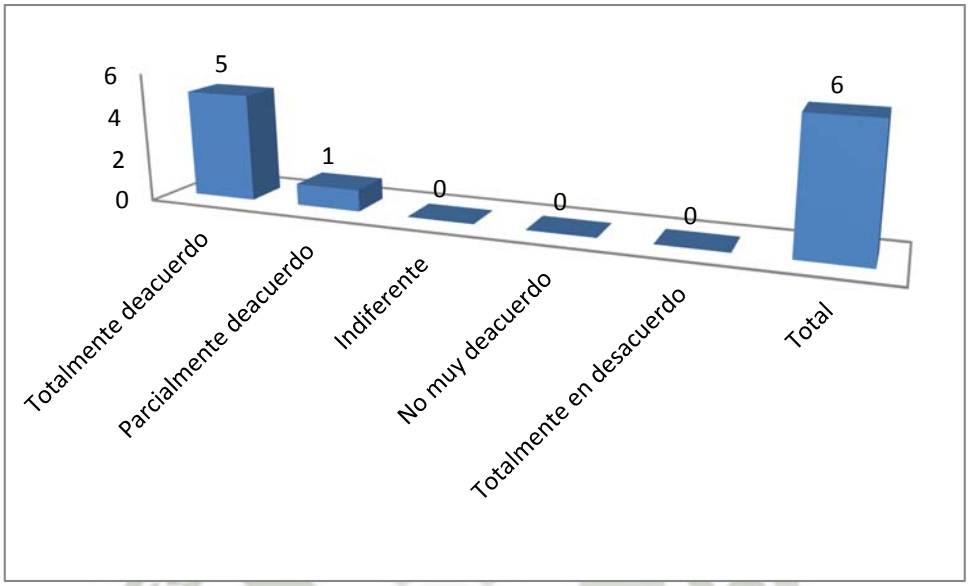


Gráfico Nº 4.4. Resultado Pregunta Nº 4.

¿Está usted de acuerdo con la siguiente afirmación: "La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows"?

Fuente (Elaboración Propia).

Cinco de 6 participantes coinciden en que el *Componente* logra integrar sistemas operativos Windows, esto dado que emplea sistemas operativos Windows bajo la plataforma WCF utilizando framework 3.0 y estos están basados en un estándar de seguridad que no es dependiente de un sistema operativo. La limitante es el sistema operativo sobre la que se ejecuta el *Componente*, solo puede ser en un ambiente Windows junto con la instalación del PostgreSql 8.4 como requisito para la ejecución del componente.

Pregunta Nº 5:

¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

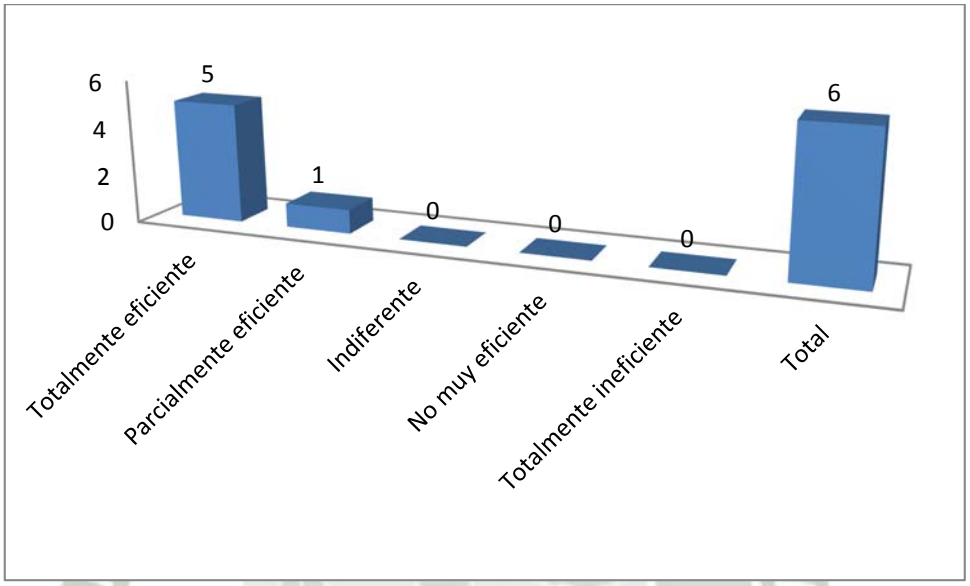


Gráfico Nº 4.5. Resultado Pregunta Nº 5.

¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?.

Fuente (Elaboración Propia).

Cinco de seis participantes consideran como totalmente eficiente el uso de BD PostgreSql 8.4 y XML elaboración de acciones en cambios en la configuración del Componente, esto porque hay actualizaciones en la base de datos PostgreSql y el manejo de ello se debe a la configuración de la versión de esta. Hay otras tecnologías que pueden ser más eficientes como. Pero corren el riesgo de tener problemas frente a cortafuegos y ser dependientes de la plataforma distinta dificultando su acción.

Pregunta Nº 6:

¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

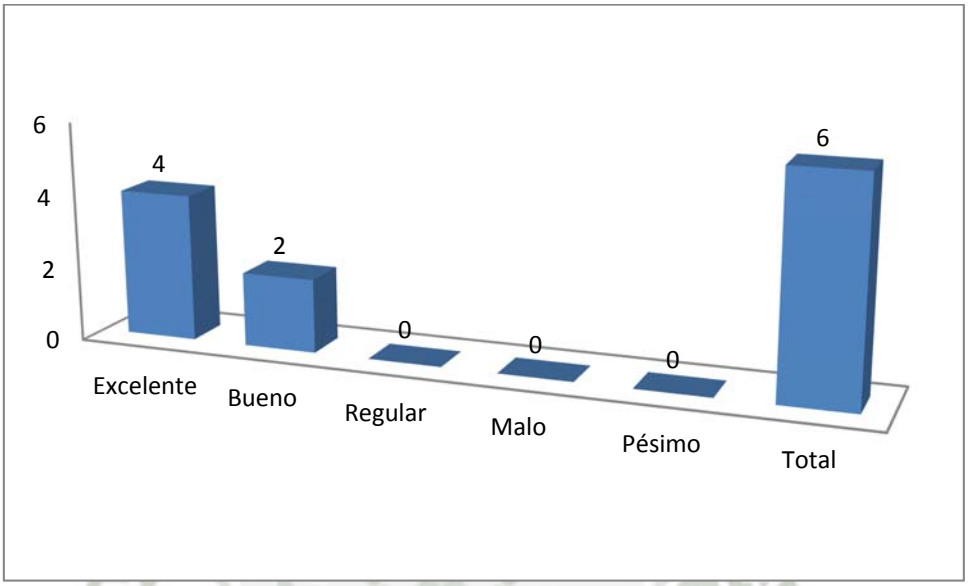


Gráfico Nº 4.6. Resultado Pregunta Nº 6.

¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

Fuente: (Elaboración Propia).

A cuatro de participantes les parece excelente y 2 participantes Bueno la sencillez de esta tecnología, ya que cualquier usuario promedio interactúa con ellas frecuentemente, por lo que la interacción con el Componente durante la ejecución de un proceso de bloqueo seria transparente.

CONCLUSIONES

- PRIMERO: Se creó un componente de software de seguridad que nos permite la reutilización del mismo basándonos en modelos comprobados.
- SEGUNDO: Se utilizó la arquitectura de servicios WCF (Windows Communication Foundation) para la comunicación entre procesos del componente de software propuesto.
- TERCERO: Se hizo uso de herramientas para desarrollo basadas en Software Libre GPL o GNU para el análisis y diseño del prototipo como son Start UML, GhostDoc, SandCastle GUI entre otras.
- CUARTO: Se desarrollaron clases y librerías reutilizables que servirán como punto de partida para el desarrollo de otros tipos de componentes pudiendo utilizar el marco de trabajo como base para otros proyectos similares.
- QUINTO: Se demostró la utilidad y facilidad de uso del componente en diferentes plataformas de sistemas operativos Windows.

RECOMENDACIONES

- PRIMERO: Se recomienda ver el DSBC como una metodología que no se puede tomar más como una metodología de desarrollo nueva, por el contrario se trata de una idea que está evolucionando de sus inicios para tomar más fuerza en el mundo de la ingeniería de software, existen ya grandes ideas, conceptos y un mundo de analistas, diseñadores que desean hacerla realidad y lo están haciendo y actualmente los proveedores de tecnologías que utilizan y proporcionan soporte al DSBC son pocas, sin embargo ya existe un mundo de desarrollo a partir de estas, ya que es definitivo que la evolución de la POO es el DSBC, y son estos mismos proveedores aquellos que definitivamente ya maduraron el paradigma de POO. Su siguiente etapa en la evolución del desarrollo de software es DSBC y sus tecnologías proveen de todos los servicios y funcionalidades para que esto ocurra.
- SEGUNDO: Con base a lo expuesto surge la necesidad no sólo de encontrar un lenguaje común referente al tema de DSBC entre desarrolladores y los usuarios finales sino también la necesidad de buscar métodos que permitan el acercamiento y faciliten la utilización de los componentes de software a cualquier tipo de usuario que desee integrarlos a los desarrollos de software.
- TERCERO: Se sugiere seguir mejorando el componente desarrollado en la presente tesis, implementando más opciones y servicios de red, que permitan hacer un monitoreo más exhaustivo, en cuanto a la información compartida por un dominio en especifico.
- CUARTO: Se deben considerar los beneficios derivados de la reutilización de software ya que los componentes de software reutilizables constituyen las unidades

fundamentales para el desarrollo de nuevas aplicaciones, debemos enfocarnos a reutilizar componentes para agilizar el tiempo de desarrollo en todos los proyectos futuros.

- QUINTO: Analizar y comparar los nuevos tipos de arquitectura de software que se plantean en esta propuesta para el DSBC viendo que mas podemos mejorar en relación a las ya existentes las cuales también hacen uso de los estándares más utilizados en la actualidad.

BIBLIOGRAFIA

[AND02] Anneliese Andrews, Sudipto Ghosh, “A model for Understandig Software Componets”, p.5. 2002.

[ARE01] C. Arévalo, J. Colmenares, N. Queipo, N. Arapé y J. Villalobos. A CORBA and Web technology based framework for the analysis and optimal design of complex systems in the oil industry. En 3rd Internacional Conference on Enterprise Information Systems (ICEIS 2001), Julio 2001.

[ATS11] Atsistemas, Soluciones, SOA (Service Oriented Architecture) [en línea], 2011 <http://www.atsistemas.com/soa.aspx> [consulta: 22 Agosto 2011].

[BACH00] F. Bachmann, L. Bass, Ch. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord y K. Wallnau. Volume II: Technical concepts of component-based software engineering, 2nd edition. Technical report, Software Engineering Institute, Carnegie Mellon University, Julio 2000.

[BEN02] PerOlof Bengtsson, Nico Lassing, Jan Bosch and Hans van Vliet,“Analyzing Software Architectures for Modifiability” . 2002.

[BER02] Bertoa Manuel F., Troya Jose M., Vallecillo Antonio. “Aspectos de Calidad en el Desarrollo de Software Basado en Componentes”. p 5.2002.

[BEU99] A. Beugnard, J-M Jézéquel y N. Plouzeau. Making components contract aware. IEEE Computer, 32(7):38-45, Julio 1999.

[BRO98] A. W. Brown and K. C.Wallnau. The current state of CBSE. IEEE Software, 15(5):37-46, Septiembre 1998.

[BUD94] T. Budd. Introducción a la programación orientada a objetos. Addison-Wesley Iberoamericana. 1994.

[CDBI99] CDBI Forum. Component based development: Using componentised software. http://www.cdbiforum.com, Mayo 1999.

[CHAP96] D. Chappell. Understanding ActiveX and OLE. Microsoft Press, 1ra edición, Enero 1996.

[DES98] Desmond Francis D'Souza, Alan Cameron Wills. Objects components and Frameworks "The Catalysis Aproach" p38.-386.1998.

[DIG99] T. Digre. Business object component architecture. IEEE Software, 15(5), 1998.

[DUK95] R. Duke, G. Rose y G. Smith. Object-Z: A specification language advocated for the description of standards. Computer Standards and Interfaces, 17:511-533, Septiembre 1995.

[FER02] Peter Ferrke y Peter Loos. "Specification of Business Components . Objects,Components, Architectures, Services and Aplications for a Networked World." , p 2-8. 2002.

[FRAN97] X. Franch. Systematic formulation of non-functional characteristics of software. In 3rd International Conference on Requirements Engineering (ICRE), pages 174-181, Colorado Springs (USA).

[FUE03-A] Lidia Fuentes, José M. Troya, Antonio Vallecillo. "Desarrollo de Software Basado en Componentes", p 2-3. 2003.

[FUE03-B] Lidia Fuentes, José M. Troya, Antonio Vallecillo "Desarrollo de Software Basado en Componentes", p 3-4. 2003.

[GAM95] E. Gamma, R. Helm, R. Johnson y J. Vlissides. Design patterns. Addison-Wesley Pub Co, Enero 1995.

[GAR98] Gartner Group. 1998.

[GON04] Rafael González. "Ontología de software para PyMEs", p 3 2004.

[GOL89] A. Goldberg and D. Robson. Smalltalk 80 : The language. Addison-Wesley Pub Co, Enero 1989.

[HER00] P. Herzum and O. Sims. Business component factory : A comprehensive overview of component based development for the enterprise. John Wiley & Sons, 2000.

[JOY98] L. Joyanes Aguilar. Programación orientada a objetos. McGraw-Hill Interamericana, Diciembre 1998.

[KOZ98] Wojtek Kozaczynski, SSA, Alan W. Brown, Kurt C. Wallnau "The Current State of CBSE".1998.

[KRU98] Philippe Krutchen. "Rational Software" 1998.

[LAN97] K. Lano, J. Bicarregui, J. Luiz Fiadeiro y T. Maibaum. Composition of reactive system components. En 1st Workshop on Component-Based Systems, Zurich, Switzerland, 1997.

[LAR05] LARMAN, GRAING Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development, Upper Saddle River, NJ : Prentice Hall, 2005

[RUM99] RUMBAUGH, JAMES The unified modeling language reference manual Reading, MA: Addison-Wesley, 1999

[CON03] CONALLEN, JIM Building Web applications with UML, Boston : Addison-Wesley, 2003

[KEN01] KENDALL, SCOTT UML Explained, Boston:Addison-Wesley, 2001

[NAN95] D. W. Nance. Pascal: Understanding programming and problem solving. West Information Pub Group, 4ta edición, Enero 1995.

[MAR02] F. Marinescu. EJB design patterns: Advanced patterns, processes and idioms. John Wiley & Sons, Febrero 2002.

[MAR96] Martin, R.H. y Odell. "Object oriented methodology: pragmatic considerations". Prentice Hall, Upper Saddle River.1996.

[MAT93] Matsuoka y Yonezawa. "Analysis of Inheritance Anomaly in Object-Oriented Concurrent Programming Languages", p 107-150.1993.

[MEY91] B. Meyer. Eiffel: The language. Prentice Hall PTR, Octubre, 1991.

[MIN02] Kim Min Sun, Kim Soo Dong. " Component metrics to measure component quality". P.421-422. 2002.

[MON00] J. Montilva, K. Hazam y M. Gharawi. The Watch Model for development business software in small and midsize organization. In IV World Multiconference on Systemics, Cybernetics and Informatics (SCI'2000), Orlando, Florida (USA), Julio 2000.

[MON03] J. Montilva and J. Barrios. A Component-Based Method for Developing Web Applications. 5th International Conference on Enterprise Information Systems (ICEIS'2003), Angers, France, 2003.

[MON03] Jonás A. Montilva C., Nelson Arapé y Juan Andrés Colmenares. "Desarrollo Basado en Componentes", p 3-4.2003.

[MOR01] J. Morris, G. Lee, K. Parker, G.A. Bundell, y C.P. Lam. Software component certification. IEEE Computer, 34(9), Septiembre, 2001.

[OMG02] Object Management Group, Inc. Common object request bróker architecture: Core specification. http://www.omg.org/docs/for mal/02-12-06.pdf, Diciembre 2002.

[OMG02] Object Management Group Inc. Corba components. http://www.omg.org/cgibin/doc?formal/ 02-06-65, Junio 2002.

[OMG04] OMG "About the Object Management Group".2004.

[PRES97] Roger Presuman. "Ingeniería de Software un enfoque practico", p 474-481. 1997.

[RIC02] Jeffrey Richter. "Microsoft .NET Framework Delivers the Platform for an Integrated,Service" 2002.

[SAM97] J. Sametinger. Software engineering with reusable components. Springer Verlag, Agosto 1997.

[SMOR94] D. Rev. Smorlarski, Research & Education Association y Dennis Chester Smolarski. The essentials of FORTRAN (essentials). Research & Education Assn, Mayo 1994.

[SOD99] J. Sodhi, y P. Sodhi. Software reuse: Domain analysis and design process. McGraw-Hill. 1999.

[SOM00] I. Sommerville. Software engineering. Addison-Wesley Pub Co, 6ta edición, Agosto 2000.

[STE00] B. Joy, G. Steele, J. Gosling y G. Bracha. The Java(TM) lenguaje specification. Addison-Wesley Pub Co, 2da edición, Junio 2000.

[STRO00] B. Stroustrup. The C++ programming language. Addison-Wesley Pub Co, 3ra edición, Febrero 2000.

[SUN01] Sun Microsystems, Inc. Enterprise javabeans specification, version 2.0. http://java.sun.com/product s/ejb/docs.html, Agosto 2001.

[SUN02] Sun Microsystems, Inc. Java remote method invocation specification. ftp://ftp.java.sun.com/docs /j2se1.4/rmi-spec-1.4.pdf

[SUN97] Sun Microsystems, Inc. Javabeans. http://java.sun.com/product s/javabeans/docs/spec.html, Agosto 1997.

[SZY97] Szyperski, C. "Component software. Beyond object-oriented programming". Addison-Wesley y ACM Press, 1997.

[SZY98-A] Clemens Szyperski. "Component Software Beyond Object – Oriented Programming". P 15 -20. 1998.

[SZY98-B] Clemens Szyperski, "Component Software Component Software Beyond Object – Oriented Programming". P 20 -22. 1998.

[SZY02] C. Szyperski. Component software: Beyond object-oriented programming. Addison-Wesley Pub Co, 2da edición, Noviembre 2002.

[THAI03] T. L. Thai and H. Lam. .NET framework essentials. O'Reilly & Associates, 3ra edición, 2003.

[XIA02] Cai Xia, Lyu Michael R., Wong Kam – Fai, "Component bases software Engineering: Technologies, Development Frameworks ans quality assurance schemes". p. 374-375. 2002

[WALL00] K. C. Wallnau and D. Plakosh. Waterbeans: A custom component model and framework. http://www.sei.cmu.edu/cbs/ cbse2000/papers/23/23.html, 2000.

[WIK11a] Wikipedia, La Enciclopedia Libre. Servicio Web, 2011 [en línea] <http://es.wikipedia.org/wiki/Web_service> [consulta: 22 Agosto 2011]

[WIK11b] Wikipedia, La Enciclopedia Libre. SOA Arquitectura Orientada a Servicios, 2011 [en línea] <http://es.wikipedia.org/wiki/SOA> [consulta: 25 Agosto 2011]

[WIK11c] Wikipedia, La Enciclopedia Libre. XML, 2011 [en línea] <http://es.wikipedia.org/wiki/XML> [consulta: 15 Julio 2011]

ANEXOS

ANEXO 1 - MANUAL DE INSTALACIÓN DE HERRAMIENTAS UTILIZADAS

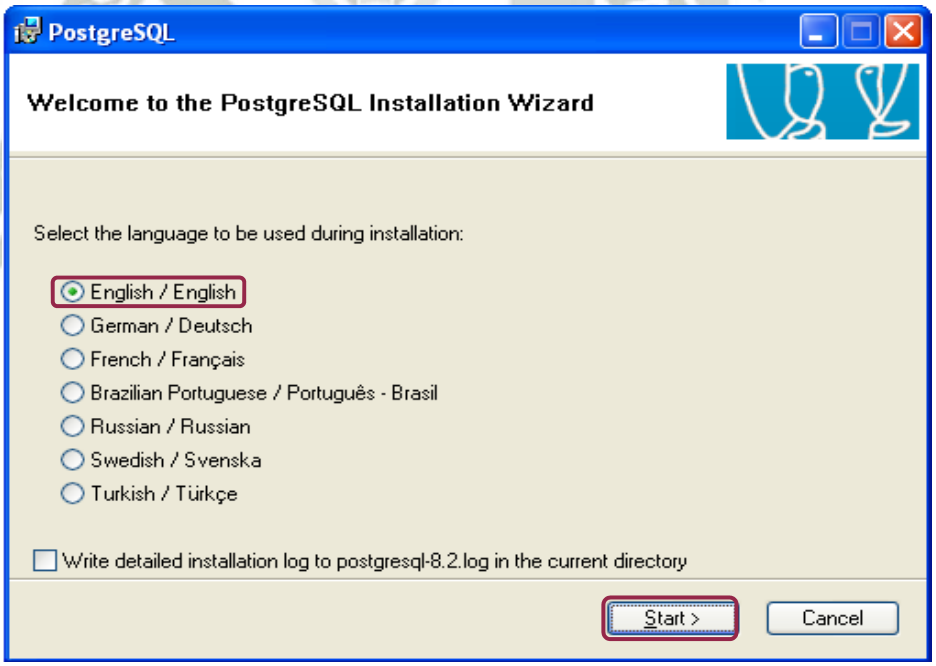
1. INSTALACIÓN Y CONFIGURACIÓN EN WINDOWS

1.1. POSTGRES

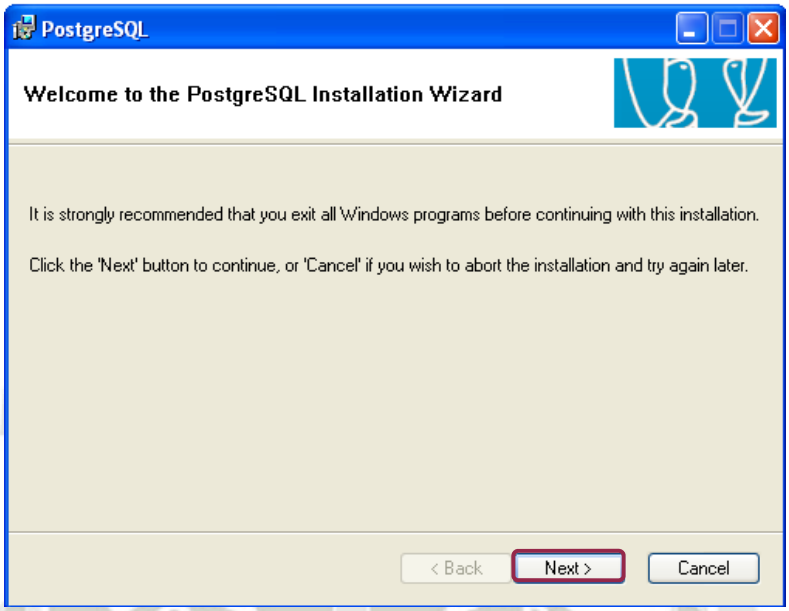
1.1.1. INSTALACIÓN

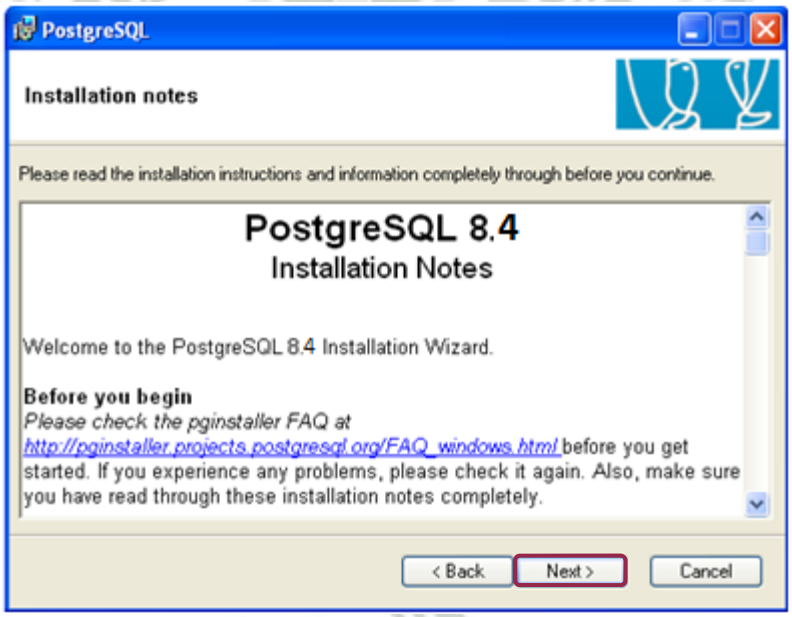
Para acceder al instalador del apache deberá dirigirse a la carpeta ***INSTALADORES*** que se encuentra en su CD de instalación. Aquí encontrara la carpeta ***POSTGRES*** en el cual encontrara el archivo **postgresql-8.4.msi y siga los siguientes pasos**

1. Elegir el idioma, por defecto ***English/English*** y pulsar ***"Start>"***

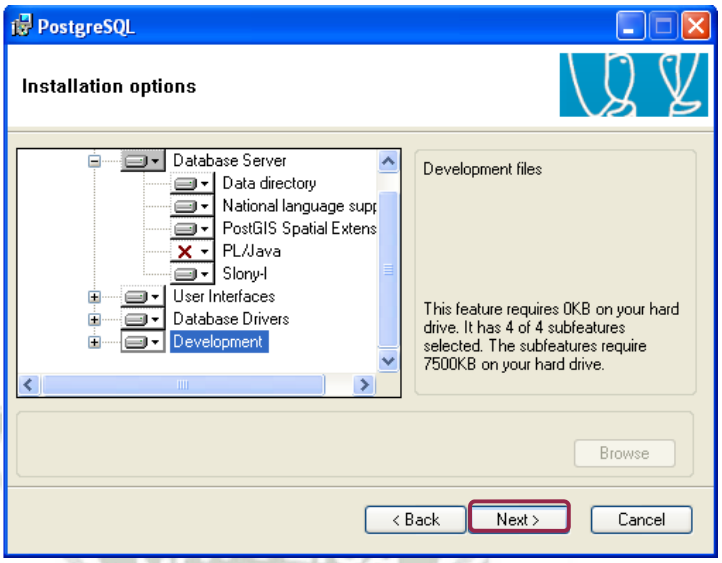


2. Aparecerá la pantalla de Bienvenida para poder iniciar la instalación de Postgresql, pulsar ***"Next >"*** y se mostrara una pantalla de Instalación notes en la cual solo deberá pulsar ***"Next >"***

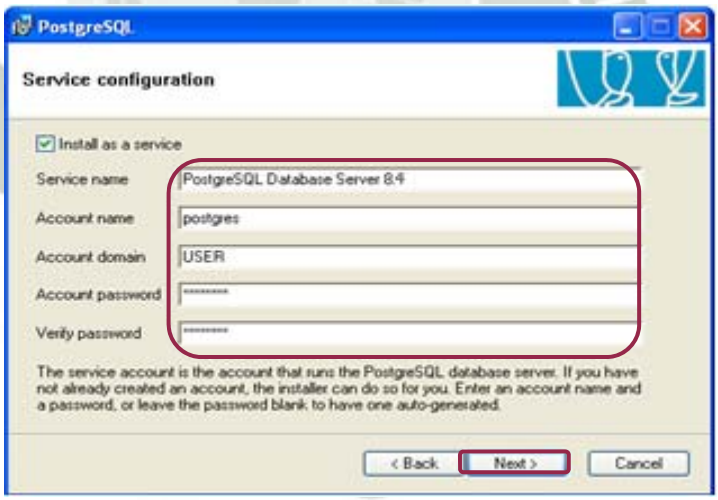




3. En este paso se deberá tener en cuenta las opciones de los paquetes que se deseen instalar, se deberá instalar todos los paquetes excepto PL/Java, pulsamos ***"Next >"***

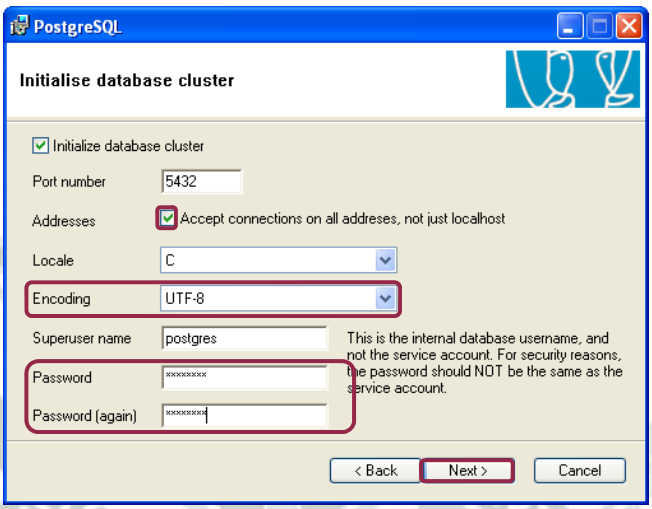


4. Aquí se debe de especificar una contraseña, en los campos ***"Account password y Verify password",*** el usuario se carga por defecto, no se deberá de cambiar esa información, seguimos y pulsamos ***"Next >"***

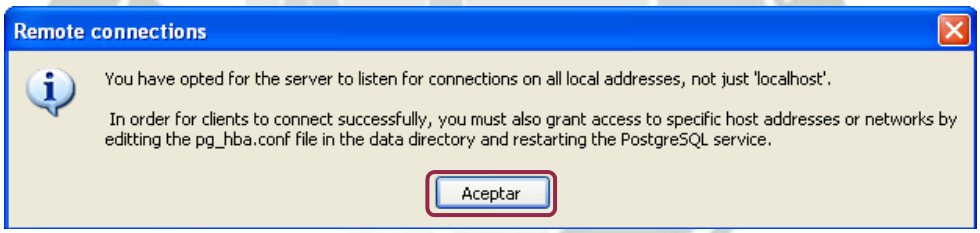


5. Paso muy importante, aquí se debe seleccionar ***Accept connections on all addreses, not just localhost*** , en el ***Encoding*** se debe seleccionar ***UTF-8***, y por ultimo poner la

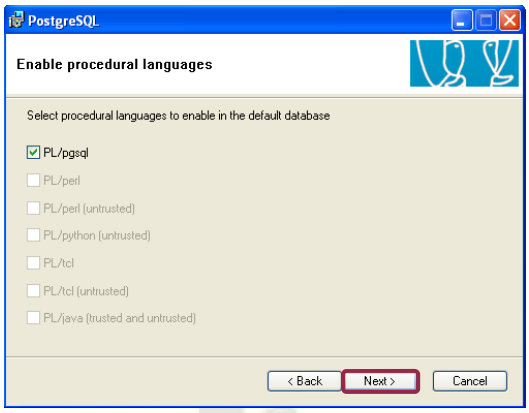
contraseña que será utilizada para conectarnos con nuestras bases de datos, una vez llenada esa información pulsar ***"Next >"***.

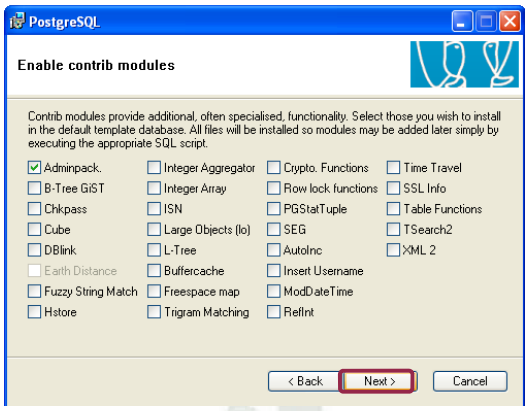


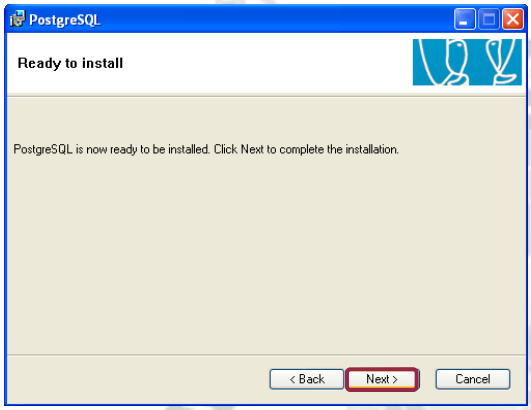
Se mostrará un mensaje de información el cual nos indica que no solo nuestro equipo puede acceder a la base de datos, el cual deberemos ***"Aceptar"***.

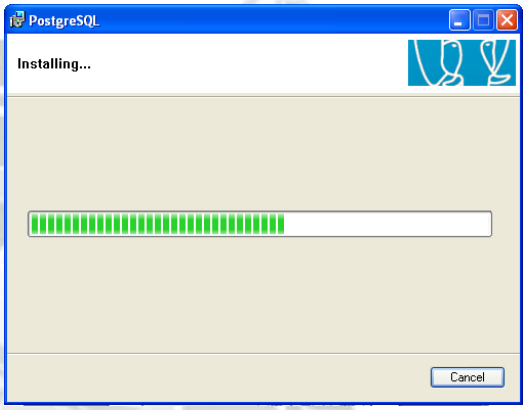


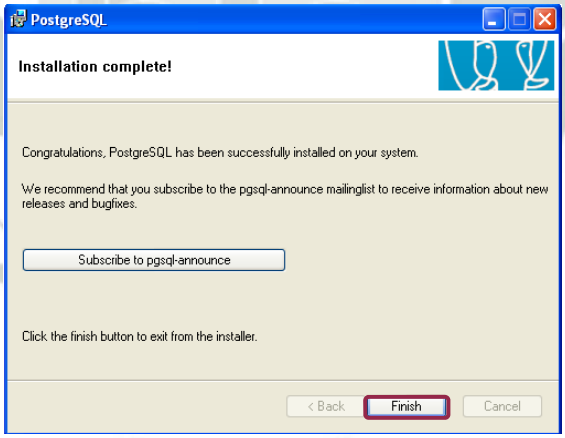
6. Dejar seleccionado las casillas que aparecen por defecto y pulsar ***"Next >"***, de este modo se continuara, hasta que nos aparezca la pantalla de ***Installation complete***, donde pulsaremos ***"Finish""***





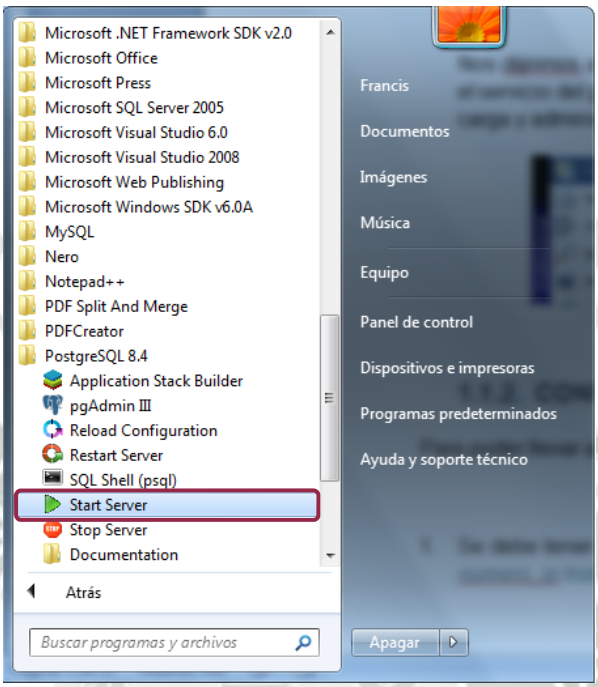






Nos dirigimos a ***Inicio-> Programas-> PostgreSQL 8.4 -> Start Server*** para poder iniciar el servicio del postgresql, aquí encontraremos las opciones, para acceder a la

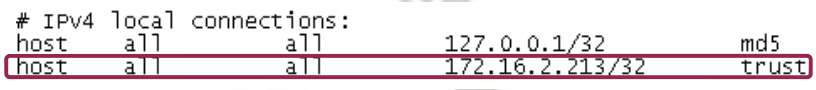
configuración, carga y administración del servicio de postgresql, así como al manejo de las bases de datos.



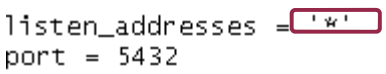
1.1.2. CONFIGURACIÓN

Para poder llevar a cabo la configuración se deberá seguir los siguientes pasos:

1. Se debe tener en cuenta el archivo *pg_hba.conf*, en el cual se deberá agregar *"host all numero_ip trust"* como se muestra en la figura.



2. Se debe tener en cuenta el archivo *postgresql.conf*, en el cual se deberá verificar la línea *"listen_addresses="*"* y *"port =5432"* como se muestra en la figura.

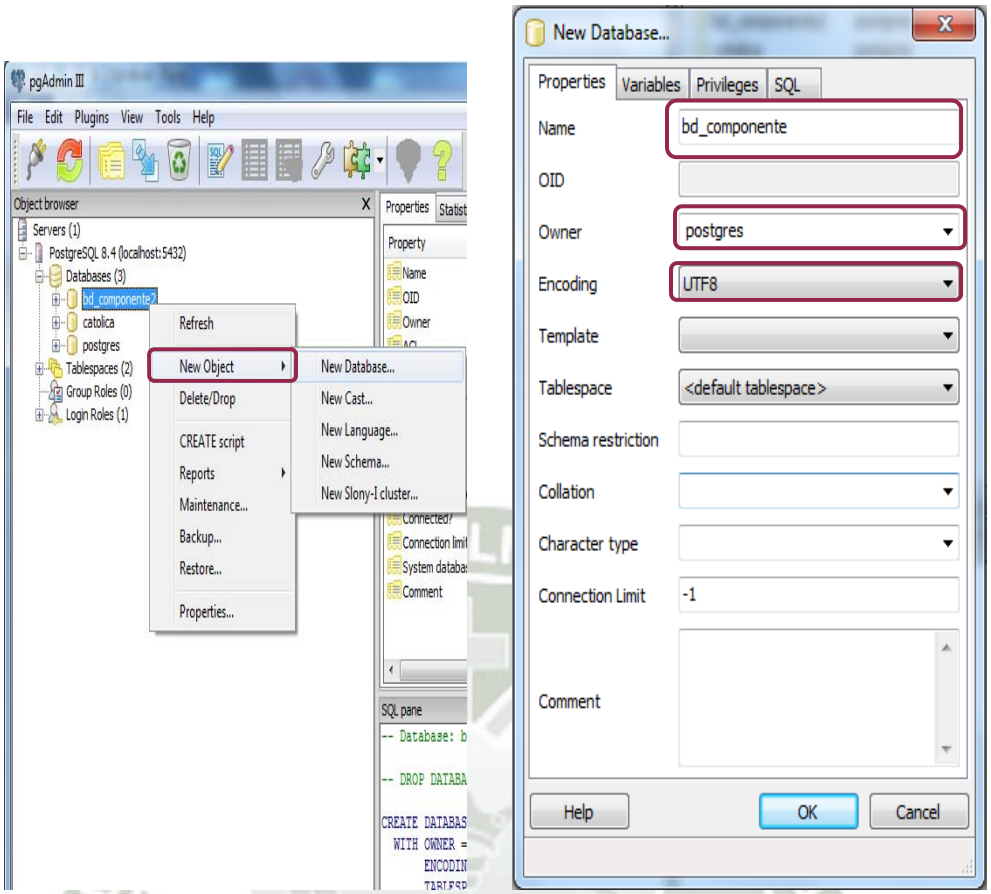


1.2. INSTALACIÓN DEL SISTEMA

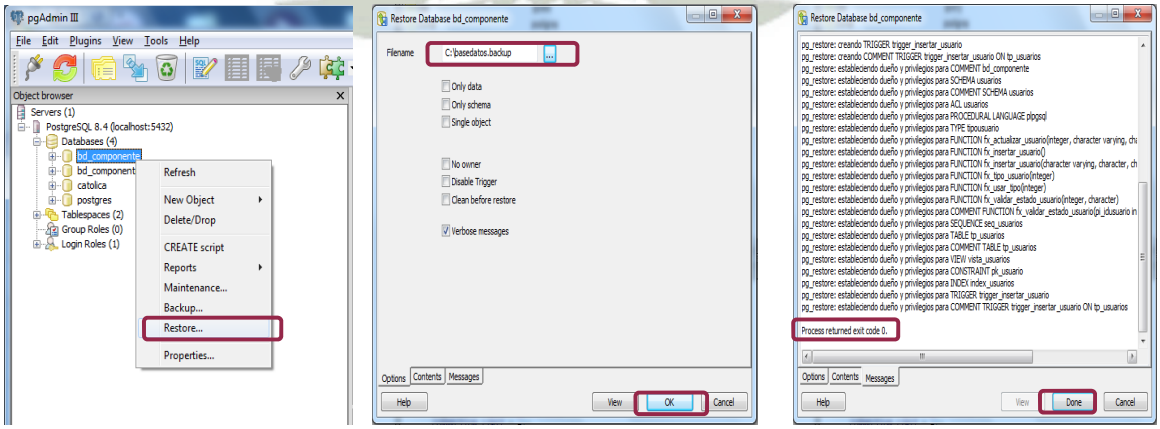
1.2.1. RESTAURACION DE BASE DE DATOS

1. Para cargar la base de datos se debe acceder al pgAdminIII, ya sea mediante el icono que se encuentra en el escritorio  o por medio de la ruta "*Inicio -> Programas -> PostgreSQL 8.2 -> pgAdminIII*". Para conectarnos debemos introducir la contraseña que colocamos al momento de la instalación

1. Para poder usar la base de datos, se debe crear una nueva base de datos, Accedemos a *Base de datos -> click derecho -> Nueva Base de Datos* como se muestra en la figura, seguidamente le damos un *Nombre* a la base de datos, verificar la *Codificación: UTF-8* y pulsar *"OK"*



2. Una vez creada la base de datos seleccionamos *Restaurar* haciendo *click derecho* sobre la base de datos, en *Nombre del Archivo:* elegimos el lugar donde se encuentra la copia de la base de datos deseada, y pulsa *"OK"* **,** esperamos a que el proceso finalice y presionamos *Cancelar*



ANEXO 2 - WINDOWS COMMUNICATION FOUNDATION

1. WCF (Windows Communication Foundation)

Windows Communication Foundation (WCF), es parte del .NET Framework (Incluido en las versiones 3.0 y 3.5), es una plataforma que los desarrolladores pueden usar para construir aplicaciones orientadas al servicio. El término orientado a las aplicaciones de servicio es una composición de una colección de servicios que se comunican solo para intercambiar mensajes.

1.1. Configuración de Servicio WCF

Todas las estructuras WCF tienen un contracto el cual es consumido para definir los siguientes tres núcleos del contrato.

- **Contrato del servicio (Service Contract).-** El cual define que operaciones del servicio están disponibles para ser invocados mediante mensajes de requerimientos que son enviados al servicio desde el cliente.
- **Contrato de data (Data Contract).-** Define la estructura de data incluido en los Payloads de los flujos de mensajes dentro y fuera del servicio.
- **Contrato del mensaje (Message contract).-** Permite controlar las cabeceras que aparecen en los mensajes y como los mensajes son estructurados.

Los tres tipos de núcleo de los contractos WCF mapean directamente para corresponder estándares de Web Services en la siguiente forma:

- Contrato de Servicio mapea a un WSDL.
- Contrato de Data mapea a un XSD.
- Contrato de mensajes mapea al protocolo SOAP.

Estos mecanismos de definición de contratos residen en un namespace (espacio de nombres) llamado System.ServiceModel, Microsoft como parte de su visión de modelado y Dominio de lenguajes específicos (DSLs) mira estas definiciones de contrato de lenguajes como modelos basados en .NET que pueden ser usados como alternativas a sus contrapartes basados en estándares.

Sin embargo el protocolo de transporte, (Hypertext Transfer Protocol [HTTP], el protocolo de control de transmisión [TCP], named pipes, o Microsoft Message Queuing [MSMQ]), son usados para cargar mensajes a y desde el endpoint (punto final) del servicio, el canal basado en la capa del Pipe WCF de presentación final representa al mensaje en la memoria como un mensaje de objeto. Los objetos *Message* son esencialmente representaciones .NET de un mensaje SOAP. Estos objetos de mensaje también puede contener encabezados de direccionamiento de conformidad con la norma WSAddressing, las cabeceras se serializan con el mensaje si el enlace admite direccionamiento.

1.1.1. ServiceContractAttribute

El ServiceContractAttribute, definido en el espacio de nombres System.ServiceModel, se puede aplicar a cualquiera de las interfaces. NET o clases NET. Se utiliza para declarar el tipo como un contrato de servicios. Este atributo no es hereditario, por lo que si la interfaz o clase que se utiliza para definir el contrato de servicio hereda de otro contrato de servicio, debe utilizar explícitamente el atributo ServiceContract al declarar el subtipo como un contrato de servicio también.

Opciones de parámetros con nombre para ServiceContractAttribute

- **Name** especifica un nombre diferente para el contrato en lugar del predeterminado, que es simplemente la interfaz o el nombre del tipo de clase. El nombre del contrato es lo que aparecerá en el nombre PortType cuando los consumidores accedan al WSDL. El espacio de nombres especifica un espacio de nombres en el WSDL para el servicio. El espacio de nombres por defecto siempre es **Http://tempuri.org**.
- **CallbackContract** asociados a otro contrato de servicio como un contrato de devolución de llamada. Esto establece la infraestructura para un MEP dúplex y por lo tanto permite que el servicio para iniciar el envío de mensajes al cliente. Los "Patrones de Intercambio de Mensajes".
- **ProtectionLeve**l activa el contrato de servicio para especificar las restricciones en la cantidad de mensajes a todas las operaciones en el contrato que están protegidos en el cable, es decir, si están firmados y cifrados.
- **ConfigurationName** Específica el nombre del atributo del elemento de servicio en un archivo de configuración. El valor predeterminado es el nombre de la clase de implementación del servicio, es decir, el tipo de servicio.
- **SessionMode** Específica si las sesiones deben ser habilitadas por el punto final (endpoint) de la exposición de este contrato de servicio. Los siguientes son algunos puntos a tener en cuenta sobre el uso de estos parámetros cuando se definen los contratos de servicios:

Utilice siempre la propiedad Namespace para proporcionar un identificador uniforme de recursos (URI) que de alguna manera identifica de forma única tanto en su organización y el dominio conceptual o negocio en que opera el servicio. Además, es una buena idea adoptar el World Wide Web Consortium (W3C) convención de usar el año y mes para diferenciar las versiones de su servicio. Con estos puntos en mente, entonces, una versión ligeramente mejorada del contrato de trabajo de Service Manager sería:

```
' VB
<ServiceContract(Name:="TaskManagerService", _
Namespace:="http://schemas.fabrikam.com/2012/04/tasks/")> _
Public Interface ITaskManagerService
' Etc...
End Interface
// C#
[ServiceContract(Name = "TaskManagerService",
Namespace = "http://schemas.fabrikam.com/2012/04/tasks/")]
public interface ITaskManagerService
{// Etc...}
```

1.1.2. OperationContractAttribute

El OperationContractAttribute, también se define en el espacio de nombres System.ServiceModel, sólo se puede aplicar a los métodos. Se utiliza para declarar el método como perteneciente a un contrato de servicio. Al igual que con el ServiceContractAttribute, la OperationContractAttribute se pueden declarar con una variedad de parámetros con nombre que proporcionan control sobre la descripción del servicio y formatos de mensaje.

Opciones de parámetros con nombre para OperationContractAttribute

- **Name** Específica un nombre diferente para la operación en lugar de usar el valor por defecto, que es el nombre del método.
- **ReplyAction** Controla el encabezado de acción para mensajes de respuesta de la operación.
- **IsOneWay** Indica si la operación es unidireccional y no tiene respuesta. Cuando las operaciones son de ida, ReplyAction no es compatible.
- **ProtectionLevel** Activa el contrato de servicio para especificar las restricciones en la cantidad de mensajes a todas las operaciones en el contrato están protegidos en el cable, es decir, si están firmados y cifrados. El valor de esta propiedad en el nivel de operación reemplaza la propiedad ProtectionLevel del ServiceContractAttribute.
- **IsInitiating** Indica si la invocación de la operación inicia una nueva Sesión entre el llamante y el servicio.
- **IsTerminating** Indica si la invocación de la operación termina una sesión existente entre el llamante y el servicio.

```
// C#
[ServiceContract]
public interface SomeCrudContract
{[OperationContract(IsOneWay = true,
Action = "urn:crud:insert")]
void ProcessInsertMessage(Message message);
[OperationContract(IsOneWay = true,
Action = "urn:crud:update")]
void ProcessUpdateMessage(Message message);
```

```
[OperationContract(IsOneWay = true,
Action = "urn:crud:delete")]
void ProcessDeleteMessage(Message message);
// The catch-all operation:
[OperationContract(IsOneWay = true,
Action = "*")]
void ProcessUnrecognizedMessage(Message message);
}
```

1.1.2.1. DataContractAttribute

El **DataContractAttribute**, definido en el namespace **System.Runtime. Serialization,** es usado para declarar a un tipo como un Data Contract. Cuando usamos este tipo, los detalles esenciales para el mismo son como sigue:

Opciones de parámetros con nombre para *DataContractAttribute*

- **Name** Determina el nombre del tipo y como este es generado en el esquema resultante. El valor por defecto es el tipo de nombre como se declara en el .NET. El Namespace Sets el objetivo namespace para el esquema.

  Este por defecto a http://schemas.datacontract.org/ 2004/07/ [CLR namespace], donde [CLRnamespace] es el namespace en el cual el tipo complejo es definido. Este puede ser aplicado solamente para enums, estructuras y clases, este no es heredable, solamente sus herederos deben ser los atributos aplicados explícitamente. Este tiene solamente dos parámetros que son:

1.1.2.2. DataMemberAttribute

El DataMemberAttribute, también es parte del namespace System.Runtime. Serialization, es aplicada solo a miembros y es usado para declarar que los miembros deberían ser incluidos en la serialización de la estructura de datos. Este atributo es proveído por muchos parámetros para controlar el resultado del esquema (schema) generado por un tipo complejo (complex type).

Opciones de parámetros con nombre para *DataMemberAttribute*

- **Name** Controla el elemento del esquema del nombre generado para el Miembro (que es, el campo o propiedad) que adorna al atributo.
- **IsRequired** Controla los atributos de minOccurs para el elemento del esquema. El valor por defecto es false, que es, cuando el elemento es opcional, el cual se traduce como minOccurs = 0.
- **Order** Controla el orden de cada elemento en el esquema, por defecto si la propiedad no es seteada explícitamente, los datos miembros aparecen alfabéticamente, seguido por los elementos para los cuales esta propiedad es seteada explícitamente.

1.1.2.3. EnumMemberAttribute

El atributo final considerado para la relación-contrato de la data es el EnumMemberAttribute. Como su nombre lo sugiere, este es usado para declarar que un elemento para un enum que ha sido declarado con un DataContractAttribute debería ser considerado como parte del Data Contract. La única propiedad o parámetro de nombre es la propiedad Value, el cual

puede ser usado para proveer un valor enumerado para ser serializado. El valor por defecto es el valor actual sin la enumeración.

1.1.3. MessageParameterAttribute

El MessageParameterAttribute, también se define en el espacio de nombres System.ServiceModel, controla los nombres de los parámetros de funcionamiento y los valores devueltos aparecen en la descripción del servicio, es decir, la forma en que se emiten como parte del WSDL para el servicio. Este atributo tiene una sola propiedad, la propiedad Name. Un caso concreto en el que este atributo puede ser útil es cuando se necesita tener una palabra clave .NET o nombre de tipo para el nombre del elemento XML que se serializa en la capa de transporte de alambre de nivel. En el siguiente fragmento de código, un MessageParameterAttribute se utiliza con este parámetro para controlar cómo tanto los parámetros y valores de retorno se serializa a petición XML y los elementos de respuesta a la capa de transporte. Es necesario utilizar la propiedad Name por el nombre de la variable como es, no puede ser utilizado en el lenguaje de programación de destino.

```
// C#
[OperationContract()]
[return: MessageParameter(Name = "responseString")]
string SomeOp([MessageParameter(Name = "string")]string s);
Contratos de mensaje (Message Contracts)
```

En el desarrollo de sus servicios WCF, los contratos de datos permiten definir la estructura de los datos que serán enviados en el cuerpo de los mensajes SOAP, ya sea en los entrantes (bajo petición) o en los mensajes de salida (respuesta).

Las siguientes son las principales razones por la que es posible que desee utilizar los contratos de mensajes:

- Para controlar cómo el cuerpo del mensaje SOAP está estructurado y, en última instancia, cómo se serializa.
- Para el suministro y acceso a los encabezados personalizados

Esta sección cubre los atributos necesarios para definir los contratos de mensaje y de las razones principales para usar contratos de mensaje.

Atributos de contrato de mensaje (Message Contract Attributes)

Para definir los contratos de mensaje, utilice los siguientes atributos: MessageContractAttribute, Message-HeaderAttribute y MessageBodyMember Attribute.

1.1.4. El ABC de los Endpoints – Address (Dirección), Binding (Enlace), Contract (Contrato)

Afortunadamente, los tres elementos que constituyen un comienzo de punto final se definen con las letras que componen una frase mnemotécnica, "ABC de endpoints", que se refiere a los tres elementos principales usados para formar un punto final. Cada extremo debe estar asociado con una dirección, un enlace y un contrato.

a) Dirección

La dirección de un punto final es un único localizador uniforme de recursos (URL) que identifica la ubicación del servicio. La dirección debe seguir el servicio Web Addressing (WS-Addressing) estándar, lo que significa que podría contener los siguientes cuatro partes:

- **Scheme.-** La parte de nivel superior de la dirección, que suele ser "http" seguido de dos puntos. Esto no es lo mismo que un protocolo, a pesar de que comúnmente utiliza las mismas letras como el protocolo.
- **Machine.-** Identifica el nombre de la máquina, que puede ser una URL pública como "www.contoso.com" o un identificador local como "localhost".
- **Port.-** El número de puerto opcional, precedido por dos puntos.
- **Path.-** La ruta de acceso utilizada para localizar los archivos de servicio. Normalmente, esto es sólo el nombre del servicio, pero el camino puede constar de más de un nivel cuando un servicio reside en una estructura de directorio.

Una dirección puede variar, dependiendo de si se está alojado en Microsoft Internet Information Services (IIS) en una red pública o están alojados localmente en un ordenador de la red interna. También puede variar, dependiendo del protocolo de los usos de unión. Por ejemplo, todos los siguientes podrían ser direcciones válidas:

- http://www.contoso.com/OrderService/

- http://localhost:8000/ServiceModelSamples/OrderService/
- net.tcp :// / localhost: 8001/OrderService /

b) Binding

El enlace de datos determina cómo el servicio puede ser accedido. Esto significa que la unión puede especificar no sólo el protocolo utilizado para acceder al servicio, sino un método de codificación usado para formatear el contenido del mensaje. La unión también puede especificar los requisitos de seguridad, como Secure Sockets Layer (SSL) o seguridad de mensajes SOAP. Para hacer las cosas más fácil para los desarrolladores, WCF proporciona un conjunto de enlaces incorporados, la unión que usted elija depende de varios factores específicos de su red y el entorno operativo. Por ejemplo, si sabe **que el servicio puede residir en un único equipo, el netNamedPipeBinding** sería la más eficiente. Alternativamente, si usted necesita comunicarse a través de computadoras, netTcpBinding o NetPeerTcpBinding puede funcionar bien. Si la interoperabilidad es fundamental y debe comunicarse con las computadoras no-WCF, usted tiene que elegir una unión tal como basicHttpBinding o el wsHttpBinding. Por último, si el servicio requiere soporte para llamadas desconectadas o en cola, debe utilizar un enlace que admita Microsoft Message Queue (MSMQ).

Descripción de Binding (Enlace)

- **basicHttpBinding** Este enlace interoperable es comúnmente usado como un reemplazo para los servicios Web basados en ASMX anteriores (Métodos Active Server). Es compatible con Hypertext Transfer Protocol (HTTP) y protocolo de transferencia de hipertexto a través de SSL (HTTPS) Los protocolos de transporte, así como los métodos de transmisión de mensajes de texto y mecanismo de optimización (MTOM) de codificación.
- **WSHttpBinding.-** Este enlace seguro e interoperable utiliza SOAP sobre HTTP y fiabilidad de los soportes, las transacciones y la seguridad en Internet. Es compatible con los protocolos HTTP y HTTPS transporte, así como los métodos de escritura y MTOM codificación.
- **WSDualHttpBinding**.- Este enlace interoperable es de uso general para los contratos de servicios dúplex ya que soporta la comunicación bidireccional.
- **WebHttpBinding.-** Este enlace seguro e interoperable envía la información directamente a través de HTTP o HTTPS sin necesidad de crear un sobre SOAP. Es una opción eficiente cuando el jabón no es requerido por el cliente.

La exposición de los servicios

El enlace que usted elija depende también de cuál método de mensaje que codifica es el necesario. Algunos enlaces pueden ser codificados como binarios, lo que puede dar mejores resultados de rendimiento. Sin embargo, la codificación binaria no está disponible con todos los enlaces. Para los servicios que requieren interoperabilidad, texto plano o codificación MTOM es necesario. Afortunadamente, usted puede especificar varios puntos finales de un servicio. Esto significa que no están atados a un solo método, y el cliente puede utilizar el mejor disponible.

c) Contrato

El elemento final en el extremo de servicio es el contrato. Esto identifica las operaciones expuestas por el servicio, y que por lo general se refiere al nombre de la interfaz, precedido por el espacio de nombres del proyecto. Al incluir el espacio de nombres, se utiliza el nombre de tipo completo del contrato. Por ejemplo, suponga que tiene un servicio llamado MyService, y la interfaz que se utiliza para definir el servicio se denomina IMyService. El archivo de clase que este servicio se encuentra dentro usa el espacio de nombres MyNamespace. En este caso, el contrato para este servicio sería MyNamespace. IMyService.

1.1.5. Creación de un Endpoint utilizando un archivo de configuración

Los extremos se pueden especificar obligatoriamente a través de código o mediante declaración en la configuración. Todo lo que se puede hacer en el código se puede hacer con un archivo de configuración, y viceversa. Esto

podría ser el archivo web.config, si su servicio está alojado en IIS o Windows Application Service (WAS), o el archivo app.config si su servicio está alojado de forma independiente con una aplicación administrada.

Siempre se deben configurar los endpoints mediante un archivo de configuración. Se considera una buena práctica usar un archivo de configuración al especificar puntos finales. Esto le permite realizar cambios en los criterios de valoración sin recompilar código costoso.

Usando el elemento System.ServiceModel

La información de configuración para un servicio está contenido dentro de un elemento System.ServiceModel. Este elemento incluye un elemento de servicios, que también puede contener uno o más elementos de servicio. Aquí es donde el endpoint se define. Debe especificar al menos un endpoint, o usted recibirá un error en tiempo de ejecución. Por ejemplo, el código siguiente especifica un extremo de servicio único para un servicio llamado OrderService.

```
OrderService:
<configuration>
<system.serviceModel>
<services>
<service name="MyNamespace.OrderService">
<endpoint address="http://localhost:8000/OrderService/"
binding="wsHttpBinding"
contract="MyNamespace.IOrderService" />
</service>
</services>
</system.serviceModel>
</configuration>
```

Uso de una Dirección Base

WCF ofrece dos opciones principales para especificar la dirección del servicio. En los ejemplos presentados hasta ahora, ustedes han visto la dirección que se especifica como una dirección absoluta. Este método es sencillo y probablemente el más sencillo de entender, pero cuando están implicados varios puntos finales, es más eficaz utilizar un método de dirección relativa. En este caso, debe utilizar una dirección base para especificar la parte común de la dirección. Especificar la dirección base en el elemento principal para cada servicio. Puede especificar una dirección base mediante el elemento add. Por ejemplo, en la configuración anterior donde varios enlaces se especifican, existe una dirección común en la forma de http://localhost:8000/OrderService/. Esta porción de la dirección se usa en los dos primeros extremos de dicha configuración. Si desea utilizar una dirección base en ese mismo ejemplo, la configuración tendría que ser cambiada como se muestra en negrita para parecerse a la siguiente:

```
<configuration>
<system.serviceModel>
<services>
<service name="OrderService">
<host>
<baseAddresses>
<add baseAddress="http://localhost:8000/OrderService/"/>
<add baseAddress="net.tcp://localhost:8001/OrderService/"/>
</baseAddresses>
```

```
</host>
<endpoint address=""
contract="MyNamespace.IOrderService"
binding="BasicHttpBinding">
</endpoint>
<endpoint address="secure"
contract="MyNamespace.IOrderService"
binding="wsHttpBinding">
</endpoint>
<endpoint address=""
contract="MyNamespace.IOrderService"
binding="NetTcpBinding">
</endpoint>
</service>
</services>
</system.serviceModel>
</configuration>
```

1.1.6. Hosteando un servicio WCF mediante un servicio de Windows

Servicios de Windows, anteriormente conocido como servicios de Windows NT, son útiles cuando se trata de larga duración servicios WCF que no requieren una interfaz de usuario. Mediante el uso de un servicio de Windows, usted puede estar seguro de que el host del servicio está siempre disponible. Un servicio de Windows puede ser configurado para iniciarse automáticamente cuando se carga el sistema operativo. También se puede pausar, detener y reiniciar mediante el uso de un complemento de MMC.

Usted puede crear un servicio de Windows utilizando una de las plantillas de Visual Studio 2008. Al igual que la aplicación de consola, el proyecto de servicio de Windows puede ser añadido a la solución para su servicio. En Visual Studio 2008, haga clic en la solución en el Explorador de soluciones, haga clic en Agregar y, a continuación, haga clic en Nuevo proyecto. A continuación, seleccione el servicio de Windows, que se encuentra bajo la categoría de Windows, como la plantilla y escriba el nombre OrderWindowsService para su aplicación host.

Antes de empezar a agregar código al proyecto, debe agregar una referencia al componente System.ServiceModel haciendo clic derecho en el nuevo proyecto y seleccionando Agregar referencia. A continuación, seleccione System.ServiceModel de la lista y haga clic en Aceptar. También debe agregar una referencia al componente System.Configuration.Install, que se utiliza para el instalador del proyecto. Después de que los componentes se seleccionan, añadir las siguientes directivas a la parte superior del archivo Service1.vb o Service1.cs:

```
// C#
using System.ServiceModel;
using System.ComponentModel;
using System.Configuration.Install;
using System.ServiceProcess;
```

Los métodos OnStart y manipuladores OnStop se añaden automáticamente cuando se crea una aplicación, utilizando la plantilla Visual Studio 2008. Además, la clase de aplicación se hereda de la clase ServiceBase. Esto es esencial para la aplicación como un servicio de Windows. Debe agregar código al principio del archivo de clase, que se declara una variable ServiceHost. Por ejemplo, el siguiente código se puede añadir al servicio Windows al final de la definición de clase:

```
// C#
public ServiceHost serviceHost = null;
```

Para ubicar a su servicio en la ventana de la consola de servicio, establezca la propiedad ServiceName en el constructor.

También debe proporcionar un punto de entrada para la aplicación host. Por ejemplo, puede proporcionar el código necesario para un servicio de Windows llamado OrderWindowsService usando el siguiente código:

```
// C#
public OrderWindowsService()
{ServiceName = "Order Service Windows Service";}
public static void Main()
{ServiceBase.Run(new OrderWindowsService());}
```

También debe agregar código al método OnStart que va a crear una instancia de ServiceHost y pase el tipo de su servicio WCF. El ServiceHost se abrirá y por lo tanto estará disponible para todas las aplicaciones cliente. Por ejemplo,

puede agregar el código siguiente al método OnStart para iniciar un servicio de WCF llamado OrderService:

```
// C#
protected override void OnStart(string[] args)
{ if (serviceHost != null)
{ serviceHost.Close();}
serviceHost = new ServiceHost(typeof(OrderService));
serviceHost.Open();}
```

El método OnStop es la encargada de cerrar la instancia ServiceHost. Por ejemplo, el código siguiente se puede utilizar para el método OnStop:

```
// C#
protected override void OnStop()
{ if (serviceHost != null)
{serviceHost.Close();
serviceHost = null;}}
```

Para que el servicio se instale como un servicio de Windows, debe crear una clase ProjectInstaller.

Esta clase hereda de la clase Installer (que forma parte del espacio de nombres System.ComponentModel) y debe estar marcado con el atributo RunInstaller. El código de la clase puede residir en el mismo proyecto de servicio de Windows creado previamente. Por ejemplo, puede agregar el siguiente código debajo de la clase OrderWindowsService:

```
// C#
[RunInstaller(true)]
public class ProjectInstaller : Installer
{private ServiceProcessInstaller process;
private ServiceInstaller service;
public ProjectInstaller(){
process = new ServiceProcessInstaller();
process.Account = ServiceAccount.LocalSystem;
service = new ServiceInstaller();
service.ServiceName = "Order Service Windows Service";
Installers.Add(process);
Installers.Add(service);}}
```

Por último, se debe agregar el código de configuración en el archivo app.config. El código de configuración debe incluir una dirección base y debe especificar todos los extremos expuestos por el servicio WCF. Una vez hecho esto, se puede compilar la aplicación para crear un ejecutable de servicio. A continuación, puede utilizar el installutil de línea de comandos para instalar el servicio de Windows. Por ejemplo, si ha creado un servicio llamado OrderWindowsService, puede ejecutar el siguiente comando desde un comando de Visual Studio 2008 del sistema:

```
installutil C:\MyProjects\bin\OrderWindowsService.exe
```

Una vez instalado, el servicio de Windows se puede acceder a través del Administrador de control de servicios, que es un complemento de MMC. Usted puede acceder al Administrador de control de servicios, escriba

services.msc desde un símbolo del sistema. Si ha seguido todos los pasos de esta sección, orden de servicio de servicio de Windows debería aparecer en la lista.

1.1.6.1. Hosteando un servicio mediante el Host WCF-Siempre

Visual Studio 2008 introdujo una nueva característica conocida como el host WCF proporciona. Este host incorporado es útil cuando se necesita implementar servicios simples rápidamente. Esto puede ser especialmente útil cuando usted está aprendiendo WCF, porque te libera de tener que añadir aplicaciones de consola para las soluciones de servicio constantemente. El host WCF-siempre es una utilidad de línea de comandos disponibles a través del ejecutable WcfSvcHost.

Para tener una idea de cómo funciona, supongamos que desea alojar un servicio llamado SimpleService. Las clases de contrato y la ejecución de este servicio se compilaron en una librería de enlace dinámico (DLL) denominada SimpleService.dll. El archivo de configuración del servicio se denomina App.config. Para utilizar el WCF proporcionan host, abra un símbolo del sistema y algo de tipo similar a lo siguiente (en sustitución de la ruta y el nombre de archivo con los específicos de su servicio):

```
WcfSvcHost.exe /service:C:\MyProject\bin\SimpleService.dll
/config:C:\MyProject\App.Config
```

Debería ver un cuadro de mensaje aparecerá cerca de la barra de tareas, que le informa que WcfSvcHost se ha iniciado. También debe obtener un servicio WCF Host que muestra información sobre el servicio que se ha cargado. En este punto, usted puede usar un cliente para acceder al servicio WCF.

1.1.6.2. Consumiendo servicios WCF

Esta lección le proporciona las herramientas que necesita para empezar a consumir servicios WCF. La mayoría de las plataformas de servicios web, como WCF incluye, proporcionar a los desarrolladores un mecanismo para la creación de un objeto que se puede utilizar para comunicarse con el servicio. Estos objetos son llamados proxies u objetos proxy, porque están actuando efectivamente como un proxy para el servicio. En esta lección, la atención se centra primero en las cuatro maneras que usted puede crear un proxy para un servicio WCF y luego en los detalles que usted necesita considerar para llamar a los servicios de la manera más eficaz de utilizar esos poderes.

a) Crear objetos proxy y clases de proxy

WCF proporciona a los desarrolladores de varios mecanismos para la creación de objetos proxy que se pueden utilizar para comunicarse con un servicio Web. En esta sección se tratan los siguientes enfoques diferentes para la generación de clases de proxy de metadatos de servicio

b) Configuración de WCF

Para algunos desarrolladores, la palabra administración se ha usado para causar aburrimiento y indiferencia Sin embargo, para los sistemas que utilizan Windows Communication Foundation (WCF), la administración significa algo más que trabajar para mantener a una aplicación que se ejecuta bien. Una de las fortalezas reales de WCF es la capacidad de posponer ciertos tipos de decisiones técnicas hasta muy tarde en el ciclo de desarrollo. En particular, usted puede decidir sobre el protocolo de transporte (servicios Web, Transmission Control Protocol [TCP], Microsoft Message Queuing [MSMQ]) y el canal de requisitos (seguridad, fiabilidad, etc.) hacia el final del proceso. Para algunos, esto puede no parecer una gran ventaja, pero en muchos casos, las decisiones arquitectónicas que se hacen al principio del ciclo de desarrollo de limitar las opciones que están disponibles en todo el resto del proceso de codificación. Aunque algunas decisiones deben hacerse al comienzo de un proyecto, algunas preguntas son más retrasadas hasta tan tarde en el proyecto como sea posible para proporcionar la máxima flexibilidad cuando se implementa la aplicación. Al elegir el mecanismo subyacente de las comunicaciones, WCF ayuda enormemente a posponer esta decisión hasta que, en algunos casos, las características de tiempo de ejecución de la aplicación se configuran. Para lograr este objetivo, algunas de las concesiones tienen que ser hechas. Por ejemplo, en el caso de WCF, la definición de los requisitos de canal debe ser movido fuera del código de la aplicación. En el mundo de. NET, "fuera del código de la aplicación" significa "los archivos de

configuración." Esto no quiere decir que WCF no se puede configurar a través de código, pero muchas de las opciones se pueden indicar mediante declaración en el archivo de configuración, así como imperativamente a través de código.

c) Configurar el extremo del cliente

Uno de los conceptos principales de la configuración de un servicio de WCF es que el servicio contiene un conjunto de puntos finales. Cada extremo puede hacer referencia a un conjunto de comportamientos de punto final. Cada extremo puede hacer referencia a una configuración de enlace predeterminada. Además, el servicio puede hacer referencia a un conjunto de comportamientos de servicio. Toda esta funcionalidad se puede expresar utilizando XML. La configuración de un extremo del cliente no es tan diferente de la configuración de un extremo de servicio. El cliente tiene un conjunto de criterios de valoración con la que puede comunicarse. Cada extremo está asociado con un enlace, y cada punto final puede tener una o más conductas. La diferencia, naturalmente, es que el cliente no apoyará los comportamientos de servicio. En su lugar, el cliente tiene la capacidad de definir comportamientos de devolución de llamada.

d) Configuración declarativa

Por ahora, los conceptos fundamentales de puntos finales (dirección, un enlace y contrato) debe ser muy familiar para usted, por lo que es lógico pensar que al configurar los puntos finales, ya sea en el cliente o el servicio, usted utiliza

estos conceptos fundamentales. A partir de los fundamentos, la configuración de extremo para un cliente se lleva a cabo en el elemento de extremo en el elemento de cliente en el elemento ServiceModel. Lo que sigue es un segmento de un archivo de configuración que muestra los atributos más utilizados.

```
<system.serviceModel>
<client>
<endpoint address="http://localhost:8080/UpdateService"
binding="wsHttpBinding"
contract="IUpdateService"
name="WSHttpBinding_IUpdateService">
</endpoint>
</client>
</system.serviceModel>
```

No es sorprendente que el elemento de configuración básica de un extremo especifica los tres componentes fundamentales de un extremo de WCF (dirección, enlace y el contrato). El otro atributo en el archivo de configuración, nombre, proporciona un mecanismo para la referencia programática.

e) Especificación de la dirección

En última instancia, la dirección tiene que evaluar a la dirección asociada con el extremo del servicio. El atributo especifica la dirección de ese valor. El valor es una cadena que representa el identificador uniforme de recursos

absoluta (URI) para el punto final del servicio, ya que la configuración del cliente no permite direcciones base y direccionamiento relativo. Esto es diferente de las capacidades del servicio de punto final, que se discuten en la lección 2. En su lugar, el URI absoluto para el punto final del servicio debe ser especificado.

f) Especificación de la unión

El enlace se especifica por su nombre. El valor del atributo de enlace es una cadena que contiene el nombre del enlace, que debe ser uno de los nombres estándar de unión o el nombre de un enlace personalizado. En este último caso, también debe definir una sección de extensión de enlace para vincular la unión a la clase que implementa el enlace. Aunque el nombre del enlace es parte de los elementos básicos de la configuración de extremo, no es la única pieza que puede ser configurado. Junto con el atributo de unión, hay un enlace de atributo-configuración. Este atributo, un valor de cadena, es el nombre de la sección de configuración de enlace que se crea una instancia cuando el cliente del punto final se crea.

g) Especificación del contrato

La tercera pieza de la dirección, un enlace, es el contrato, para lo cual el elemento de extremo incluye un atributo contrato. El valor de este atributo es el nombre de la interfaz que el servicio que se espera de implementar. El cliente debe ser consciente de los métodos y propiedades expuestas por la interfaz. Se puede encontrar esta información en un número de maneras. Una

de las técnicas más comunes es el uso de la utilidad svcutil para generar el proxy basado en los metadatos expuestos por el servicio. La utilidad svcutil también es utilizado por Visual Studio 2008 cuando la opción Agregar referencia de servicio seleccionado. El programa consulta el servicio a determinar los métodos y propiedades que se exponen a continuación, genera una clase de interfaz para su uso por la aplicación cliente.

Aunque el nombre del enlace es parte de los elementos básicos de la configuración de extremo, no es la única pieza que puede ser configurado. Junto con el atributo de unión, hay un enlace de atributo-Configuración. Este atributo, un valor de cadena, es el nombre de la sección de configuración de enlace que se crea una instancia cuando el cliente del punto final se crea.

h) Configurar Bindings

El propósito principal de un enlace es para expresar la intención. Específicamente, se define cómo cada componente en la aplicación de mensajería interactúa con los otros componentes. En su aplicación, los enlaces de crear una fuente de generador de canales o los objetos de canal de escucha, lo que significa que el enlace es una fábrica.

Para los enlaces, la zona de impacto es el canal. Desde una perspectiva de la configuración, los enlaces se exponen a través de la capa de modelo de servicio. Sin embargo, los objetos que se crean afectan o se inyectan directamente en la tubería de canales. Las instancias de estos elementos de

enlace proporcionan los medios por los que los clientes WCF implementan un conjunto particular de funcionalidad de mensajería. En definitiva, un enlace es el mecanismo promotor a usar para encapsular el tiempo de ejecución de funcionalidad de mensajería de WCF. Fuera de la caja, WCF soporta un gran número de medios de transporte, codificación de mensajes, protocolos y seguridad, transacciones, y las opciones de simultaneidad. El número total de combinaciones es muy grande, tan grande, de hecho, que el equipo de desarrollo de WCF ha trabajado duro para hacer que sea más fácil de usar para los desarrolladores. Ellos incluido combinaciones comunes a un conjunto de enlaces predefinidos. Al especificar el punto final, utilice el nombre del enlace en el atributo obligatorio. Entonces, los valores predeterminados correspondientes para las distintas propiedades se utilizarán. Como se mencionó anteriormente, es posible proporcionar información de configuración personalizado para un enlace determinado. El punto de partida es para especificar el nombre de la configuración de enlace en el atributo bindingConfiguration del punto final.

i) Configurar los comportamientos

Se utiliza el mismo estilo de sistema de referencia para configurar los comportamientos del cliente. Los comportamientos del cliente están relacionados con los puntos finales definidos en otros lugares en el archivo de configuración. Se espera que el atributo behaviorConfiguration en el punto

final se mencione el nombre de uno de los elementos de comportamiento en la sección endpointBehaviors.

j) Construcción de una Dirección

El proceso de construcción o modificación de la dirección de punto final de un proxy de servicio no es complicado. La clase básica implicada se llama EndpointAddress, y una instancia de esa clase puede ser pasada al constructor de la clase proxy. El código siguiente muestra este escenario fundamental.

```
' VB
Dim endpoint As New
EndpointAddress("http://localhost:8080/UpdateService")
Dim proxy As New UpdateServiceProxy(New WSHttpBinding(), endpoint)
// C#
EndpointAddress endpoint = new
EndpointAddress("http://localhost:8080/UpdateService");
UpdateServiceProxy proxy = new UpdateServiceProxy(new
WSHttpBinding(), endpoint);
```

Observe que la especificación del punto final, como mínimo, consiste en nada más que identificar el URI que indica la ubicación del servicio. Una segunda forma del constructor para el proxy lleva el nombre de un elemento de configuración variable como parámetro. Esto no es exactamente lo mismo que la configuración dinámica, pero sí se proporciona para re direccionar el mismo proxy para los diferentes parámetros en tiempo de ejecución. Por ejemplo, considere el siguiente elemento del archivo de configuración.

```
<system.serviceModel>
<client>
<endpoint address="http://localhost:8080/UpdateService"
binding="wsHttpBinding"
contract="IUpdateService"
name="WSHttpBinding_IUpdateService">
</endpoint>
</client>
</system.serviceModel>
```

El código siguiente le diría al proxy para utilizar la dirección y el enlace se especifica en este extremo. Supongamos, para este ejemplo, que la clase UpdateServiceProxy es la clase generada cuando una referencia de servicio se añade a través de Visual Studio 2008 o mediante el uso de la SVCUTIL de línea de comandos.

k) Encabezados de dirección

WCF proporciona un mecanismo compatible con WS-Addressing, para dar cabida lógica sofisticada de enrutamiento y despacho. La idea subyacente es la de suministrar información adicional con cada solicitud. Esta información puede ser utilizada tanto por los dispositivos intermedios o por el punto final del servicio en sí para determinar enrutamiento o el procesamiento de la lógica. Una de las preguntas obvias acerca de encabezados de dirección es por qué usted debe preocuparse. ¿No sería más fácil incluir sólo la información que se utiliza para determinar la ruta o transformación en el cuerpo del mensaje? De esta manera, el proxy de cliente sería consciente de ello y

simplemente incluir cualquier detalle adicional es necesario. Sin embargo, incluso esta información en el contrato no permite un número de escenarios comunes. Supongamos, por ejemplo, el cliente no tiene la información necesaria. Si el objetivo es determinar si un nivel superior o nivel de servicio debe responder, ¿por qué el cliente puede confiar para proporcionar la respuesta? Además, supongamos que la información de enrutamiento se adjunta al mensaje mediante un proceso intermedio. Supongamos, además, que un intermediario recibe la solicitud, añade la información de encaminamiento, y luego transfiere la solicitud. En ambos de estos casos, incluyendo el mensaje en el contrato no es apropiado. La clase AddressHeader (y la propiedad Headers en el punto final) son los puntos de entrada expuestos al mecanismo de WCF que se acaba de describir. Un encabezado de dirección se crea utilizando el método CreateAddressHeader estática en la clase AddressHeader. Este método toma dos parámetros de cadena: el nombre de la cabecera y la información a incluir en la cabecera.

```
// C#
AddressHeader header =
AddressHeader.CreateAddressHeader("premium",
"http://tempuri.org/ServiceLevel", null);
EndpointAddress endpoint = new
EndpointAddress(new Uri("http://localhost:8080/UpdateService"),
header);
UpdateServiceProxy proxy = new UpdateServiceProxy(new
WSHttpBinding(),
endpoint);
```

l) La construcción de un Binding

El código para crear instancias de un proxy para un servicio WCF incluye tanto un objeto EndpointAddress y un objeto de enlace. Es a través de este objeto de enlace que la información de enlace necesario para determinar la forma de comunicarse con un servicio WCF se especifica. Como recordatorio, el código para crear y utilizar un enlace es el siguiente.

```
// C#
EndpointAddress endpoint = new
EndpointAddress("http://localhost:8080/UpdateService");
WSHttpBinding binding = new WSHttpBinding();
UpdateServiceProxy proxy = new UpdateServiceProxy(binding,
endpoint);
```

En este caso sencillo, definiendo una unión para su uso con un servicio de WCF es sencillo. Basta con crear una instancia del tipo de encuadernación deseado y pasarlo al constructor para la clase proxy. En el caso no tan simple, un número de variaciones sobre este tema permiten una gran flexibilidad en el uso de WCF. Todas las clases de enlace de WCF incluidos con. NET Framework 3.5 se derivan de un tipo común abstracto, la clase de unión en el espacio de nombres System.ServiceModel.Channels. Debido a esto, todos los enlaces comparten una serie de características comunes. La buena noticia es que la jerarquía de herencia de la clase Binding es muy poco profunda. Se deriva directamente de System.Object y sólo implementa el interfaz IDefaultCommunicationTimeout.

El objeto base Binding tiene un constructor que toma dos parámetros de cadena: nombre y espacio de nombres. Estos parámetros representan el nombre y espacio de nombres XML del enlace que se va a crear. Los valores de estos parámetros son distintos del nombre de la propia unión.

Para el funcionamiento normal de los enlaces, los valores no son importantes. En cambio, cuando se utilizan las capacidades de la unión necesita ser representado como metadatos XML, tal como con WSDL.

Debido a que la clase Binding es un abstracto, este constructor no se es el único implementado en los enlaces que se proporcionan con WCF. De hecho, cada una de las clases de enlace especializados tiene al menos un constructor que toma parámetros adicionales. En las secciones siguientes se describen los constructores para cada clase de enlace.

m) La construcción de un comportamiento

La idea de que usted está construyendo un comportamiento es un poco de un nombre inapropiado. Un comportamiento, en el contexto de un cliente de WCF, es simplemente establecer las propiedades en el enlace para apoyar un nivel deseado de funcionalidad, por lo que cuando una instancia del objeto de unión deseada (y antes de que lo asocien con un punto final), sólo tiene que ajustar el propiedades en el objeto de enlace con los valores necesarios. El mayor desafío en la construcción de una conducta es garantizar que el enlace utilizado por el servicio también es compatible con las propiedades deseadas.

Los detalles de las propiedades que se utilizan en cada uno de los enlaces fuera de la caja son:

- **BasicHttpBinding Binding**

  El punto de partida es el enlace predeterminado para WCF, el utilizado para la mensajería básica HTTP. Esta unión tiene cuatro modos de seguridad, que se definen mediante la propiedad Security.Mode en el enlace. El modo de seguridad afecta a los elementos de enlace utilizados. Para empezar, consideremos el caso más simple en el que no se proporciona ningún tipo de seguridad.

- **netTcpBinding Binding**

  El atributo netTcpBinding utiliza, como es lógico, TCP como base subyacente. Como resultado, las clases BindingElement que están involucrados en este sentido. En este caso, tres elementos de unión están involucrados. La clase BinaryMessageEncoding BindingElement es responsable de convertir el mensaje en un formato binario para la transmisión. Esta clase no tiene propiedades más allá de las propiedades que se encuentran en la clase TransportBindingElement. El segundo elemento de unión de la pila es la clase TransactionFlowBinding Element. La única propiedad de interés en esta clase es TransactionProtocol. Esta propiedad específica el número de transacciones se hacen fluir desde el cliente hasta el servicio. El valor por

defecto es OleTransaction, pero WCF también es compatible con WS-Atomic Transactions.

- **NetNamedPipeBinding Binding**

  Ahora se empieza a ver algo común a medida que construye su pila de elementos de enlace. Hay tres elementos de unión en la pila estándar. El TransactionFlowBindingElement primero, segundo, BinaryMessage EncodingBinding-Element, y el tercer elemento es Nombrado PipeTransport BindingElement.

ANEXO 3 - CUESTIONARIO APLICADO

1. CUESTIONARIO PARA LA EVALUACIÓN A PARTICIPANTES

Propuesta de Componente

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

   • Totalmente apropiado _____

   • Parcialmente apropiado _____

   • Indiferente _____

   • No muy apropiado _____

   • Totalmente desapropiado _____

2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

   • Excelente _____

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente _____

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?

   • Totalmente de acuerdo _____

   • Parcialmente de acuerdo _____

   • Indiferente _____

   • No muy de acuerdo _____

   • Totalmente en desacuerdo_____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

   • Totalmente Eficiente _____

   • Parcialmente Eficiente _____

   • Indiferente _____

   • No muy eficiente _____

   • Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

   • Excelente _____

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

ANEXO 4 - CUESTIONARIOS RESUELTOS

1. CUESTIONARIO PARA LA EVALUACIÓN A PARTICIPANTES

Resultado N°1

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

   • Totalmente apropiado ____X__

   • Parcialmente apropiado _____

   • Indiferente _____

   • No muy apropiado _____

   • Totalmente desapropiado _____

2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

   • Excelente ___X___

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente ___X___

   • Bueno _____

   • Regular _____

• Malo _____

• Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?

• Totalmente de acuerdo ___X___

• Parcialmente de acuerdo _____

• Indiferente _____

• No muy de acuerdo _____

• Totalmente en desacuerdo _____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

• Totalmente Eficiente ___X___

• Parcialmente Eficiente _____

• Indiferente _____

• No muy eficiente _____

• Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

• Excelente _____

• Bueno ___X___

• Regular _____

• Malo _____

• Pésimo _____

Resultado N° 2

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

   • Totalmente apropiado _____

   • Parcialmente apropiado ___X___

   • Indiferente _____

   • No muy apropiado _____

   • Totalmente desapropiado _____

2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

   • Excelente ___X___

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente _____

   • Bueno ____X__

- Regular _____
- Malo _____
- Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?
   - Totalmente de acuerdo ___X___
   - Parcialmente de acuerdo _____
   - Indiferente _____
   - No muy de acuerdo _____
   - Totalmente en desacuerdo_____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?
   - Totalmente Eficiente ___X___
   - Parcialmente Eficiente _____
   - Indiferente _____
   - No muy eficiente _____
   - Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?
   - Excelente ____X__

- Bueno _____
- Regular _____
- Malo _____
- Pésimo _____

Resultado N°3

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)
   - Totalmente apropiado ___X___
   - Parcialmente apropiado _____
   - Indiferente _____
   - No muy apropiado _____
   - Totalmente desapropiado _____
2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?
   - Excelente ____X__
   - Bueno _____
   - Regular _____
   - Malo _____
   - Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente _____

   • Bueno ___X___

   • Regular _____

   • Malo _____

   • Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?

   • Totalmente de acuerdo ___X___

   • Parcialmente de acuerdo _____

   • Indiferente _____

   • No muy de acuerdo _____

   • Totalmente en desacuerdo_____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

   • Totalmente Eficiente ___X___

   • Parcialmente Eficiente _____

   • Indiferente _____

   • No muy eficiente _____

   • Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

   • Excelente ____X__

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

Resultado N°4

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

   • Totalmente apropiado _____

   • Parcialmente apropiado __X____

   • Indiferente _____

   • No muy apropiado _____

   • Totalmente desapropiado _____

2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

   • Excelente ___X___

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente _____

   • Bueno ___X___

   • Regular _____

   • Malo _____

   • Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?

   • Totalmente de acuerdo ____X__

   • Parcialmente de acuerdo _____

   • Indiferente _____

   • No muy de acuerdo _____

   • Totalmente en desacuerdo_____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

   • Totalmente Eficiente ____X__

   • Parcialmente Eficiente _____

   • Indiferente _____

   • No muy eficiente _____

   • Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

   • Excelente ____X__

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

Resultado N°5

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

   • Totalmente apropiado ___X___

   • Parcialmente apropiado _____

   • Indiferente _____

   • No muy apropiado _____

   • Totalmente desapropiado _____

2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

   • Excelente _____

   • Bueno ____X__

   • Regular _____

   • Malo _____

   • Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente ____X__

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?

   • Totalmente de acuerdo ___X___

   • Parcialmente de acuerdo _____

   • Indiferente _____

   • No muy de acuerdo _____

   • Totalmente en desacuerdo_____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

   • Totalmente Eficiente ___X___

   • Parcialmente Eficiente _____

   • Indiferente _____

   • No muy eficiente _____

   • Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

   • Excelente _____

   • Bueno ____X__

   • Regular _____

   • Malo _____

   • Pésimo _____

Resultado N°6

1. Que le parece el tiempo de implementación para la coordinación de un equipo informático (computadora)

   • Totalmente apropiado _____

   • Parcialmente apropiado _____

   • Indiferente ___X___

   • No muy apropiado _____

   • Totalmente desapropiado _____

2. ¿Qué le parece el tiempo de respuesta del Componente bajo la plataforma WCF?

   • Excelente _____

   • Bueno ____X___

   • Regular _____

   • Malo _____

   • Pésimo _____

3. ¿Qué le parece los dos modos de Utilidad del componente dentro de un equipo informático (Sistema - Sistema y Sistema - Usuario)?

   • Excelente ____X__

   • Bueno _____

   • Regular _____

   • Malo _____

   • Pésimo _____

4. ¿Está usted de acuerdo con la siguiente afirmación: “La propuesta de Componente hace posible la seguridad de distintos sistemas operativos Windows”?

   • Totalmente de acuerdo _____

   • Parcialmente de acuerdo __X_____

   • Indiferente _____

   • No muy de acuerdo _____

   • Totalmente en desacuerdo_____

5. ¿Le parece eficiente el uso de la Base de datos BD Posgresql 8.4 y XML para la elaboración de acciones en cambios en la configuración del Componente?

   • Totalmente Eficiente _____

   • Parcialmente Eficiente ____X__

   • Indiferente _____

   • No muy eficiente _____

   • Totalmente Ineficiente _____

6. ¿Le parece sencillo el uso de administración del programa clipboard para el bloqueo de archivos las actividades con un usuario?

   • Excelente _____

   • Bueno ___X___

   • Regular _____

   • Malo _____

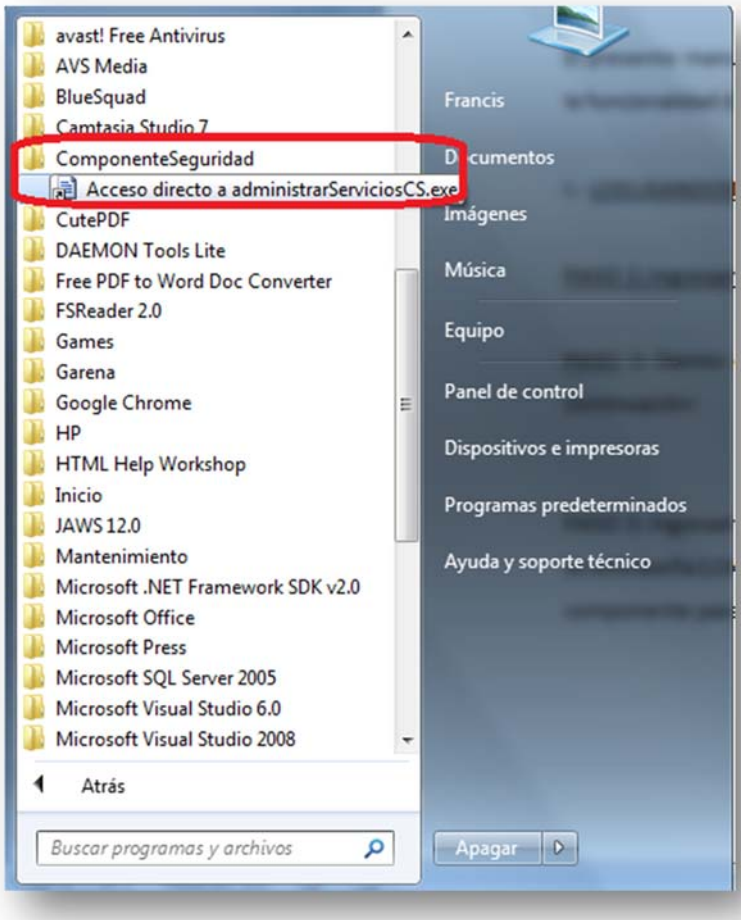
   • Pésimo _____

ANEXO 5 – MANUAL DE USUARIO

El presente manual de usuario básico servirá como herramienta de ayuda al usuario final para describir la funcionalidad del componente y explicar paso a paso como administrarlo.

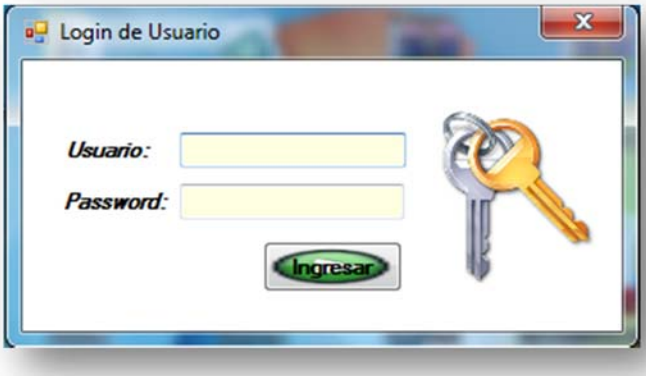
I.- Logueando al sistema de administración del componente

PASO 1: Ingresamos al **Menú Inicio->ComponenteSeguridad->administrarServiciosCS.exe**, luego de esto aparecerá un icono en la barra de notificación.

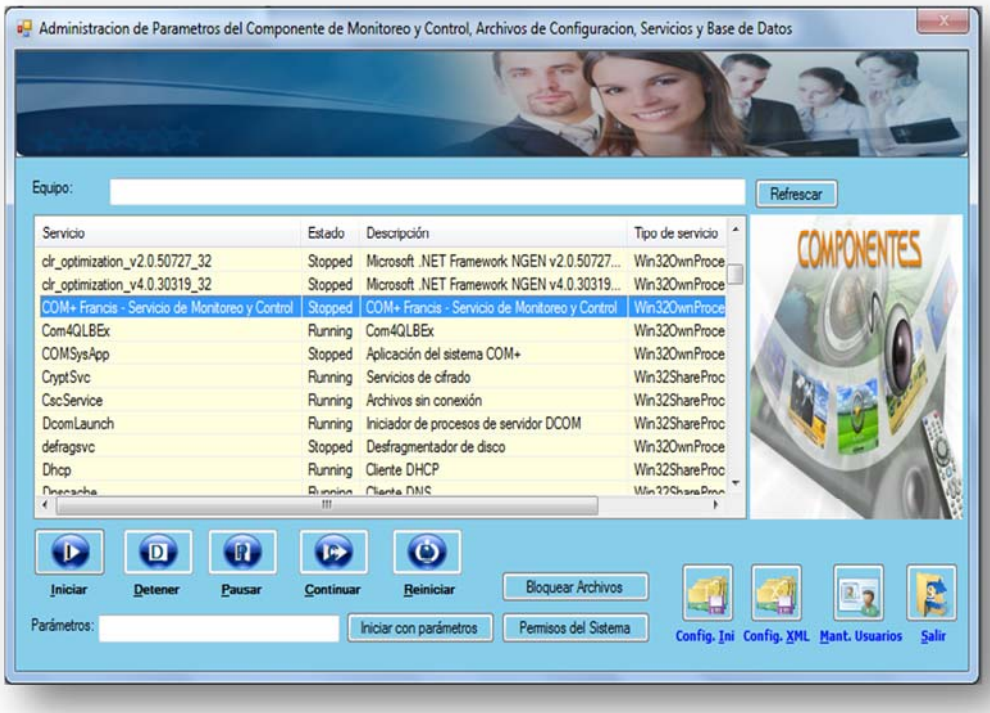


PASO 2: Damos doble clic al icono de la barra de iconos de notificación como se muestra a continuación:

PASO 3: Ingresamos el Usuario y Password respectivo: Para nuestro caso el usuario seria francis y la contraseña 12345, si el logueó se hizo correctamente se habrá activado también internamente el componente para base de datos encargado de controlar a los usuarios con perfil administrador.

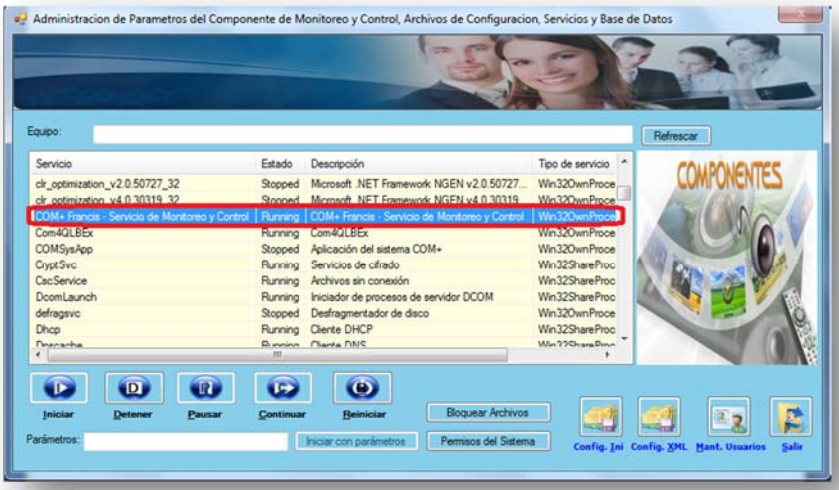


PASO 4: Una vez logueados se nos presentara la pantalla de administración del componente.

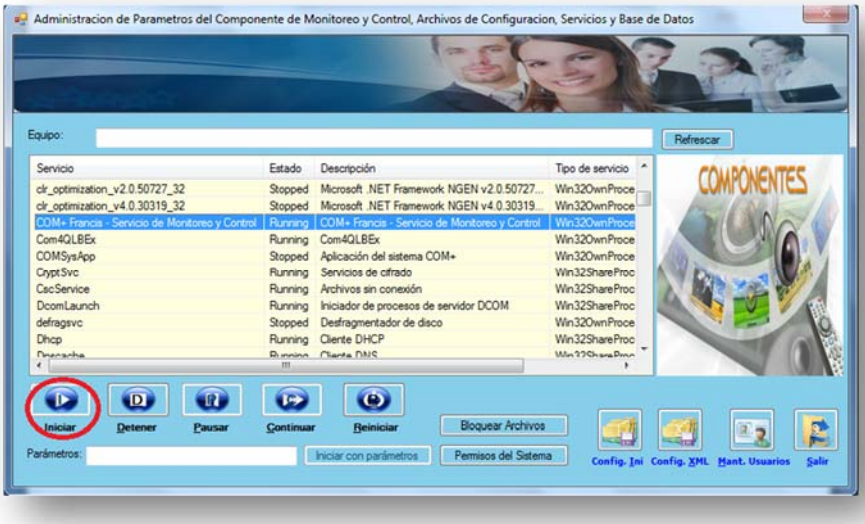


II.- <u>Inicializando el servicio y la Interfaz de Control de Tiempo del Componente</u>

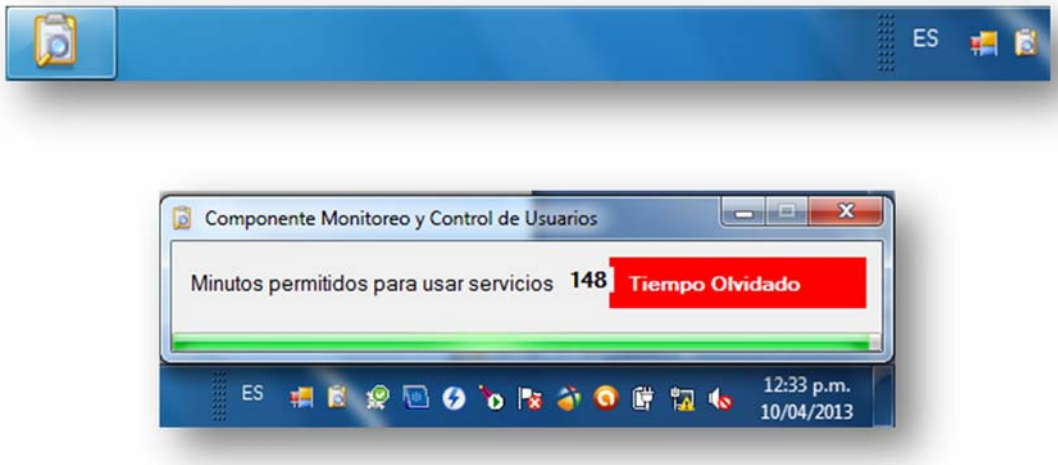
<u>PASO 1:</u> Buscamos y seleccionamos nuestro servicio que se llama "COM+ Francis – Servicio de Monitoreo y Control" en la grilla de servicios que se nos muestra y que es la misma del componente de servicios del sistema.



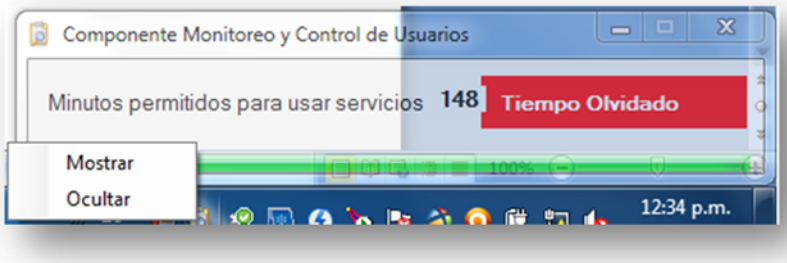
<u>PASO 2:</u> Damos un doble clic al botón Iniciar, para que el servicio cree el host servidor para el cliente (usuario logueado en el sistema).



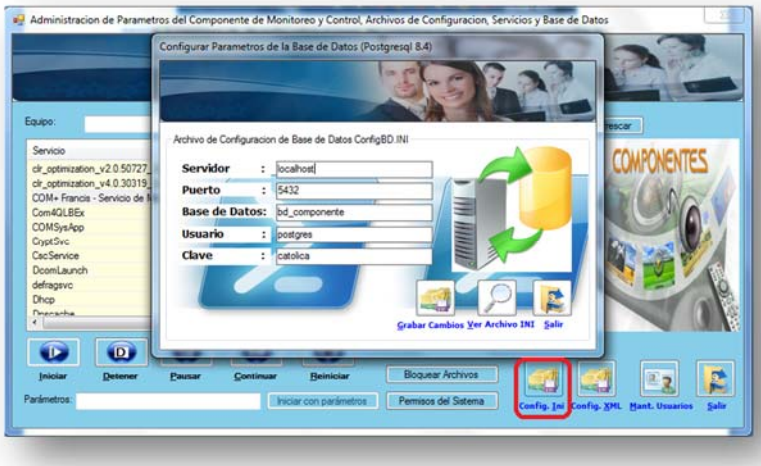
PASO 3: Una vez que hemos inicializado el servicio se activara el componente para Monitoreo y Control en la barra de herramientas y también en la barra de iconos de notificación además aparecerá la interfaz de control de tiempo del componente.



PASO 4: Si hacemos clic derecho sobre el icono del componente podremos mostrar/ocultar la interfaz de control de tiempo.



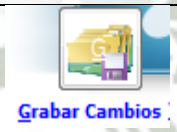
III.- Configurar Parámetros de la Base de Datos (archivo ConfigBD.ini)
PASO 1: Damos clic en el botón Config.Ini y nos aparecerá la pantalla de configuración.



PASO 2: Tenemos los parámetros para la configuración de la Base de datos que son:

- **Servidor:** nombre del servidor de la base de datos (localhost para postgresql).
- **Puerto:** Puerto de la base de datos (5432 por defecto).
- **Base de Datos:** nombre de base de datos.
- **Usuario:** postgres.
- **Clave:** católica.

Esos son los datos que podremos modificar de acuerdo a la configuración que mejor nos parezca conveniente, para grabar los cambios usaremos el botón Grabar Cambios y para ver el archivo original usaremos el botón Ver Archivo INI.

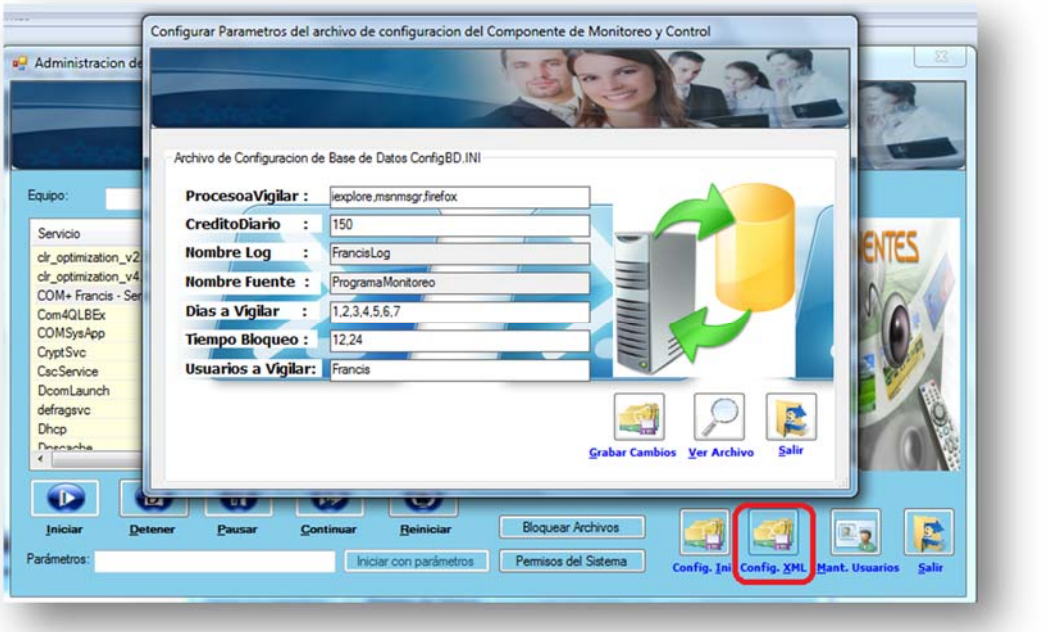

Botón para Grabar Cambios

Botón para ver archivo de configuración


Botón para Salir de la pantalla

III.- Configurar Parámetros del componente

PASO 1: Damos clic en el botón Config XML y nos aparecerá la pantalla de configuración.



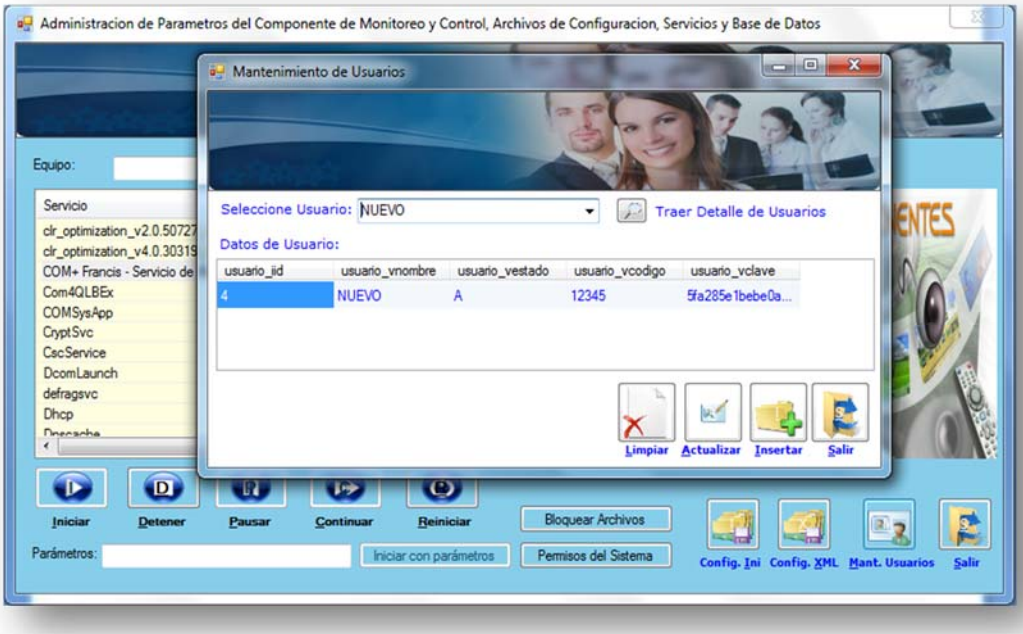
PASO 2: Tenemos los parámetros para la configuración de la Base de datos que son:

- **ProcesoaVigilar:** Procesos a monitorear (ejecutables calc, explore, firefox).
- **Credito diario:** Numero de minutos para usar los procesos a vigilar.
- **NombreLog:** Nombre de log de sucesos (solo lectura).
- **NombreFuente:** Nombre de la fuente de log de sucesos (solo lectura).
- **Dias a Vigilar:** Días de monitoreo (Lunes=1, Martes=2, Miércoles = 3…Domingo = 7).
- **Tiempo Bloqueo:** Rango de tiempo en formato de 24 horas en que no se puede usar ninguno de los procesos a vigilar (1, 16, 18, 24 bloqueado desde la 1pm hasta las 4pm y luego desde las 6pm a las 12pm horas), van separados por comas pero se toman los valores de par en par.
- **Usuarios a Vigilar:** Lista de nombres de usuarios que se loguean a la maquina, van separados por comas.

Para grabar los cambios usaremos el botón Grabar Cambios y para ver el archivo original usaremos el botón Ver Archivo INI.

III.- Mantenimiento de Usuarios del Sistema

PASO 1: Damos clic en el botón Mantenimiento de usuarios y nos aparecerá la pantalla de administración de usuarios.



En esta pantalla tendremos los siguientes botones que son:


Botón para Insertar un nuevo usuario.

Botón para Actualizar los datos de un usuario.

Botón para Limpiar los datos de la grilla de usuarios.

Botón para Salir de la pantalla.

IV.- <u>Bloqueando/Desbloqueando archivos del Sistema</u>

<u>PASO 1:</u> Damos clic al botón Bloquear/Archivos.

<u>PASO 2:</u> En la pantalla tendremos el icono de play/pause que nos permitirá bloquear/desbloquear el copiado de archivos en el sistema, haciendo que no se pueda copiar ningún archivo de cualquier formato.

V.- <u>Cambiando directivas de grupo en el sistema</u>

<u>PASO 1:</u> Damos clic al botón Permisos del sistema.

<u>PASO 2:</u> En la pantalla tendremos la posibilidad de cambiar las diferentes directivas de grupo local del sistema, para dar o quitar permisos a los usuarios de acuerdo al perfil con el que cuenten.