TABLA DE CONTENIDO

LISTA DE CUADROS Y GRAFICOS

1 INTRODUCCIÓN

1.1 ANTECEDENTES

La arquitectura de los sistemas ha evolucionado con el tiempo, acompañado de cambios tecnológicos y de paradigmas en su diseño que aun se encuentran vigentes en muchas instituciones.

Entre las tecnologías más conocidas e iníciales se conocen las centralizadas, Cliente/Servidor, aplicaciones de n capas, y la que hoy en día ha revolucionado los métodos de trabajo, que son las aplicaciones orientadas a servicios que realmente es altamente desacoplada.

El SOA es un estilo arquitectónico de software, no una tecnología determinada. Como estilo arquitectónico, define a los servicios como unidades de partición para dar respuesta a los requerimientos del negocio dentro de una organización brindando una metodología y un marco de trabajo para mapear procesos de negocio a componentes de software en forma de servicios. Las aplicaciones ya no son más elementos aislados sino parte integrante del negocio.

Tecnológicamente, estos servicios tienen la particularidad de ser:

- ✓ Encapsulados
- ✓ De bajo acoplamiento
- ✓ Reutilizables
- ✓ Sin información de estado (stateless)
- ✓ Localizables
- ✓ Independientes de una plataforma o lenguaje

La evolución de las arquitecturas ha tenido pasos agigantados en su accionar en los negocios y aplicaciones, aportando con funcionalidades cada vez mas optimas para el rendimiento de cada una de sus arquitecturas en su debido momento.

Las arquitecturas más comunes y relevantes que se tiene en la actualidad, se mantienen vigentes principalmente por la monotonía de trabajo y a la vez el desconocimiento de nuevas tendencias de arquitectura que aportarían con el crecimiento tecnológico de la empresa y la aplicación de nuevos horizontes para su clientes.

En las arquitecturas que se mantienen vigentes se puede detallar las siguientes:

1.1.1 Centralizadas (Monolíticas)

También conocidas como "Mainframe Architecture que viene hacer toda la inteligencia del procesamiento de negocio e interfaces, se centraliza en un host principal.

Con lo que los usuarios interactúan a través de terminales tontas, es decir que esta captura la iteración del teclado, se envía la información al host, este lo procesa y se retorna a la pantalla a la terminal, esta iteración se la puede realizar en terminales Unix, Dos/Windows u otras.

Este modelo fue y es todavía muy exitosa en ámbitos donde se realiza procesamientos intensivos de datos, pero a la par, sus limitantes se enfocan al no soportar interfaces graficas y el difícil acceso a repositorios de datos distribuidos.

1.1.2 Intercambio de archivos

Conocido como "file sharing architectures", que tiene su punto de partida con la popularización de las redes de área local.

En esta, el servidor (o el PC) descarga en su espacio local archivos que se encuentran en el servidor, el procesamiento solicitado por el cliente, es realizado en el espacio de procesamiento del mismo.

Esto funciona cuando el volumen compartido es bajo, la contención de actualizaciones es baja y el volumen de transferencia es también bajo.

Un ejemplo palpable de esto es un sistema desarrollado en Visual Basic y Access, donde los clientes acceden a través de los programas (almacenado en el servidor en un disco compartido) a una base de datos/archivos indexados compartida.

A medida que el uso concurrente aumenta, disminuye la escalabilidad de este tipo de soluciones ya que el limitante principal es la decenas de usuarios van acceder a esta carpeta a realizar una acción.

1.1.3 Cliente/Servidor (2 niveles)

Como una evolución de las anteriores, surge este modelo donde el servidor de archivos es reemplazado por una base de datos (relacional) en donde se tienen dos partes claramente diferenciadas, el cliente y el servidor.

El cliente emite consultas, las cuales son respondidas por un servidor y en este caso, se recibe solo la respuesta, en vez de un archivo compartido ya que el procesamiento es dividido entre el cliente y el servidor, balanceando la carga entre ambo.

1.1.4 En múltiples capas (3 o más niveles)

También conocidas como arquitecturas multi-capa (que no están limitadas a 3), Una capa intermedia se añade entre el ambiente del cliente y el servidor de base de datos, que puede implementarse de múltiples formas, pudiendo ser:

- ✓ Servidores de aplicación.
- ✓ Monitores transaccionales.
- ✓ Sistemas de mensajería.

Con esta capa intermedia se puede facilitar el encolado de peticiones, ejecutar aplicaciones y lógica de negocio en forma de programas, permitiendo aumentar e performance en sistemas con grandes volúmenes de uso.

Una limitación importante de este tipo de arquitecturas, es su complejidad intrínseca ya que el proceso de diseño, desarrollo y especialmente testeo de este tipo de soluciones es muy complejo.

1.2 PLANTEAMIENTO DEL PROBLEMA

Banco Promerica, empresa financiera cuyo producto principal es el servicio al cliente y la banca tiene sus propios productos que fueron diseñados en arquitecturas de tres capas en su debido momento, lo que hoy en la actualidad por su constante crecimiento y demanda de productos por clientes y la constante competencia con las demás instituciones financieras en prestación de servicios bancarios, se ha visto la necesidad buscar nuevas tendencias de arquitectura de los sistemas actuales y nuevos, por lo que se ve la necesidad de desarrollar un MOTOR TRANSACCIONAL EN WINDOWS, que permita la prestación de servicios para los sistemas actuales así como la seguridad de acceso y uso de transacciones de cada sistema que tenga la responsabilidad de realizar una acción con el servicio proporcionado.

De lo anteriormente mencionado surgen las siguientes interrogantes:

¿Es posible diseñar un motor transaccional que pueda brindar servicios para interactuar con sistemas?

¿Ayudará el sistema desarrollado a cambiar el concepto de la arquitectura actual de trabajo de los sistemas?

¿El nuevo software será la herramienta que garantice la ejecución de transacciones y la seguridad de acceso a datos?

1.3 Sistematización

1.3.1 Diagnóstico.

Dentro del Banco Promerica, al no contar con una arquitectura optimizada para los sistemas que se encuentran activos, hace que el proceso de desarrollo e implementación de sistemas, utilice componentes ya desarrollados que cumple las funciones de interconexión entre las interfaces desarrolladas y las capas de negocio

Este proceso se repite constantemente, cada vez que la empresa necesita brindar un servicio a sus clientes, ocasionando que se rediseñe los componentes y se adapté para el nuevo requerimiento del servicio a prestar.

Esto ocasiona retrasos en la entrega de los productos y falta de seguridad en el acceso y la ejecución de transacciones que se encuentran ligados con actividades bancarias.

A continuación el flujo que se tiene actualmente para el desarrollo de un sistema nuevo (producto).

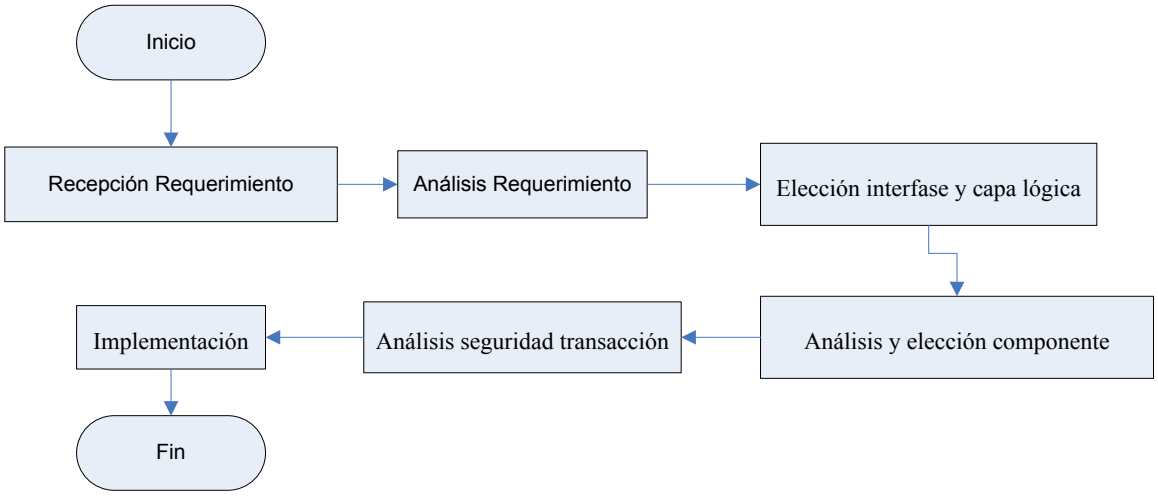


Grafico 1-1 Flujo desarrollo sistema

De igual manera en la tabla se refleja el tiempo de desarrollo invertido en la creación de un nuevo producto.

Procesos	Tiempo en horas	%
Recepción requerimiento	1	0.74
Análisis requerimiento	3	2.10
Elección interface /capa lógica	14	9.79
Análisis componentes	14	9.79
Análisis seguridad	16	10.49
Implementación	96	67.13
	143	100.00

Cuadro 1-1 Tiempo invertido en desarrollo nuevo producto

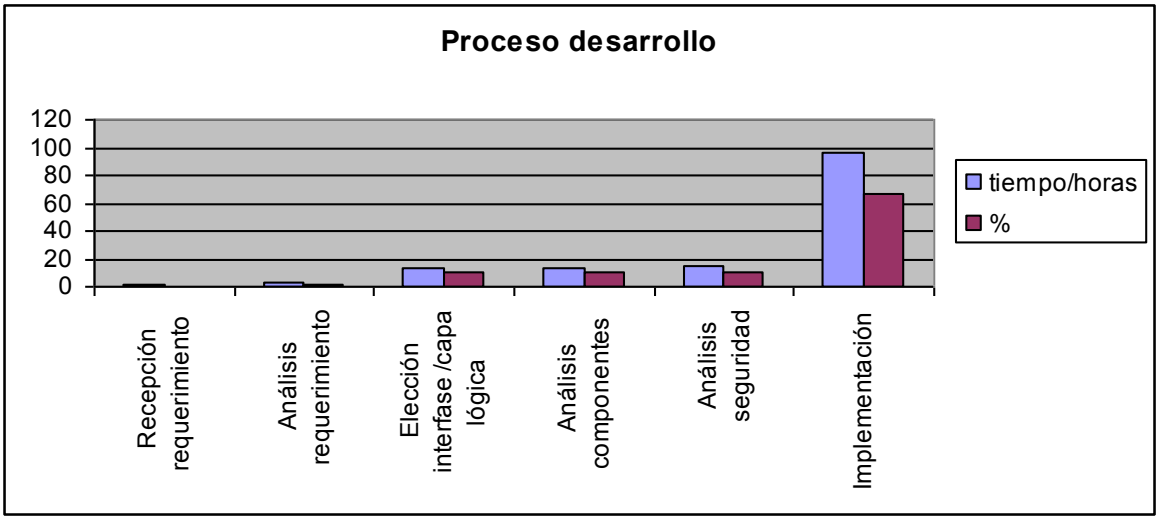


Grafico 1-2 Proceso de desarrollo

En los procesos de desarrollo definidos en la **Cuadro1-1,** y tal como muestra la Grafico 1-2, se puede observar que el gasto en tiempo de análisis de componentes y de seguridad es alto, ya que las horas dedicadas en su implementación puede ser enfocado en otras actividades o lo que es más importante, que su salida a producción puede ser relativamente corto

Con este antecedente, y el no contar con un sistema que permita la elección de un servicio para su pronto consumo en la parte de desarrollo, y el hecho de no contar con un sistema de seguridad para las transacciones hace que se presente los siguientes problemas:

- ✓ El análisis e implementación del componente, así como su seguridad, provoca un gasto de horas y dedicación en su implementación, el cual podría ser canalizado en otra actividad de desarrollo.
- ✓ El análisis e implementación de un sistema nuevo, tiene su demora en desarrollarse.
- ✓ Dificultad para seleccionar un protocolo de comunicación.
- ✓ No existe un control de acceso a los aplicativos.
- ✓ No existe un control de ejecución de transacciones.
- ✓ No existen reportes de actividades de transacciones:
  - ✓ Hora ejecución.
  - ✓ Métodos invocados.
- ✓ No existe un modulo de seguridad debidamente implementado para manejo de transacciones.

1.3.2 Pronóstico.

Si se continúa con el actual proceso de desarrollo, en el que la implementación de un aplicativo para la empresa, conlleva un gasto de tiempo de análisis de componentes, y seguridad, no se va poder agilizar la prestación de productos de negocios hacia clientes propios del banco.

El tiempo dedicado en la implementación del componente, así como la seguridad de uso y ejecución, ocasionara la demora en salida del producto a producción y un gasto de tiempo y recurso para la empresa.

De no optimizarse la reutilización de los servicios para diferentes aplicativos, ocasionara que el tiempo de desarrollo sea más extenso.

Al no contar con un control de ejecución de transacciones, ocasionará en un momento dado tener resultados de acciones en la base de datos, que no lo haría un aplicativo normal en las horas establecidas de trabajo.

1.3.3 Control del Pronóstico

Para la construcción de sistemas distribuidos Microsoft tiene ya distintas tecnologías, cada una con sus ventajas y desventajas, con la **implementación del motor transaccional** se pretende incorporar una de ellas y "esconderlas" detrás de un modelo de servicio y seguridad.

Esto significa que podrá evitar la decisión clave de escoger el tipo de transporte al momento de diseñar la arquitectura de una solución y el poder tener la oportunidad de usar una de ellas sin necesidad de conocer mucho de la tecnología y disponer de la capacidad de tener sus componentes por separado para su futuro mantenimiento y reutilización de los mismos.

Esto se lo realizara con la ayuda de un motor transaccional, diseñado en WCF, que podrá usar una de estas tecnologías (uso compartido de puertos Net Tcp) en nuestros servicios, brindando la posibilidad de aprovechar los puntos fuertes de esta tecnología y combinarla con en el entorno de nuestras aplicaciones.

Entendiéndose el ahorro de horas dedicadas al dominio de cada tecnología en particular, ya que el desarrollador se enfocara en diseñar las interfaces y las capas lógicas

Aprovechar el comportamiento de los servicios que provee Windows Comunication Foundation, se puede extender las opciones de trabajo y por medio del uso de la tecnología, tener la posibilidad de extender, según se desee, el comportamiento de un servicio y una operación que ofrece un servicio en un mismo puerto, o técnicamente conocido como Endpoint.

Entendiéndose que un Endpoint representa un punto de acceso donde un servidor recibe clientes, y lo ata a una regla nemotécnica conocida como "ABC", en el que se indica en donde está el servicio, como se accede, y que métodos y operaciones expone.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar un monitor de transacciones para aplicaciones distribuidas sobre plataforma Windows que brinde servicios a clientes (sistemas), por medio del protocolo TCP, compartiendo puertos net.tcp en varios procesos de usuario, y disponer de un sistema de seguridad para la configuración de los principios básicos de las transacciones disponibles en el servicio, tales como horario de acceso, uso de aplicativos, y transacciones permitidas.

1.4.2 Objetivos específicos

Como objetivos específicos se busca:

- ✓ Desarrollar una aplicación en consola que hará las funciones de Motor Trasnacional.
- ✓ Implementar un diseño de carga dinámica de librerías para el Motor Transaccional, que serán las encargadas de describir las operaciones que ofrecen el servicio y las estructuras de datos manejadas por las operaciones del servicio
- ✓ Implementar un modelo de datos que permita almacenar las transacciones que pase por el motor transaccional.
- ✓ Implementar un modelo de datos que permita almacenar la configuración de seguridad de las transacciones que pase por el motor transaccional.
- ✓ Implementar una aplicación que usando el motor transaccional desarrollado, permita administrar la seguridad del mismo por medio de la conexión del estándar Net TCP.

1.5 Justificación

1.5.1 Justificación teórica

El enfocarse en el uso de nueva tecnología y la unificación de API´s para sistemas distribuidos, es una motivación personal e intelectual, ya que la investigación y el desarrollo del mismo, permitirá conocer las tendencias de una arquitectura orientada a servicios (SOA).

De igual manera el poder trabajar con WCF, permitirá obtener conocimientos sobre aplicaciones distribuidas, así como aprovechar el mecanismo de seguridad que brinda a nivel de enlace, integración con infraestructuras de seguridad existentes, métodos de autentificación y la seguridad a nivel de transporte de mensaje.

1.5.2 Justificación Metodológica

El Sistema de Motor transaccional para el Banco Promerica, permitirá poner en manifiesto todos los conocimientos adquiridos durante la carrera y permitirá sentar las bases para otras aplicaciones que necesite la Empresa.

El Sistema de Motor transaccional, será realizado con la programación orientada a Objetos. Cómo proceso para desarrollar el Sistema de Motor Transaccional se utilizará el Proceso Unificado de Desarrollo, porque es un proceso que soporta técnicas de programación orientada a objetos y se basa en la relación entres las clases y los objetos por medio de la notación UML que es común.

El Proceso Unificado de Desarrollo puede verse como una metodología adaptable. Esto quiere decir que se puede modificar para adaptarlo al sistema concreto que se va a desarrollar en cada momento ya que es iterativo, centrado en la arquitectura y dirigido por casos de uso y riesgos. Por definición el Proceso Unificado de Desarrollo utiliza buenas prácticas de desarrollo, siendo adaptable a un amplio rango de aplicaciones y sistemas.

Este proceso no sólo considera aspectos de desarrollo de un sistema, sino también los de gestión del mismo.

El Proceso Unificado es un marco de desarrollo compuesto por las siguientes fases:

- ✓ Inicio
- ✓ Elaboración
- ✓ Construcción
- ✓ Transición

Donde cada una de las fases se divide en una serie de iteraciones que ofrecen como resultado un incremento del producto desarrollado, que añade o mejora las funcionalidades del sistema en desarrollo.

Como herramienta para el adecuado modelamiento de datos, el Sistema de Motor Transaccional utilizará el Lenguaje de Modelamiento Unificado (UML).

El lenguaje unificado de diagrama o notación UML sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o como diseñar el sistema, sino que simplemente ayuda a visualizar el diseño y hacerlo más accesible para otros.

1.5.3 Justificación práctica

El aportar con una herramienta para el control de transacciones y prestación de servicio (uso de compartido de puertos Net. Tcp) sin necesidad de rediseñar la estructura arquitectónica del sistema actual, permitiendo inicialmente al desarrollador enfocarse en diseñar una interface de presentación del sistemas solicitado, definir la capa de negocio y obtener una aplicación distribuida sin tener que aprender el modelo de programación Net Remoting.

A la par no se puede olvidar que el disponer de una arquitectura independiente de la plataforma, del lenguaje, de los objetos y del mecanismo de llamada, permite a la institución competir en la prestación de servicios propios de la institución de manera ágil y competitiva.

1.6 Alcance

La implementación del sistema propuesto tendrá como base de trabajo en sistema operativo Windows, permitiendo almacenar sus peticiones y respuestas del motor transaccional en una base de datos Sql2005.

El motor transaccional permitirá la conexión usando el estándar y el protocolo de red basado en TCP (Net. Tcp)

El uso de los servicios se regirá a un modelo de seguridad basado en fecha, horario, usuario, aplicaciones, fortalecido por un sistema que permita realizar la configuración y programación de estos parámetros.

Disponer de un aplicativo en ambiente Windows, que será el encargado de consumir exclusivamente el servicio por el protocolo TCP (Net. Tcp), para la demostración de uso de este servicio.

1.7 Estudios de Factibilidad

1.7.1 Factibilidad Técnica

Se cuenta con los recursos necesarios tanto de hardware y software para la implementación y diseño del proyecto con esto es posible llevar a buen término el desarrollo del proyecto motor transaccional para Windows.

Este estudio se enfoca al Hardware y Software disponible:

Hardware

Lo que corresponde a hardware, específicamente para el servidor de base de datos donde se va almacenar la base de datos de seguridad, se dispone de:

- ✓ Procesador Intel 2 GHz
- ✓ 4 GB de Memoria RAM
- ✓ Disco Duro de 600 GB.
- ✓ Tarjetas de Red.
- ✓ Tarjeta de Vídeo.
- ✓ Monitor SVGA.
- ✓ Teclado.
- ✓ Mouse.
- ✓ Protección UPS.

De igual manera se dispone de un servidor donde se instalara el aplicativo que será el encargado de levantar los servicios que se requieran trabajar.

- ✓ Procesador Intel 2 GHz
- ✓ 4 GB de Memoria RAM
- ✓ Disco Duro de 300 GB.
- ✓ Tarjetas de Red.
- ✓ Tarjeta de Vídeo.
- ✓ Monitor SVGA.
- ✓ Teclado.
- ✓ Mouse.
- ✓ Protección UPS.

De acuerdo al Hardware existente en cuanto al servidor de base de datos y de aplicación, la institución no requiere realizar inversión adicional para la adquisición de un nuevo servidor, ni tampoco para realizar actualizaciones de equipos existentes, ya que el mismo

cumple con los requerimientos establecidos para el desarrollo e implementación del motor transaccional en Windows

A continuación se detalla el hardware que dispone actualmente la institución.

HARDWARE DISPONIBLE	
Cantidad	Descripción
6	Servidores en matriz. Pentium IV Intel, 2 GB Memoria RAM, Disco Duro 600 GB, Tarjetas de Video, red, Monitor, teclado, mouse, UPS.
20	Estaciones de trabajo área administrativa. Pentium IV Intel Pentium , 1 gb Memoria RAM, Disco Duro 80 GB, Tarjetas de Video, red, Monitor, teclado, mouse, regulador de voltaje
50	Estaciones de trabajo agencias. Pentium IV Intel Pentium , 1 gb Memoria RAM, Disco Duro 80 GB, Tarjetas de Video, red, Monitor, teclado, mouse, regulador de voltaje
1	Red Ethernet Topología Estrella
10	Switchs 16 puertos
10	Switchs 10 puertos
3	Rack
8	Impresora HP LaserJet 9000 PCL 6

Cuadro 1-2 Hardware Disponible

Cada agencia cuenta con un enlace dedicado con la matriz y las transacciones y procesos se los realiza en línea, con lo que el motor transaccional utilizara esta infraestructura de conectividad para brindar los servicios de recepción de peticiones y entrega de respuestas al aplicativo que lo solicite.

Todas las estaciones de trabajo se conectan a los servidores a través del enlace dedicado que lo provee Andinadatos que es la conexión principal y a la vez se cuenta con un enlace de contingencia con GlobaCrossing.

Esta conectividad permite que los equipos instalados tanto en las agencias como en el área administrativa de la institución, puedan utilizar el sistema de motor transaccional para Windows.

Software

En cuanto al software, la institución dispone de todas las aplicaciones que permitirán el desarrollo e implementación del Sistema de Motor Transaccional en Windows, por lo tanto no es necesario que se realice inversión alguna para la adquisición de las aplicaciones adicionales.

Actualmente el servidor principal utiliza el sistema operativo Windows 2003, así como el servidor de base de datos se encuentra actualmente trabajando con SQL 2005, como plataforma de desarrollo se utilizará Punto Net 2008 y las estaciones de trabajo, operan bajo Sistema Operativo Windows.

SOFTWARE DISPONIBLE	
Cantidad	Descripción
4	Sistema Operativo Microsoft Windows Server 2003
50	Herramientas de escritorio Office 2003
50	Antivirus McAfee
1	SQL 2005
1	Punto Net 2008
1	Dev Express 2008

Cuadro 1-3 Software disponible

Base de datos

MATRIZ COMPARATIVA REPOSITORIO DE DATOS							
Características	**%**	**SQL**		**ORACLE**		**MYSQL**	
		Valor	**Total**	**Valor**	**Total**	**Valor**	**Total**
Integración con Microsoft	20	4	0.80	2	0.40	3	0.60
Soporte .Net Framework	20	4	0.80	2	0.40	1	0.20
Importación/exportación	20	4	0.80	4	0.80	2	0.40
Licenciamiento	25	1	0.25	1	0.25	4	1.00
Manejo desencadenadores	15	4	0.60	3	0.25	1	0.15
Valores	**100%**		**3.25**		**2.30**		**2.55**

Cuadro 1-4 Matriz comparativa repositorio de datos

En la matriz comparativa se puede observar que la integración de sql server 2005 con Framework .net y sus cualidades de configuración, importación/exportación de datos lo hace un repositorio de datos adecuado para el almacenamiento de las transacciones y configuraciones del motor transaccional, ya que al trabajar sobre la plataforma Microsoft, una evolución de cualquiera de estos componentes implicaría una actualización de Framework y no de programación.

Lenguaje de programación

MATRIZ COMPARATIVA PLATAFORMA DE DESARROLLO							
Características	**%**	**.Net**		**Java**		**MYSQL**	
		Valor	**Total**	**Valor**	**Total**	**Valor**	**Total**
Compatibilidad Microsoft	25.00%	4	1.00	3	0.75	3	0.75
Desarrollo de aplicaciones Web	5.00%	4	0.20	2	0.10	3	0.15
Desarrollo de aplicaciones distribuidas	22.50%	4	0.90	2	0.45	2	0.45
Gestión de memoria	5.00%	4	0.20	4	0.20	1	0.05
Manejo de ensamblados	22.50%	4	0.90	4	0.90	2	0.45
Independencia del lenguaje	5.00%	4	0.20	1	0.05	1	0.05
Licenciamiento	15%	1	0.15	4	0.60	4	0.60
Valores	**100%**		**3.55**		**3.05**		**2.53**

Cuadro 1-5 Matriz comparativa plataforma de desarrollo

Al ser un aplicativo que va trabajar completamente en ambiente Windows se opto por un ambiente de desarrollo 100% compatible para la plataforma seleccionada, en donde se puede observar que Visual Studio .net es un ambiente compatible y natural para desarrollar aplicativos sobre Windows, en donde se identifica claramente las características básicas para el desarrollo de aplicaciones, como es su compatibilidad con Microsoft, desarrollo web y la factibilidad para diseñar aplicaciones distribuidas.

Framework´s

MATRIZ COMPARATIVA FRAMEWORK DE DESARROLLO							
Características	**%**	**Framework .NET 3.5**		**Spring Framework(java**		**Framework Prado(PHP)**	
		Valor	**Total**	**Valor**	**Total**	**Valor**	**Total**
Múltiple lenguaje de programación	25.00%	4	1.00	3	0.75	3	0.75
Biblioteca de clase	5.00%	4	0.20	2	0.10	3	0.15
Desarrollo de aplicaciones distribuidas	22.50%	4	0.90	2	0.45	2	0.45
Gestión de memoria	5.00%	4	0.20	4	0.20	1	0.05
Manejo de ensamblados	22.50%	4	0.90	4	0.90	2	0.45
Integración con Microsoft Windows Installer	5.00%	4	0.20	1	0.05	1	0.05
Valores	**100%**		**3.40**		**2.00**		**1.6**

Cuadro 1-6 Matriz comparativa plataforma de desarrollo

De acuerdo al análisis de factibilidad técnica que dispone la institución, se puede determinar que en los actuales momentos cuenta con la infraestructura tecnológica (hardware y Software) para el desarrollo en implementación del sistema de motor transaccional en Windows.

Es decir que el Framework de desarrollo es el adecuado para la implementación del sistema ya que este cuenta actualmente con las características principales de acoplamiento de ensamblados (Dll's), compatibilidad con Windows Installer para elaborar un instalador si así lo desee, y una cualidad que el desarrollador podrá elaborar el aplicativo o ensamblado en el lenguaje que considere más adecuado para su implantación, sin preocuparse por las librerías ya que estas están embebidas en el Framework.

1.7.2 Factibilidad Operativa

En la actualidad existen diferentes tipos de sistemas integradores de aplicativos que realizan la función de receptar y procesar solicitudes de transacciones, a continuación una breve descripción de dos de ellas.

BIZTALK SERVER 2006

Conocido como un servidor de integración de procesos empresariales, BizTalk Server permite conectar distintos programas de software para generar un macro-sistema de procesos, personas e información y de ese modo maximizar la eficiencia de la organización.

BizTalk Server para gestionar esta integración de sistemas diversos, se basa en un modulo de Mensajería que es el encargado de efectuar la comunicación entre diferentes aplicaciones, y que tiene la siguiente características.

- ✓ BizTalk Server recibe los mensajes por medio de un Adaptador de Recibimientos. Hay un adaptador para cada mecanismo de comunicación, por ejemplo, adaptador de Web Services o adaptador para leer de un archivo de texto plano
- ✓ El mensaje entrante es procesado por el Conducto de Conversión de Recibimientos. El conducto puede contener muchos sub-componentes para convertir el mensaje o bien para validar firmas
- ✓ El mensaje es enviado a una base de datos, definido como Caja de Mensajes, la cual se implementa por medio Microsoft SQL Server
- ✓ El mensaje se deriva al módulo Orquestación que ejecuta las acciones requeridas por el proceso de negocios
- ✓ Comúnmente el módulo de Orquestación devuelve otro mensaje que se graba también en la Caja de Mensajes

- ✓ BizTalk Server aplica un Conducto de Conversión de Envío si es necesario convertir el formato o firmar el archivo
- ✓ El mensaje ya convertido y firmado se envía por el Adaptador de Envíos que usa un mecanismo apropiado de comunicación para interactuar con la aplicación destino

ENTERPRISE SERVICE BUS (ESB).

Este producto es galardonado y líder en la industria ya que facilita la comunicación de los servicios Web, aplicaciones comercializadas y personalizadas y sistemas heredados a través de una Arquitectura orientada a servicios (SOA). WebMethods ESB transforma servicios técnicos muy precisos y diferentes funcionalidades en servicios de negocio de alta rentabilidad que confieren mayor agilidad a sus TI.

Un bus de servicios de empresa (ESB) consiste en un combinado de arquitectura de software que proporciona servicios fundamentales para arquitecturas complejas a través de un sistema de mensajes (el bus) basado en las normas y que responde a eventos. Los desarrolladores normalmente implementan un ESB utilizando tecnologías de productos de infraestructura de middleware que se basan en normas reconocidas.

Un ESB generalmente proporciona una capa de abstracción construida sobre una implementación de un sistema de mensajes de empresa que permita a los expertos en integración explotar el valor del envío de mensajes sin tener que escribir código. Al contrario que sucede con la clásica integración de aplicaciones de empresa (IAE) que se basa en una pila monolítica sobre una arquitectura hub and spoke, un bus de servicio de empresa se construye sobre unas funciones base que se dividen en sus partes constituyentes, con una implantación distribuida cuando se hace necesario, de modo que trabajen armoniosamente según la demanda.

Punto de vista operativo

Con estos antecedentes de los sistemas, que su cualidad principal es la prestación de servicios y su reutilización para ser consumidos por aplicativos, se ha considerado el desarrollo de un sistema, que aporte con la característica principal de cada uno de los sistemas, ya que el hecho de contar con un sistema que se adapte directamente al requerimiento del negocio, y el disponer del código fuente para futuras rediseños del motor transaccional, lo hace una propuesta factible para la empresa.

Centrándonos en la característica principal de cada uno de estos productos, que es la capacidad de brindar servicios, el Motor Transaccional tendrá la facultad de brindar servicios en el protocolo TCP y compartir estos servicios en un mismo puerto, si así se lo considere.

Refiriéndonos al hospedaje en protocolo TCP, se orienta a mejorar el rendimiento con aplicativos de la misma empresa, ya que su transmisión binaria lo hace rápido, confiable, en el envío y recepción de paquetes entre las interfaces de usuario y las capas lógicas.

Considero que el impacto del nuevo sistema sobre el ambiente de desarrollo y el brindar a los operadores una herramienta para identificar el flujo de transacciones, será un cambio positivo y sin grandes trabas debido a los siguientes ítems.

- ✓ En primera instancia, la idea surge de una necesidad detectada por los profesionales de desarrollo que trabajan implementando los sistemas requeridos. Por lo cual, éste sistema se enfoca a resolver un problema concreto (unificar interfaces cliente con lógica negocio) y que fija un punto de partida a la resolución de los problemas por ellos planteado.

- ✓ Por otro lado, la implementación del mismo no representa un cambio radical en los procesos de desarrollo de un aplicativo, así como la capacitación del operador para la identificación de transacciones y configuración de los servicios.
- ✓ El sistema presentará una interfaz en consola y Windows (configuración de servicios), muy intuitiva que solo requerirá en concepto de conocimientos previos, estar familiarizado con una PC y la iteración con aplicativos en ambiente Windows, conceptos con los que, hoy en día, la gente está cada vez más en contacto tanto en el hogar como durante sus tareas laborales.

De todas formas, evaluando el personal que se verá afectado por el software notamos lo siguiente:

- ✓ En el caso del operador, este mantendrá la misma labor y vera que el cambio será mínimo, debido que el banco de por sí cuentan con distintos sistemas informáticos con características similares. Lo que facilita la adaptación al nuevo sistema y sus correspondientes opciones.
- ✓ Desde el punto de vista del desarrollador, se habla de personal capacitado, quien inclusive durante sus estudios debió interesarse por nuevas tecnologías de aplicaciones distribuidas. Tampoco se verá abrumado por este nuevo proyecto, y mucho menos teniendo en cuenta como se mencionó anteriormente que la idea surge de ellos, por lo tanto sentirán que sus palabras fueron escuchadas. Con esto último, se contempla la motivación que hace posible de manera más fácil la implementación de ésta nueva propuesta.

1.7.3 Factibilidad Económica

En el estudio de la factibilidad económica para el desarrollo del Sistema motor transaccional se determinan los recursos para desarrollar, implantar, y mantener en operación el sistema de motor transaccional.

Análisis Costos-Beneficios

En base a este análisis se puede hacer una comparación de los costos que conlleva el procedimiento, y los costos que tendrían la implantación del sistema de motor transaccional.

Basados en el estudio de factibilidad técnica, la empresa cuenta con las herramientas necesarias para la puesta en marcha del Sistema Informático, por lo que no es necesario una inversión inicial.

A continuación se presenta un resumen de los costos que conlleva actualmente los costos de operación.

Costos del Sistema.

El diseño de un motor transaccional para un sistema con múltiples clientes en Windows Comunication Foundation, involucra los siguientes costos de manera aproximada.

Costos Generales

Al realizar un motor transaccional se lograría optimizar tiempos y recursos, agilizando el flujo de información y prestación de servicios.

Se espera agilizar un servicio, sin importar el canal a prestar y el cliente externo o interno porque al contar con una Intranet los servicios serán consumidos de manera directa.

Costos de Hardware y Software

Al contar con los equipos y recursos técnicos necesarios dentro de la empresa, no se necesita realizar ningún tipo de inversión en esta área, para la debida implantación del sistema de motor transaccional.

Costo de Personal

El Sistema de motor transaccional, no representará variaciones en cuanto al personal, pero si requerirá de actividad adicional, por parte de los operadores y desarrolladores.

El equipo de operadores deberá, recibir una capacitación interna para el respectivo monitoreo del Motor Transaccional este monitoreo se centrara en que el servicio este levantado, y seguimiento de la base datos del motor y del sistema de seguridad, que es el encargado de controlar los accesos de hora del servicio.

El sistema propuesto no incluyo variaciones en cuanto al personal bajo cuya responsabilidad está la operación y/o funcionamiento del sistema.

El equipo de desarrollo no genera inversión ya que al trabajar en la misma institución se considero que es un proyecto que beneficiara a la institución en la transición del desarrollo de sistemas, pero a nivel personal se realizo una inversión de tiempo y materiales para la investigación y desarrollo del motor transaccional el cual se partió de un presupuesto base.

Beneficios Tangibles

Los beneficios tangibles que aportará el sistema están dados por los siguientes aspectos:

- ✓ Reducción de costos de tiempo desarrollador de aplicativos.
- ✓ Reducción de costos de documentación.

Beneficios Intangibles

Entre los beneficios intangibles del sistema se pueden incluir los siguientes:

- ✓ Optimizar la interconexión de clientes internos con las capas lógicas a través de los servicios.
- ✓ Un control y seguimiento de las transacciones que generen las interconexiones.
- ✓ Log de transacciones efectuadas desde y hacia el componente.
- ✓ Selección de API de aplicación distribuida y facilidad de cambio al mismo.
- ✓ Contar con estándares de implementación de aplicativos.

- ✓ Aprovechar al máximo los recursos tecnológicos con los que cuenta la empresa

2 MARCO DE REFERENCIA

2.1 Marco teórico

El diseño de un sistema que provea una plataforma empresarial que implemente interfaces estandarizadas para comunicación, conectividad, transformación y seguridad, conlleva analizar la factibilidad y disponer de todo esto integrado en uno solo sistema.

El Motor transaccional ocupara un lugar intermedio entre distintos sistemas de la empresa, proporcionado mecanismos de comunicación y transformación a través de protocolo TCP.

El Motor Transaccional se basara en la noción de comunicación basada en mensajes y todo aquello que se pueda modelar como un mensaje (por ejemplo, un mensaje por puerto compartido Net TCP que va de tipo binario). Para de esta manera habilitar un puerto escucha con varios servicios y así logar que los aplicativos se conecten con este.

El consumo de los servicios se basara en clientes, que son aplicaciones que inician la comunicación y servicios, que esperan a que los clientes se comuniquen con ellos y respondan a esa comunicación, de tal manera que una única aplicación puede actuar como cliente y como servicio.

Con respecto a los mensajes, estos se enviaran entre extremos, ya que los extremos son los lugares donde los mensajes se envían o reciben (o ambos), y definen toda la información requerida para el intercambio de mensajes. Un servicio expondrá uno o más extremos de la aplicación, y el cliente genera un extremo que es compatible con uno de los extremos del servicio.

El extremo describirá de una manera basada en estándar dónde se deberían enviar los mensajes, cómo se deberían enviar y qué aspecto deberían tener los mensajes.

Para el motor transaccional, la pila de comunicación será el protocolo de transporte de los mensajes, ya que los mensajes se podrán enviar a través de intranet por medio del protocolo TCP y compartir un mismo puerto para varios servicios, proporcionado un

punto común de control para los administradores del firewall, al tiempo que permite a los desarrolladores de aplicaciones minimizar el costo de implementación de generar nuevas aplicaciones que pueden utilizar la red.

Los servicios serán creados y configurados en WCF (Windows Comunication Fundation), basándose en Contracts (contrato), para describir su estructura, sus acciones, y de qué manera se enlazaran o la estructura de datos, todo y cada uno de estos contratos serán fundamentales a la hora de trabajar con el Motor Transaccional

Ya que cada tipo de información del servicio proporcionado, tiene una clasificación de contratos, tales como la descripción de las operaciones que ofrece el servicio, la estructura de datos manejados por las operaciones del servicio, y el contenido de un mensaje, ya sea esta cabecera o cuerpo del mensaje.

El sistema de seguridad del Motor Transaccional será desarrollado en una arquitectura orientada a servicios (SOA), y las herramientas de UML para describir los métodos y procesos del sistema.

2.2 Marco conceptual

2.2.1 Andinadatos

Es la unidad de telecomunicaciones avanzadas de Andinatel S.A (CNT, corporación nacional de telecomunicaciones), implementada con el fin de ofrecer soluciones integrales en la transmisión de datos.

Esta unidad se soporta en una plataforma tecnológica de última generación que permite ofrecer confiabilidad y máxima velocidad para cumplir con óptima calidad las

Necesidades de transmisión de datos.

2.2.2 GlobalCrossing.

Empresa mundial que brinda servicios de IP y tienen alcance global y conectan a empresas, gobiernos y operadores de todo el mundo con sus clientes, empleados y socios ubicados por todo el mundo, a través de un ambiente seguro que es ideal para las aplicaciones de negocios basadas en IP que facilitan el desarrollo del comercio electrónico. Esta compañía ofrece una completa gama de productos administrados de datos y voz

2.2.3 WCF (Windows Comunication Foundation)

Las aplicaciones distribuidas se vuelven cada día más complejas, y es primordial que los negocios estén conectados.

Esta globalización informática requiere la interacción y la conectividad entre diferentes plataformas e incontables dispositivos. Windows Comunication Foundation fue concebido con el objetivo de simplificar el desarrollo de aplicaciones distribuidas

Windows Comunication Foundation es un conjunto de tecnologías .NET para la creación y puesta en marcha de sistemas interconectados, el cual basado en el modelo de programación unificado de Microsoft, se puede generar aplicaciones orientadas a servicios ya que fue creado con el fin de permitir una programación rápida de sistemas distribuidos permitiendo a los programadores generar soluciones con transacción seguras y de confianza.

Una de sus principales características es que cuenta con un modelo de programación unificado, ya que es la unificación de numerosas capacidades que antes podíamos encontrar en distintas tecnologías. Esto nos libra de tener que estar utilizando más de una tecnología para cumplir satisfactoriamente los requerimientos. De este modo, los desarrolladores harán una tarea de una única manera.

Las comunicaciones se enriquecen notablemente, ya que Windows Comunication Foundation da la libertad al desarrollador de utilizar múltiples transportes, distintos tipos de formatos de mensajes y diversos patrones de mensajes.

2.2.4 Contratos (Contract)

Los contractos son donde se describe los servicios, su estructura, sus acciones, y de qué manera se enlazan a la estructura de datos, especificando que el contrato es un acuerdo entre las partes involucradas, que define la manera en que se va a trabajar y con qué elementos.

En este caso se especificará qué funciones se van a ofrecer y qué tipos de datos se van a recibir o enviar, y cómo van a ser enviados.

Los contratos son conceptos fundamentales a la hora de trabajar con WCF. Ya que estos permiten que los clientes y los servicios tengan el mismo conocimiento de las operaciones ofrecidas, las estructuras de datos y la estructura del mensaje

Para cada tipo de información del servicio proporcionado, existe una clasificación de contratos:

- ✓ **Contrato de Datos:** describe las estructuras de datos manejadas por las operaciones del servicio, donde también es posible especificar la estructura del tipo de datos que intercambia el servicio.
- ✓ **Contrato de Servicio:** describe qué es lo que hace el servicio, ya que dentro de este se encuentran los métodos y funciones que definen acciones; este conjunto de métodos y funciones se denomina “service operations” u “operaciones del servicio”, donde un servicio debe tener por lo menos un contrato de servicio.
- ✓ **Contrato de Mensaje** define el formato del mensaje enviado y recibido por una operación de un servicio.

- ✓ **Contrato de Error (FaultContract):** Los servicios WCF reportan los errores usando fault objects. El tipo de contrato permitirá viajar el mensaje por el servicio desde el motor hacia el cliente, para su respectivo procesamiento.

Titulo: Windows Comunication Foundation Descripción de la Arquitectura (WCF)

http://msdn.microsoft.com/en-us/library/aa480210.aspx

Consultado el 16 de Junio del 2009

2.2.5 Terminologías Motor Transaccional

Mensaje:

Un mensaje es una unidad autónoma de datos que pueden estar compuestos de varias partes, incluyendo un cuerpo y encabezados.

Servicio

Un servicio es una construcción que expone uno o más extremos, y en la que cada extremo expone una o más operaciones de servicio.

Extremo

Un extremo es una construcción en la que se envían o reciben mensajes (o ambos). Comprende una ubicación (una dirección) que define dónde se pueden enviar mensajes, una especificación del mecanismo de comunicación (un enlace) que describe cómo se deberían enviar los mensajes, y una definición para un conjunto de mensajes que se puede enviar o se puede recibir (o ambos) en esa ubicación (un contrato de servicio) que describe qué mensajes se pueden enviar.

Un servicio se expone al mundo como una colección de extremos.

Dirección

Una dirección especifica la ubicación donde se reciben los mensajes. Se especifica como un identificador uniforme de recursos (URI). La parte del esquema URI nombra el mecanismo de transporte que se ha de utilizar para alcanzar la dirección, como HTTP y

TCP. La parte jerárquica del URI contiene una ubicación única cuyo formato depende del mecanismo de transporte.

Operación de servicio

Una operación de servicio es un procedimiento definido en el código de un servicio que implementa la funcionalidad de una operación. Esta operación se expone a los clientes como métodos en un cliente. El método puede devolver un valor y puede tomar un número opcional de argumentos, o no tomar ningún argumento y no devolver ninguna respuesta.

Contrato de servicio

El contrato de servicio une varias operaciones relacionadas en una unidad funcional única. El contrato puede definir ajustes de servicio, tales como el espacio de nombres del servicio, un contrato de devolución de llamada correspondiente y otros ajustes de este tipo.

Contrato de mensaje

Un contrato de mensaje describe el formato de un mensaje. Por ejemplo, declara si los elementos del mensaje deberían ir en encabezados frente al cuerpo, qué nivel de seguridad debería aplicarse a qué elementos del mensaje

Alojamiento

Un servicio se debe alojar en algún proceso. Un host es una aplicación que controla la duración del servicio. Los servicios pueden auto alojarse o un proceso de alojamiento existente puede administrarlos.

Aplicación de cliente

Una aplicación de cliente es un programa que intercambia mensajes con uno o más extremos. La aplicación de cliente comienza creando una instancia de un cliente del Motor Transaccional y llamando métodos del cliente. Es importante tener en cuenta que una única aplicación pueda ser cliente y servicio.

Metadatos

Los metadatos de un servicio describen las características del servicio que una entidad externa necesita entender para comunicarse con el servicio.

Los metadatos expuestos por el servicio incluyen documentos de esquema XML, que definen el contrato de datos del servicio, y documentos WSDL, que describe los métodos del servicio.

Titulo: MSDN Library (Español)

http://msdn.microsoft.com/es-es/library/ms734772.aspx

Consultado el 10 de septiembre del 2009

2.2.6 Uso compartido de puertos Net TCP.

El servicio de uso compartido de puertos Net.TCP es un servicio de Windows de modo de usuario que acepta conexiones net.tcp:// en nombre de los procesos de trabajo que se conectan a través de él.

Que cuando una conexión de socket llega, el servicio de uso compartido de puertos inspecciona la secuencia del mensaje entrante para obtener su dirección de destino. Basándose en esta dirección, el servicio de uso compartido de puertos puede enrutar la secuencia de datos a la aplicación que finalmente lo procesa.

La arquitectura del uso compartido de puertos en WCF tiene tres componentes principales:

- ✓ **Un proceso de trabajo**: Es cualquier proceso que se comunica sobre net.tcp://, utilizando los puertos compartidos.
- ✓ **El transporte TCP de WCF**: que es el implementa el protocolo net.tcp://.
- ✓ **El servicio de uso compartido de puertos Net.TCP**: Que es encargado de permitir, que varios procesos de trabajo compartan el mismo puerto TCP.

Cuando un servicio de WCF que utiliza el uso compartido de puertos net.tcp:// se abre, la infraestructura de transporte TCP de WCF no abre directamente un socket TCP en el proceso de la aplicación. En su lugar, la infraestructura de transporte registra el

Identificador uniforme de recursos (URI) de la dirección base del servicio con el servicio de uso compartido de puertos Net.TCP y espera a que el servicio de uso compartido de puertos escuche mensajes en su nombre.

Este servicio de uso compartido de puertos, entrega mensajes direccionados al servicio de la aplicación (sistema), según van llegando.

Algo importante que se debe considerar al hacer uso del servicio del uso compartido de puertos Net.TCP (**capa de procesamiento entre las aplicaciones y la red**), es que las aplicaciones que utilizan el uso compartido de puertos, deben estar protegidas como si estuvieran realizando escuchas directamente en la red. Es decir que las aplicaciones al utilizar el uso compartido de puertos deberían evaluar los privilegios de procesos bajo los que se ejecutan. Como el considerar ejecutar la aplicación utilizando la cuenta de servicio de red integrada, que se ejecuta con el conjunto mínimo de privilegios de procesos requeridos para la comunicación por red.

Titulo: Uso compartido de Puerto Net. TCP

http://msdn.microsoft.com/es-es/library/ms734772.aspx

Consultado el 27 de agosto del 2009

2.2.7 Aplicaciones distribuidas

Un sistema distribuido es una colección de computadores conectados por una red de comunicaciones, que el usuario percibe como un solo sistema (no necesita saber qué cosas están en qué máquinas). El usuario accesa los recursos remotos de la misma manera en que accesa recursos locales, o un grupo de computadores que usan un software para conseguir un objetivo en común.

Los sistemas distribuidos deben de ser muy confiables, ya que si un componente del sistema se descompone otro componente debe de ser capaz de reemplazarlo.

El tamaño de un sistema distribuido puede ser muy variado, ya sean decenas de hosts (Local Área Network), centenas de hosts (Metropolitan Area Network), y miles o millones de hosts (Internet).

Todo sistema distribuido debe ser diseñado y construido bajo los siguientes aspectos de calidad:

- ✓ **Heterogeneidad**: mezclar e integrar todos sus componentes (hardware y Software) como un todo, debe poderse ver como un gran sistemas, sin importar el numero y ubicación de los elementos que lo conforman.
- ✓ **Extensibilidad**: poseer la facultad de poder integrar sin mayores riesgos nuevos elementos, esto no debería afectar significativamente el desempeño, la seguridad y objetividad del sistema.
- ✓ **Seguridad**: tener en cuenta aspectos como la Convivialidad, Integridad y Acceso o disponibilidad (Modelo C.I.A) a la información.
- ✓ **Escalabilidad**: permanecer estable en la medida que aumenten o disminuyan los recursos que utiliza y los elementos (Hardware y Software) que lo conforman.
- ✓ **Tratamiento de fallos**: Tener control absoluto de posibles fallos, evitando el colapso total del sistema cuando fallen algunos de sus elementos.
- ✓ **Concurrencia**: poseer la capacidad y habilidad de poder realizar múltiples tareas (conexión, desconexiones, peticiones, procesamiento de peticiones, generación de respuestas, análisis y control de recurso) de forma simultánea.
- ✓ **Transparencia**: abstraer y esconder al usuario su funcionamiento interno y su estructura.

Titulo: Diseñando Aplicaciones Distribuidas

http://es.wikipedia.org/wiki/Interoperatividad

Consultado el 27 de agosto del 2009

2.2.8 UML

Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software rehusadles.

UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Sin embargo, hay que tener en cuenta un aspecto importante del modelo: no pretende definir un modelo estándar de desarrollo, sino únicamente un lenguaje de modelado.

Anteriormente existían diversas técnicas y métodos orientados a objetos con características comunes pero utilizando distintas notaciones, con lo que se presentaban inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc., lo que genero la creación del UML como estándar de modelamiento de sistemas de software principalmente, pero con la gran posibilidad de que se pueda aplicar a todo tipo de proyecto.

Titulo: Lenguaje Unificado de Modelado

http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

Consultado el 9 de septiembre del 2009

2.3 Marco espacial

El sistema de Motor transaccional en Windows está enfocado a proporcionar una herramienta de desarrollo para el programador y enlace de conexión con clientes (sistemas desarrollados y nuevos), para aplicar el concepto de aplicaciones distribuidas dentro de cualquier institución que quiera innovar su línea de negocio por la parte informática.

El resultado del proyecto se lo espera culminar en el lapso de 9 meses los cuales se dividen 5 meses de investigación y 4 meses de implementación y desarrollo.

2.4 Marco Legal

Dentro del marco legal se basa en el concepto de propiedad intelectual del sistema desarrollado basado en los conceptos vigentes en la ley.

Art.1. El Estado reconoce, regula y garantiza la propiedad intelectual adquirida de conformidad con la ley, las Decisiones de la Comisión de la Comunidad Andina y los convenios internacionales vigentes en el Ecuador

La propiedad intelectual comprende:

1. Los derechos de autor y derechos conexos.
2. La propiedad industrial, que abarca, entre otros elementos, los siguientes:
   a. Las invenciones;
   b. Los dibujos y modelos industriales;
   c. Los esquemas de trazado (topografías) de circuitos integrados;
   d. La información no divulgada y los secretos comerciales e industriales;
   e. Las marcas de fábrica, de comercio, de servicios y los lemas comerciales;
   f. Las apariencias distintivas de los negocios y establecimientos de comercio;
   g. Los nombres comerciales;
   h. Las indicaciones geográficas; e,
   i. Cualquier otra creación intelectual que se destine a un uso agrícola, industrial o comercial.

Art.28. Los programas de ordenador se consideran obras literarias y se protegen como tales. Dicha protección se otorga independientemente de que hayan sido incorporados en un ordenador y cualquiera sea la forma en que estén expresados, ya sea en forma legible por el hombre (código fuente) o en forma legible por la máquina (código objeto), ya sean

programas operativos y programas aplicativos, incluyendo diagramas de flujo, planos, manuales de uso, y en general, aquellos elementos que conformen la estructura, secuencia y organización del programa.

3 METODOLOGIA

3.1 Metodología de investigación

3.1.1 Unidad de Análisis

Para la implementación del proyecto se toma como unidad de análisis el área de sistemas del banco Promerica, ya que se enfoca en la necesidad de prestar servicios de manera ágil y rápida, se conseguirá un sistema real y factible que demuestre que la utilización del sistema implementado favorecerá en la prestación de servicios de manera ágil y rápida.

3.1.2 Tipo de Investigación.

Este proyecto es de tipo aplicada, con el objetivo de aplicar y utilizar los conocimientos adquiridos en el proceso de investigación.

La aplicación de los conocimientos adquiridos será recopilada en base a la investigación de conceptos del negocio y experiencias propias de elaboración de proyectos anteriores que han requerido de su progresivo crecimiento y han motivado a la investigación de nuevas tendencias tecnológicas, cuya fuente de información serán datos en el Internet y personas que tengan un vasto conocimiento de la tecnología.

3.1.3 Métodos

Debido a la naturaleza de este proyecto, el desarrollo del mismo se dará en diferentes etapas, cada una de las cuales utilizar su propia metodología.

El método deductivo será el que permita partir de los datos generales aceptados como valederos, para deducir por medio del razonamiento lógico, varias suposiciones, es decir;

partirá de verdades previamente establecidas como principios generales, para luego aplicarlo a casos individuales y comprobar así su validez.

Y el método inductivo cuando de los hechos particulares se obtendrá proposiciones generales, es decir, establecer un principio general una vez realizado el estudio y análisis de hechos y fenómenos en particular.

La primera etapa de la investigación es de tipo documental, puesto que permitirá la sustentación básica en cuanto a conocimientos para el desarrollo del sistema informático objeto de investigación. Para la selección de la información se utilizara metodología bibliografía y deductiva.

En la segunda etapa se realizara el diseño e implementación, debiendo utilizar el método analítico y experimental obteniendo prototipos que serán perfeccionados a través de continuos procesos.

3.1.4 Fuentes y Técnicas para la recolección de información

Lo que concierne a fuentes primarias, será la recolección de información por Internet, libros y requerimiento de la empresa, y fuentes secundarias será el dialogar con personal con vasto conocimiento teórico tecnológico del tema.

Como técnica para la recolección de información se utilizara las encuestas, que se lo realizar de modo tradicional, con cuestionario papel, en la empresa.

Esta encuesta ayudara a la recolección de información para conceptualización y posterior implementación del sistema propuesto.

3.2 Metodología informática

3.2.1 Metodología Orientada a Objetos

La metodología de análisis y diseño orientados a objetos es la más madura y eficiente que existen en la actualidad. La gran virtud que aporta esta metodología es su carácter de abierta (no propietaria), que le permite ser de dominio público y, en consecuencia le permite sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software.

El análisis orientado a objetos (AOO) se basa en un conjunto de principios básicos:

- ✓ Se modela el dominio de la información.
- ✓ Se describe la función del módulo.
- ✓ Se representa el comportamiento del modelo.
- ✓ Los modelos se dividen para mostrar más detalles.
- ✓ Los modelos iníciales representan la esencia del problema mientras que los últimos aportan detalles de la implementación.

El propósito de la metodología orientad a objetos es definir todas las clases (y las relaciones y comportamientos asociados con ellas) que son relevantes al problema que se va a resolver.

El objetivo del análisis orientado a objetos es desarrollar una serie de modelos que describan el software de computadora al trabajar para satisfacer un conjunto de requisitos definidos por el cliente.

El modelo de análisis ilustra información, funcionamiento y comportamiento dentro del contexto de los elementos del modelo de objetos (clases y objetos, atributos operaciones y mensajes).

3.2.2 Proceso de Desarrollo

El Sistema de motor transaccional se basa en el proceso de desarrollo que sigue las especificaciones del (UML), Proceso Unificado de Desarrollo, el mismo que es un conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software.

Las características del Proceso Unificado son las siguientes:

- ✓ Dirigido por casos de uso
- ✓ Centrado en la arquitectura.
- ✓ Iterativo-Incremental.
- ✓ Modelos de procesos.

3.2.3 Planificación proceso de ingeniería

El proceso unificado de desarrollo está conformado por las siguientes fases.

- ✓ Fase de inicio o planificación
- ✓ Fase de elaboración
- ✓ Fase de construcción
- ✓ Fase de transición

Cada una de estas fases se divide en iteraciones o mini-proyectos. Y cada una de estas iteraciones pasa por los cinco flujos de trabajo (requisitos, análisis, diseño, implementación y pruebas). Tal y cual muestra en el grafico 2, con la estructura de las fases de un ciclo junto con los flujos de trabajo por los que tiene que pasar cada fase.

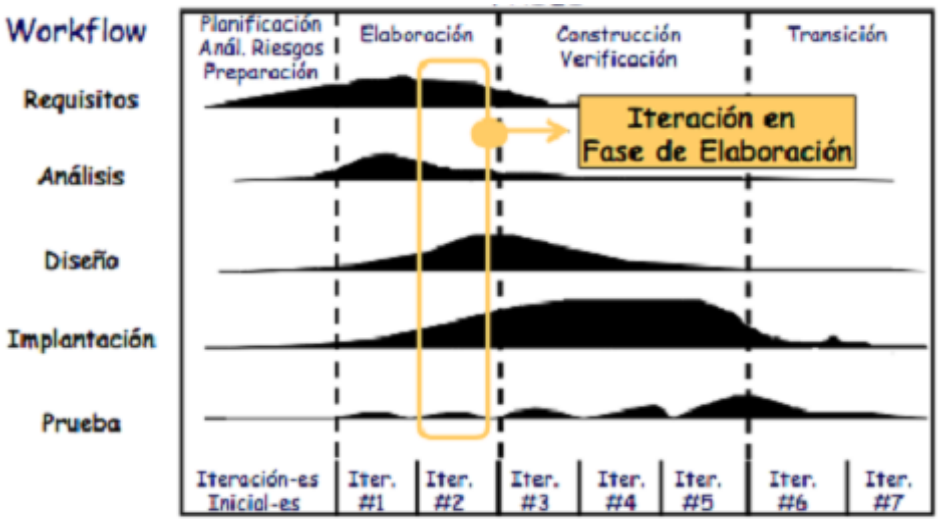


Grafico 3-1 Proceso unificado de desarrollo

Fase de inicio o planificación

Se define el ámbito y el objetivo del proyecto, así como la funcionalidad y capacidades del producto.

El analista construye un modelo del dominio del problema, mostrando sus propiedades más importantes. El modelo de análisis es una abstracción resumida y precisa de lo que debe de hacer el sistema deseado y no de la forma en que se hará, los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos. Un buen modelo debe poder ser entendido y criticado por expertos en el dominio del problema que no tengan conocimientos informáticos.

En esta fase se responde a las siguientes preguntas:

- ✓ ¿Cuáles son las principales funciones del sistema? La respuesta se encuentra en un modelo de casos de uso simplificado que contenga los casos de uso más críticos.
- ✓ ¿Cómo podría ser la arquitectura del sistema? Es un simple bosquejo que muestra los subsistemas más importantes.

- ✓ ¿Cuál es el plan de proyecto y cuánto costará desarrollar el producto? En esta fase se identifican los riesgos y priorizan los más importantes, se planifica en detalle la fase de elaboración y se estima el proyecto de manera aproximada.

En esta fase se obtiene la siguiente documentación

- ✓ La versión inicial del modelo de dominio.
- ✓ La versión inicial del modelo de negocio.
- ✓ La versión inicial del modelo de los artefactos de los requisitos (diagrama de casos de usos).

Fase de elaboración

Tanto la funcionalidad como el dominio del problema se estudian en profundidad, así como su arquitectura básica, y se planifica el proyecto considerando recursos disponibles

Es donde el diseñador de objetos construye un modelo de diseño basándose en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centra en las estructuras de datos y algoritmos que son necesarios para implementar cada clase, en esta fase se describe la forma en que el diseño puede ser implementado en distintos lenguajes (orientados y no orientados a objetos, bases de datos, etc.).

Por lo tanto, en esta fase la arquitectura se expresa en forma de vistas de todos los modelos del sistema, los cuales juntos representan al sistema entero. Esto implica que hay vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación (ésta incluye componentes para probar que la arquitectura es ejecutable) y del modelo de despliegue. Al finalizar la fase de elaboración, se estará en la disposición de planificar las actividades y estimar los recursos necesarios para terminar el proyecto

El objetivo de la fase se pueden enumerar como:

- ✓ Tanto la funcionalidad como el dominio del problema se estudian en profundidad.

- ✓ Se define la arquitectura básica.
- ✓ Se planifica el proyecto considerando recursos disponibles.

Fase de construcción

En esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo. La descripción evoluciona hasta convertirse en un producto preparado para ser entregado al usuario. La mayor parte de los recursos requeridos se emplean en esta fase de desarrollo. Al finalizar esta fase, el producto contendrá todos los casos de uso que se han acordado para el desarrollo de esta versión. Sin embargo, puede que no esté libre de defectos. Muchos de estos defectos se descubrirán y solucionarán durante la fase de transición.

El producto se desarrolla a través de iteraciones donde cada iteración involucra tareas de análisis, diseño e implementación, las fases de estudio y análisis dan una arquitectura básica que es aquí refinada de manera incremental conforme se construye (se permiten cambios en la estructura).

Gran parte del trabajo es programación y pruebas y se documenta tanto el sistema construido como el manejo del mismo, con lo que la fase proporciona un producto construido junto con la documentación

Fase de transición

Es la fase final del ciclo, en donde la fase cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número de usuarios con experiencia, realizan las pruebas del producto e informan de defectos y deficiencias. El desarrollador corrige los problemas e incorporan algunas mejoras.

La fase de transición conlleva actividades como la fabricación, formación del cliente, proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren tras la entrega.

Se libera el producto y se entrega al usuario para un uso real, donde se incluyen tareas de marketing, empaquetado atractivo, instalación, configuración, entrenamiento, soporte, mantenimiento, etc.

Los manuales de usuario se completan y refinan con la información anterior, estas tareas se realizan también en iteraciones

3.2.4 Flujos de trabajo del proceso

El Proceso Unificado identifica a los flujos de trabajo fundamentales que se producen durante el proceso de desarrollo de software. Estos flujos incluyen el modelado de negocio, requerimientos, análisis, diseño, implementación y testing. Los flujos no son secuénciales y serán realizados preferentemente durante las cuatro fases. Los flujos son descriptos separadamente en el proceso por claridad, pero de hecho se ejecutan en forma concurrente, interactuando y utilizando los artefactos que cada uno genera.

En RUP se definen nueve flujos de trabajo distintos, separados en dos grupos. Los flujos de trabajo de 'ingeniería' son los siguientes:

Modelado del negocio.

Con este flujo de trabajo pretende llegar a un mejor entendimiento de la organización donde se va a implantar el producto. Los principales motivos para esto son los siguientes:

- ✓ asegurarnos de que el producto será algo útil, no un obstáculo;
- ✓ conseguir que encaje de la mejor forma posible en la organización;
- ✓ tener un marco común para los desarrolladores, los clientes y los usuarios finales.

Este flujo de trabajo no será siempre necesario. Si sólo añadimos funcionalidad que no verán los usuarios directamente, no hará falta.

Requisitos.

- ✓ Este es uno de los flujos de trabajo más importantes, porque en él se establece QUÉ es lo que tiene que hacer exactamente el sistema que diseñe. En esta línea los

requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.

Análisis y diseño.

- ✓ El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos.

Implementación.

- ✓ En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer los test de unidad: cada implementador es responsable de testear las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

Test.

- ✓ Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Despliegue.

- ✓ El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:
  - ✓ Testear el producto en su entorno de ejecución final.
  - ✓ Empaquetar el software para su distribución.
  - ✓ Distribuir el software.
  - ✓ Instalar el software.

- ✓ Proveer asistencia y ayuda a los usuarios.
- ✓ Formar a los usuarios y al cuerpo de ventas.
- ✓ Migrar el software existente o convertir bases de datos.

Los flujos de trabajo de apoyo son:

Administración del proyecto.

El objetivo de la administración de un proyecto es conseguir equilibrar el completar los objetivos, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requisitos de los usuarios.

Configuración y control de cambios.

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

Entorno.

La finalidad de este flujo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Es decir tener a punto las herramientas que se vayan a necesitar en cada momento, así como definir la instancia concreta de proceso unificado que se va a seguir. En concreto las responsabilidades de este flujo de trabajo incluyen:

- ✓ Selección y adquisición de herramientas.
- ✓ Establecer y configurar las herramientas para que se ajusten a la organización.
- ✓ Configuración del proceso.
- ✓ Mejora del proceso.
- ✓ Servicios técnicos.

3.2.5 Artefactos

Se tiene un conjunto de artefactos definidos en cada una de las disciplinas y utilizadas dentro de ellas por los actores para la realización de las mismas.

RUP en cada una de sus fases realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema estos artefactos son los siguientes:

Inicio

- ✓ Documento visión
- ✓ Especificación de requerimiento.

Elaboración

- ✓ Diagrama de casos de uso

Construcción

- ✓ Documento Arquitectura que trabaja con las siguientes vistas:

Vista Lógica:

- ✓ Diagrama de clases
- ✓ Modelo E-R (Si el sistema así lo requiere)

Vista de Implementación:

- ✓ Diagrama de Secuencia
- ✓ Diagrama de estados
- ✓ Diagrama de Colaboración

Vista Conceptual:

- ✓ Modelo de dominio

Vista física:

- ✓ Mapa de comportamiento a nivel de hardware

3.2.6 Modelos

Cada flujo de trabajo está relacionado con uno o varios modelos, que representan las decisiones relacionadas con la visualización, especificación, construcción y documentación de un sistema:

- ✓ **Modelo de negocio**. representa el modelo lógico de la empresa.

- ✓ **Modelo de dominio**. representa el contexto en el que se incluye el Sistema.
- ✓ **Modelo de casos de uso**. representa los requisitos funcionales del sistema
- ✓ **Modelo de análisis**: representa el diseño de las ideas.
- ✓ **Modelo. de diseño**: establece el vocabulario del problema y de su solución.
- ✓ **Modelo de proceso** (opcional): define los mecanismos de concurrencia y sincronización del sistema.
- ✓ **Modelo de despliegue**: define la topología hardware para la ejecución del sistema.
- ✓ **Modelo de implementación**: define los elementos a ensamblar y el sistema físico.
- ✓ **Modelo de pruebas**: define cómo validar y verificar el sistema.

Los distintos flujos de trabajo dan lugar a diversos modelos que pueden ser expresados mediante diagramas UML

3.2.7 Vistas

Los sistemas deben ser vistos desde diferentes perspectivas para comprender mejor el diseño por lo que la arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema, abstrayéndose de los demás. Para RUP, todas las vistas juntas forman el llamado modelo 4+1 de la arquitectura, el cual recibe este nombre porque lo forman las vistas lógica, de implementación, de proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas.

3.2.8 Presupuesto

Costos del Proceso Actualmente

A continuación se detalla el presupuesto para el motor transaccional en donde se determina los recursos para investigar, desarrollar, implantar, y mantener en operación el sistema informático.

Recursos Institucionales:

- Banco Promerica

Recursos Humanos:

- El investigador del proyecto con la asesoría del tutor.

Recursos Técnicos:

- Visual Estudio Team System 2008
- Componentes Developer Express
- SQL Server 2005
- Windows 2000
- Windows Xp

Recursos Materiales:

- Materiales de escritorio y otros suministros.

Costos Generales.

Los gastos generales se encuentran representados por todos aquellos gastos en accesorios y el material de oficina de uso diario necesario para realizar los procesos, pero de acuerdo a la finalidad y objetivo del sistema de motor transaccional.

Costos de Material de Oficina y Papelería			
Gastos Generales	**Costo Aproximado**	**Cantidad a utilizar**	**Total (Usd)**
Material de Oficina	100	1 unidad	100
Consumo internet mensual	28	6 meses	168
Material Bibliográfico	55	1 unidad	55
Toner de Impresora	80	1 unidad	80
Transporte mensual	22	6 meses	132
Papel Bond	4	3 resmas	12
Total			**547**

Cuadro 3-1 Costos de material de oficina y papelería

Costo de Personal

El costo de personal está enfocado a los generados por el recurso humano, bajo cuya responsabilidad está en el monitoreo y soporte al sistema.

Costos de Salario del Personal. Proceso Actual		
Recurso Humano	**Salario Mensual**	**Salario Anual**
DBA	1000	12000
Analista de Sistemas	670	8040
Operador	600	7200
Total	2270	27240

Cuadro 3-2 Costos de salario del personal – Proceso actual

Descripción de los costos de personal

Contara con una persona para la investigación y el desarrollo

Investigador	**Función**	**Tiempo (h/semana)**	**Duración**	**Costos**			
				U.T. Adm Central	**Fondo de invest UTS**	**Otros**	**Total**
Napoleón Chávez	Investigador principal	8h	40	15	0	0	15

Cuadro 3-3 Personal investigación y desarrollo

Descripción de los equipos requeridos:

Se dispone con un equipo para el desarrollo

No. Equipo	**Función**	**No. Unidad**	**Adquisición**	**Arriendo**	**Costos**			
					U.T. Adm Central	**Fondo de invest UTS**	**Otros**	**Total**
1	Equipo de desarrollo	1	1	0	600	0	150	750

Cuadro 3-4 Equipo disponible desarrollo

Descripción de Insumos y Materiales:

Nombre	Uso	Cantidad	Costos			
			U.T. Adm Central	Fondo de invest UTS	Otros	Total
Flash memory	Almacenamien to información	1	600	0	150	750
Libro WCF	Investigación	1	70	0	0	70
Internet	Investigación	1	25	0	0	25

Cuadro 3-5 Insumos y materiales

4 Proceso de ingeniería

4.1 Fase de Inicio

Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones.

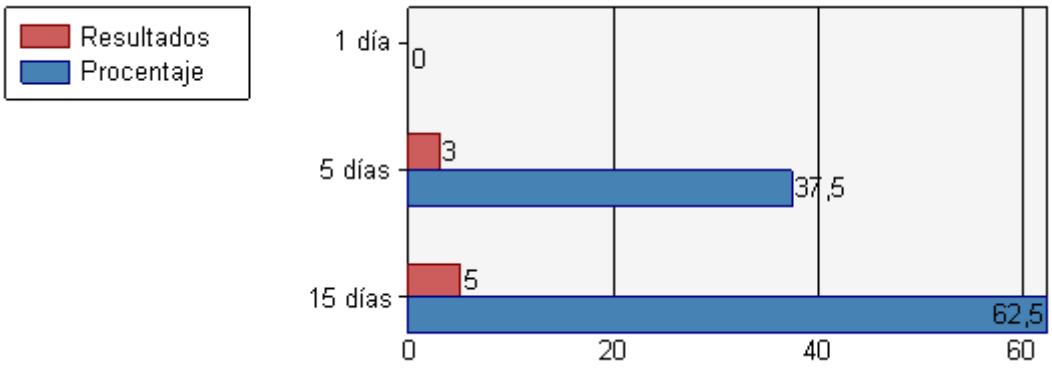
4.1.1 Análisis situación actual

Para tomar la decisión de desarrollar un motor transaccional que ayude y agilice el desarrollo de aplicaciones en plataforma Windows, se opto por la investigación y consulta de tecnologías para crear un motor transaccional, con herramientas que proporcione Microsoft , tomando la decisión de trabajar en Windows Comunication Foundation y aprovechar su extensibilidad para personalizar el funcionamiento de un conector de aplicativos con capas lógicas, de igual manera se consulto a personal de desarrollo y jefes de área por medio de una encuesta, ya que cada uno está en constante relación con los sistemas actuales al momento de la recepción del requerimiento y la implementación de la misma.

Los resultados de la encuesta son las siguientes.

En la actualidad que tiempo lleva la implementación de un proyecto desde su etapa inicial.

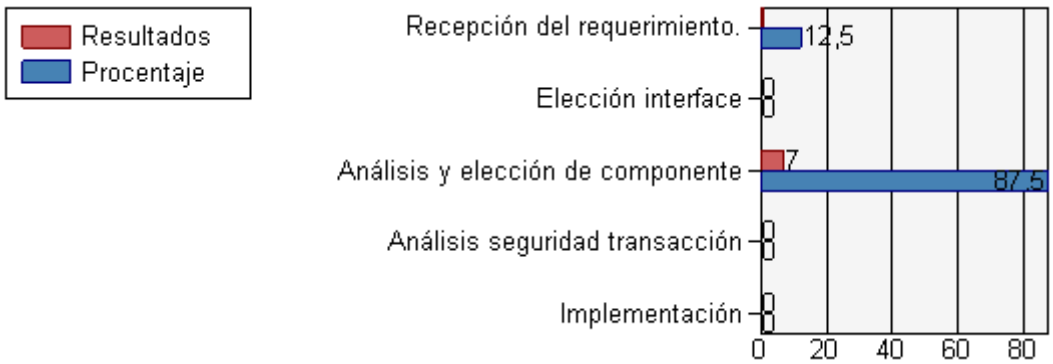
- ✓ 1 día
- ✓ 5 días
- ✓ 10 días



En la pregunta realizada y su resultado se puede observar que la implementación de un sistema lleva un tiempo aproximado de 15 días, esta pregunta se enfoca e manera general en identificar el tiempo de desarrollo de un aplicativo.

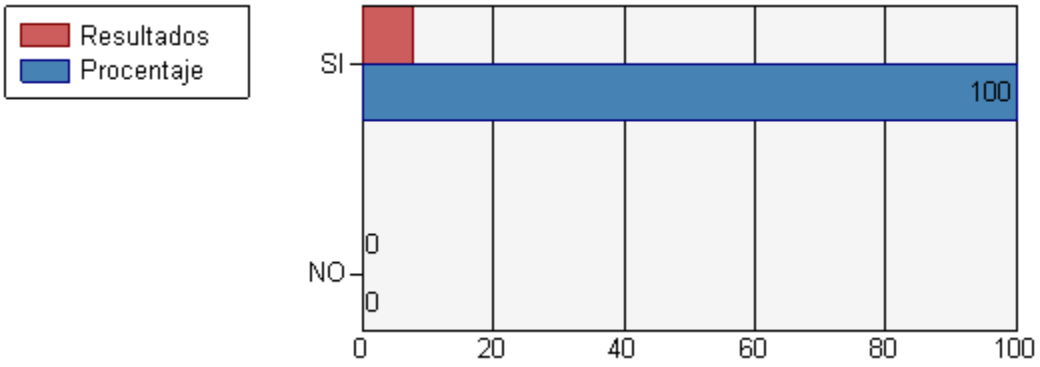
Que proceso es el que lleva dedicación de tiempo en la fase de desarrollo.

- Recepción del requerimiento.
- Elección interface.
- Análisis y elección de componente.
- Análisis seguridad transacción.
- Implementación.



La elección del componente de interconexión implica un conocimiento técnico más amplio para el desarrollador.

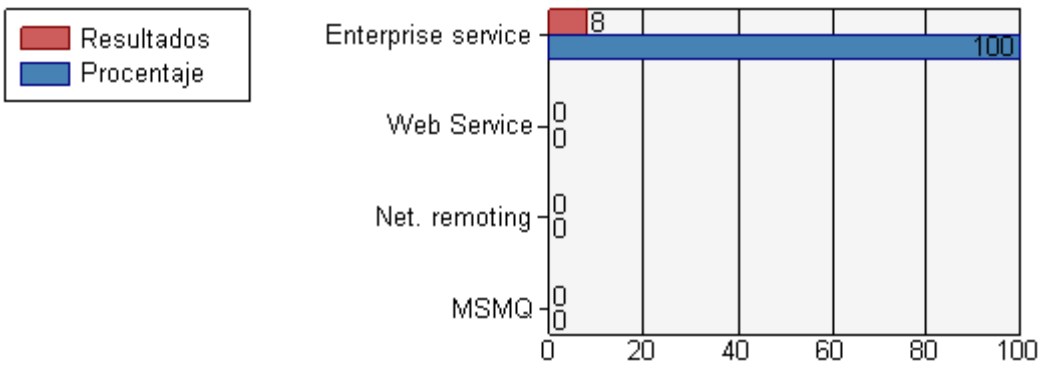
- Si
- No



Estas preguntas se enfocan en identificar cual etapa implica tiempo y conocimiento de una tecnología para la elaboración de aplicaciones distribuidas, con lo que se pudo identificar que la etapa de **análisis y elección de componente** de interconexión lleva un tiempo significativo, el cual se podría enfocar en otra fase de la elaboración, así como el diseño de interface o simplemente reducir el tiempo de entrega del sistema solicitado.

A nivel de conocimiento técnico para el desarrollador, este debería tener conocer una de los componentes de conexión distribuida.

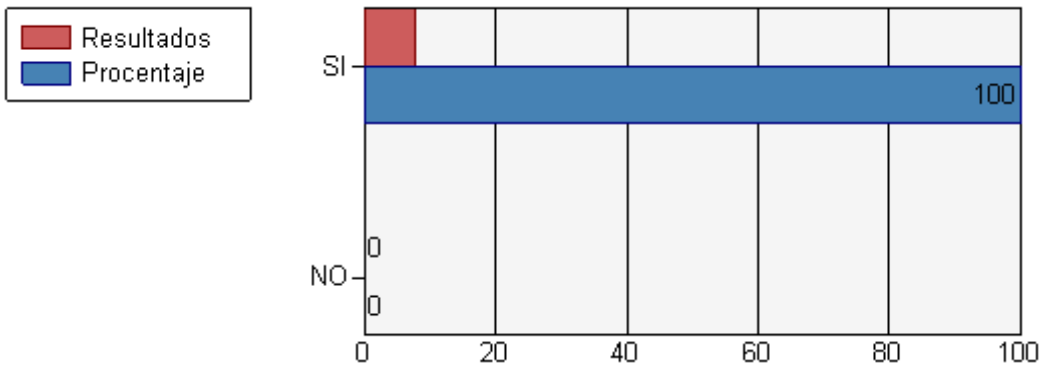
- Enterprise service
- Web Service.
- Net. Remoting
- MSMQ (colas).



El resultado obtenido en esta pregunta, se puede observar que el personal de desarrollo debe poseer un conocimiento sobre Enterprice Service (COM+) , ya que esta es la tecnología que el banco utiliza para desarrollar sus aplicaciones.

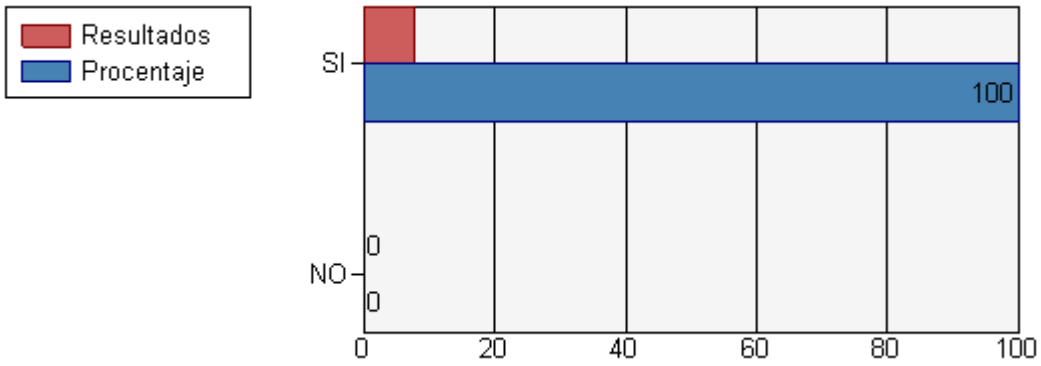
Considera que como desarrollador, el contar con una herramienta que se centre únicamente en la configuración de puerto y la creación de interfaces de conexión, agilizara la entrega de proyectos.

- Si
- No



El desarrollo de un motor transaccional para la interconexión de sistemas con las capas lógicas, considera que sería una herramienta viable para cubrir los requerimientos solicitados de manera más rápida.

- Si
- No



Con estas dos últimas preguntas se pudo determinar la importancia y relevancia que tendría el cambiar la dinámica de desarrollo de un sistema, ya que la estructura de los sistemas se

verán acompañados por una capa intermedia la uqe cumplirá las funciones de interconexión y monitoreo.

4.1.2 Resultado situación actual

En base a los resultados de la encuesta se determino e identifico que al momento de desarrollar un sistema, este involucra un grado de conocimiento técnico con relación al uso de una de las tecnologías para aplicaciones distribuidas (Enterprice Service) y que este conocimiento no siempre está disponible en los desarrolladores de sistemas.

De igual manera se relaciona que el análisis y aprendizaje del componente involucra dedicación de tiempo en la fase de desarrollo, lo que ocasiona que la entrega de los aplicativos lleve más tiempo de lo adecuado.

4.1.3 Análisis de riesgo

En esta fase se analiza, que el riesgo más relevante se da, en la aceptación y comprensión de la funcionalidad del motor transaccional por parte de los desarrolladores y operadores.

Ya que por parte del desarrollador el interactuar con una nueva tecnología y su desconocimiento ocasionará resistencia para el acoplamiento y asimilación de los principios básicos de uso del motor transaccional.

Por parte del operador, se centra en que el proceso de monitoreo es nuevo y las soluciones a problemáticas de servicio, como es la conectividad, permiso de puertos, y configuración de servicios es relativamente nuevo.

Para mitigar los posibles riesgos se considera el capacitar al personal operativo y proporcional manual de uso para su comprensión.

Con relación al desarrollador se considera el capacitar en aspectos básicos de configuración del motor transaccional y proveer de un manual técnico, el que contendrá los estándares de programación y requerimientos de diseño para los servicios.

4.1.4 Modelo de Negocio

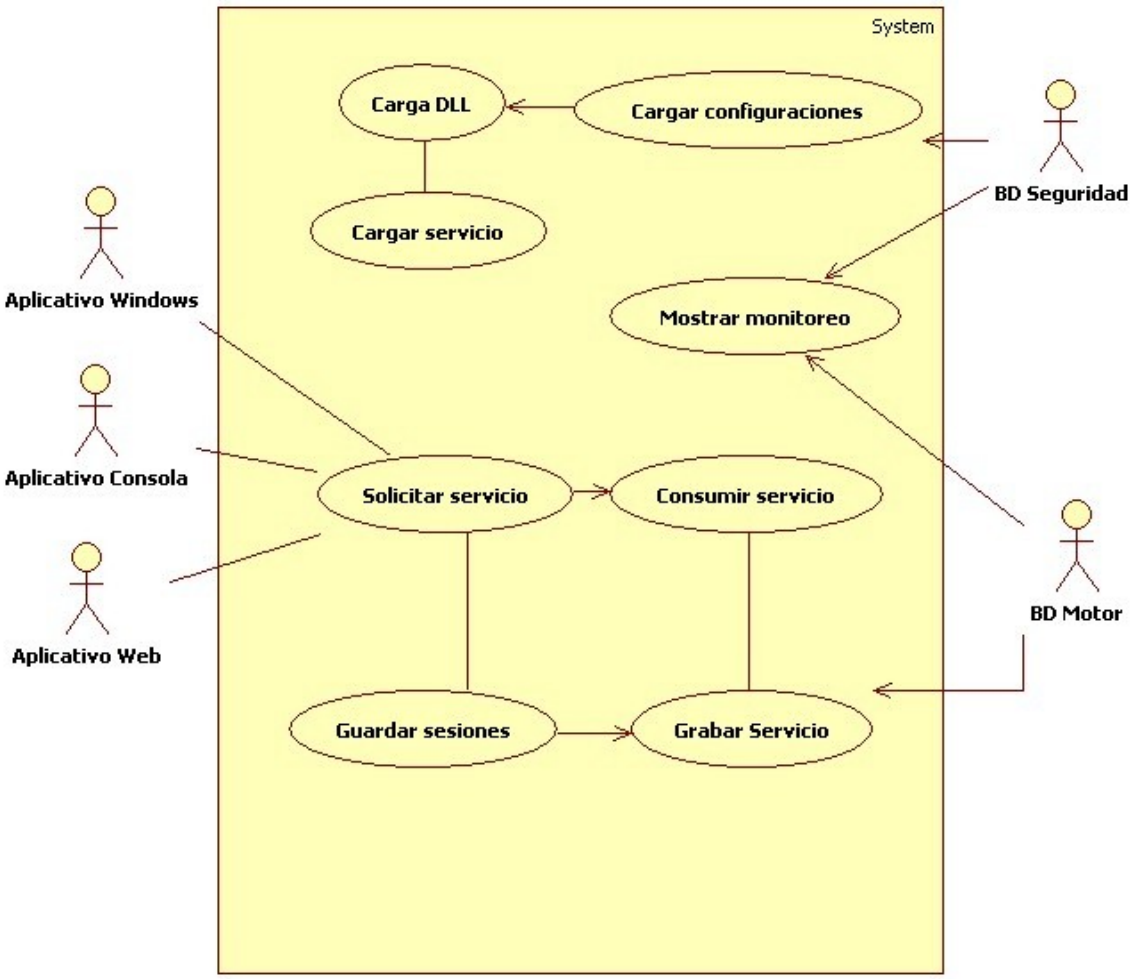


Grafico 4-1 Caso de uso del negocio

Una vez analizados los procesos para el motor transaccional, se decidió utilizar el paradigma orientado a objetos, con el fin de obtener un sistema estable, reutilizable y tolerante al cambio.

En este sentido se decidió utilizar UML como notación común durante el desarrollo, que proporciona una serie de modelos que se basan en los conceptos de objetos, clases y en las relaciones entre ellos.

Para realizar los diferentes diagramas de UML usamos StarUML como herramienta CASE, con el objetivo de producir un software de mayor calidad que satisfaga las necesidades de los usuarios finales.

Inicialmente se implementa el caso del negocio en donde se identifica los procesos principales del motor transaccional, tales como la lectura de ensamblados, configuraciones, almacenamiento de solicitudes y el proceso de consumo del servicio por parte de un aplicativo.

4.1.5 Descripción de los requisitos funcionales

- ✓ Para que los servicios sean levantados por el motor transaccional deben estar debidamente creados en la base de datos, y cada ensamblado debe poseer los atributos necesarios para ser reconocidos por el sistema y levantarlos de acuerdo a la configuración en la base de datos.
- ✓ Se debe mantener una base de datos actualizada para el correcto funcionamiento del motor transaccional.
- ✓ El acceso al modulo de seguridad debe brindar las seguridades de acceso para salvaguardar los datos de la base de datos.
- ✓ Los clientes deben tener definidos los contratos de datos y funciones para consumir los servicios levantador por el motor transaccional.

- ✓ Los servicios deben estar correctamente configurados para poder estar en escucha y responder a las peticiones recibidas.

4.2 Fase de elaboración

En la fase de inicio se plantea los requisitos para la elaboración del sistema, los mismos que sirven de base para iniciar el modelo del sistema con los diferentes diagramas que propone UML, de esta manera se recopila y se obtiene toda la información necesaria para poder comenzar con la construcción del sistema.

En esta fase se elabora un análisis más profundo del proyecto, donde se realiza lo siguiente:

- ✓ Modelo de análisis, primer iteración
    - o Modelo de casos de uso
    - o Diagrama de secuencia
    - o Diagrama de actividad
- ✓ Modelo de diseño, segunda iteración.
    - o Modelo de componentes.
        - ▪ Diagrama de despliegue
        - ▪ Diagrama de arquitectura
        - ▪ Diagrama de componente
    - o Modelo de clases
        - ▪ Diagrama de clases.

4.2.1 Modelo de análisis

4.2.1.1 Diagrama de casos de uso

El Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Diagrama de caso de uso - Carga de servicios

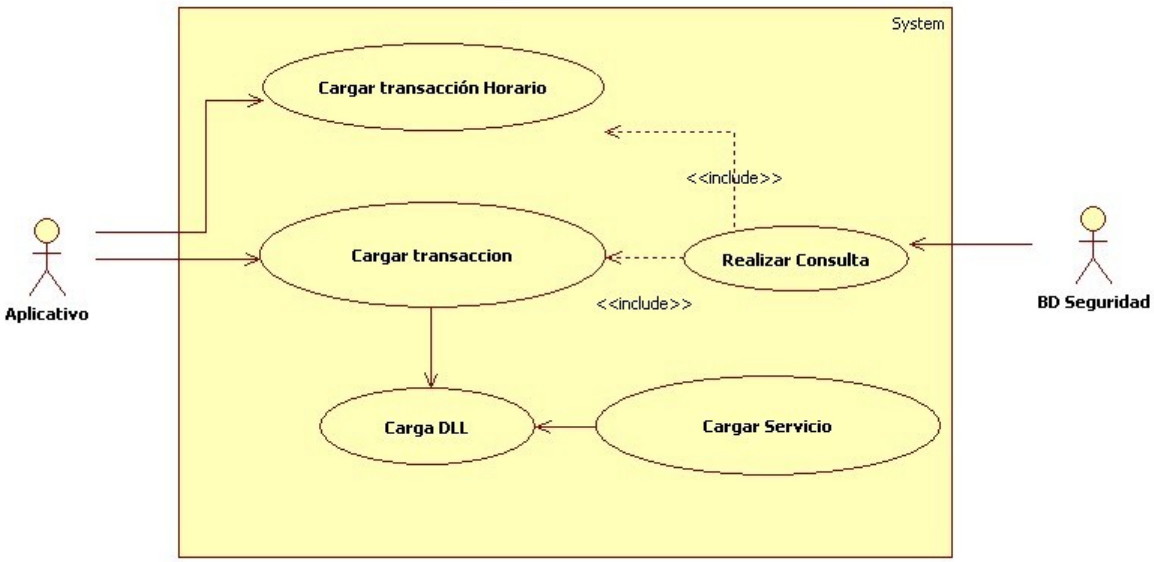


Grafico 4-2 Caso de uso - Carga de Servicio

Detalle Diagrama Casos de Uso - Carga de servicios

Actores

Actor:	Aplicativo
Caso de Uso:	Cargar transacción horario, cargar transacción, realizar consulta
Tipo:	Primario
Descripción:	Es el actor principal y representa al motor transaccional cuando va levantar la lista de transacciones por hora programada, lista de transacciones y dll`s o capas lógica que son las que tienen el servicio

Cuadro 4-1 Actor - Aplicativo

Actor:	**Base datos seguridad**
Caso de Uso:	Realizar consulta, Cargar transacción horario, cargar transacción
Tipo:	Secundario
Descripción:	Es un actor secundario y representa a la base de datos donde se guarda toda la información relacionada con los horarios de trabajo de las transacciones

Cuadro 4-2 Actor - Base datos seguridad

Casos de Uso

Caso de Uso:	**Cargar transacción horario**
Actores	Aplicativo
Tipo:	Inclusión
Propósito:	Cargar las transacciones que se programen con un horario de trabajo
Resumen	Este caso de uso es iniciado por el propio aplicativo para poder definir si una transacción especifica tiene su configuración de hora
Precondiciones	Cada transacción debe ser creada en la base de datos seguridad definiendo el código de aplicativo, transacción y función que deben ser únicos, y luego de ser creada la transacción se debe

	configurar la transacción en el grupo de transacciones regidas por hora de trabajo, para que salga en la respectiva lista.
Flujo Principal	Al iniciar el motor transaccional realiza una consulta de transacciones programadas a trabajar en un horario específico. Si existen transacciones configuradas, almacena el horario en memoria para luego ser utilizado al momento de tratar de ejecutar una transacción especifica. Si no existen transacciones programadas al realizar la consulta sigue el proceso de cargada de transacciones si condición de horario de trabajo.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo. Cuando las claves principales de la transacción son duplicadas se emite el mensaje de excepción y se detiene el flujo de cargada.

Cuadro 4-3 Caso de uso - Cargar transacción horario

Caso de Uso:	**Realizar Consulta**
Actores	Aplicativo, base de datos seguridad, cargar transacción horario, cargar transacción
Tipo:	Inclusión
Propósito:	Permite al aplicativo consultar transacciones configuradas con horario y sin horario.
Resumen	Este caso de uso es iniciado por el propio aplicativo para cargar consultar las transacciones disponibles
Precondiciones	Es necesario haber ejecutado los casos de uso cargar transacción horario, cargar transacción
Flujo Principal	Proporcionar una lista de transacciones disponibles para trabajo.

Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo.

Cuadro 4-4 Caso de uso - Realizar Consulta

Caso de Uso:	**Cargar transacción**
Actores	Aplicativo
Tipo:	Inclusión
Propósito:	Cargar las transacciones que no tienen un horario de condición de trabajo
Resumen	Este caso de uso es iniciado por el propio aplicativo para cargar en memoria una lista de transacciones de trabajo
Precondiciones	Ser ingresada en la base de datos seguridad con un código de aplicativo, transacción y función que debe ser único en la base de datos.
Flujo Principal	Al iniciar el motor transaccional realiza una consulta de transacciones que pueden trabajar sin horario. Las transacciones debidamente creadas, serán almacenadas en memoria para definir las transacciones básicas de trabajo. Realizado la carga de transacciones, continua con la carga de dll`s.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo. Cuando las claves principales de la transacción son duplicadas se emite el mensaje de excepción y se detiene el flujo de cargada.

Cuadro 4-5 Caso de uso - Cargar transacción

Caso de Uso:	**Cargar DLL**
Actores	Aplicativo
Tipo:	Inclusión
Propósito:	Cargar la capa lógica que serán invocadas por el motor transaccional a través de los endpoint´s que son los encargados de prestar los servicios
Resumen	Este caso de uso es iniciado por el propio aplicativo para iniciar la carga de las dll´s
Precondiciones	Haber sido definido como servicio y estar plenamente identificadas sus operaciones.
Flujo Principal	Cuando se cargo las transacciones de horario y sin horario se realiza una lectura de la DLL lógica. Se identifica configuración de servicio y operación Se recorre configuración de interface y se levanta servicio de acurdo al archivo configuración del aplicativo.
Excepciones	Si el servicio no es plenamente configurado, se emite el mensaje de que la transacción no posee configuración correcta de parámetros.

Cuadro 4-6 Caso de uso - Cargar DLL

Caso de Uso:	**Cargar Servicio**
Actores	Aplicativo
Tipo:	Inclusión
Propósito:	Poner en escucha los servicios en TCP
Resumen	Este caso de uso es iniciado por el propio aplicativo para iniciar la carga de los servicios.
Precondiciones	Debidamente configurados en archivo de configuración, identificando. • Dirección IP y puerto escucha para el servicio • Nombre de servicio • Interface de servicio

	Las dll`s que vas a ser leídas deben poseer los requerimiento y condiciones de un servicio a prestar
Flujo Principal	El aplicativo realizara la lectura de las dll`s y de acuerdo a la configuración de archivo ira levantado el Endpoint con el servicio relacionado.
Excepciones	Cuando no se definió la respectiva configuración en el archivo, se emitirá el mensaje de servicio no configurado para trabaja.

Cuadro 4-7 Caso de uso - Cargar Servicio

4.2.1.2 Diagrama de Caso de Uso Inicio de sesión en Motor Transaccional

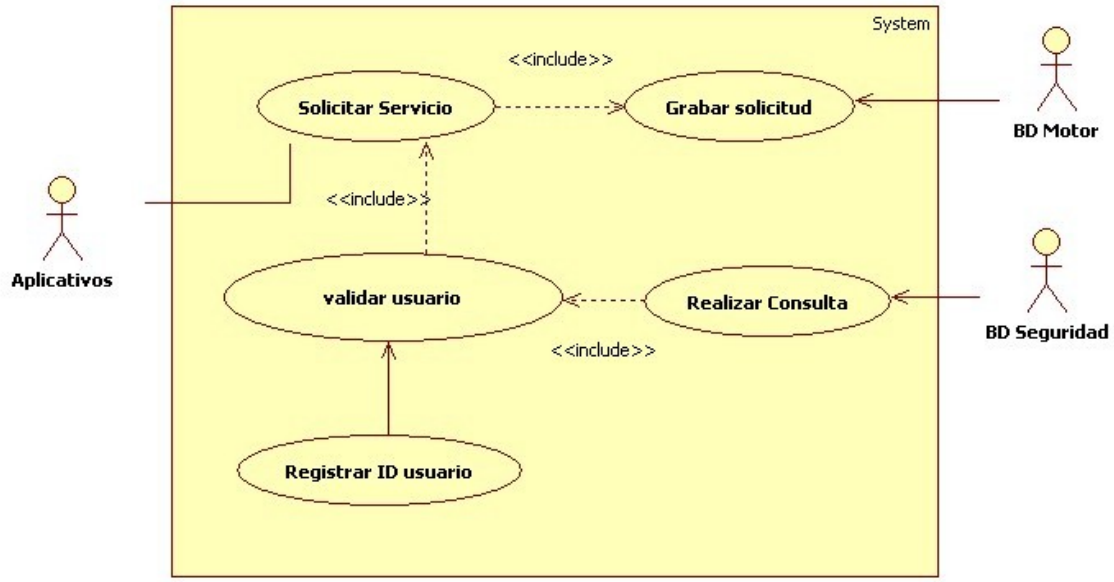


Grafico 4-3 Inicio de sesión en Motor Transaccional

Detalle diagrama Casos de Uso - Inicio de sesión en Motor Transaccional

Actores

Actor:	**Aplicativo**
Caso de Uso:	Solicitar Servicio, validar usuario
Tipo:	Primario
Descripción:	Es el actor principal y representa al usuario cuando realiza la petición de login por primera vez

Cuadro 4-8 Actor - Aplicativo

Actor:	**Base datos seguridad**
Caso de Uso:	Realizar consulta, Validar usuario
Tipo:	Secundario
Descripción:	Es un actor secundario y representa a la base de datos donde se encuentran almacenados los usuarios.

Cuadro 4-9 Actor - Base datos seguridad

Actor:	**Base datos motor**
Caso de Uso:	Grabar solicitud
Tipo:	Secundario
Descripción:	Es un actor secundario y representa a la base de datos donde se grabara la información de solicitudes de transacciones.

Cuadro 4-10 Actor - Base datos motor

Casos de Uso

Caso de Uso:	**Solicitar servicio**
Actores	Aplicativo
Tipo:	Inclusión
Propósito:	Identificar la transacción que se está solicitando
Resumen	Este caso de uso es iniciado por el propio aplicativo para poder definir si una transacción especifica se encuentra definido y hábil para su uso.
Precondiciones	El motor transaccional se debe encontrar en ejecución.

	El servicio debe estar en escucha.
Flujo Principal	El aplicativo inicia la petición de login enviando el usuario y la clave solicitante. Se inicia la identificación de la transacción solicitada. Se graba los datos de la petición en la base de datos del monitor transaccional. Se realiza la consulta de los datos recibidos y se procede a realizar la validación Validado los datos se procede a guardar en memoria el id usuario para conservar la identificación y relación de peticiones de transacciones.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo. Cuando el servicio no existe o no este en escucha se emitirá el mensaje de servicio no disponible.

Cuadro 4-11 Caso de uso - Solicitar servicio

Caso de Uso:	**Validar usuario**
Actores	Aplicativo, solicitar servicio
Tipo:	Inclusión
Propósito:	Validar a un usuario ya registrado para que éste pueda hacer uso de los diferentes servicios
Resumen	Este caso de uso es iniciado por el Usuario. El usuario ingresa su nombre de usuario y su contraseña, los cuales son comparados con el nombre de usuario y contraseña registrados anteriormente, si es correcto puede acceder a los diferentes servicios.
Precondiciones	Si el Usuario aún no se ha registrado, requerirá ser ingresado en la base de datos de seguridad
Flujo Principal	Se identifica porque servicio está solicitando (login o petición de transacción). Si la petición es Login, se ejecuta el caso de uso realizar consulta Al verificar los datos correctos se ejecuta el caso de Registrar ID usuario.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo. Cuando el usuario no existe en la base de datos de seguridad, sea este usuario o clave.

Cuadro 4-12 Caso uso - Validar usuario

Caso de Uso:	**Grabar Solicitud**
Actores	Base de datos monitor , solicitar servicio
Tipo:	Inclusión
Propósito:	Grabar las peticiones la motor transaccional, almacenando el usuario que lo solicita
Resumen	Este caso de uso es iniciado por el caso de uso Solicitar servicio y es el encargado de almacenar todos las peticiones al motor transaccional (consultas, y transacciones)
Precondiciones	Para el uso de todos los servicios debe estar plenamente logiado y asignado ID de sesión.
Flujo Principal	Se recibe los datos de la petición. • Username • Id usuario • Aplicación • Transacción • Función de ejecución. Si la petición es de inicio de sesión el Id usuario es cero, esto es por el caso de identificado.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo.

Cuadro 4-13 Caso de uso - Grabar Solicitud

Caso de Uso:	**Registrar Id usuario**
Actores	Validar Usuario
Tipo:	Inclusión
Propósito:	Conservar el ID usuario en la memoria del motor transaccional
Resumen	Este caso de uso es iniciado por el caso de uso Validar usuario cuando el login de usuario fue correcto y almacenará en la memoria del motor para su respectiva relación de petición de usuario solicitante y usuario registrado, para el uso de la transacción o consulta.
Precondiciones	Que el caso de uso Validar usuario envía el id usuario para su respectivo almacenamiento.
Flujo Principal	Recibe los datos de acceso y almacena en memoria el usuario
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo.

Cuadro 4-14 Caso uso - Registrar Id usuario

Diagrama de Caso de Uso Consumir Servicio.

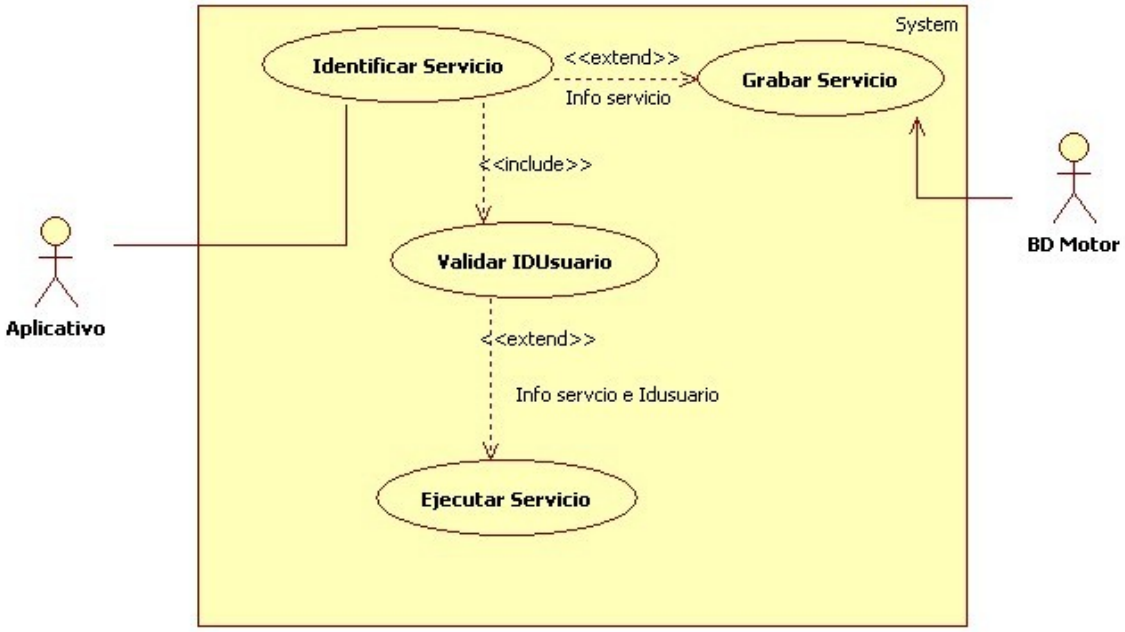


Grafico 4-4 Caso de Uso - Consumir Servicio

Detalle Diagrama Casos de Uso- Consumir Servicio

Actores

Actor:	**Aplicativo**
Caso de Uso:	Identificar Servicio, Validar Id Usuario
Tipo:	Primario
Descripción:	Es el actor principal y representa al aplicativo cuando realiza la petición de un servicio.

Cuadro 4-15 Actor – Aplicativo

Actor:	**Base datos motor**
Caso de Uso:	Grabar solicitud
Tipo:	Secundario
Descripción:	Es un actor secundario y representa a la base de datos donde se grabara la información de solicitudes de transacciones y el responsable que invoco al servicio.

Cuadro 4-16 Actor - Base datos motor

Casos de Uso

Caso de Uso:	**Identificar Servicio**
Actores	Aplicativo
Tipo:	Inclusión
Propósito:	Identificar la transacción que se está solicitando
Resumen	Este caso de uso es iniciado por el propio aplicativo para poder definir si una transacción especifica se encuentra definido y hábil para su uso.
Precondiciones	El motor transaccional se debe encontrar en ejecución. El servicio debe estar en escucha. El servicio debe estar definido en algún protocolo Debe estar plenamente identificado con uno Id usuario
Flujo Principal	El aplicativo inicia la petición de consumo de servicio, enviando los parámetros de Id usuario (obligatorio) y los parámetros del servicio que desea consumir. Si no se envía el Id usuario, no se podrá hacer uso del servicio por no constar como identificado en el motor transaccional.

	Se graba los datos de la petición en la base de datos del monitor transaccional. Se ejecuta el servicio y se recibe la respuesta del servicio Se graba los datos de salida en la base de datos del monitor transaccional.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo. Cuando el servicio no existe o no este en escucha se emitirá el mensaje de servicio no disponible.

Cuadro 4-17 Caso de uso - Identificar Servicio

Caso de Uso:	**Grabar Solicitud**
Actores	Base de datos monitor , Identificar servicio
Tipo:	Extendido
Propósito:	Grabar las peticiones hechas al motor transaccional, almacenando el Usuario que lo solicita y a qué servicio se invoco
Resumen	Este caso de uso es iniciado por el caso de uso identificar servicio y es el encargado de almacenar todas las peticiones al motor transaccional (consultas, y transacciones) junto con el id usuario responsable de la petición.

Precondiciones	Para el uso de todos los servicios debe estar plenamente logiado y asignado ID de sesión si no lo esta se almacenara el intento de petición.
Flujo Principal	Se recibe los datos de la petición. • Username • Id usuario • Aplicación • Función de ejecución. Si la petición es de inicio de sesión el Id usuario es cero, esto es por el caso de identificado. Si el Id usuario es diferente de cero se almacena los datos de la petición junto con el id usuario.
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo.

Cuadro 4-18 Caso de uso - Grabar Solicitud

Caso de Uso:	**Validar Usuario**
Actores	Caso de uso Identificar servicio
Tipo:	Inclusión
Propósito:	Identificar el responsable de petición de servicio
Resumen	Este caso de uso es iniciado por el caso de uso identificar servicio y es el encargado de identificar el responsable de la petición en base a su id usuario previamente asignado en su petición de de inicio de sesión
Precondiciones	Debe estar asignado un inicio de sesión y debidamente

	identificado en el motor transaccional para hacer uso de los servicios
Flujo Principal	Se recibe los datos de la petición. • Username • Id usuario Se verifica que este registrado en el motor transaccional (iniciado sesión) Si se encuentra debidamente identificado en el motor transaccional, permite continuar en la invocación del servicio Si no se encuentra debidamente identificado en el motor transaccional, no se permitirá el paso a la ejecución del servicio y se revocara la petición
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo.

Cuadro 4-19 Caso de uso - Validar Usuario

Caso de Uso:	**Ejecutar Servicio**
Actores	Caso de uso Validar Id usuario
Tipo:	Extendido
Propósito:	Invocar el servicio solicitado
Resumen	Este caso de uso se ejecuta cuando el caso de uso Validar Id usuario ha sido satisfactorio y por consecuencia permite su ejecución y es el encargado de invocar los servicios disponibles por el motor transaccional

Precondiciones	Debe estar asignado un inicio de sesión y haber sido validado por el caso de uso Validar Id usuario par su respectiva ejecución.
Flujo Principal	Se recibe los datos de la petición. • Username • Id usuario • Aplicativo • Transacción • Función Se verifica la existencia de la transacción solicitada Se verifica la disponibilidad del servicio. Se ejecuta la invocación del servicio Se recibe la respuesta del servicio
Excepciones	Cuando no existe conexión: Cuando se pierde la conexión con el motor transaccional , se mostrará un mensaje de error de conexión con el motor Cuando no existe conexión: Cuando se pierde la conexión con la base de datos, se mostrará un mensaje de error de conexión en la consola y se detendrá el flujo.

Cuadro 4-20 Caso de uso - Ejecutar Servicio

4.2.2 Diagrama de secuencia

El diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En donde se muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

4.2.2.1 Diagrama de secuencia Validar Usuario

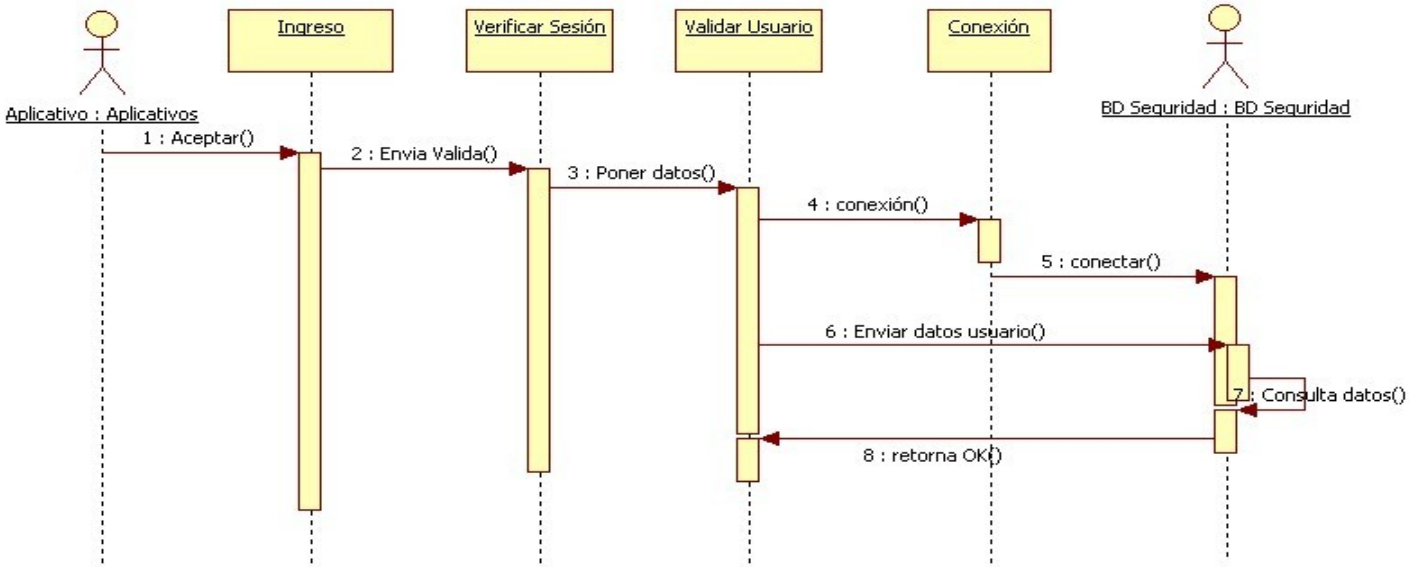


Grafico 4-5 Diagrama de secuencia - Validar Usuario

4.2.2.2 Diagrama de Secuencia Solicitar Servicio

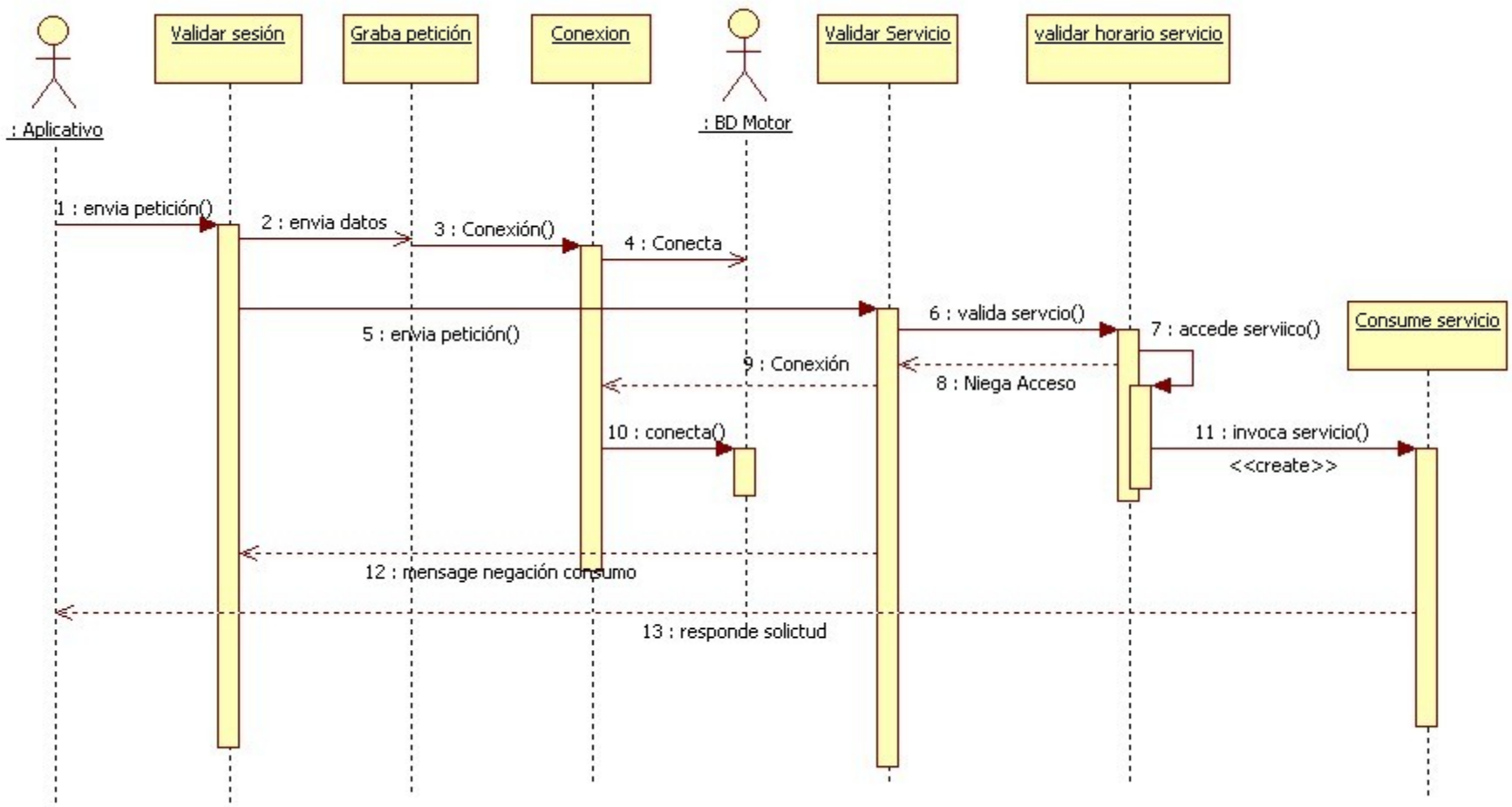


Grafico 4-6 Diagrama de Secuencia Solicitar Servicio

4.2.2.3 Diagrama de Secuencia Graba Servicio

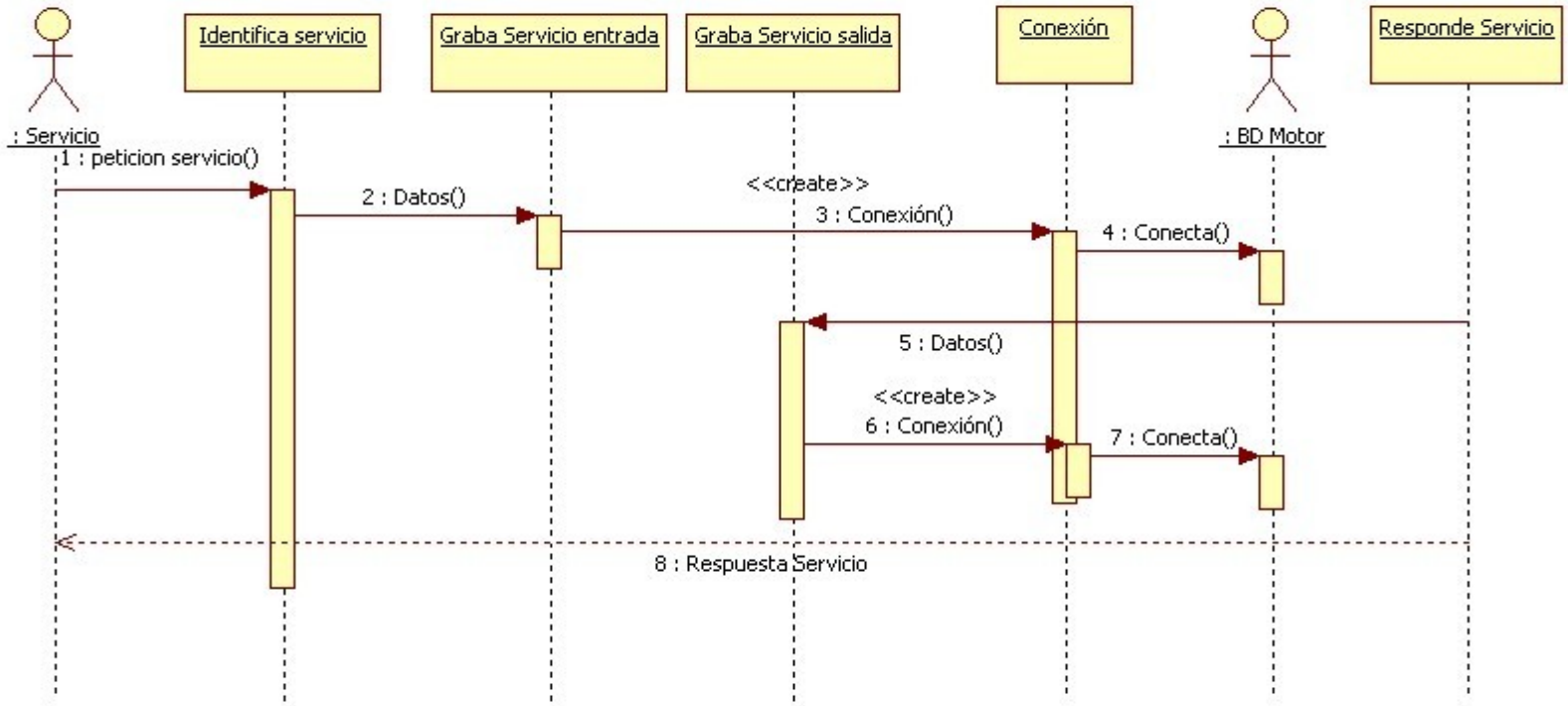


Grafico 4-7 Diagrama de Secuencia Graba Servicio

4.2.2.4 Diagrama de Secuencia Registra Id Usuario

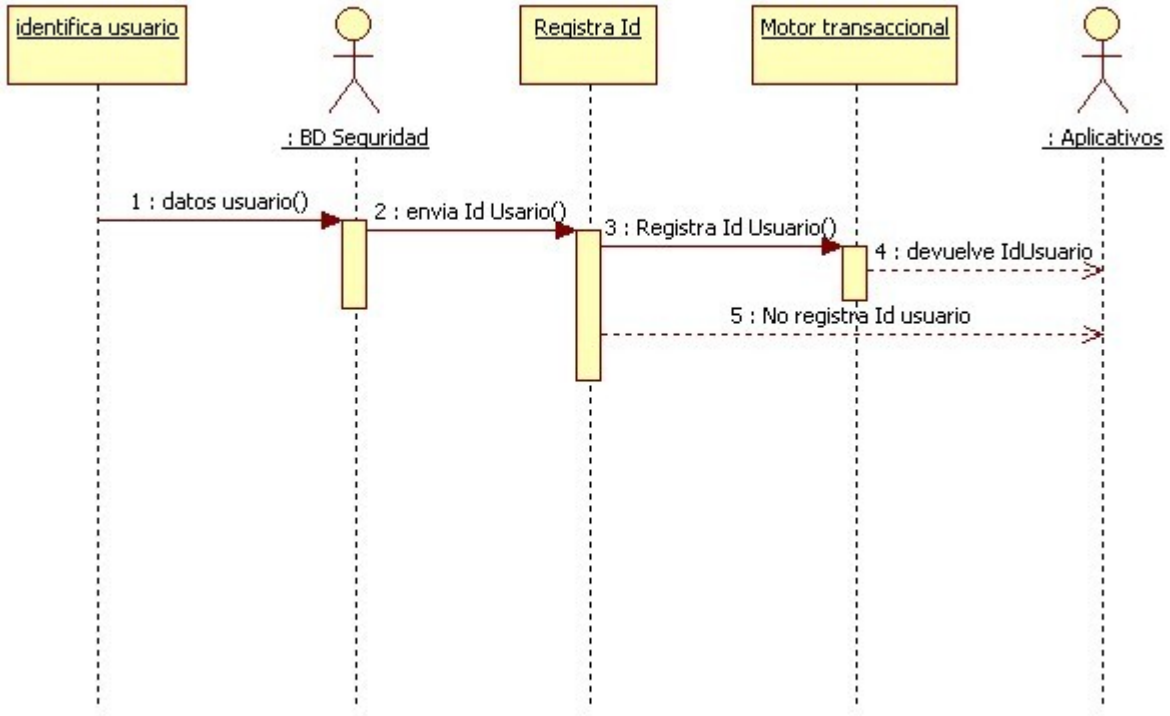


Grafico 4-8 Diagrama de Secuencia Registra Id Usuario

4.2.2.5 Diagrama de Secuencia Realizar Consulta Usuarios

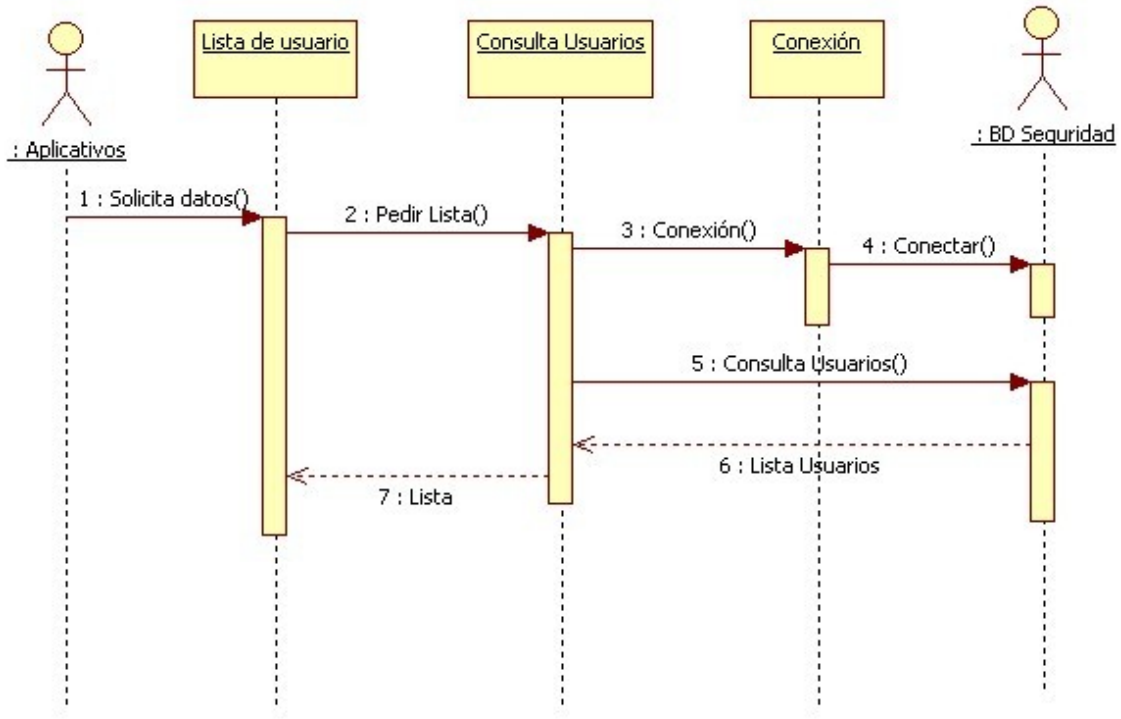


Grafico 4-9 Diagrama de Secuencia Realizar Consulta Usuarios

4.2.2.6 Diagrama de Secuencia Cargar Transacción Hora

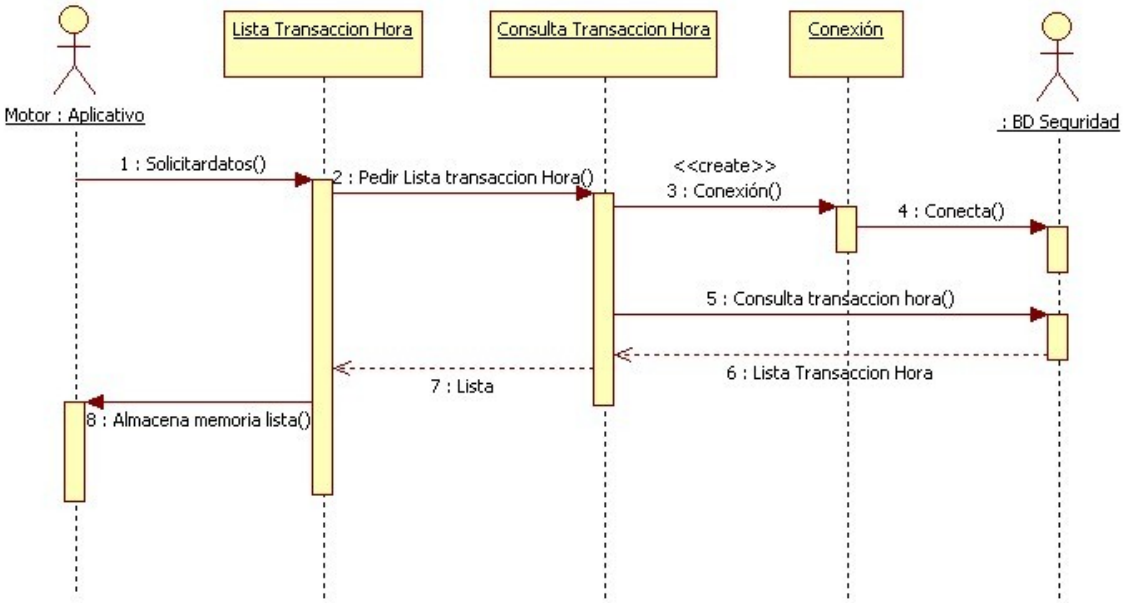


Grafico 4-10 Diagrama de Secuencia Cargar Transacción Hora

4.2.2.7 Diagrama de Secuencia Cargar Transacción

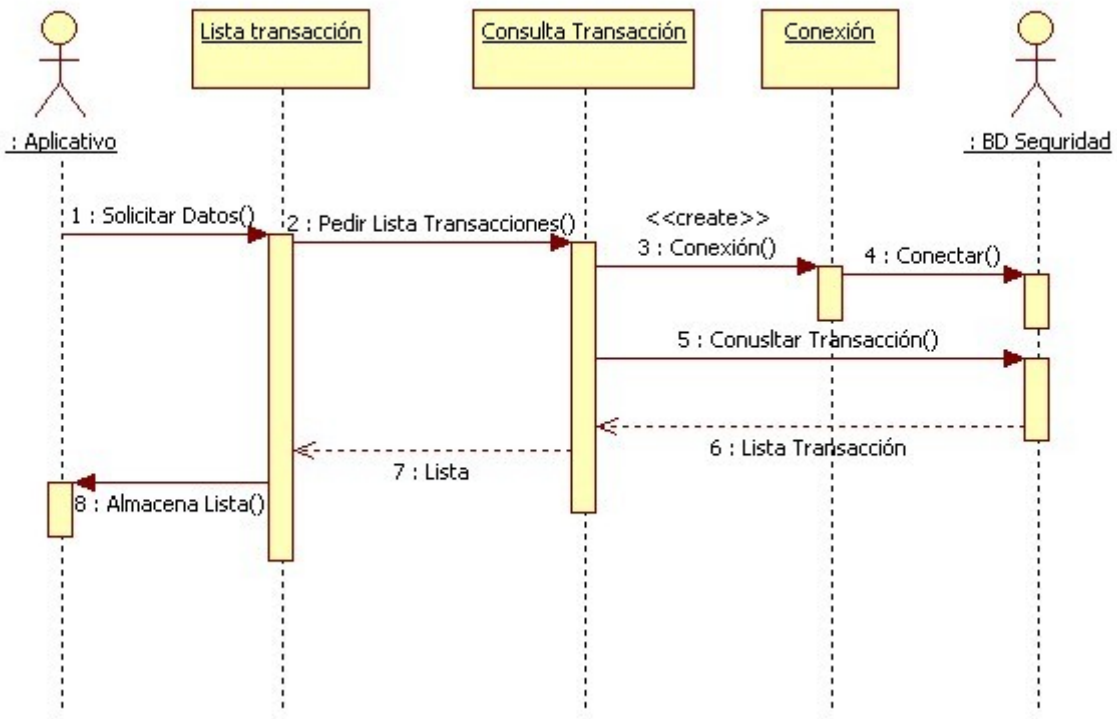


Grafico 4-11 Diagrama de Secuencia Cargar Transacción

4.2.3 Diagrama de colaboración

El Diagrama de Colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los Diagramas de Secuencia, los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia.

4.2.3.1 Diagrama de colaboración Validar Usuario

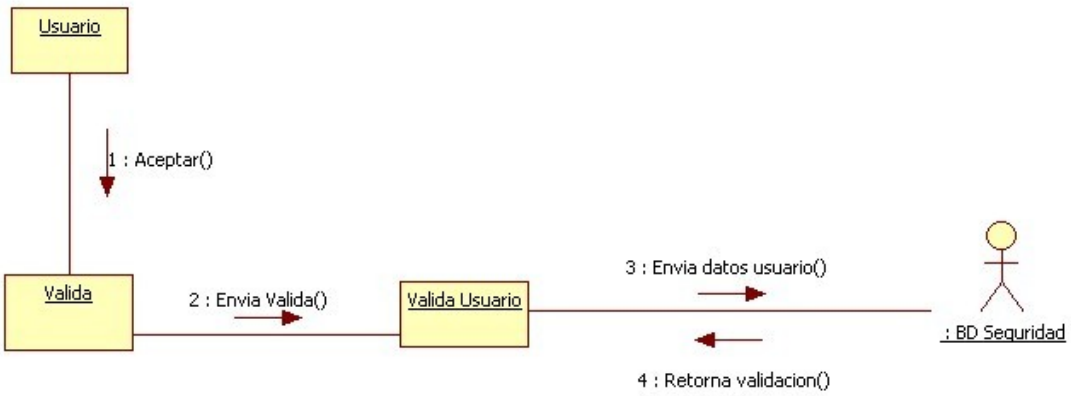


Grafico 4-12 Diagrama de colaboración - Validar usuario

4.2.3.2 Diagrama de colaboración Solicitar Servicio

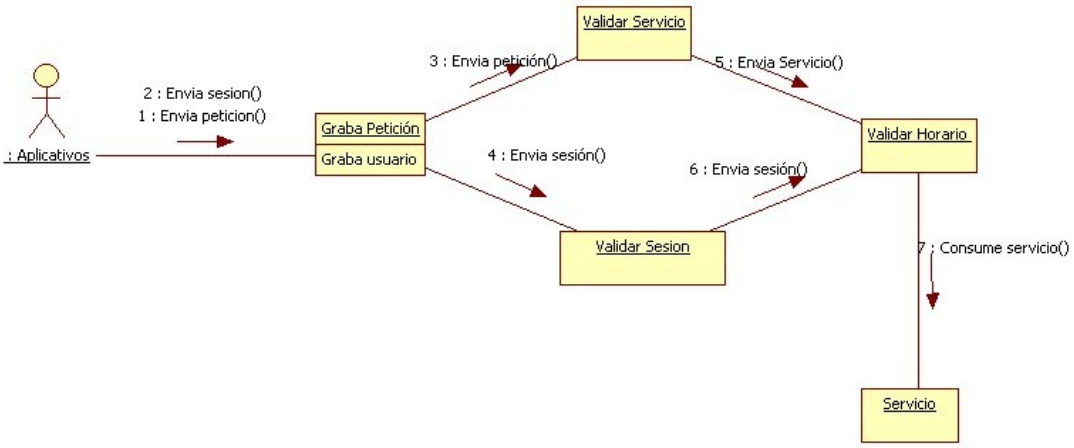


Grafico 4-13 Diagrama de colaboración - Solicitar Servicio

4.2.3.3 Diagrama de colaboración graba servicio

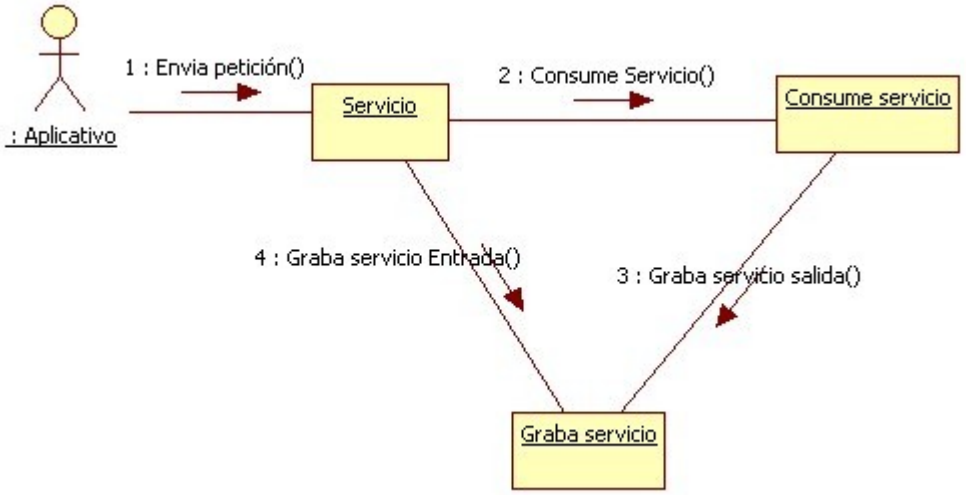


Grafico 4-14 Diagrama de colaboración - graba servicio

4.2.3.4 Diagrama de colaboración Registra Id usuario

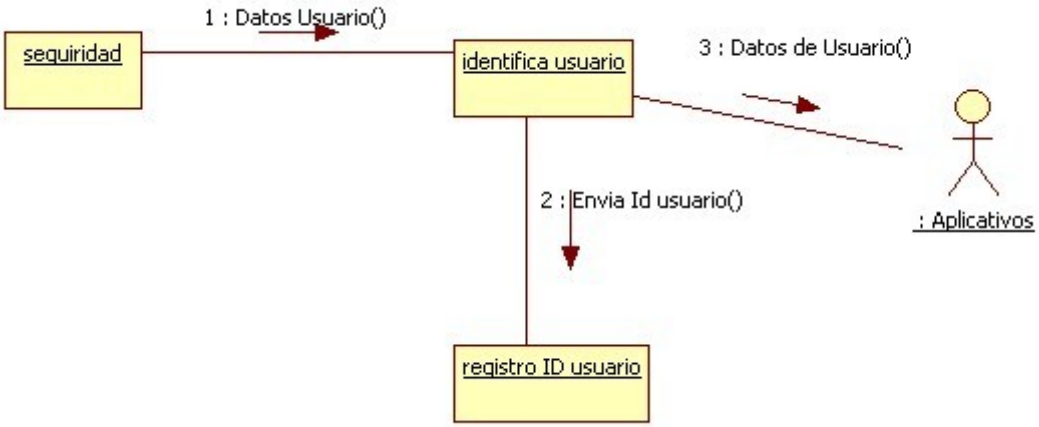


Grafico 4-15 Diagrama de colaboración Registra Id usuario

4.2.3.5 Diagrama de colaboración Consulta usuario

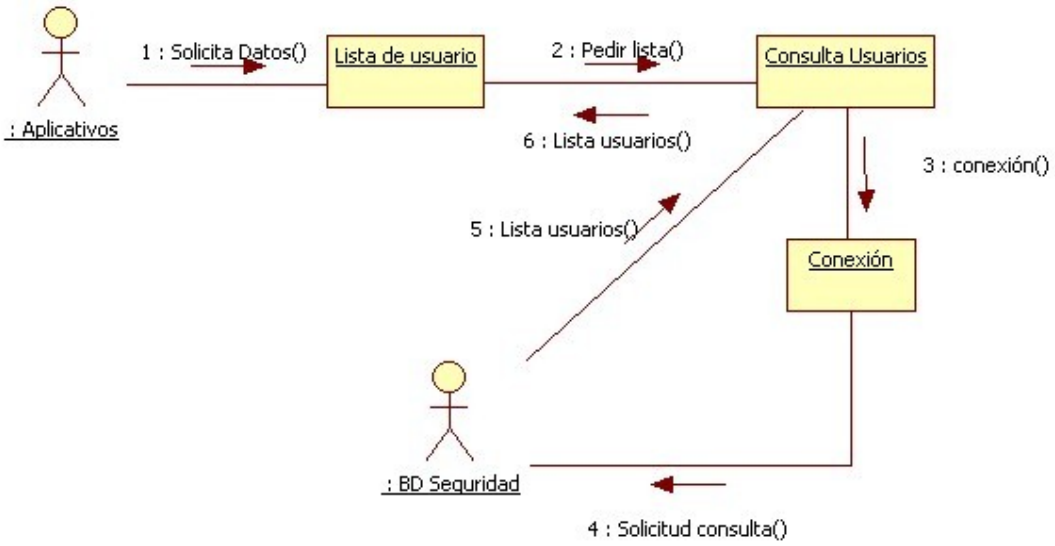


Grafico 4-16 Diagrama de colaboración Consulta usuario

4.2.3.6 Diagrama de colaboración carga transacción hora

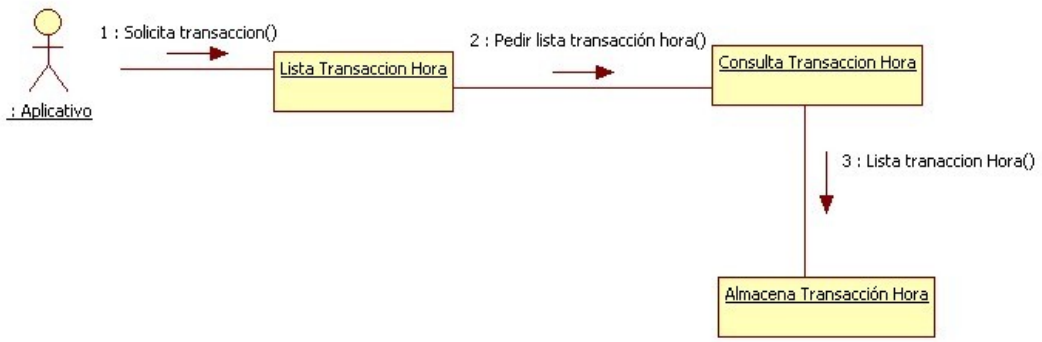


Grafico 4-17 Diagrama de colaboración carga transacción hora

4.2.3.7 Diagrama de colaboración carga transacción

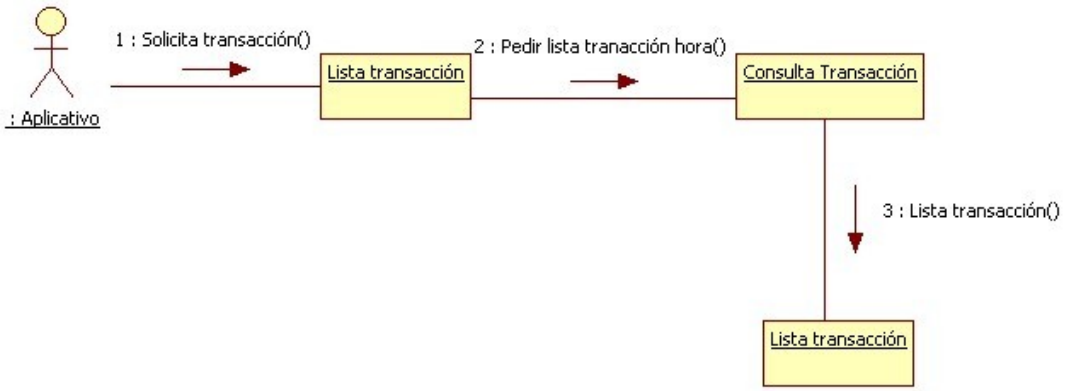


Grafico 4-18 Diagrama de colaboración carga transacción

4.2.4 Diagrama de actividad

Los diagramas de actividad describen la secuencia de las actividades en un sistema.

4.2.4.1 Diagrama de actividad cargar servicio

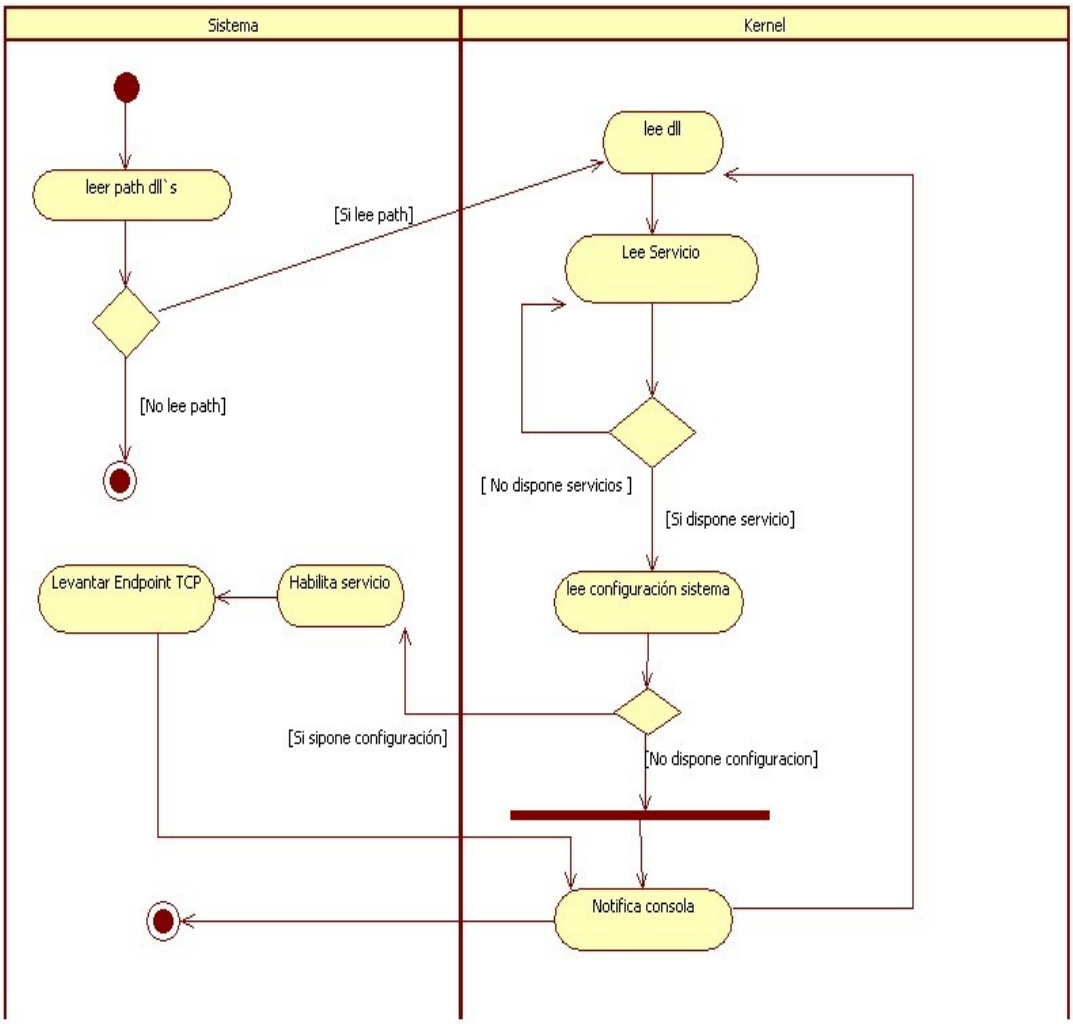


Grafico 4-19 Diagrama de actividad cargar servicio

4.2.4.2 Diagrama de actividad Iniciar sesión servicio

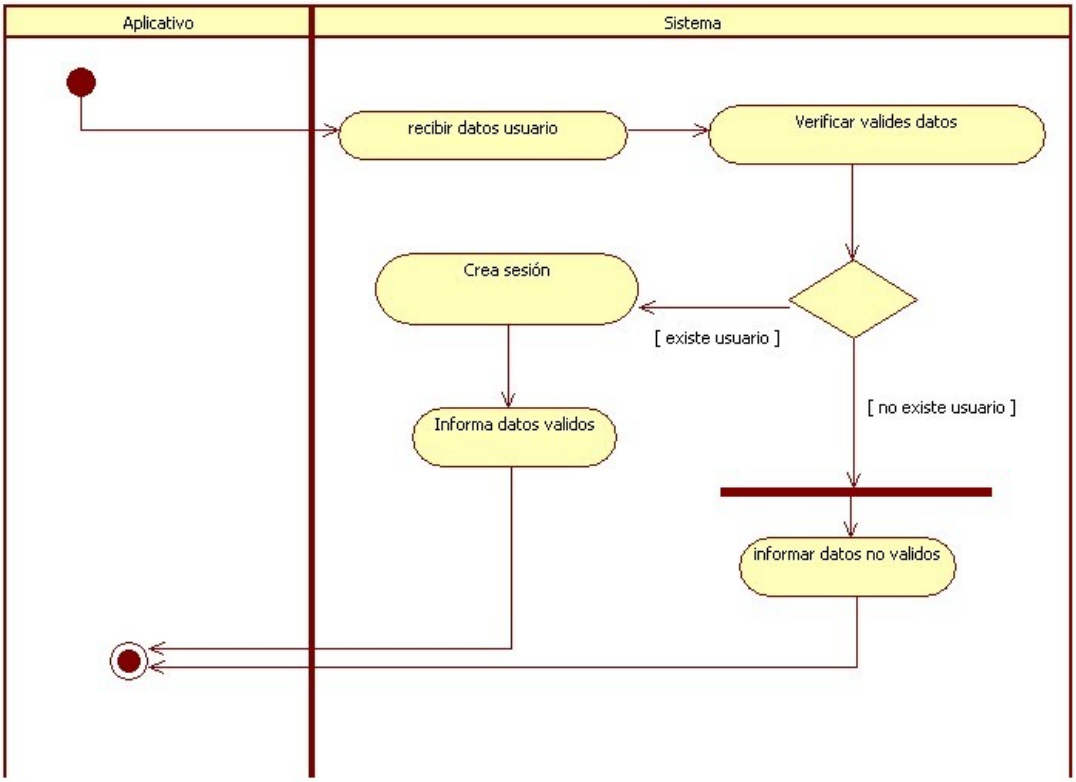


Grafico 4-20 Diagrama de actividad Iniciar sesión servicio

4.2.4.3 Diagrama de actividad Consumir servicio

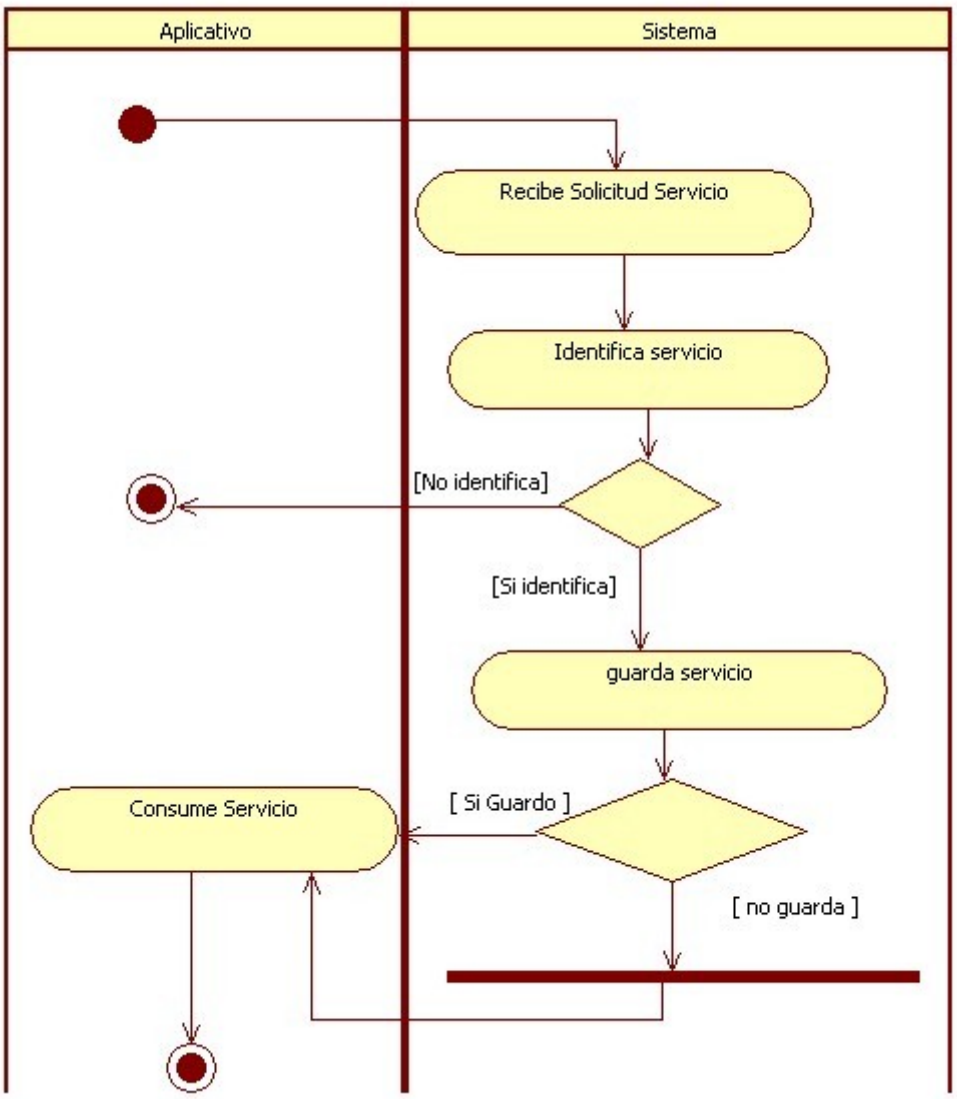


Grafico 4-21 Diagrama de actividad Consumir servicio

4.2.5 Modelo de diseño

4.2.5.1 Modelo de componente

Se modela los diagramas de componente, los cuales están enfocados a la implementación del sistema, donde se identifica las capas, paquetes, componentes y su relación.

4.2.5.2 Diagrama de despliegue

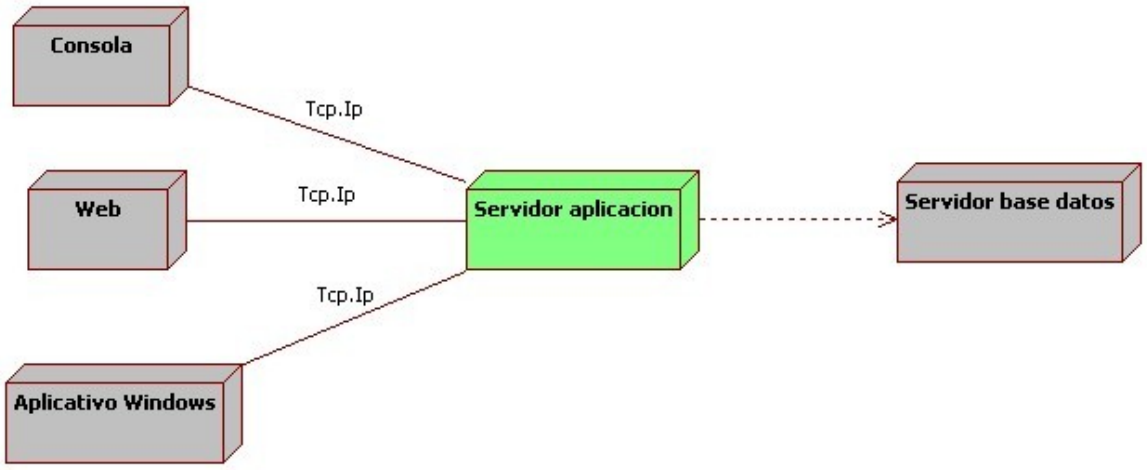


Grafico 4-22 Diagrama de despliegue

4.2.5.3 Diagrama de arquitectura

Grafico 4-23 Diagrama de arquitectura

4.2.5.4 Diagrama de componentes

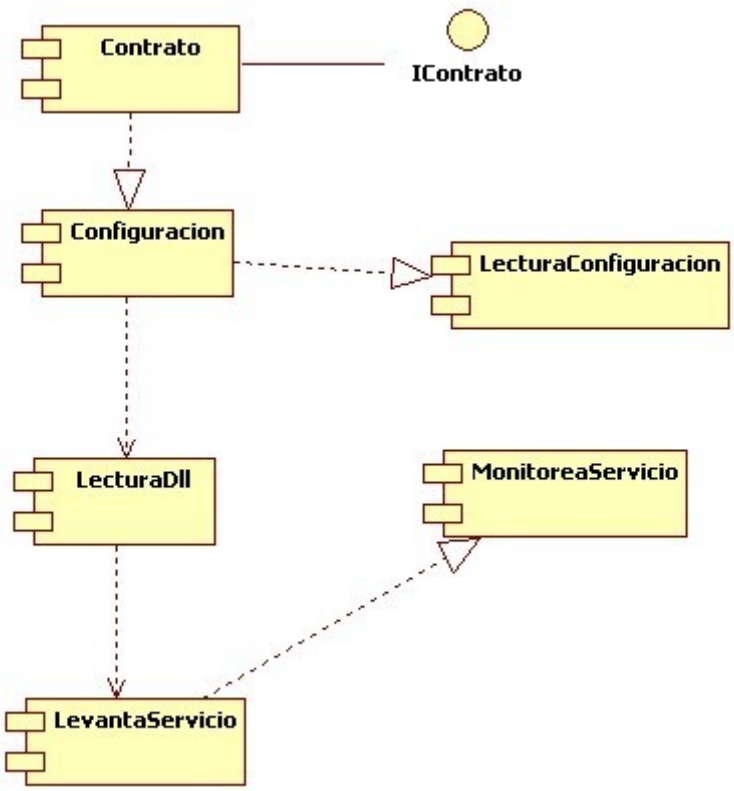


Grafico 4-24 Diagrama de componente

4.2.6 Modelo de clases

4.2.6.1 Diagrama de clases - diseño

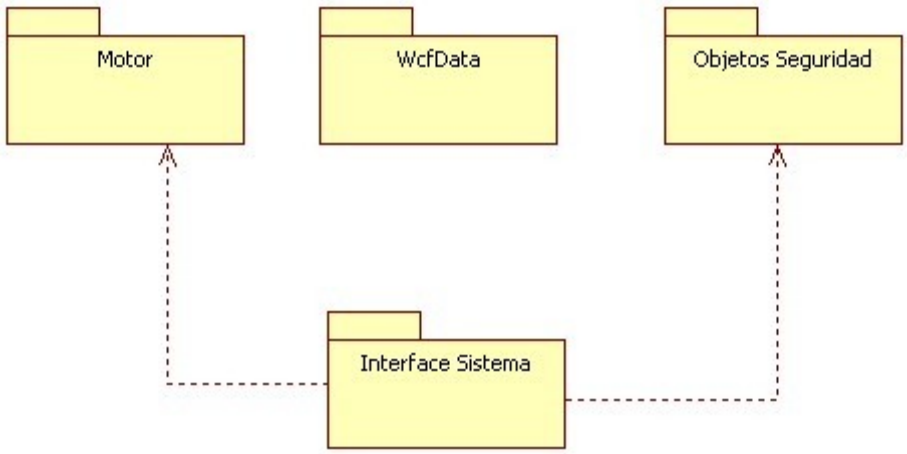


Grafico 4-25 Diagrama de clase – Diseño

4.2.6.2 Diagrama de clase – Paquete motor

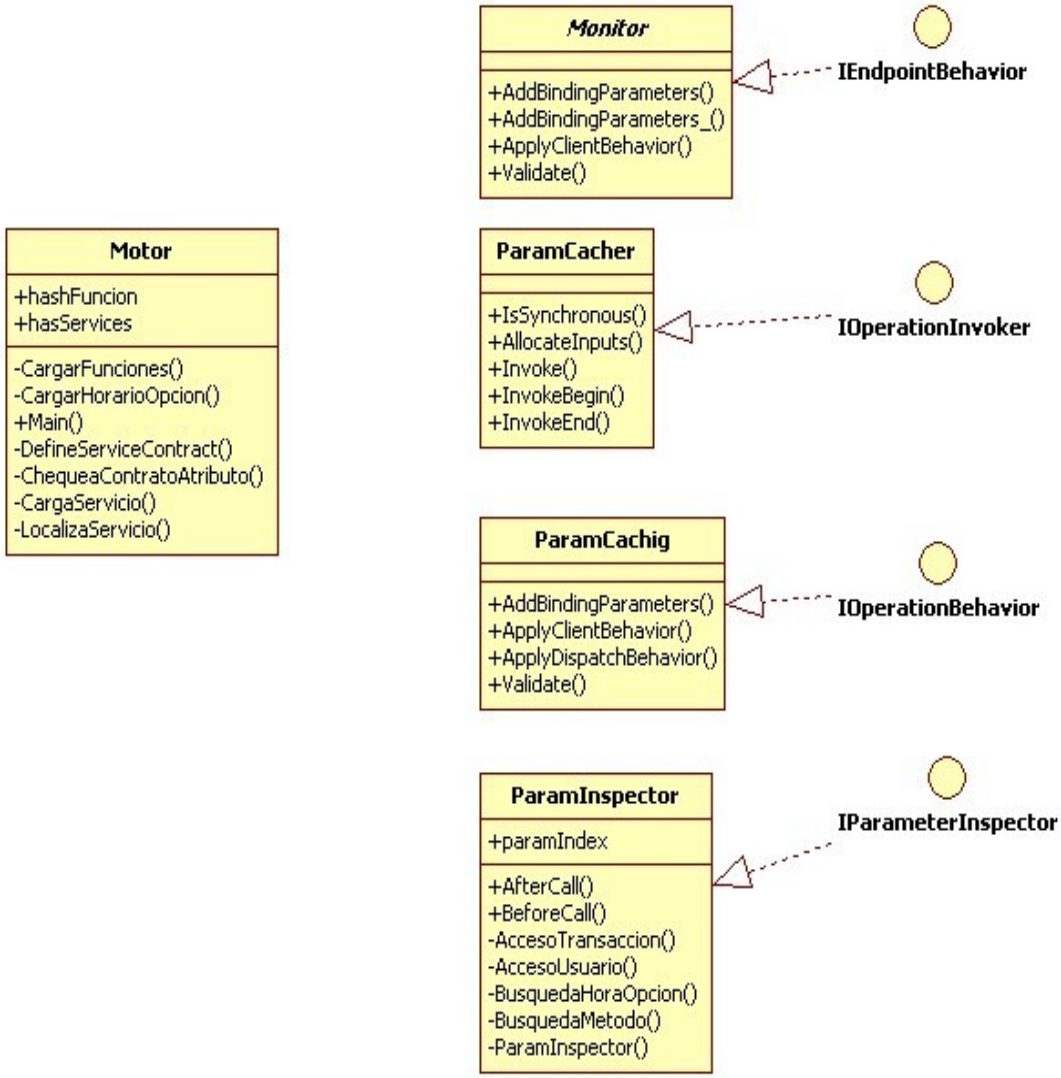


Grafico 4-26 Diagrama de clase – Paquete motor

4.2.6.3 Diagrama de clase – Paquete WcfData

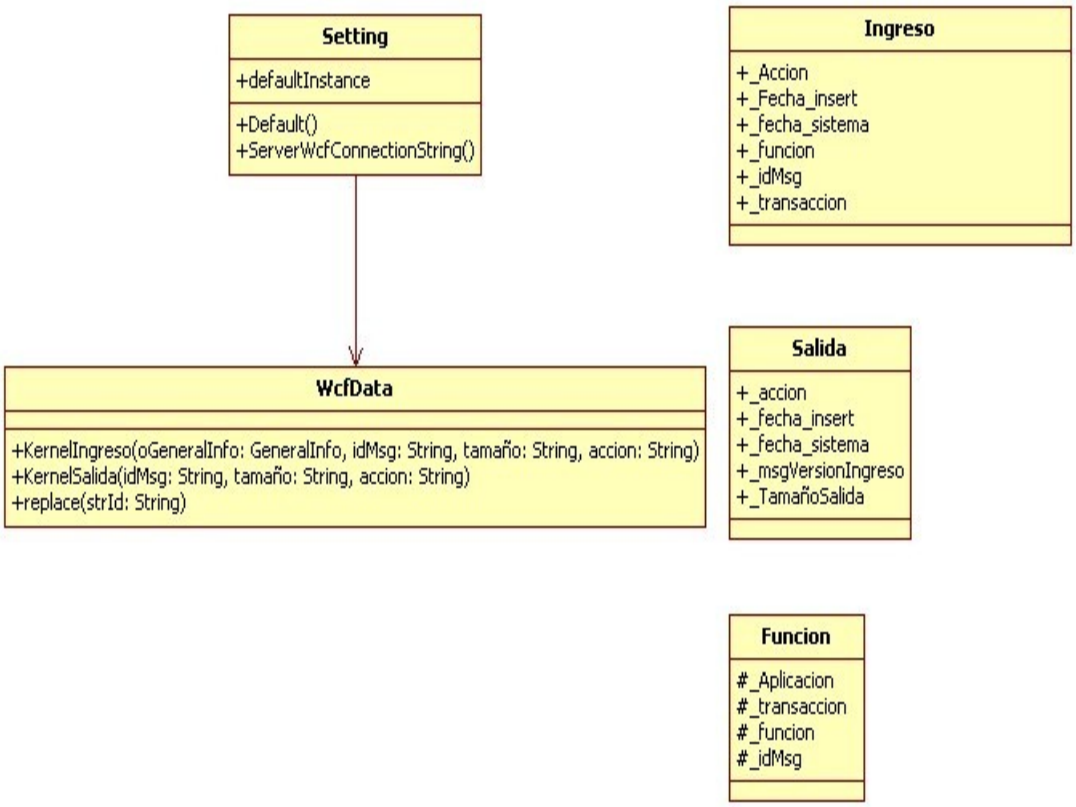


Grafico 4-27 Diagrama de clase – Paquete WcfData

4.2.6.4 Diagrama de clase – Paquete objetos seguridad

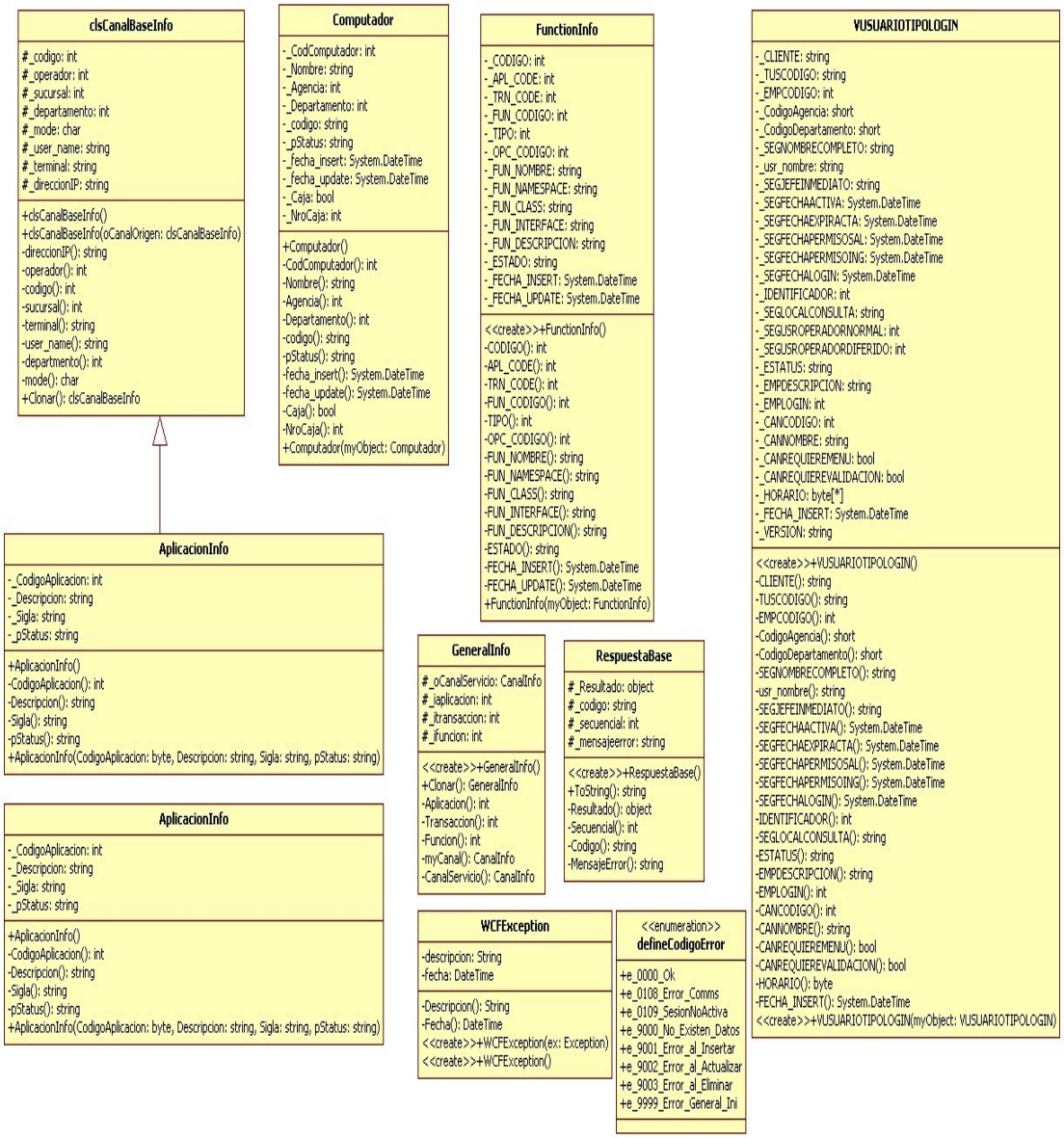


Grafico 4-28 Diagrama de clase – Paquete objetos seguridad

4.2.6.5 Diagrama clase – Interface sistema

Grafico 4-29 Diagrama clase – Interface sistema

4.3 Fase de construcción

Durante la fase de construcción, el foco del producto se mueve de la arquitectura de base a un sistema lo suficientemente completo como para llevarlo al usuario. El modelo de la arquitectura crece en complejidad y se convierte en un sistema completo.

4.3.1 Base de datos transacción motor

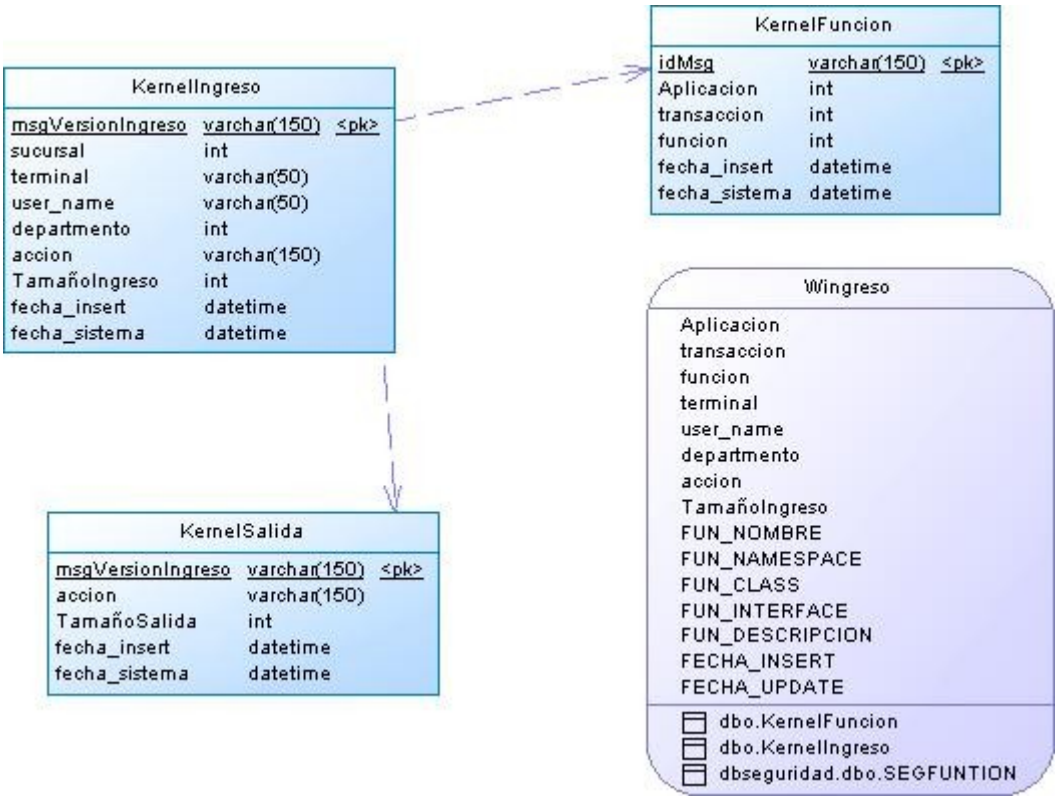


Grafico 4-30 Base de datos motor

4.3.2 Base de datos Seguridad

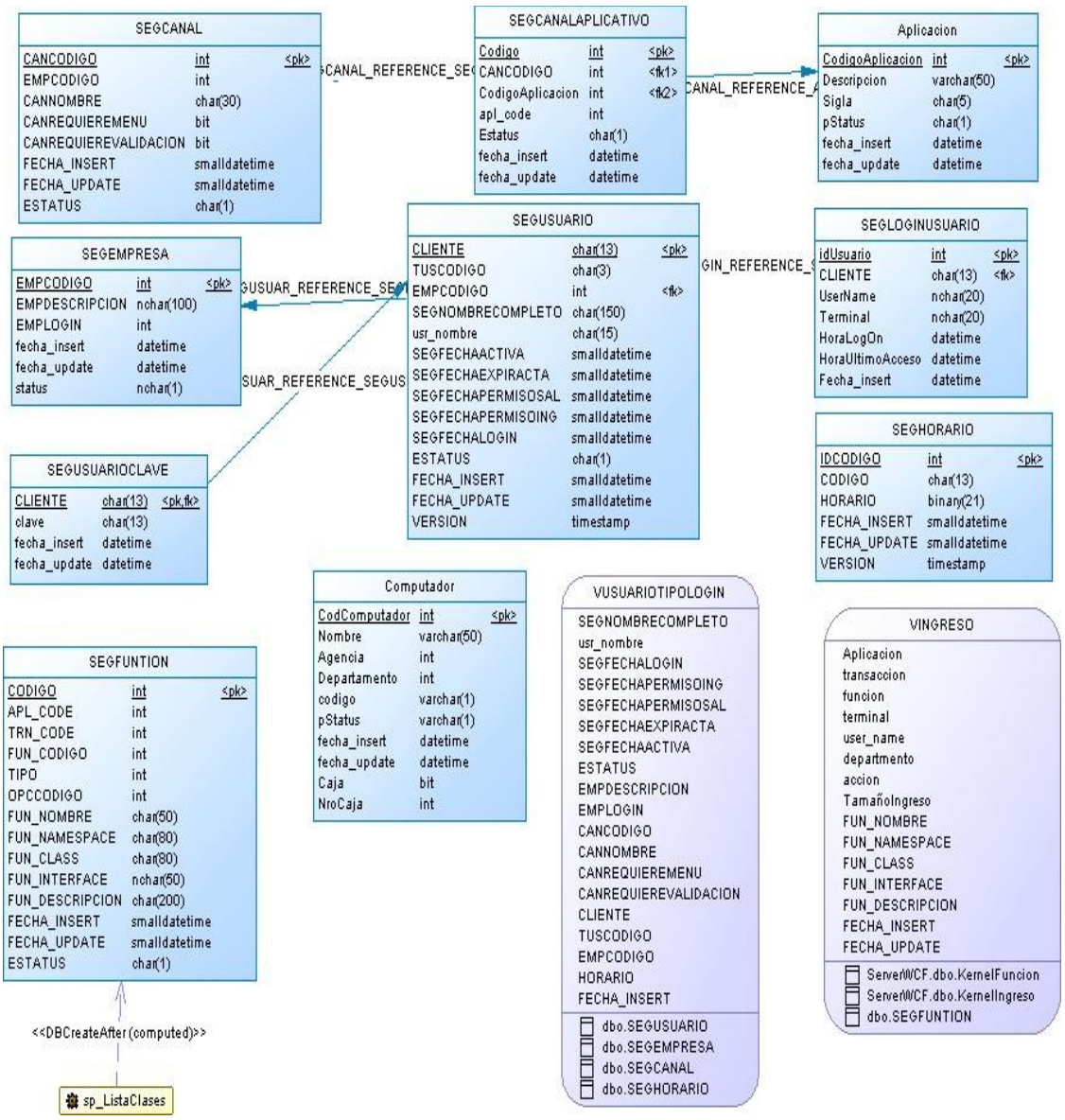


Grafico 4-31 Base de datos seguridad

4.3.3 Pruebas de sistema.

Para las pruebas del software desarrollado se utilizó el paradigma de pruebas tipo caja negra, y caja blanca, y para probar el rendimiento del software se utilizo pruebas de unidad, ya que el objetivo que se busco, fue el de asegurar que el sistema fuera capaz de levantar un servicio y recibir una determinada peticiones de servicio y realizar su respectiva respuesta.

4.3.3.1 Pruebas caja negra

Nro.	Nombre	Entrada	Salida	Resultado
1	Ingreso sistema	Datos usuario	Pantalla menú	Ok
2	Ingreso función	Datos función	Datos función	Ok
3	Ingreso horario función	Función-hora trabajo	Datos hora-función	Ok
4	Ingreso usuario	Usuario-hora	Datos hora-usuario	Ok
5	Vista log	Fecha consulta	Log's	Ok
6	Filtro log	Usuario-fecha	Log's	Ok
7	Ingreso computador	Datos computador	Datos computador	Ok
8	Levantar configuración	Configuración	Datos configuración	Ok
9	Levantar Horario	Configuración	Datos Horario trabajo	Ok

Cuadro 4-21 Pruebas caja negra

4.3.3.2 Pruebas caja blanca

Nro.	Nombre	Entrada	Salida	Resultado
1	Manipulación Base datos	Datos	Datos, creación, eliminación, actualización.	Ok
2	Validación sesión	Username, pc	Información validación	Ok
3	Consumo servicio	Solicitud servicio	Respuesta servicio	Ok
4	Control accesos	Usuario-hora	Control acceso	Ok

Cuadro 4-22 Pruebas caja blanca

4.3.3.3 Prueba de carga

Las pruebas se realizaron con la ayuda de Visual Estudio Team System 2008, orientado a pruebas de carga, es decir que están enfocadas en consumir servicios levantados por el motor transaccional.

Las pruebas de carga de datos, se realizaron en un computador Pentium Dual Core, velocidad de procesador de 2.20 GHZ y memoria RAM de 2 GB

En donde la cantidad de datos de rendimiento recopilados durante la prueba de carga se dio por la duración de la ejecución de prueba (30 minutos), del intervalo del muestreo, del número de equipos sometidos a prueba (25 virtuales), y del número de contadores que se utilizaron, que en este caso los identificadores más importantes para .Net son: el porcentaje de uso de la CPU, los recuentos de bloqueo de SQL Server.

Cada equipo virtual realiza su conexión y consumo de servicio, lo que la herramienta genera una media de resultados que son interpretados de la siguiente manera:

- ✓ Tiempo de prueba 30 minutos.
- ✓ Usuarios virtuales de carga, 25.
- ✓ Test fallidos 388, de 5000 iteraciones. Que equivale al 0.89% de las iteraciones.
- ✓ Media de transacciones por segundo. 0.012 (Avg. Transaction Time (sec))
- ✓ Transacciones SQL.
    - o Media de transacción ADD, 0.021
    - o Media de transacción SELECT, 0.018.
- ✓ Memoria utilizada por el equipo de prueba 690 mgb

El resultado de las pruebas muestra la flexibilidad del sistema para acoplarse a peticiones continuas, sin que afecte su rendimiento, ya que al mantener una media de trabajo en tiempo, relativamente bajo, beneficia en su uso en medio de futuras aplicaciones.

4.4 Fase de transición

En esta fase el objetivo es garantizar que los requisitos se han cumplido, con la satisfacción de las partes interesadas. Esta fase se inicia con una versión beta de la aplicación

4.4.1 Manual de usuario

Se realiza una guía enfocado al operador, en el que se detalla cómo utilizar el sistema de seguridad, como es pantallas, opciones, menús y configuraciones básicas para el correcto funcionamiento del modulo.

4.4.2 Manual técnico

De igual amanera se realiza una guía escrita con los estándares de configuración y programación para la creación de los servicios.

4.4.3 Plan de capacitación

En esta etapa se coordina con jefatura de área un plan de capacitación (horario, fechas, lugar, participantes) y como refuerzo de la capacitación se entrega el manual de usuario, así como el manual técnico a los que corresponda.

4.4.4 Plan de mantenimiento

Se establece un tiempo en que se empieza a dar un plan de mantenimiento al sistema, este puede incluir nuevas implementaciones, cambios sugeridos por los programadores u operadores en base a nuevos requerimientos.

5 Conclusiones y recomendación

5.1 Conclusiones

El objetivo fundamental de esta tesis es el proveer un sistema computacional que permita primero al desarrollar agilizar su trabajo en la implementación de aplicaciones, y a la organización el de disponer de una herramienta fiable para el control de aplicaciones, por lo que en la fase de análisis y desarrollo se obtiene el siguiente conjunto de conclusiones.

- ✓ La arquitectura de enlaces del motor transaccional basado en Windows Comunication Foundation ofrece un modelo elegante para configurar lo que sucede en la conexión. Sólo debe especificar un enlace y el motor transaccional lo usa como guía para generar la pila de canales en tiempo de ejecución.
- ✓ Una aplicación del motor transaccional está compuesta por:
    - Clientes: Son aplicaciones que inician la comunicación.
    - Servicios: Son aplicaciones que esperan los mensajes de los clientes y responden a los mismos.
    - Los mensajes son enviados entre endpoints. Un endpoint es un lugar donde un mensaje es enviado, o recibido, o ambos.
    - Un servicio expone uno o más endpoints, y un cliente genera un endpoint compatible con uno de los endpoints de un servicio dado.
- ✓ El hecho de que un cliente interactúe con un proxy, significa que el motor transaccional siempre está presente entre el servicio y el cliente, interceptando la llamada y llevando a cabo pre procesamiento y post procesamiento.
- ✓ La implementación del motor transaccional basado en consola, brinda la facilidad de visualizar los eventos y peticiones a cada uno de los servicios diseñados para un fin determinado

- ✓ El proporcionar la carga dinámica de las librerías que contienen los estándares de diseño, brinda una herramienta dinámica de lectura y hospedaje de servicios.
- ✓ El monitorear y almacenar las peticiones a los servicios levantados, proporciona la capacidad de realizar identificación y seguimientos de transacciones, identificándose el origen de llamada y el responsable.
- ✓ El tener la capacidad de configurar la ejecución de los métodos diseñados y almacenados en un repositorio de datos, brinda la dinámica de funcionamiento y optimización de control de transacciones permitidas en un proceso.
- ✓ El utilizar una de las bondades de Windows Comunication Foundation, como es el protocolo de red basado en TCP (net.tcp://) combinado con el nuevo componente de sistema, el servicio de uso compartido de puertos Net.TCP, permite compartir puertos net.tcp en varios procesos de usuario.

5.2 Recomendaciones

Dentro del proyecto que se orienta a optimizar y utilizar la tecnología de aplicaciones distribuidas, se recomienda a las personas que tengan interés en este proyecto, investigar sobre las demás tecnologías que posee Windows Comunication Foundation, como son la creación de servicios Web ASP.NET, Enterprise Services y así poder extender y fortaleces el concepto de modelo unificado de programación cuyo objetivo principal es que los desarrolladores logren hacer distintos tipos de aplicaciones distribuidas sin tener que aprender distintos modelos de programación.

Se recomienda el uso de Framework 2.0 en adelante, ya que en este se encuentra las librerías esenciales de Windows comunication foundation.

Se debe mantener los estándares de programación sugeridos en la sección de anexos, ya que son fundamentales para la lectura y el hospedaje de un servicio.

BIBLIOGRAFIA

Tema: Alojamiento y consumo de servicios WCF

URL: http://www.programar.net/directory/?fid=26

Tema: Concepto métodos

URL: http://www.monografias.com/trabajos11/metods/metods.shtml

Titulo: Desarrollo Orientado a Objetos con UML

URL: http://www.clikear.com/manuales/uml/index.aspx

Titulo: Manual WCF. NET

URL: http://www.programar.net/directory/?fid=26

Titulo: Programación 5 estrellas (manuales)

URL: http://www.mslatam.com/latam/msdn/comunidad/dce2005/

Titulo: Uso compartido de puertos Net.TCP

URL: http://msdn.microsoft.com/es-es/library/ms734772.aspx

Titulo: Transacciones

URL: http://msdn.microsoft.com/es-es/library/ms190612.aspx

Titulo: Síntesis del Proceso Unificado de Desarrollo de Software

URL: http://www.siliconkernel.com/ingenieria-de-software/1-sintesis-del-proceso-unificado-de-desarrollo-de-software-i

Titulo: Windows Communication Foundation

URL: http://msdn.microsoft.com/es-es/library/ms735119.aspx

Libro:

Improving Web Service Security (Microsoft)

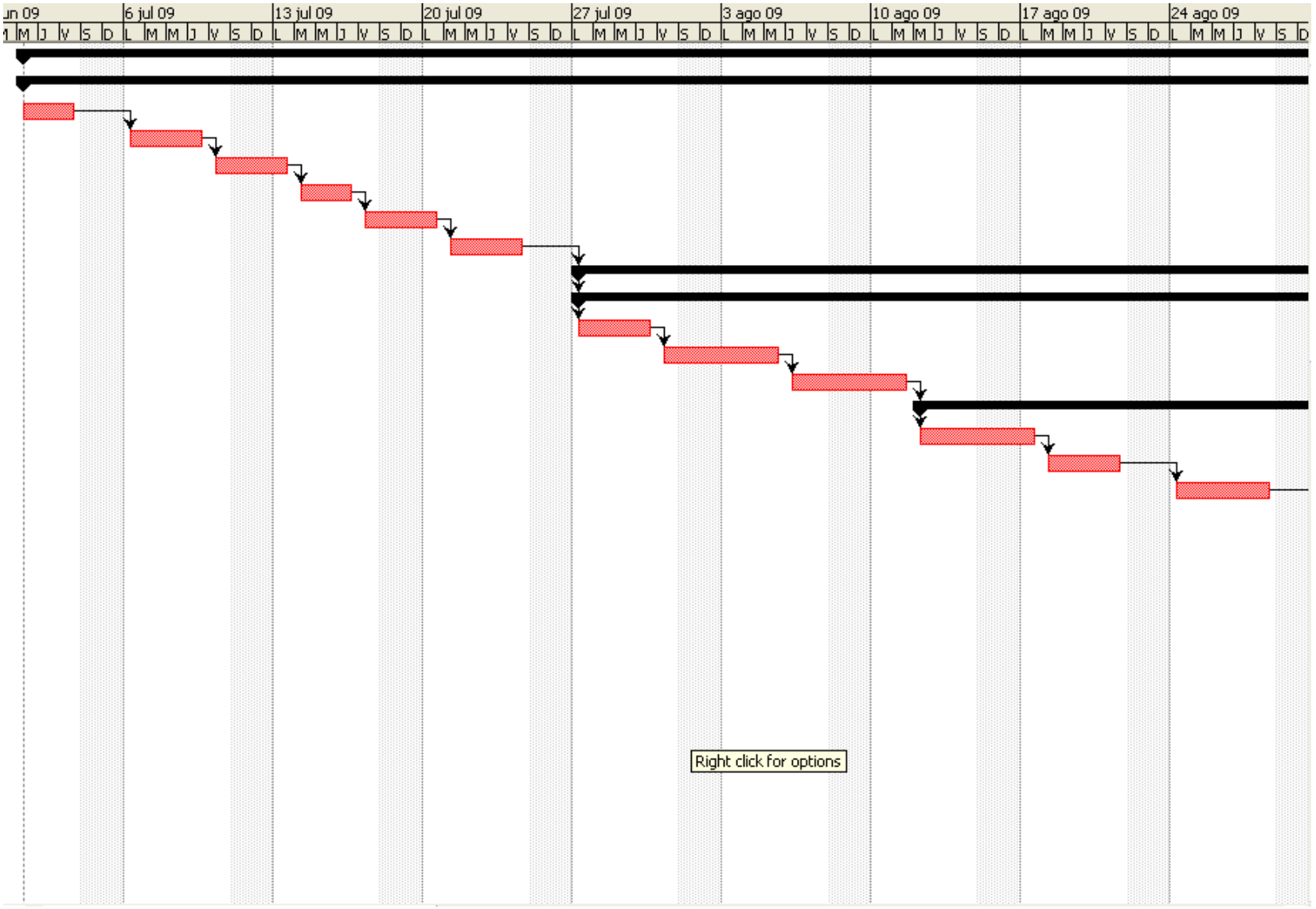
Scenarios and Implementation Guidance for WCF (Patterns & Practices)

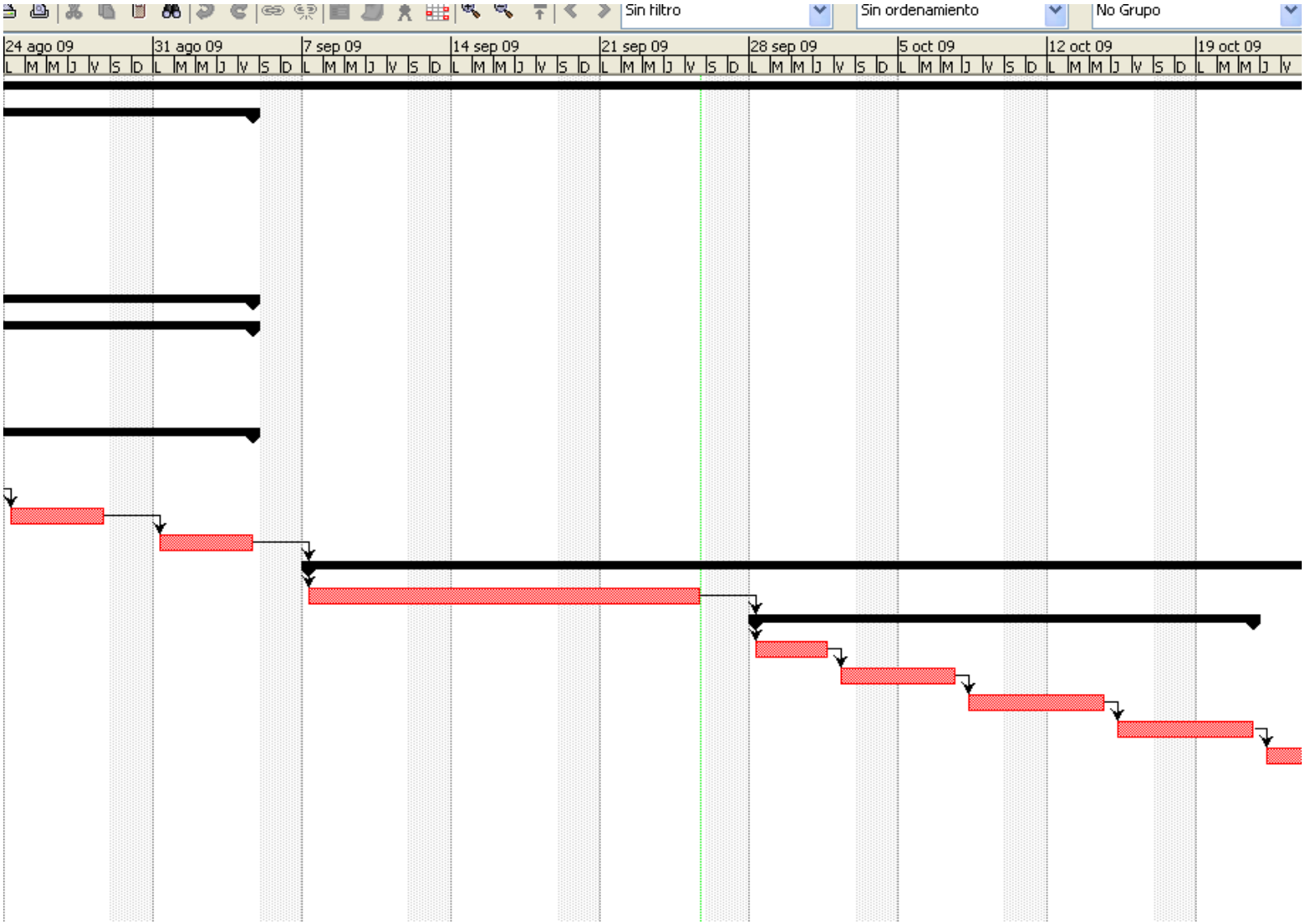
J.D. Meier /Carlos Farre

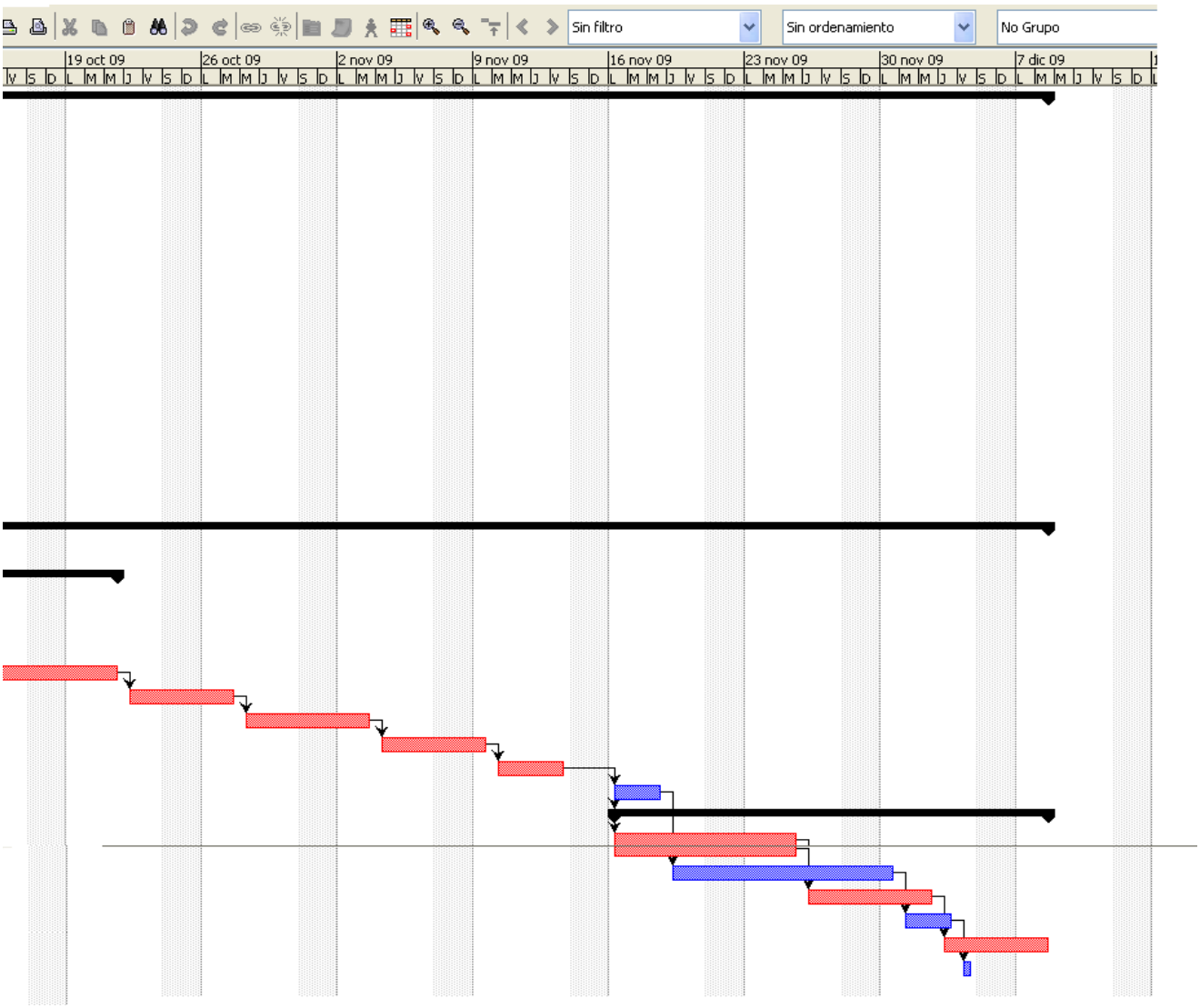
ANEXOS

Cronograma

	Nombre	Duracion	Inicio	Terminado
1	**Cronograma de desarrollo**	115 days?	**01/07/09 08:00 AM**	**08/12/09 05:00 PM**
2	**Fase de Planificación**	48 days	**01/07/09 08:00 AM**	**04/09/09 05:00 PM**
3	Estudio de viabilidad	3 days	01/07/09 08:00 AM	03/07/09 05:00 PM
4	Investigación Preliminar	4 days	06/07/09 08:00 AM	09/07/09 05:00 PM
5	Definicion de Requisitos	2 days	10/07/09 08:00 AM	13/07/09 05:00 PM
6	Definir Casos de Uso	3 days	14/07/09 08:00 AM	16/07/09 05:00 PM
7	Definir el Modelo Conceptual-Borrador	2 days	17/07/09 08:00 AM	20/07/09 05:00 PM
8	Definir coronograma de trabajo	4 days	21/07/09 08:00 AM	24/07/09 05:00 PM
9	**Fase de Elaboración**	30 days	**27/07/09 08:00 AM**	**04/09/09 05:00 PM**
10	**Levantamiento de información**	30 days	**27/07/09 08:00 AM**	**04/09/09 05:00 PM**
11	identificacion de procesos	4 days	27/07/09 08:00 AM	30/07/09 05:00 PM
12	identificaion de recursos	4 days	31/07/09 08:00 AM	05/08/09 05:00 PM
13	requisitos	4 days	06/08/09 08:00 AM	11/08/09 05:00 PM
14	**Diseño**	18 days	**12/08/09 08:00 AM**	**04/09/09 05:00 PM**
15	estructura de datos	4 days	12/08/09 08:00 AM	17/08/09 05:00 PM
16	modelamiento de interfas	4 days	18/08/09 08:00 AM	21/08/09 05:00 PM
17	definir protocolos de comunicacio	5 days	24/08/09 08:00 AM	28/08/09 05:00 PM
18	definir componenentes	5 days	31/08/09 08:00 AM	04/09/09 05:00 PM
19	**Fase de Construcción**	67 days?	**07/09/09 08:00 AM**	**08/12/09 05:00 PM**
20	analisis y diseño	15 days	07/09/09 08:00 AM	25/09/09 05:00 PM
21	**modelo de datos**	18 days	**28/09/09 08:00 AM**	**21/10/09 05:00 PM**
22	modelo logico	4 days	28/09/09 08:00 AM	01/10/09 05:00 PM
23	modelo conceptual	4 days	02/10/09 08:00 AM	07/10/09 05:00 PM
24	diccionario de datos	5 days	08/10/09 08:00 AM	14/10/09 05:00 PM
25	modelo fisico	5 days	15/10/09 08:00 AM	21/10/09 05:00 PM
26	Interfas	4 days	22/10/09 08:00 AM	27/10/09 05:00 PM
27	pantallas	5 days	28/10/09 08:00 AM	03/11/09 05:00 PM
28	menus	4 days	04/11/09 08:00 AM	09/11/09 05:00 PM
29	pruebas	4 days	10/11/09 08:00 AM	13/11/09 05:00 PM
30	instacion de version beta	3 days	16/11/09 08:00 AM	18/11/09 05:00 PM
31	**Fase de Transición**	17 days?	**16/11/09 08:00 AM**	**08/12/09 05:00 PM**
32	plan implantacion sistema	8 days	16/11/09 08:00 AM	25/11/09 05:00 PM
33	manual tecnico del sistema	8 days	19/11/09 08:00 AM	30/11/09 05:00 PM
34	manual de usuario	5 days	26/11/09 08:00 AM	02/12/09 05:00 PM
35	instalacion del sistema	3 days	01/12/09 08:00 AM	03/12/09 05:00 PM
36	pruebas finales	4 days	03/12/09 08:00 AM	08/12/09 05:00 PM
37	acata de entrega	1 day?	04/12/09 08:00 AM	04/12/09 05:00 PM


un 09
6 jul 09
13 jul 09
20 jul 09
27 jul 09
3 ago 09
10 ago 09
17 ago 09
24 ago 09
Right click for options


Sin filtro
Sin ordenamiento
No Grupo
24 ago 09
31 ago 09
7 sep 09
14 sep 09
21 sep 09
28 sep 09
5 oct 09
12 oct 09
19 oct 09


Sin filtro
Sin ordenamiento
No Grupo
19 oct 09
26 oct 09
2 nov 09
9 nov 09
16 nov 09
23 nov 09
30 nov 09
7 dic 09

Formulario encuesta personal técnico del banco.

En la actualidad que tiempo lleva la implementación de un proyecto desde su etapa inicial.

- 1 día
- 5 días
- 10 días

Que proceso es el que lleva dedicación de tiempo en la fase de desarrollo.

- Recepción del requerimiento.
- Elección interface.
- Análisis y elección de componente.
- Análisis seguridad transacción.
- Implementación.

La elección del componente de interconexión implica un conocimiento técnico más amplio para el desarrollador.

- Si
- No

A nivel de conocimiento técnico para el desarrollador, este debería tener conocer una de los componentes de conexión distribuida.

- Enterprise service
- Web Service.
- Net. Remoting
- MSMQ (colas).

Considera que como desarrollador, el contar con una herramienta que se centre únicamente en la configuración de puerto y la creación de interfaces de conexión, agilizara la entrega de proyectos.

- Si
- No

El desarrollo de un motor transaccional para la interconexión de sistemas con las capas lógicas, considera que sería una herramienta viable para cubrir los requerimientos solicitados de manera más rápida.

- Si
- No

Manual de usuario – Seguridad

El sistema de seguridad es un cliente que consume los servicios del Motor transaccional, ya que en base a su configuración se da el funcionamiento de:

- ✓ Creación de empresa.
- ✓ Creación de canal de trabajo
- ✓ Creación de aplicativo
- ✓ Configuración canal-aplicativo
- ✓ Creación de usuario
- ✓ Configuración de horario de trabajo de un usuario
- ✓ Definición de funciones (transacciones) que estarán disponibles en los servicios.
- ✓ Configuración de horario de trabajo de una función (transacción)

Ingreso modulo seguridad

Grafico 5-1 Ingreso modulo seguridad

Usuario: Admotor

Clave: 1

Creación de empresa

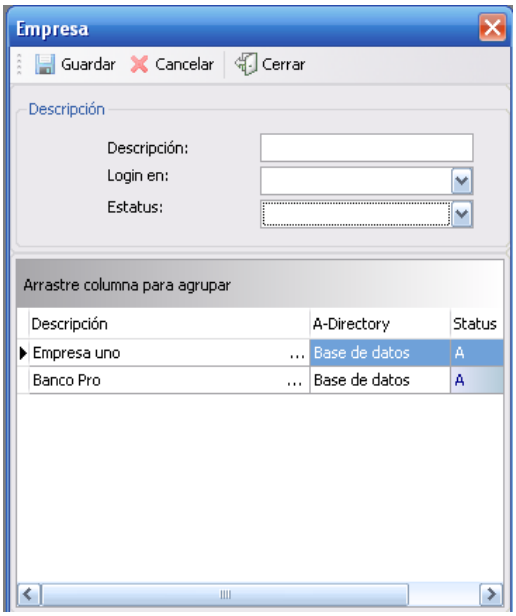


Grafico 5-2 Creación de empresa

Descripción. Nombre de la empresa

Login en: el modo login se lo tiene en base de datos, que es lo que obliga a crear el usuario en seguridad

Estatus: Estado actual de la empresa creada.

Creación de canal de trabajo

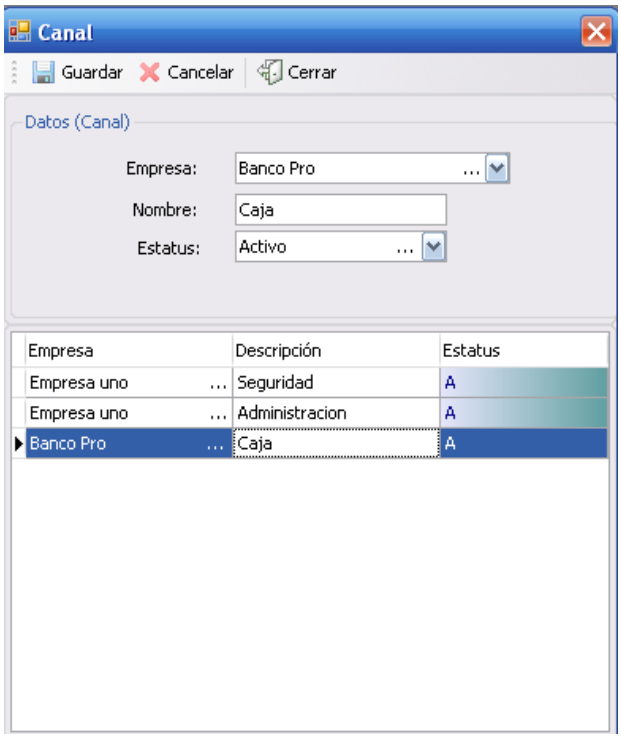


Grafico 5-3 Creación canal de trabajo

Empresa: Empres a la que se definirá el canal de trabajo

Nombre: Nombre que se desea dar al canal

Estatus: estado del canal

Creación de aplicativo

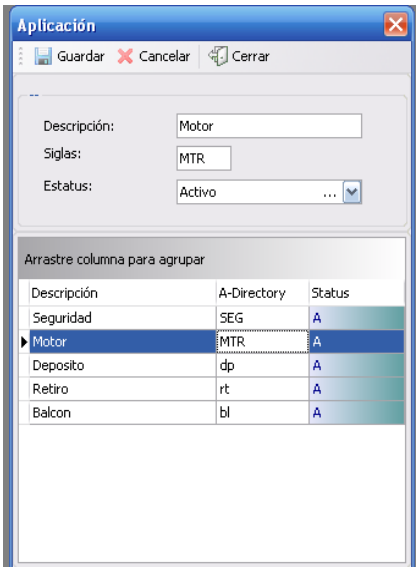


Grafico 5-4 Creación de aplicativo

Descripción: Descripción de aplicativo

Siglas: tres letras descriptivas del aplicativo

Estatus: estado del aplicativo.

Configuración canal-aplicativo

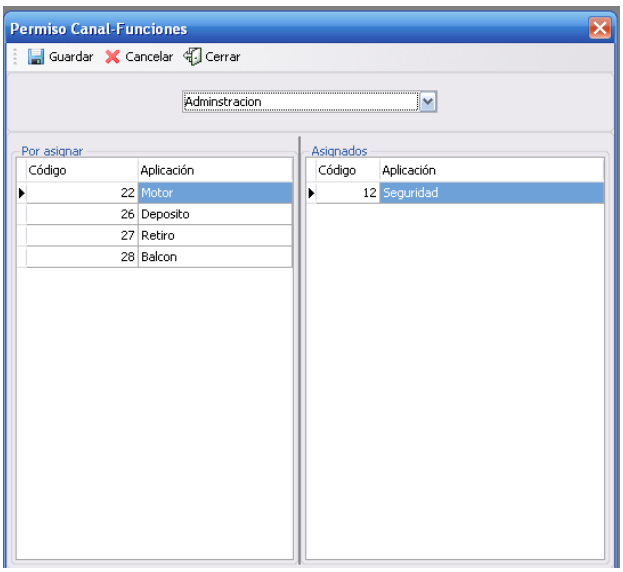


Grafico 5-5 Configuración canal-aplicativo

En esta pantalla se configura el permiso de canales con aplicativos, es decir para canal **administración** se habilita a que trabaje con el aplicativo **seguridad.**

Creación de usuario

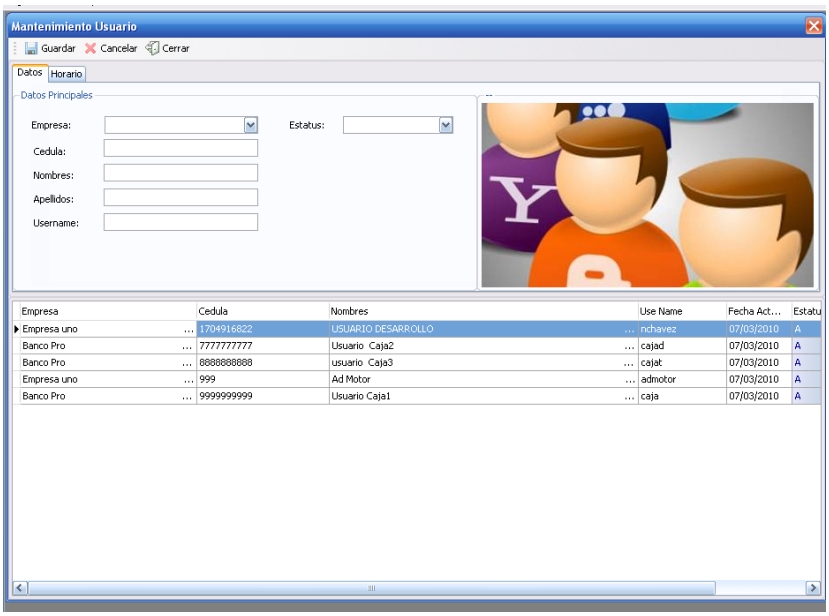


Grafico 5-6 Creación de usuario

Se define un usuario con sus datos básicos como:

- ✓ Cedula
- ✓ Nombres
- ✓ Apellidos
- ✓ Clave

Este usuario se crea tiene la capacidad de crear una sesión de trabajo para el uso de los servicios del motor.

Configuración de horario de trabajo de un usuario

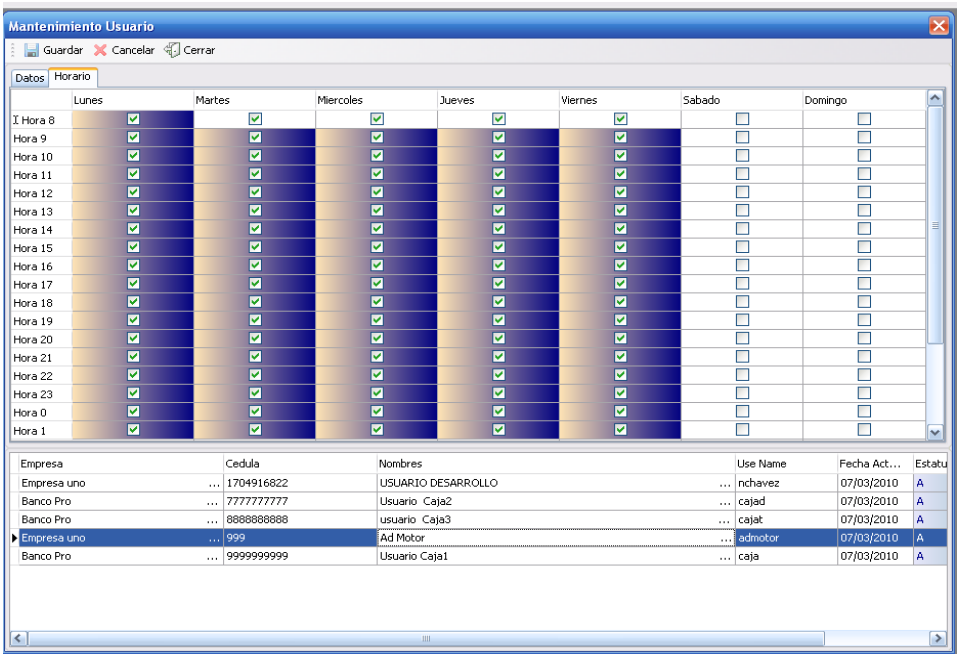


Grafico 5-7 Configuración de horario de trabajo de un usuario

En la misma de ventana de usuarios en la viñeta de horario se selecciona y nos da la facilidad de asignar horario de trabajo, por default el usuario se crea con 8 horas de trabajo, de 8 a 5 de la tarde.

Definición de funciones (transacciones), disponibles en los servicios.

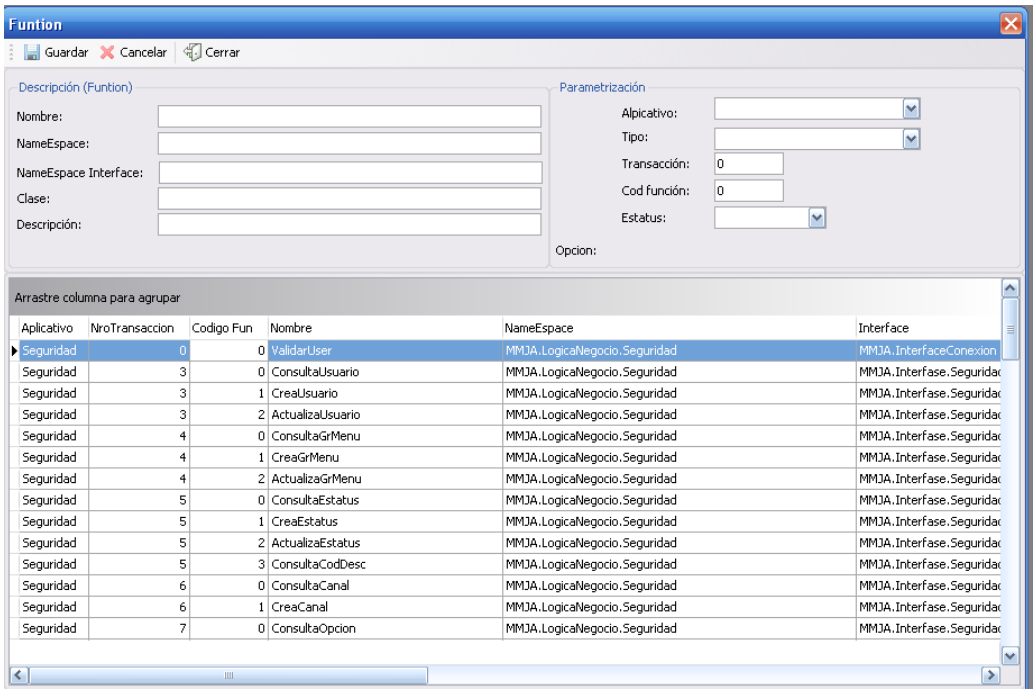


Grafico 5-8 Definición de funciones (transacciones), disponibles en los servicios

Se define:

Nombre: es la definicion del metodo dentro de la clase que se creo.

- ✓ Ej. ValidaUsuario

NameEspace: es el espacio de trabajo definido en la creacion de la clase.

- ✓ Ej. LogicaNegocio.Seguridad

NameEspace Interface: es el nombre de la clase que contiene la interfas definido en el ambiente de desarrollo.

- ✓ Ej. Interface.seguridad

Clase: Nombre de la clase que se creó y contiene los métodos declarados.

- ✓ Ej. clsRoles.
- ✓ Algo muy importante en la definición de la clase, es que siempre debe anteponer la palabra **cls.**

Descripción: es una definición no técnica para poder relacionar la actividad del método o función insertada.

- ✓ Ej. Proceso de acreditación sueldos quincenal.

Aplicativo: se debe definir el aplicativo al que pertenece la función, ya que con esta opción se ata, los métodos a los aplicativos declarados

Tipo: se define si es consulta o transacción.

Transacción: definición de número de transacción dentro del aplicativo.

Función: definición de numero de función dentro del aplicativo y si se relaciona, puede ser dentro de la transacción.

Ejemplo.

- ✓ Aplicativo: Cajas
- ✓ Transacción: 1 (estos pueden incrementar o permanecer únicos)
- ✓ Función: 1 (este valor debe incrementar, caso contrario no permite guardar)

Nota: la configuración de aplicativo, transacción y función deben ser únicos.

Estatus: estado que se pone a la función para su ejecución.

Configuración de horario de trabajo de una función (transacción)

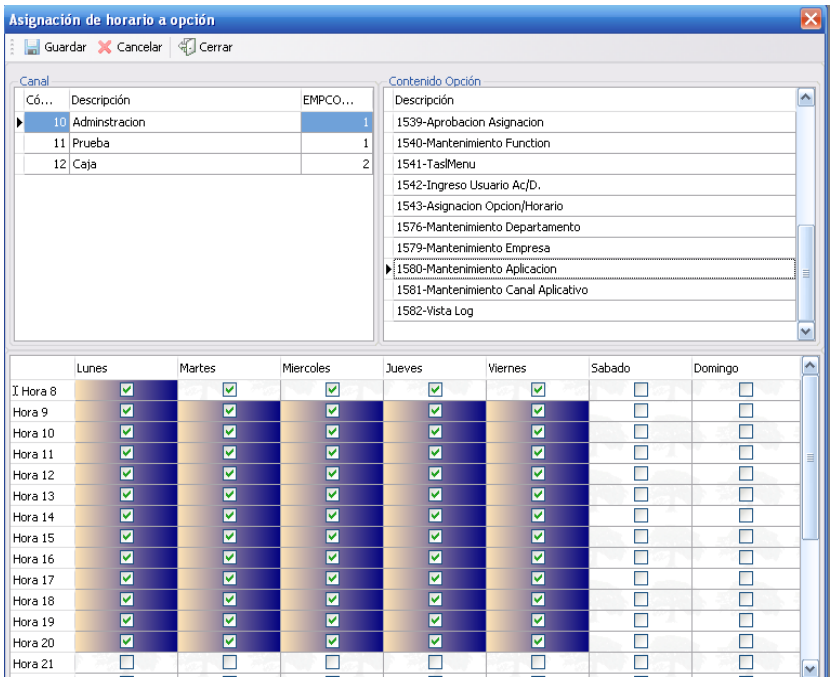


Grafico 5-9 Configuración de horario de trabajo de una función (transacción)

Se configura el horario de trabajo del método o función, definido en la venta (**Definición de funciones**), los métodos por default se crean sin horario de trabajo, es decir que cuando el motor levante los servicios, este levantara una lista de funciones con horario y una lista de funciones que trabajen normalmente.

Manual técnico.

Estándar programación servicios

Para la creación de los servicios que el Motor transaccional levanta dinámicamente, las clases deben cumplir el siguiente estándar y contener atributos necesarios en el momento de su creación.

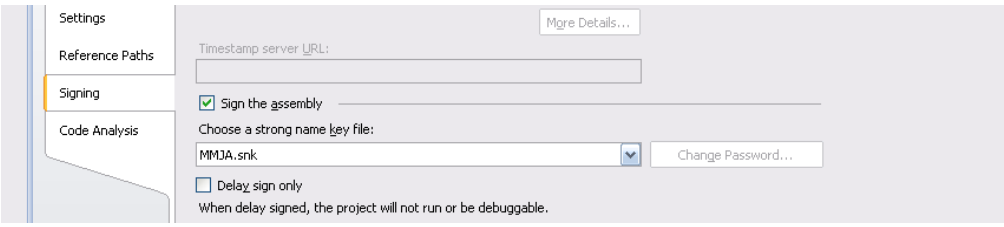
- ✓ Las clases deben ser creadas en .Net.
- ✓ Se debe usar Framework 2.0 o superior.
- ✓ El proyecto debe hacer referencia a los espacios de trabajo System.ServiceModel y System.Runtime.Serialization.

En el siguiente paso se creara una aplicación muy básica que servirá para agregar en la carpeta de librerías del motor transaccional.

La estructura básica de un servicio consta de dos partes: un contrato de servicio y el servicio propiamente dicho.

Contratos de servicio y de datos

Se debe definir una clase tipo interfaz con los métodos y asignarle un nombre seguro a la librería.



La interface deberán tener los atributos de ServiceContract y OperationContract.

En donde:

El atributo **ServiceContract** indica que el interfaz es un contrato de servicio WCF, mientras que **OperationContract** indica que la operación está expuesta como parte del servicio. Estos atributos están definidos en el ensamblado **System.ServiceModel**, de modo que es necesario tenerlo referenciado.

```
using System;
using System.ServiceModel;
namespace Guia.Servicio.Interface
{
   [ServiceContract]
   public interface IclsTestService
   {
     [OperationContract]
     Dato getUserData (int userId);
   }
 }
```

Un dato importante es que el servicio puede requerir contratos de datos. Se trata de tipos definidos (clases, tipos enumerados, estructuras) para el intercambio de datos con los clientes., estos deben tener los siguientes atributos especiales:

```
using System;
using System.Runtime.Serialization;
namespace Guia.Servicio.Contracts
{
        [DataContract]
        public class UserData
        {
                 [DataMember]
                public int userId {get; set ;}
                [DataMember]
                public string userName {get; set ;}
        }
}
```

El atributo DataContract indica que la clase es un contrato de datos WCF y DataMember que la propiedad a la que se aplica es un miembro de datos expuesto. Ambos están definidos en el ensamblado System.Runtime.Serialization, que debe estar referenciado.

Lo más conveniente es poner los contratos en un ensamblado aparte, a fin de facilitar el posterior mantenimiento. Para ello hay dos opciones:

- ✓ Crear un nuevo proyecto de librería de servicios WCF (y posteriormente eliminar el fichero de configuración y de servicio creados por el asistente).
- ✓ Crear un nuevo proyecto de librería de clases y agregar manualmente referencias a los ensamblados System.ServiceModel y (de ser necesario) System.Runtime.Serialization.

Implementar el servicio

El servicio es una clase que implementa el contrato de servicio correspondiente. No se ejecuta por sí solo, sino que debe ser hospedado por otra aplicación, que en este caso es el motor transaccional.

```
using System;
using Guia.Servicio.Interface;
namespace Guia.Servicio.Services
{
        public class clsTestService : IclsTestService
        {
                public UserData getUserData(int userId)
                {
                        if (userId == 1)
                {
                        UserData data = new UserData() { userId = 1, userName = "David
                        Liñán" };
                        return data;
                }
                else
                        return null;
                }
         }
}
```

El servicio debería estar en un ensamblado aparte para que pueda ser hospedado por el motor transaccional.

Hospedar el servicio

Para hospedar el servicio se debe copiar la librería del servicio, la interface, y el contrato de datos en la carpeta configurada para este fin.

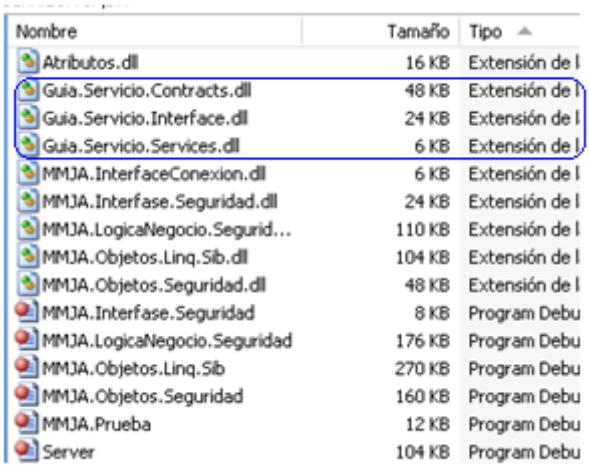


Grafico 5-10 Ubicación física librería

Para que el servicio se levante se debe configurar las líneas del servicio en el App.config del motor transaccional.

```
    <service name="MMJA.LogicaNegocio.Seguridad.clsAutentificacionAD">
      <endpoint address="net.tcp://localhost:8002/clsAutentificacionAD"
        binding="netTcpBinding" bindingConfiguration="BindingTcp" name="clsAutentificacionAD"
        contract="MMJA.Interfase.Seguridad.IclsAutentificacionAD" />
    </service>
    <service name="Guia.Servicio.Services.clsTestService">
      <endpoint address="net.tcp://localhost:8002/clsTestService"
        binding="netTcpBinding" bindingConfiguration="BindingTcp" name="clsTestService"
        contract="Guia.Servicio.Interface.IclsTestService" />
    </service>
    <endpoint address="net.tcp://localhost:8002/Retiro" binding="netTcpBinding"
      bindingConfiguration="BindingTcp" name="Retiro" contract="LogicaNegocioBBDD.Interface.IRetiro" />

  </service>
  <service name="LogicaNegocioBBDD.clsBalcon">
    <endpoint address="net.tcp://localhost:8002/Balcon" binding="netTcpBinding"
      bindingConfiguration="BindingTcp" name="Balcon" contract="LogicaNegocioBBDD.Interface.IBalcon" />
  </service>
</services>
system.serviceModel>
```

Grafico 5-11 App.config - motor

En donde:

Service Name: es el la unión del namespace con el nombre de la clase

```
<service name="Guia.Servicio.Services.clsTestService">
```

Endpoint address: es la ubicación Tcp y el puerto compartido con el nombre de la clase

```
<endpoint address="net.tcp://localhost:8002/clsTestService"
          binding="netTcpBinding" bindingConfiguration="BindingTcp"
```

Name: nombre de la clase que contiene el servcio (**muy importante**).

```
name="clsTestService"
```

Contract: es la union del namespace con el conmbre de la interface

```
contract="Guia.Servicio.Interface.IclsTestService" />
</service>
```

Implementando el cliente

La aplicación cliente necesita conocer tres cosas del servicio creado: su contrato, su dirección y el tipo de canal a usar.

El contrato se lo obtine haciendo una referencia al ensamblado del contrato (interface). Y las otras dos se las define en el fichero de configuración de la aplicación cliente y sus

valores deben coincidir con los definidos en el fichero de configuración del servicio (para ser exacto, de la aplicación que lo hospeda). El cliente debe tener además una referencia a System.ServiceModel.

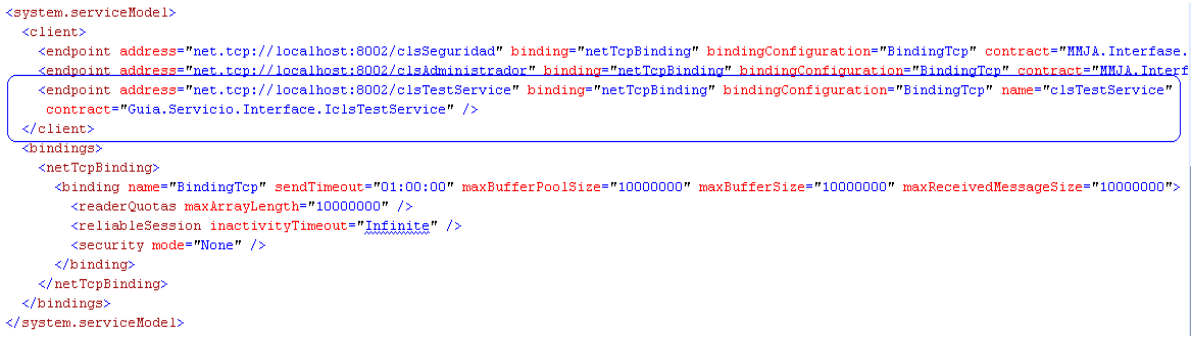


Grafico 5-12 App.config - Cliente

Definicion del cliente:

```
<system.serviceModel>
  <client>
   <endpoint address="net.tcp://localhost:8002/clsTestService"
binding="netTcpBinding" bindingConfiguration="BindingTcp"
name="clsTestService"
       contract="Guia.Servicio.Interface.IclsTestService" />
   </client>
</system.serviceModel>
```

Configurando el motor:

El correcto funcionamiento del motor se basa en su archivo de configuración.

Definir el path de las librerias.

```
  <appSettings>
    <add key="DropPath" value="D:\servcios\Dll"/>
  </appSettings>
```

Definir configuracion de servicios dentro de area system.serviceModel, esto debe ser por cada servcio que se dese eimplementar.

```
<service name="Guia.Servicio.Services.clsTestService">
   <endpoint address="net.tcp://localhost:8002/clsTestService"
    binding="netTcpBinding"bindingConfiguration="BindingTcp"
name="clsTestService"
contract="Guia.Servicio.Interface.IclsTestService" />
```

Al realizar la correcta configuracion se proce a ejecutar la consola de motor.

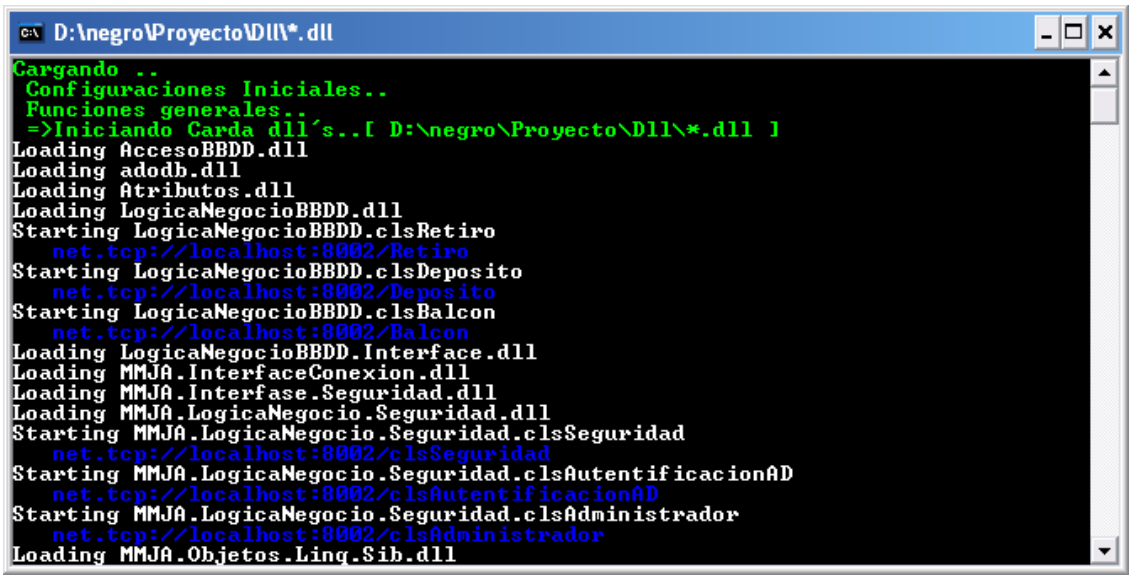


Grafico 5-13 Pantalla consola en ejecución

El motor transaccional tiene la cualidad de levantar los servios diseñados en sus librerias y publicarlos en un puerto TCP de aucerdo al fichero de configuración, por lo que su correcta configuracion nos presentara una ventana con las siguiente respuesta.

Donde indica la librería y sus servicios programados y configurados en el archivo de del motor.