UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO INFORMÁTICO

TÍTULO DE LA TESIS:	Integrador de Sistemas Heredados, Una Solución para la Integración de Información
NOMBRE Y APELLIDOS DEL GRADUADO:	Edison Francisco Muñoz Recuay
CICLO EN EL QUE EGRESO:	2004 – II
CICLO EN EL QUE INGRESO:	2000 – I
DOCENTE ASESOR:	Dr. David Mauricio Sánchez
OBJETIVO DE LA TESIS:	Otorgar un aporte teórico y práctico para lograr la integración de información de sistemas heredados.

LIMA - PERÚ

2007

Resumen

Los sistemas de información legacy o heredados fueron creados con la finalidad de automatizar procesos que antes de la invención de la informática se hacían de forma manual, estos sistemas han cobrado importancia con los años porque las organizaciones han ido dependiendo cada vez más de ellos y la información que generan y administran son de gran valor. Hoy en día con las nuevas tendencias y paradigmas hacen que los negocios y organizaciones basen la mayoría de sus procesos en los sistemas de información y es imprescindible que la nueva tecnología conviva con los sistemas legacy o heredados; por tal motivo la información debe ser integrada. Bajo este contexto, es necesario plantear una alternativa de solución que permita integrar la información de diversos sistemas legacy o heredados.

En la presente tesis, se analiza el origen del problema, se describe la importancia de los sistemas legacy o heredados frente a las nuevas tendencias tecnológicas actuales, se analiza las herramientas con las que podemos lograr integración de información, y se propone una alternativa de solución al problema mediante un sistema integrador de información multiplataforma adaptable a cualquier tipo de negocio. Adicionalmente, se propone una metodología de integración que deja una base establecida para realizar el desarrollo de un nuevo sistema que reemplace al sistema legacy o heredado. La solución propuesta ha sido probada mediante un caso de estudio en una empresa del sector turismo, donde se pudo verificar: que es posible integrar la información de un sistema que maneja archivos planos, mostrar la información en una interfase Web por ejemplo, y empezar la migración hacia una nueva base de datos relacional. Después de realizadas las pruebas de la solución se puede afirmar que esta puede integrar información sin importar el sistema operativo y que puede ser usado en distintas organizaciones o empresas.

Palabras clave: *Integración, XML, Web Services, Sockets, JMS, Java, .Net, UML, Abstract Factory, Patrones, Legacy.*

Abstract

The Legacy information systems were created with the purpose of automate process that before of the informatics invention were made in a manual way, these systems have gained prominence over the years because organizations have become increasingly dependent of them, and the information they generate and manage have a great value. Nowadays new tendencies and paradigms make that business and organizations base the majority of their process in information systems and it is imperative that new technology can works together with legacy systems, for this reason the information must be integrated. Therefore under this context, is necessary to raise an alternative solution for integrating information of different legacy systems.

In this thesis, is examined the problem, is described the importance of legacy systems in contrast with new current technological tendencies, besides is analyzed tools which we can achieve information integration and is purposed an alternative solution to problem through a multiplatform information system integrator, additionally, it proposes a integration methodology, leaving an established base to perform the development of a new system in order to replace the legacy system. The proposed solution has been proven through a case study in a business tourism sector, which was able to verify: that it is possible to integrate information from a system that handles flat files, display the information on a web interface for example, and start migration to a new relational database. After completing the proof of solution, we can say that this can integrate information regardless of the operating system and can be used in different organizations or companies.

Key words: *Integration, XML, Web Services, Sockets, JMS, Java, .Net, UML, Abstract Factory, Patterns, Legacy.*

Ficha Catalográfica

Autor:	Muñoz Recuay, Edison Francisco
Título	Integración de Sistemas Heredados, Una Solución para la Integración de Información
Tesis:	Tesis para adoptar el título de Ingeniero Informático
Lugar:	Universidad Particular Ricardo Palma
	Lima - Perú
Fecha	14 de Diciembre de 2007
Número de páginas:	188 páginas

Dedicatoria:

A mi Madre Elsa y hermana Miriam por su amor y cariño,
y por darme motivos para ser feliz,
a mis profesores por brindarme su sabiduría y conocimientos
y a mis amigos de la URP por ser las mejores
personas que la vida me dio la oportunidad de conocer
Este trabajo es para Ustedes.

Agradecimientos:

A Dios que ilumina mi camino y está presente siempre en nuestras vidas, al Dr. David Mauricio Sánchez por su amistad y todo el apoyo brindado en la consecución de esta tesis, a los Sres. Jurados, Ing. Edgard De Olazábal, Ing. Víctor Beltrán gracias por sus críticas y comentarios que ayudaron a mejorar el trabajo final, al Dr. Silverio Bustos por su amistad y apoyo en mis años en la universidad, al Lic. Joel Francia por sus enseñanzas y amistad, al Ing. Joel Moreno por su ayuda desinteresada, a mis amigos Giovana Montano por darle forma y orden al inicio de la tesis, a Cristina Palomino por tu amistad y todo tu apoyo al ayudarme muchas veces gracias realmente, Srta. Alida Pardo por su ayuda en los trámites, al Ing. Leonardo Alcayhuamán, gracias por interesarse en mi tesis, José Socola gracias por la explicación de la plataforma J2EE a Carlos Linares mi amigo, gracias por tu ayuda desinteresada compadre, Iván Campos y Marco Salazar por su ejemplo, apoyo y amistad, fue un honor haber trabajado ustedes, les deseo muchos éxitos, así como también a Giancarlo Vidal, Jorge Ramírez, Cesar Egoavil, Edwin Suarez y Javier Alvarado, el gran equipo de desarrollo, a Delia San Miguel, gracias por tu amor y cariño, a mis amigos de la URP, David Guadalupe, Lis Laurente, Gustavo Ramos, Jimmy Quintana, Edgard Alejos, Francis Chaupis, Luis Zerga, Maria Esther Dominguez, Rómulo Panaqué, Juan Carlos Santa Cruz, Fiorela Benavides, Paola Battistolo, Jorgen Bontemps, Jesús Ramirez, Raúl Villagarcia, Jorge Polo, Carlos Tamayo, Ricardo Vilchez, Carlos Garcia, Karen Huamán, Kenji Kato en la distancia que estes muy bien, a la familia Recuay, a mis primos Nelly, José y Jesús Recuay, Aldo Flores, Mitchael Durán mi tío Humberto Bullón, gracias por tus enseñanzas y tu apoyo en todo momento, mis tíos Marcelo, Benigno y Bertha Recuay, a todos ustedes gracias por su amistad y su ayuda, por compartir grandes momentos, tal vez puedo olvidar alguno en estas líneas, pero no en el corazón.
Un agradecimiento especial a la Universidad Ricardo Palma, por los bonitos años de vida y sabiduría que me brindó

Muchas gracias por todo.

Índice

ÍNDICE DE TABLAS Y FIGURAS

TABLAS

FIGURAS

Capítulo I

Introducción

El avance tecnológico en las ciencias de la información y los sistemas informáticos, ha dejado rápidamente relegados grandes sistemas de información que sostuvieron los negocios no hace más de dos décadas; el mundo de la computación la tecnología avanza rápidamente, es por ello que cualquier tecnología que aparezca será antigua pasado un año, por esta razón los negocios que aún conservan sistemas de información legacy o heredados tienen una desventaja tecnológica frente a competidores nuevos y con tecnología más reciente.

El problema cobra importancia por la información que manejan los sistemas legacy o heredados, esta información, que en la mayoría de los casos es almacenada en archivos de texto o planos, representa muchas veces el trabajo diario de las organizaciones, y reemplazar estos sistemas por uno nuevo significa en la mayoría de veces un alto costo que las organizaciones no pueden afrontar, además en muchos casos estos sistemas son altamente confiables y eficientes que realizar un cambio implicaría un riesgo alto, sobre todo cuando son sistemas que implican factor crítico de éxito, pero la integración de la información que manejan es necesaria para darle un debido tratamiento.

El planteamiento de solución a este problema es la de desarrollar una herramienta de integración de información que sea adaptable, flexible y que se acomode a las necesidades de las empresas, y a un costo menor adecuado, todo ello con tecnología basada en herramientas de última generación, aplicando patrones y utilizando estándares del desarrollo actual de aplicaciones software; una principal característica de esta solución es también su capacidad multiplataforma, es decir, que se puede implementar en diferentes sistemas operativos.

La construcción de la solución usa la metodología Proceso Unificado de desarrollo de software, está basada en orientación de objetos, usando patrones de diseño; y la Arquitectura de la solución se fundamenta en plataformas de software libre y tecnología propietaria. La solución es instalada en un servidor de aplicaciones de tecnología libre para pueda ejecutarse ininterrumpidamente. Todo este conjunto de tecnologías y herramientas soportan el desarrollo y producción del proyecto de integración de información de sistemas que manejan archivos planos, así como también bases de datos basados relaciones. En los capítulos siguientes se darán a conocer cómo es aprovechada la utilidad de cada uno de ellos.

La prueba de esta contribución es realizada en la empresa CTM Tours del Grupo Costamar, que cuenta con un origen de datos en Texto Plano, para su sistema de gestión paquetes y destinos turísticos, y requiere que la información sea integrada con nuevos orígenes de datos y nuevas aplicaciones, el capitulo de Caso de Estudio describe el problema de la empresa y las pruebas realizadas.

Por otro lado la estructura de la tesis es de la siguiente manera: en el segundo capítulo encontramos las generalidades de los sistemas legacy o heredados, el estado del arte respecto a integración se encuentra detallado en el tercer capítulo; en el cuarto capítulo se presenta un análisis de la tecnología orientada a la integración de información, en el quinto capítulo se describe la contribución como alternativa de solución para integración de integración; luego en el sexto capítulo , como se dijo anteriormente, se describe el caso de estudio, aplicado a la solución, y para terminar, en el sétimo capítulo se listan las conclusiones y recomendaciones

que se obtuvieron después de realizada la investigación.
Dando una opción innovadora de integración de información, solucionando un problema común de muchas empresas se presenta a continuación el trabajo realizado, compilado y organizado para su entendimiento, y que sirva de aporte a la sociedad y a las personas involucradas en tecnologías de la información y que puedan servir también como base para futuras investigaciones.

1.1 Objetivos Generales

Plantear, diseñar e implementar una solución integradora de información para las organizaciones, que sirva de apoyo para darle un mejor uso a la información que manejan desde sus sistemas legacy o heredados, tratando de la solución tenga un costo mínimo accesible a la mayoría de las empresas, pero que a la vez sea fácilmente configurable y escalable.

Plantear la solución a través de la investigación de la tecnología actual, estándares y herramientas de desarrollo que pueden ser útiles para la integración de información de sistemas legacy o heredados.

1.2 Objetivos Específicos

Diseñar e implementar una arquitectura de integración de información que soporte diferentes orígenes de datos, sin alterar en gran medida el código que ya ha sido desarrollado.

Diseñar una solución que sea fácilmente configurable y que pueda adaptarse rápidamente a la mayoría de los escenarios de negocios. Además esta debe ser fácilmente y rápidamente modificable, si es que el usuario desea integrar más orígenes de datos, o agregar más archivos o tablas a ser integradas.

Orientar una de las opciones de integración a archivos planos DBF porque sistemas construidos en Fox Pro, Visual Fox Pro, Clipper, DBase y entre otros, trabajan con este tipo de archivo, y por esta razón la solución podría tener un alcance a varios tipos de sistemas legacy o heredados que fueron construidos bajo estos entornos de desarrollo.

Diseñar el servicio que brindara la solución bajo plataforma libre de tal forma que tenga bajo costo y que sea consumible desde diferentes sistemas sin importar la plataforma de desarrollo o sistema operativo.

Realizar pruebas mediante un caso de estudio en una organización real que permita establecer la factibilidad de la solución planteada.

Aprovechar las bondades de XML mediante su uso adecuado para la integración de información entre diferentes plataformas tecnológicas, así como también al construir una solución generadora de código que acelere el proceso de desarrollo y configuración de la solución de integración.

Capítulo II

Generalidades de Sistemas Legacy o Heredados

Los sistemas legacy o heredados son un frecuente dolor de cabeza para los gerentes de sistemas de información dentro de las empresas, dado que muchas veces es necesario mantenerlos por su eficacia, utilidad y usabilidad; pero también es una limitante al momento de planear nuevas soluciones que incluyan usar la información que manejan estos sistemas.

En el presente capitulo se brinda información sobre como surgieron, cuales son sus características, y cuál ha sido su evolución.

2.1 Sistemas legacy o heredados

Cuando surge la era de la información, impulsada por las necesidades que traía la segunda guerra mundial, era necesario hacer grandes cálculos matemáticos para poder diseñar formulas y técnicas que pudieran impulsar la construcción de armas y artefactos que ayudarían a tener ventaja estratégica ante el enemigo, la computadora por esta época se veía solo como una herramienta de cálculos, que nos brindaba la ventaja de realizar operaciones que humanamente serían imposibles de hacer, así surge la necesidad de procesamiento y de respuesta rápida. Después de terminada la guerra y con el gran avance rápido que se había logrado, se llevo el uso de la computadora al campo civil. Este nuevo uso de las computadoras se dio en las grandes empresas y corporaciones que apostaron por la tecnología para cubrir sus necesidades de procesamiento y almacenamiento de datos generados por sus operaciones. Estos sistemas estaban basados en tarjetas perforadas las cuales se constituyeron en uno de los primeros medios para alimentar a las computadoras con información para el procesamiento, cada tarjeta representaba una línea de código o de datos del programa, la que después alimentaría a la computadora para que ésta la procesara, se procesaba y se imprimían los resultados. El proceso de leer información y procesarla por completo se conoce como procesamiento por lotes**.**

Luego en la década de los sesenta comenzó a florecer un nuevo tipo de servicio de red comercial conocido como tiempo compartido**.** El tiempo compartido permitió que se instalaran las terminales en lugares geográficamente aisladas a la computadora anfitriona, en locales de negocio o en centros de cómputos específicos, desde donde podrían servir para acceder a los recursos de cómputo de la computadora anfitriona. En esta etapa los archivos que se manejaban eran propios del fabricante, como por ejemplo IBM o Xerox, IBM y American Airlines, con el sistema SABRE. Por esa época también surgieron muchos sistemas construidos en leguajes como Assembler, Fortran, Cobol etc. En 1981, IBM introdujo la IBM PC y puso en escena lo que sería el futuro de la computación personal. Aunque ya habían sido introducidas varias computadoras personales (también llamadas microcomputadoras) unos cuantos años antes que la IBM PC, esto trajo sistemas en las empresas, muy comunes, los de contabilidad y de personal, en lenguajes como Pascal, C, FoxPro, Basic, entre otros, cada uno con una forma diferente de manejar sus archivos y repositorios de información.

Pero tras todo lo dicho sobre la evolución tecnológica en sistemas de información, tenemos que tomar en cuenta el concepto de evolución tecnológica en las empresas y

cabe hacer mención al Modelo NOLAN. En la década de los setenta, Richard Nolan, profesor de la Escuela de Negocios de Harvard, desarrolló una teoría que impactó el proceso de planeación de los recursos y las actividades de la informática.

Según Nolan [43], la función de la Informática en las organizaciones evoluciona a través de ciertas etapas de crecimiento, las cuales se explican a continuación:

- Comienza con la adquisición de la primera computadora y normalmente se justifica por el ahorro de mano de obra y el exceso de papeles.
- Las aplicaciones típicas que se implantan son los Sistemas Transaccionales tales como nóminas o contabilidad.
- El pequeño Departamento de Sistemas de información depende en la mayoría de los casos del área de contabilidad.
- El tipo de administración empleada es escaso y la función de los sistemas suele ser manejada por un administrador que no posee una preparación formal en el área de computación.
- El personal que labora en este pequeño departamento consta a lo sumo de un operador y/o un programador. Este último podrá estar bajo el régimen de honorarios, o bien, puede recibirse el soporte de algún fabricante local de programas de aplicación.
- En esta etapa es importante estar consciente de la resistencia al cambio del personal y usuario que están involucrados en los primeros sistemas que se desarrollan, ya que estos sistemas son importantes en el ahorro de mano de obra.
- Esta etapa termina con la implantación exitosa del primer Sistema de Información. Cabe recalcar que algunas organizaciones pueden vivir varias etapas de inicio en las que la resistencia al cambio por parte de los primeros usuarios involucrados aborta el intento de introducir el computador a la empresa.

Etapa de contagio o expansión.

- Se inicia con la implantación exitosa del primer Sistema de Información en la organización. Como consecuencia de lo anterior, el primer ejecutivo usuario se transforma en el paradigma o persona que habrá que imitar.
- Las aplicaciones que con frecuencia se implantan en esta etapa son el resto de los Sistemas Transaccionales no desarrollados en la etapa de inicio, tales como facturación, inventarios, control de pedidos de clientes y proveedores, cheques, etc.
- El pequeño departamento es promovido a una categoría superior, donde depende de la Gerencia Administrativa o Contraloría.
- El tipo de administración empleado está orientado hacia la venta de aplicaciones a todos los usuarios de la organización; en este punto suele contratarse a un especialista de la función con preparación académica en el área de sistemas.
- Se inicia la contratación de personal especializado y nacen puestos tales como analista de sistemas, analista-programador, programador de sistemas, jefe de desarrollo, jefe de soporte técnico, etc.
- Las aplicaciones desarrolladas carecen de interfaces automáticas entre ellas, de tal forma que las salidas que produce un sistema se tienen que alimentar en forma manual a otro sistema, con la consecuente irritación de los usuarios.
- Los gastos por concepto de sistemas empiezan a crecer en forma importante, lo que marca la pauta para iniciar la racionalización en el uso de los recursos

computacionales dentro de la empresa. Este problema y el inicio de su solución marcan el paso a la siguiente etapa.

Etapa de control o formalización.

Para identificar a una empresa que transita por esta etapa es necesario considerar los siguientes elementos:

- Esta etapa de evolución de la Informática dentro de las empresas se inicia con la necesidad de controlar el uso de los recursos computacionales a través de las técnicas de presupuestación base cero (partiendo de que no se tienen nada) y la implantación de sistemas de cargos a usuarios (por el servicio que se presta).
- Las aplicaciones están orientadas a facilitar el control de las operaciones del negocio para hacerlas más eficaces, tales como sistemas para control de flujo de fondos, control de órdenes de compra a proveedores, control de inventarios, control y manejo de proyectos, etc.
- El departamento de sistemas de información de la empresa suele ubicarse en una posición gerencial, dependiendo del organigrama de la Dirección de Administración o Finanzas.
- El tipo de administración empleado dentro del área de Informática se orienta al control administrativo y a la justificación económica de las aplicaciones a desarrollar. Nace la necesidad de establecer criterios para las prioridades en el desarrollo de nuevas aplicaciones. La cartera de aplicaciones pendientes por desarrollar empieza a crecer.
- En esta etapa se inician el desarrollo y la implantación de estándares de trabajo dentro del departamento, tales como: estándares de documentación, control de proyectos, desarrollo y diseño de sistemas, auditoria de sistemas y programación.
- Se integra a la organización del departamento de sistemas de información, personal con habilidades administrativas y preparadas técnicamente.
- Se inicia el desarrollo de interfaces automáticas entre los diferentes sistemas.

Etapa de integración.

Las características de esta etapa son las siguientes:

- La integración de los datos y de los sistemas de información surge como un resultado directo de la centralización del departamento de sistemas bajo una sola estructura administrativa.
- Las nuevas tecnologías relacionadas con base de datos, sistemas administradores de bases de datos y lenguajes de cuarta generación, hicieron posible la integración.
- En esta etapa surge la primera hoja electrónica de cálculo comercial y los usuarios inician haciendo sus propias aplicaciones. Esta herramienta ayudó mucho a que los usuarios hicieran su propio trabajo y no tuvieran que esperar a que sus propuestas de sistemas fueran cumplidas.
- El costo del equipo y del software disminuyó por lo cual estuvo al alcance de más usuarios.
- En forma paralela a los cambios tecnológicos, cambió el rol del usuario y del departamento de Sistemas de Información. El departamento de sistemas evolucionó

hacia una estructura descentralizada, permitiendo al usuario utilizar herramientas para el desarrollo de sistemas de información.

- Los usuarios y el departamento de sistemas de información iniciaron el desarrollo de nuevos sistemas, reemplazando los sistemas legacy o heredados, en beneficio de la organización.

Etapa de administración de datos.

Entre las características que destacan en esta etapa están las siguientes:

- El departamento de tecnologías de información reconoce que la información es un recurso muy valioso que debe estar accesible para todos los usuarios.
- Para poder cumplir con lo anterior resulta necesario administrar los datos en forma apropiada, es decir, almacenarlos y mantenerlos en forma adecuada para que los usuarios puedan utilizar y compartir este recurso.
- El usuario de la información adquiere la responsabilidad de la integridad de la misma y debe manejar niveles de acceso diferentes.

Etapa de madurez.

Entre los aspectos sobresalientes que indican que una empresa se encuentra en esta etapa, se incluyen los siguientes:

- Al llegar a esta etapa, la Informática dentro de la organización se encuentra definida como una función básica y se ubica en los primeros niveles del organigrama (dirección).
- Los sistemas que se desarrollan son Sistemas de Manufactura Integrados por Computadora, Sistemas Basados en el Conocimiento y Sistemas Expertos, Sistemas de Soporte a las Decisiones, Sistemas Estratégicos y, en general, aplicaciones que proporcionan información para las decisiones de alta administración y aplicaciones de carácter estratégico.
- En esta etapa se tienen las aplicaciones desarrolladas en la tecnología de base de datos y se logra la integración de redes de comunicaciones con terminales en lugares remotos, a través del uso de recursos computacionales.

En el párrafos anteriores se dio un panorama general acerca de la implantación de sistemas de información desde los inicios de la informática en la en el mundo organizacional, y la evolución que han tenido en las últimos 60 años, Nolan nos da una perspectiva de cómo evoluciona la informática en las empresas, y esto nos conlleva más adelante a situarnos en el problema de integración de información.

2.1.1 Definición

Los sistemas Legacy o heredados, en nuestro idioma, son nombrados así porque representan aquellas herramientas concebidas décadas anteriores a la actual, y son el soporte de las operaciones centrales de muchas empresas, pero que han quedado relegadas en el tiempo, por sus limitaciones. Según Alfredo Rodríguez en "La Gestión de la evolución del software" [14]: *El eterno problema de la mayor parte de los grandes sistemas de información que hoy están funcionando en las empresas, fueron desarrollados en los años ochenta. La irrupción de las*

tecnologías relacionadas con Internet, el paradigma de objetos, los componentes distribuidos y la nueva mentalidad empresarial que intenta ofrecer mejores servicios a sus clientes y durante más tiempo, han provocado que la información que permanecía en los viejos sistemas y que es totalmente aprovechable, sea objeto de diversos tratamiento para su recuperación.

2.1.2 Tipo de Sistemas Legacy o Heredados

Puede hacerse una tipificación de los sistemas Legacy o heredados de la siguiente manera:

2.1.2.1 Sistemas transaccionales

Son aquellos sistemas que cuyo objetivo fue el de llevar a cabo las tareas de los procesos de soporte a dar valor a la organización como son de personal, logística, contabilidad, finanzas, gestión de ventas, entre otros. Entre ellos también se pueden contar a los sistemas ERP (Enterprise Resource Planning) que son los sistemas integrados que constan de todos los módulos principales para grandes organizaciones, mayormente estos sistemas son vendidos por compañías grandes como People Soft y SAP, también compañías peruanas como Novatronic, Sonda, Ofisis, todos con un esquema propio y al cual las empresas tienen que adaptarse, no así los sistemas singulares que cada uno tiene sus propias formas y modelos de trabajo, con sus propios formatos. Los sistemas transaccionales registran el día a día de las organizaciones, los más comunes son los sistemas de contabilidad, mayormente casi todas las empresas tienen al menos un sistema de contabilidad.

2.1.2.2 Sistemas de procesos automatizados

Los sistemas de procesos automatizados, son usados junto con los equipos electrónicos o mecánicos, en su mayoría están hechos con lenguajes de bajo nivel como C, Assembler, Pascal, entre otros. Estos sistemas registran datos.

Como medidas de tiempo, de potencia, de producción, calor, nivel de combustible, en el caso de motores, controlan el accionar de las maquinas por medio de programas predefinidos y repetitivos y pueden o no guardar información.

2.1.2.3 Lenguajes en que fueron desarrollados

En el siguiente cuadro se muestra una lista de los lenguajes de programación, y su evolución desde el inicio de la informática:

Tabla 2.1. Evolución de los Lenguajes de Programación desde sus Inicios [30]

AÑO	LENGUAJE	INVENTOR	DESCRIPCION
1900s	BINARIO	Bool	Primer lenguaje
1946	Plankalkul	Konrad Zuse	Creado para jugar al ajedrez
1949	Short Code		Lenguaje traducido a mano
1950	ASM (ensamblador)		Lenguaje ensamblador
1951	A-0	Grace Hopper	Fue el primer compilador
1952	AUTOCODE	Alick E. Glennie	Compilador muy rudimentario
1956	FORTRAN	IBM	Sistema de Traducción de Formulas matemáticas
1956	COBOL		Compilador
1958	ALGOL 58		
1960	LISP		Interprete orientado a la Inteligencia Artificial
1961	FORTRAN IV	IBM	Sistema de Traducción de Fórmulas matemáticas
1961	COBOL 61 Extendido		
1960	ALGOL 60 Revisado		
1964	PASCAL	Niklaus Wirth	Programación estructurada
1964	BASIC	Universidad de Dartmouth (California)	Beginners All Purpose Symbolic Instruction Code
1965	SNOBOL		
1965	COBOL 65		
1966	PL/I		
1966	FORTRAN 66	IBM	Sistema de Traducción de Fórmulas matemáticas
1967	SIMULA 67		
1968	ALGOL 68		
1968	SNOBOL4		
1970s	GW-BASIC		Antiguo y clásico BASIC
1970	APL/360		
1972	SMALLTALK	Centro de Investigación de Xerox en Palo Alto	Pequeño y rápido
1972	C	Laboratorios Bell	Lenguaje con tipos
1974	COBOL 74		
1975	PL /I		Lenguaje sencillo

1977	FORTRAN 77	IBM	Sistema de Traducción de Fórmulas matemáticas
1980s	SMALLTALK/V	Digitalk	Pequeño y rápido
1980	C con clases	Laboratorios Bell	Lenguaje con clases
1981	PROLOG	Ministerio Japonés de Comercio Internacional e Industria (MITI)	Lenguaje estándar para la Inteligencia Artificial
1982	ADA	Ministerio de Defensa de los EE.UU	Lenguaje muy seguro
1984	C++	AT&T Bell Laboratories (Bjarne Stroustrup)	Compilador
1985	CLIPPER		Compilador para bases de datos
1985	QuickBASIC 1.0	Microsoft®	Compilador de BASIC
1986	QuickBASIC 2.0	Microsoft®	Soporte de tarjeta gráfica EGA
1987	QuickBASIC 3.0	Microsoft®	43 lineas con la tarjeta EGA
1987	QuickBASIC 4.0	Microsoft®	Tarjetas Hércules,

En el Anexo 2 se describen a más detalle algunos de los más importantes lenguajes de programación de la tabla anterior.

2.1.2.4 Evolución de Sistemas Legacy o Heredados

Hablar de evolución involucra varios aspectos, desde corrección de fallas o problemas presentados, el acostumbrado mantenimiento cada cierto tiempo, pasando por una adaptación y a nuevas necesidades, agregar nuevas funcionalidades y nuevos procesos y terminando con el desecho total de todo el sistema, es decir su dado de baja y/o reemplazo por uno nuevo totalmente desde cero.

En el caso de resolver problemas y fallas del sistema de información, es una situación que depende de factores como la calidad de la aplicación, que es un elemento crucial ya que en ello muchas veces radica la negativa de las organizaciones a reemplazar sistemas antiguos, como puede ser el caso de aplicaciones hechas en lenguaje Assembler, y que funcionan todo el tiempo con una eficiencia normal a lo largo de los años desde su implantación. También puede depender del entorno, cuando el hardware que soporta al sistema cambia, la tecnología de red y entre otras cosas influyen en el desempeño de la aplicación. Sin embargo si la calidad no es tan confiable entonces se tienen que arreglar errores, problemas, y realizar actualizaciones continuas con la respectiva compilación y revisión de la documentación, si el sistema la tuviera.

Históricamente, [14] el mantenimiento se puede considerar como la primera forma de evolución de los sistemas de información. La

mayoría de las empresas que realizaron sus grandes desarrollos hace 25 años, necesitan, sobre todo si la aplicación sigue resultando efectiva, la realización de estas actividades.

El ANSI/IEEE da la siguiente definición de mantenimiento:

Las modificaciones de los productos software después de su entrega para corregir fallos, mejorar rendimiento u otros atributos o adaptar el producto a un cambio de entorno.
Una definición similar es dada por ISO/IEC12207:

> *Un producto software soporta una modificación en el código y su documentación asociada para la solución de un problema o por la necesidad de una mejora. Su objetivo es mejorar el software existente manteniendo su integridad.*

El mantenimiento representa el porcentaje más alto del costo de todo el ciclo de vida de un sistema

Lientz y Swanson proponen dividir en cuatro los tipos de mantenimiento:

- *Adaptativo para adaptar el software a un cambio de entorno*
- *Perfectivo para suplir nuevos requerimientos*
- *Correctivo para solucionar errores puntuales de programas*
- *Preventivo para prever la aparición de problemas*

Según el estudio de Lientz y Swanson, los dos primeros consumen el 75% de los recursos y el Correctivo el 21%. Otros estudios proporcionan resultados similares. Todos muestran que, es la incorporación de nuevos requerimientos de usuario la causa principal de la evolución y el mantenimiento del software.

Si los cambios se pudieran anticipar en el diseño, se podrían prever en forma de parametrizaciones. El problema fundamental es que, el elevado número de cambios que el software soporta a lo largo de su ciclo de vida, hace imposible una previsión en el diseño del sistema. Por tanto, el mantenimiento debe tener una consideración especial porque:

- *Consume una gran parte del coste total del ciclo de vida del software.*
- *Imposibilita un cambio rápido y fiable en el software haciendo perder oportunidades de negocio.*
- *Dificulta la adopción de nuevas tecnologías.*
- *Ocasiona daños en la integridad del sistema.*

Por ello, es previsible que la investigación en el campo del mantenimiento del software vaya en aumento en los próximos años, complementada por la reingeniería de sistemas.

Aunque algunos autores enfatizan el ciclo de mantenimiento hasta el punto de asimilarlo a la evolución de un sistema de información, en la comunidad del software se admite que el mantenimiento es sólo una estrategia de micro evolución [14].

Una adaptación a nuevas necesidades del cliente dentro de la empresa o de los usuarios que operan el sistema, eso hace que se modifique la concepción original de la aplicación creándole nuevas funcionalidades que cumplan con el cometido y solucionen el problema o vacío presentado, pero esto a su vez causa ya que el sistema se haga cada vez más amorfo, y ya su mantenimiento cada vez se haga más tedioso y sin una documentación precisa y disciplinada se puede perder el control de la situación hasta acabar con un desorden total. Este es el peligro de colocar funcionalidad tras funcionalidad, a un sistema que al final termina por perder el objetivo con el cual fue concebido.

Una última alternativa es la de cambiar el sistema actual por uno nuevo, ya sea por desarrollo desde cero o por compra de un sistema integrado, pero se debe tener en cuenta que ello involucra la migración y adaptación de los datos al nuevo sistema, teniendo en cuenta formatos, longitudes de cadenas, tipos de datos, etc. Para que el nuevo sistema pueda empezar a explotar la data, ahora esta migración puede ser paulatina, es decir escalonada, de acuerdo a las necesidades que tenga el nuevo sistema de la información del sistema Legacy o heredado, o puede ser total en un solo paso, con el riesgo de llevar también data no consistente, o de que se pierdan datos o se transformen en sus formatos y tipo al momento de migrar, para ello se debe realizar un Plan de Migración que verifique cuáles son los riesgos que habría que afrontar.

Capítulo III

Estado del Arte de Integración

La integración es la búsqueda de la unión, de poder enlazar elementos que están separados, para la informática es importante estar interconectado así se gana valor agregado para realizar innovaciones y se puede brindar soluciones en conjunto. En el presente capítulo se muestra los aspectos actuales en lo referente a integración de la información.

3.1 Definiciones

En el caso de los sistemas de información, la idea de integración es la de unir partes que han sido automatizadas no necesariamente que sean antiguas o de estos últimos años, es más bien la de unir arquitecturas tecnológicas, comunicar sistemas y plataformas, compartir datos, recursos; y que unidos puedan desempeñarse como si fuera un solo sistema brindando una funcionalidad superior a los usuarios, cumpliendo los objetivos que son necesarios para la empresa u organización que requieran la unión de los sistemas de información.

3.2 Integración

Del latín*" integratĭo, -ōnis",* la integración es el proceso por el cual las cosas se unen y se adhieren a otras para constituir un todo, completar un todo con las partes que faltaban, hacer que alguien o algo pase a formar parte de un todo. Aunar, fusionar dos o más conceptos, corrientes, entidades, etc., divergentes entre sí, en una sola que las sintetice.

3.3 Middleware [44]

Middleware es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El Middleware nos abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API (Application Programming interface) para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipo de servicios de middleware.

Por lo general el middleware del lado cliente está implementado por el Sistema Operativo subyacente, el cual posee las librerías que implementan todas las funcionalidades para la comunicación a través de la red.

3.4 Antecedentes

Tras la difusión, de Internet, las tendencias de E-Business, el Comercio Electrónico, y los productos Digitales; hacen que las organizaciones apuesten por tener presencia en el mercado actual, y el hecho de no perder competitividad dado que tienen sistemas de

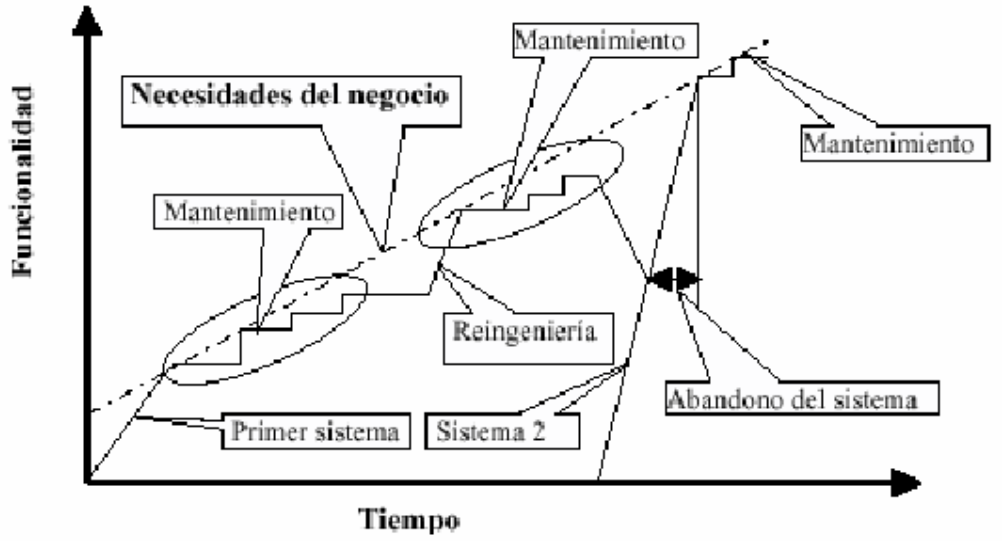
las décadas del 60, 70 y 80 las cuales guardan la información más valiosa y permanentemente en transacciones, dificulta que puedan integrarse al nuevo estándar de negocios y aplicaciones empresariales de hoy en día. Es por ello que surge la necesidad de buscar formas de integrar la data procesada por los sistemas heredados o legacy.

Veamos cómo se origina el problema:

La curva de crecimiento del software que se construye, de tiempo en tiempo se le da mantenimiento, esto también incluye hacer modificaciones hasta que el sistema ya no se pueda modificar más y se le da en abandono y nuevamente se empieza por un nuevo desarrollo.

En la figura 3.1 se muestra que en primera instancia se agrega nuevas funcionalidades al sistema para que pueda adaptarse a los nuevos requerimientos que van surgiendo, esto se va haciendo de tiempo en tiempo, hasta que el sistema conceptualizado en un primer momento pierde el objetivo para el cual fue creado, hasta que llega un momento en que se da al abandono al sistema y se empieza a construir uno totalmente nuevo.

Figura 3.1 Ciclo de Vida de un sistema de Información [14].



En el primer caso de agregar nueva funcionalidad se comparte la información de los procesos antiguos del sistema, esto no es una tarea compleja ya que ambas corren bajo el mismo sistema operativo y manejan el mismo tipo y formato de orígenes de datos.

Una segunda forma se da cuando, se realiza un sistema más moderno que el anterior, y que maneja sus propios orígenes de datos con formatos distintos y/o, bajo diferentes plataformas, entonces es necesario realizar las siguientes formas de integración de información:

3.5 Definición del Problema de Integración de Información

El problema de integración de información radica en la necesidad que tienen las organizaciones y las personas de explotar sus recursos, en este caso su principal recurso son los datos, generados de diversas formas, es decir, el problema es integrar en uso, a diferentes orígenes de información, antiguos y los de esta época, pero cobra mayor importancia integrar los orígenes de datos antiguos porque son los que contienen la información desde los inicios de la compañía, además si están actualmente operando seguramente lo están haciendo en procesos centrales importantes, como los de contabilidad, los de recursos humanos, comercialización, producción, entre otros.

Y si la data no está integrada a las nuevas aplicaciones existentes no se podrán hacer mayores cosas que duplicar la información, y estar tomando decisiones sobre supuestos y no sobre data confiable, además del tiempo que ello implica muy vital tener la información cuando más se la necesita y tomar las decisiones en el tiempo preciso, eso a su vez se transforma en eficiencia y rentabilidad para la organización.

Además están los motivos económicos de las empresas que no pueden sostener altos costos de un nuevo desarrollo de sistemas, es por eso que la integración de información es una alternativa válida de solución.

3.6 Tipos de Integración

En la integración de software informático se tocara 2 clases de dichos sistemas que están enteramente relacionados con el tema de la Tesis:

3.6.1 Integración de Sistemas Informáticos

La integración de sistemas informáticos, es la integración de sistemas enteros o de componentes, o partes del sistema el cual es de interés que esté unido a un sistema principal, para que sean parte del conjunto entero de procesos.

Es decir, cuando se integran partes de sistemas se hace que funcionen conjuntamente sin diferencia, este hecho es transparente para el usuario.

La integración de sistemas también puede a dar a entender que se están integrando partes de un mismo sistema bajo una misma tecnología, esto por ejemplo se da con los componentes COM, COM+ y CORBA, y en si estaría bien que se les llame integración, pero también se llama integración al enlace que se da entre sistemas de diferentes tecnologías y cuyos componentes puedan interactuar de forma normal a pesar de sus propias maneras de trabajar y sus propios esquemas de administración.

3.6.2 Integración de Información

La integración de información se da cuando se desea captar información que maneja otro sistema o proceso. Este sistema puede estar compuesto de software o tal vez solo hardware, ya que también puede darse el caso de querer obtener los datos por ejemplo de un sensor de movimiento que da el número de

revoluciones de un motor, entonces se puede aprovechar esta información de forma que se explote y genere un valor agregado en los procesos.

Por otro lado, si la información es manejada por Software, entonces integrar información de dicho sistema podría parecer a simple vista más factible, pero esto muchas veces no es así, ya que depende de varios factores que lo restringen como pueden ser:

- El tipo de computadora en la que fue creada la información, por ejemplo archivos de Mac y PC Compatible.
- El Sistema operativo en el que fue creada la información, por ejemplo archivos de Mac OS y Windows XP, UNIX, Linux, etc.
- Por ende ya los programas que son la fuente de los archivos así sean de la misma plataforma los crean en formato distinto.

La integración de información es la capacidad de poder compartir información entre diferentes dispositivos, plataformas y programas, para que cada uno de estos los manejen de acuerdo a su propios formatos.

3.7 Variantes

Entre las variantes que se encontraron al problema de integración de información se puede mencionar las siguientes:

3.7.1 Integración de Información de las Aplicaciones Industriales

Según Madnick, 1998 [19], una categoría importante en las aplicaciones está relacionada con el vínculo intercorporativo y la integración intracorporativa. Las aplicaciones de esta categoría requieren que varios sistemas funcionen juntos. Esta categoría de los sistemas de información recibe el nombre de sistemas de información compuestos (CIS).

3.7.2 Integración de Datos de Sensores Electrónicos

Los sensores son sistemas fotográficos u óptico-electrónicos capaces de detectar y registrar, en forma de imágenes o no, el flujo de energía radiante reflejado o emitido por objetos distantes. El gran desarrollo que ha tenido la tecnología en esta área ha permitido al hombre ampliar su capacidad visual.- Ejemplos de tales detectores son la placa fotográfica, la película fotográfica infrarroja, fotodetectores, foto multiplicadores y cámaras de televisión. Muchos de éstos son baratos y de gran eficiencia, y algunos incluso proporcionan imágenes en forma cuantitativa, es decir, en forma numérica, de tal manera que la información resultante puede ser almacenada una origen de datos para proceder a un análisis matemático posterior.

Es por ello que también es importante un aplicación en estos casos en que el software junta con componentes electrónicos, que generan información y necesitan ser analizados, y estudiados de una forma más óptima.

3.8 Aplicaciones

Las aplicaciones de la Integración de Información se da en:

3.8.1 Integración de información de las Aplicaciones Empresariales [37]

Es el sector en el cual se necesitan más la integración de información de contenida o manipulada por sistemas heredados, aquí se necesitan bastante los datos de las aplicaciones de por ejemplo:

- **Contabilidad**

  Este sistema es el que casi primeramente se ha informatizado en la mayoría de las empresas, la contabilidad que antes era por libros y documentos se mejoro con la implementación de sistemas, incluso yendo más allá en antigüedad en los sistemas de tarjetas perforad; el sistema de contabilidad típico cuenta con las siguientes funcionalidades:

- **Recursos Humanos**

  Este es otro sistema muy común en las empresas que generan gran cantidad de información y que debido a su antigüedad no puede aprovecharse esos datos y poder explotarlos de una forma adecuada, entre las funcionalidades principales que tiene un sistema de Recursos Humanos común son:

  Control de personal, terminación de los vínculos del empleado con la organización [37].

3.8.2 Aplicaciones de Tratamiento de Imágenes [37]

Las imágenes obtenidas de un mismo objeto con diferentes técnicas de microscopía: petrográfica, metalografía, de fluorescencia y electrónica de barrido y su digitalización en un ordenador, permiten medir numerosos parámetros y obtener nuevos datos.

Este proceso también se utiliza con imágenes normales, digitalizadas con una cámara de vídeo o un escáner, o imágenes especiales de termografía, teledetección, fotogrametría, etc.

Se ha dado un repaso a unos ejemplos de aplicaciones de la integración de información, pasando por la necesidad de las empresas con sistemas heredados, que tienen datos que necesitan ser integrados, también con los sistemas industriales con dispositivos que generan data, y las nuevas aplicaciones a la Medicina, dando a conocer un nuevo proyecto realizado en España, y las aplicaciones de tratamiento de imágenes, que es una forma de integración de información no necesariamente desde sistemas Legacy o heredados.

3.9 Soluciones o Aplicativos Existentes para Integración de Información

En cuanto a aplicaciones de integración de información existen diversos fabricantes y empresas que se dedican a la construcción de soluciones muchas de ellas son soluciones a detalle del problema presentado; se escogieron una muestra de las más importantes:

3.9.1 SYBASE

La propuesta de SYBASE, tomada de la dirección http://www.sybase.es/productos.htm, nos muestra toda una suite de productos orientados a la integración de datos:

*Las [38] soluciones **Unwired Enterprise** de Sybase incluyen software de gestión de información basadas en estándares abiertos, desarrollo, integración y movilidad, que trabaja dentro de la infraestructura existente del cliente para hacer posible un acceso en tiempo real a la información corporativa y a las aplicaciones, tanto en la oficina como fuera de ella.*

Gestión de la información

Las soluciones de gestión de la información de Sybase ofrecen software integrado para ayudar a las empresas a mejorar el tiempo de respuesta, reducir costes y reducir la complejidad de gestionar sus heterogéneos entornos de información.

Servidores de información y datos

*- **Sybase Adaptive Server Enterprise** es un sistema de gestión de bases de datos relacionales de clase empresarial.*

Opción de servicio en tiempo real: hace posible la agilidad en los negocios proporcionando una visión dinámica de la información con alertas y notificaciones en tiempo real.

Otras opciones: alta disponibilidad, gestión de transacciones, XML y servicios Web.

*- **Sybase IQ 12.5**, base de datos relacional altamente escalable optimizada para un rendimiento aún no superado en entornos de business intelligence y data warehouses.*

Opciones de integración de información

*-**Sybase RepConnect 2.01**: captura y transforma los acontecimientos informativos en XML para facilitar el transporte.*

*- **Sybase OpenSwitch**: incrementa la calidad de los datos al monitorizar y gestionar las transacciones y comunicaciones a una conclusión segura.*

*- **Sybase Information Middleware**: acceso seguro a sistemas de información líderes de IBM, Oracle, Microsoft y cualquier ODBC a una dócil base de datos.*

*- **Desarrollo e integración**: Las soluciones Sybase para el diseño, desarrollo, despliegue e integración de aplicaciones permite a los equipos de desarrollo responder a las necesidades de la línea de negocios con una rapidez sin precedentes. Las innovaciones patentadas incrementan la productividad de los desarrolladores al proporcionar unas herramientas significativamente más fáciles de usar, que reducen el tiempo y la complejidad del proyecto mientras mejora la colaboración entre los negocios y los miembros del equipo técnico [38].*

3.9.2 Microsoft

Se tomó información del producto de Microsoft para integración empresarial de sistemas e información, muy completo muy atrayente ya que cumple con todos los requisitos de un gran servidor de integración pero con un costo elevado.

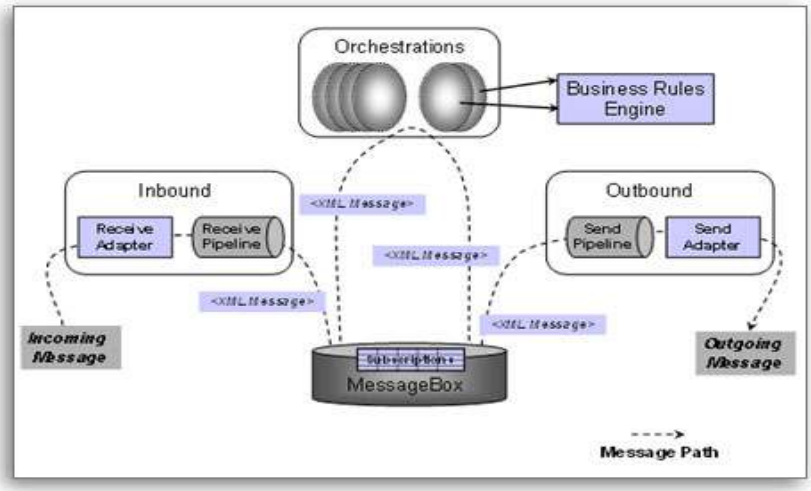
BizTalk Server [39] es un producto de Windows Server System que permite a los clientes integrar sistemas, empleados y asociados comerciales con más eficacia y rapidez que nunca.

En el actual marco de economía globalizada, las empresas están obligadas a integrar aplicaciones, sistemas y tecnologías muy diferentes. Para facilitar esta integración, Microsoft ofrece BizTalk Server 2004, una tecnología de integración de última generación, que además amplía la oferta directa de aceleradores y adaptadores industriales y a través de asociados [39].

Capturas de pantalla

El cambio más notable con respecto a BizTalk Server 2002 es que la interfaz de usuario se ha consolidado en Visual Studio .NET.

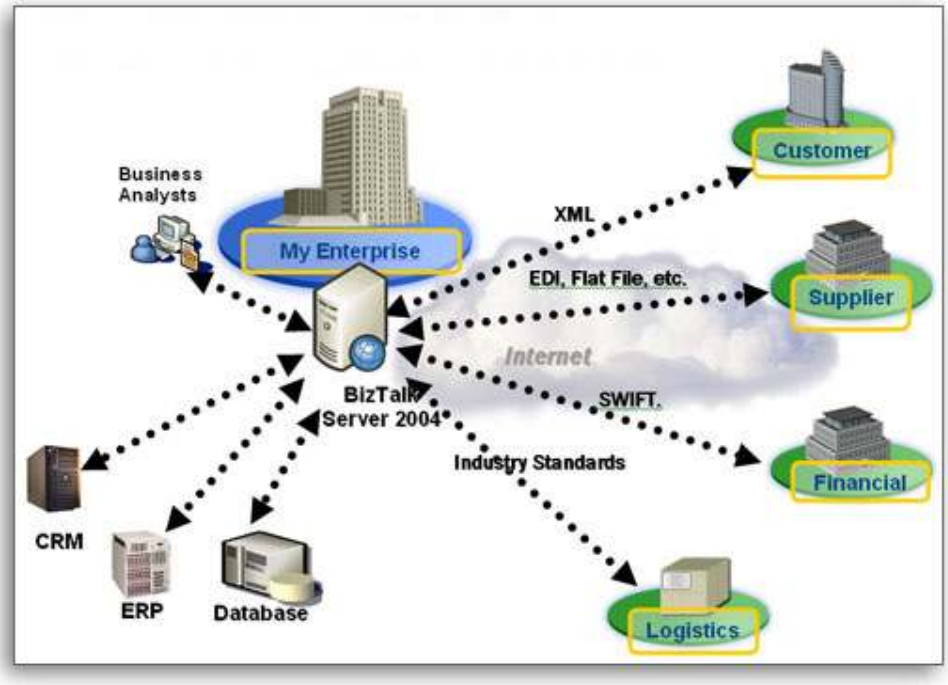
Figura 3.2 Arquitectura BizTalk Server *[39]*



Ventajas empresariales

La conexión de socios comerciales y la integración de sistemas han dejado de ser el objetivo a largo plazo de la integración empresarial. En la actualidad, las empresas necesitan funciones de administración de procesos empresariales altamente automatizadas con la flexibilidad necesaria para incorporar un toque humano en determinadas fases del flujo de trabajo. Y esto es precisamente lo que proporcionan los servicios de instrumentación de BizTalk Server 2004. Además, con el motor de reglas de BizTalk Server 2004, las empresas pueden implementar reglas empresariales muy flexibles que estarán a disposición de los expertos de la información [39].

Figura 3.3 Orquestación Empresarial con BizTalk Server *[39]*



3.9.3 IBM

DB2 Content Manager

IBM DB2 Content Manager proporciona una base para la gestión, el acceso y la integración de la información empresarial más importante "on demand". Le permite integrar todos los formatos de contenido (documentos, web, imágenes, soportes enriquecidos) en los diversos procesos y aplicaciones empresariales, como Siebel, PeopleSoft y SAP. Content Manager se integra con las inversiones en hardware y software existentes, tanto de IBM como de otras marcas, lo que permite a los clientes aprovechar la infraestructura común, conseguir un coste más bajo de propiedad y ofrecer información y servicios potentes y nuevos a los clientes, socios y empleados, dónde y cuándo los necesiten. DB2 Content Manager for Multiplatforms V8.2 [40].

- *DB2 Content Manager for Multiplatforms V8.2*
- *DB2 Content Manager Express Edition*
- *Content Manager for iSeries (US)*
- *Content Manager for OS/390 (US)*
- *DB2 ImagePlus for z/OS (US)*
- *DB2 Content Manager OnDemand (US)*
- *DB2 Common Store*
- *DB2 Document Manager (US)*
- *DB2 Record Manager*

DB2 Information Integrator

IBM DB2 Information Integrator representa la próxima generación de software de integración de información. Proporciona una base para el e-business "on demand", lo que permite a las empresas tener acceso integrado, en tiempo real, a información variada y distribuida. Como resultado, las empresas pueden aumentar su eficacia, atender mejor a los clientes y a los proveedores, tomar mejores decisiones y responder con más rapidez a las nuevas oportunidades o amenazas. IBM DB2 Information Integrator V8.1 proporciona acceso integrado en tiempo real a una diversidad de datos, como si fuese una única base de datos, independientemente de su ubicación. DB2 Information Integrator V8.1 [40].

- *DB2 Information Integrator Classic Federation for z/OS (US)*
- *DB2 Information Integrator for Content*
- *DB2 DataPropagator (US)*

El párrafo anterior para Integración de IBM, fue tomado del sitio Web: http://www-306.ibm.com/software/es/data/

3.9.4 Novatronic

Las soluciones que provee Novatronic generalmente se enmarcan en el desarrollo de aplicaciones bajo arquitectura Cliente/Servidor, de Sistemas Distribuidos, Multiplataforma y de Internet (Web), utilizando tecnología de Bases de Datos, de Seguridad, de Comunicaciones y de Ambiente Visual [41].

La familia de productos SIX está dividida en herramientas y aplicaciones que facilitan la integración de sistemas multiplataforma, el acceso remoto, el acceso vía tecnología Web y el proceso de transacciones bajo estándares internacionales [41].

Por el lado de herramientas se cuenta con los siguientes productos:

SIX/TCL Teleacceso de Clientes

*El **SIX/TCL** opera en el esquema cliente/servidor y provee de una serie de funciones que permiten que el programa aplicativo cliente pueda solicitar una determinada gama de servicios (transferencia de archivos, transacciones, entre otros) al **SIX/TCL** server.*

Las instituciones que desean brindar servicios de acceso remoto a su información, tal es el caso de Bancos, Compañías de seguros, Centrales de Información, Centrales de órdenes de pedido.

Empresas que desean integrar sus operaciones con sus clientes y proveedores en un esquema de comercio electrónico interempresarial.

Empresas que deseen establecer redes privadas de intercambio de información (transferencia electrónica de archivos), en la forma de mensajes, datos o imágenes.

Cualquier otra institución que requiera proveer información, atender transacciones y/o intercambiar información (archivos) con sus clientes, proveedores, vendedores, representantes, así como su propio personal.

SIX/EFT Switch

Hoy en día las instituciones financieras enfrentan el reto de satisfacer una creciente demanda de productos que permitan a sus usuarios acceso electrónico a sus fondos en forma confiable y segura. Asimismo el desafío de incorporar avances tecnológicos tales como la tecnología de tarjetas inteligentes (smart cards), y los canales de entrega de servicios como kioscos multimedia, Internet, Cajeros Automáticos, Puntos de Venta POS, Sistemas de respuesta por voz (IVR) y acceso remoto transaccional hace necesario que se cuente con una plataforma única e integrada para brindar estos servicios.

Para brindar la atención a estos requerimientos, Novatronic ha desarrollado una plataforma de servicios transaccionales financieros basada en el estándar ISO 8583, denominada SIX/EFT Switch.

SIX/EFT Switch provee a su institución de una infraestructura que le permite desplegar en forma modular sus aplicaciones de misión crítica. SIX/EFT Switch establece un grupo de funciones que son requeridas para brindar los servicios de Cajeros Automáticos SIX/ATM, SIX/Internet Banking, SIX/Kioscos, SIX/PPV (Prepago Virtual), SIX/OLC (Recaudaciones en Línea) y asimismo se pueden integran otras aplicaciones propietarias.

SIX/WebLink

El SIX/WebLink es un producto desarrollado utilizando la última tecnología disponible tanto en los aspectos de seguridad, cifrado de información, comunicaciones y control, garantizando de esta manera la ejecución de transacciones en forma segura y facilitando por ende la implementación de páginas de información dinámica sobre Internet.

Estas facilidades, se integran con los sistemas de seguridad empleados por su institución, tales como son los Firewalls y la tecnología de socket seguros [41].

Capítulo IV

Análisis de la Tecnología orientada a la integración de Información

Las herramientas de desarrollo, las tecnologías desarrolladas hoy en día son importantes y sirven de mucha ayuda en la consecución de la unión y comunicación entre aplicaciones, entre las principales tecnologías usadas destaca el lenguaje extensible de marcas (XML), también los servicios Web y los servidores de aplicaciones.

Todo ello hace posible diseñar una solución de integración de información también cabe mencionar que varias de estas herramientas son de plataforma de software libre y esto facilita la implementación multiplataforma de la solución.

4.1 Servicios web

Los servicios Web XML permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado, esto nos da una gran facilidad, porque a partir de esta concepción de los Servicios Web es que se da inicio a la realización de muchas arquitecturas que involucran su utilización.

La implementación y consumo de los servicios Web tienen diferentes maneras de acuerdo a la plataforma de desarrollo que se esté utilizando para realizar estas tareas, cada plataforma tiene ciertas peculiaridades para construir y publicar a los Servicios Web, entonces la utilización de herramientas de software libres, no es la excepción cada una de ellas tiene una complejidad determinad por la instalación, versiones y configuraciones.

Pero una vez que se logró instalar correctamente este tipo de sistemas funcionaban con total normalidad y en buena forma, es el caso de:

- AXIS (plug ins para SOAP)

Utilitarios para XML como:

- XDOCLET

4.2 Servidores de Aplicaciones

Los servidores de aplicaciones del entorno de software libre son variados, tanto en configuración como en versiones, ya que las versiones se van liberando cada vez más rápidamente, pero dentro de esta amplia gama de servidores existen 2 que destacan y que actualmente tiene bastante acogida:

- Apache TomCat
  El más conocido Servidor de Aplicaciones o Servidor Web, basado en la Maquina Virtual de Java, en donde pueden ejecutarse páginas JSP, Archivos .java, JAR, EAR, WAR.

- JBoss
  Un servidor, basado en la Tecnología de Apache, pero con funcionalidades más ligeras y con más facilidad de configurar, está tomando cada vez más fuerza debido a que muchas aplicaciones que antes se ejecutaban en Apache están migrando a JBoss, aplicaciones empresariales basadas en EJB (Enterprise Java Beans) y Web Services.

- Sun System Application Server 9.0

  Sun Java System Application Server es parte de Sun Solaris, es posible realizar una descarga de este servidor de aplicaciones en forma gratuita, pero si se desea soporte y mantenimiento tiene un costo de licenciamiento [45].

  A partir de la versión 9, SJSAS Plataform Edition se encuentra siendo desarrollada bajo el proyecto de código libre GlassFish y bajo las licencias de CDDL y GPL. Este proyecto incluye código de otras compañías tales como Oracle y bases para el JEE5 [45].

4.3 Interpretes (parsers) de XML

4.3.1 SAX

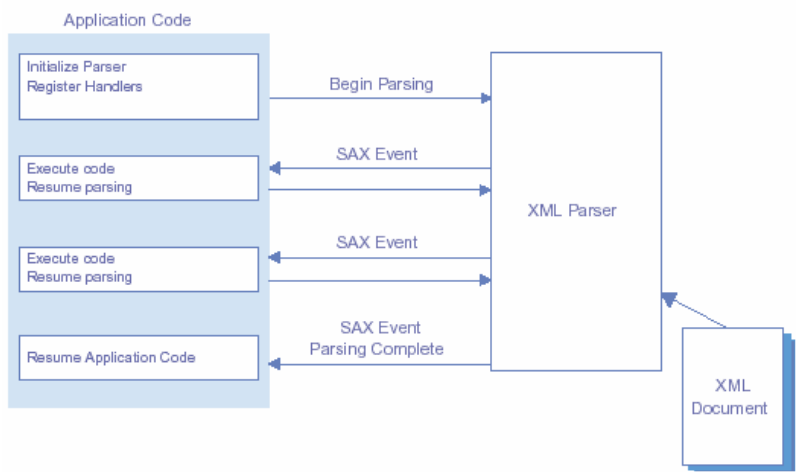
Es uno de los intérpretes más populares, puede ser usado en 2 modos distintos, según como se requiera en la aplicación. El primer modo es un modelo basado en eventos llamado Simple API para XML (SAX). Usando SAX, el intérprete lee en origen de datos XML y realiza una llamada (callback) hacia la aplicación cliente donde este se encuentre en cualquier sección distinta del Documento XML. Por ejemplo, un evento SAX es lanzado siempre que ha sido encontrado el final de un elemento XML. EL evento incluye el nombre del elemento que ha finalizado.

Para usar SAX, se debe implementar un manejador de eventos (event handler) para que el intérprete lo use mientras recorre el documento XML. Este manejador de eventos es una máquina de estado que agrega data como si estuviera siendo interpretada y manejada en un conjunto de datos en un subdocumento independientemente uno del otro. El uso de SAX es descrito en la Figura 4.1. SAX es el método más rápido de interpretación de XML, y es apropiado para manejar grandes documentos los cuales no podrían ser leídos en memoria todos al mismo tiempo [20].

Una de las dificultades al usar SAX es la inhabilitación para poder ver el documento mientras es interpretado. El manejador SAX es una máquina de estado que puede operar solo en la porción del documento XML ya ha sido interpretado.

Otra desventaja es la falta de relaciones predefinidas entre nodos dentro del documento. El orden de desarrollo y lógica basada en nodos padres e hijos, se deben describir manualmente las relaciones entre los nodos.

Figura 4.1 Descripción del modo de Trabajo del Parser SAX [20].

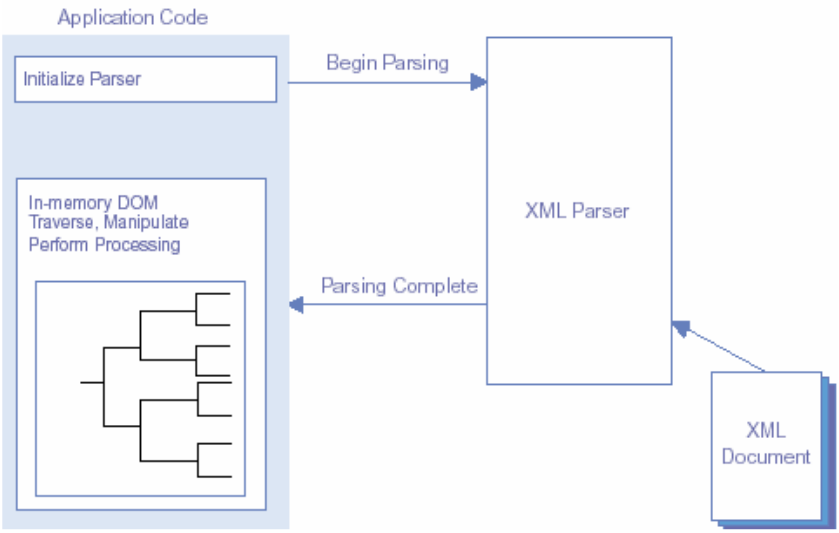


4.3.2 DOM

El otra forma de interpretar XML es usando el Modelo de Objetos de Documento (DOM Document Object Model), en vez de SAX. En el modelo DOM, el intérprete leerá enteramente el origen de datos XML y construirá una representación de tipo de árbol en memoria. Bajo DOM, un apuntador hacia el documento entero es retornado a la aplicación que la llama. La aplicación puede entonces manipular el documento, por medio de los nodos, agregando y eliminando contenido si fuera necesario. El uso de DOM es mostrado en la Figura 4.2.

Mientras que DOM es bastante fácil de implementar, este un algo lento y utiliza más recursos que SAX. DOM puede ser usado efectivamente con estructuras XML no muy grandes en situaciones de gran velocidad no es muy recomendable para las aplicaciones, es decir, usando DOM nativo. Pero existen derivados de DOM que permiten la utilización eficiente de DOM con documentos XML de gran cantidad de información.

Figura 4.2 Descripción del modo de Trabajo del Parser DOM [20]



4.3.2.1 DOM4J

Dom4J, es un intérprete de XML basado en DOM, disponible para el sitio Web para poder descargar el archivo es www.dom4j.org, permite la interpretación (construcción/lectura) de XML en estructuras de Árbol, es bastante portable, y su puesta en servidor es ligera, ya que no consume bastantes recursos, la desventaja de DOM4J es que aún no soporta el uso de esquemas para la validación de documentos XML.

4.4 Servicio de Encolamiento de Mensajes

Para el servicio de colas la especificación JMS, es la adecuada en java, ya que tiene todas las funcionalidades de un servidor de colas, este tiene persistencia y visibilidad de los mensajes puestos en cola.

Viendo un poco a lo que se refiere JMS y su terminología podemos decir que mensajes, son pedidos asíncronos, reportes y eventos que son consumidos por aplicaciones, no por humanos. Contienen información vital información necesaria para coordinar entre los sistemas. Contienen más precisamente data con formato que describe una específica acción de negocio. A través de este intercambio de mensajes cada aplicación se consolida los procesos asíncronos de la organización.

Si JMS nos brinda una unión de todas las características existentes de los sistemas de mensajes; estos podrían ser muy complicados para el entendimiento de los usuarios. En otras manos, JMS es más que una intersección de características de mensajería común a todos los productos. Es crucial que JMS incluya la funcionalidad necesaria para implementar sofisticadas aplicaciones empresariales.

4.4.1 Open JMS

Implementación de un servidor de mensajería (JMS) basado en las especificaciones 1.0.2 JMS. Tiene una licencia libre y las características son las típicas de estos productos: varios tipos de mensajería, envío garantizado, herramientas gráficas de administración, filtros de mensajes, persistencia con JDBC, etc.
Se puede descargar del sitio Web http://openjms.sourceforge.net [42].

4.5 Uso Apropiado de XML en Representación de Datos para Integración de Información

4.5.1 Fortalezas

XML nos da una gran facilidad para poder representar conjuntos de datos de acuerdo a las necesidades, este es un punto importante ya que no es necesario aprender la construcción de un esquema de datos complejo, porque XML puede ser interpretado a simple vista.

Otra fortaleza de XML es que es un estándar normalizado por la W3C (www.w3c.org), y por ello todas las tecnologías que utilicen y desarrollen con XML tienen que respetar los estándares, esto lleva a uniformidad de criterios, y como consecuencia de ello, XML puede ser interpretado por diferentes tecnologías y distintas plataformas, esto nos da un amplio espectro para poder desarrollar aplicaciones multiplataforma comunicándose a través de un lenguaje común XML.

Otra fortaleza de XML, es su facilidad de manejo, de interpretación y escritura, todas las plataformas tienen una manera de interpretar y escribir XML.

XML es texto plano, es decir no son datos binarios, es por eso su gran éxito en las aplicaciones orientados a Internet, ya que se puede pasar gran cantidad de información, sin el peligro de que pueda ser interrumpido por la seguridad de las redes como Firewalls, lo que no pasaba por ejemplo con los componentes tipo COM, los cuales si eran datos binarios y eran interceptados por las seguridades de las redes como posibles sospechosos de código malicioso.

XML promueve y hace factible el desarrollo de las arquitecturas orientadas al servicio, mediante los servicios Web, que están totalmente basados en XML, desde la descripción del servicio (WSDL Web Service Description Lenguaje), la estructura del mensaje SOAP, y el mensaje propiamente dicho.

4.5.2 Debilidades

Una de las posibles debilidades de XML es el tratamiento particular que le dan algunas al darle algunos agregados a las etiquetas, y eso que están bajo el mismo esquema, esto implica realizar una tarea extra para emparejar los mensajes bajo un mismo formato, pero este problema es no significativo.

4.6 Uso de Flujo de Bytes

4.6.1 Fortalezas

El uso de Streams o de Flujo de Bytes es una forma de comunicación bastante utilizada en los sistemas de información que involucran comunicación, dado que brindan toda una plataforma de soporte para poder manejar las conexiones, los puertos y las direcciones.

La mayoría de los streams de datos fluyen en una sola dirección, como sus contrapartidas en el mundo real, los ríos. Un stream de datos es o un stream de entrada o un stream de salida, y frecuentemente abrimos dos streams a la vez: uno para lectura y chequeo de validación (entrada) y otro para escritura (salida).

El flujo de Streams no es propio de un solo entorno de desarrollo, estos son usados por diferentes plataformas desde las más antiguas hasta las más recientes como J2EE y .NET

A continuación listamos las ventajas de utilizar este tipo de comunicación:

- Rapidez en la comunicación, ya que se trabaja a nivel de capa de transporte, utilizando el protocolo TCP/IP.
- Indiferente a los tipos de datos, todos los caracteres son interpretados de una forma utilizando el mismo encoding, es por ello que se puede pasar XML como stream.
- Facilidad que brinda tanto .NET como Java con clases especializadas en el tratamiento de flujo de datos.
- Puede establecerse en un servidor de aplicaciones, para que el servicio este atento todo el tiempo a las llamadas de pedido de data de los clientes.

4.6.2 Debilidades

El límite de flujo es algo que se debe manejar en la comunicación utilizando Sockets, porque se manejan buffers que almacenan la información en la maquina cliente que envía el pedido de data al servidor.

Primeramente se debe considerar la cantidad de flujo de datos que se enviarán entre los nodos para poder adecuar el tamaño del buffer y no se pierda información o colapse el sistema.

4.7 Cola de Mensajes Usados en Integración de Información

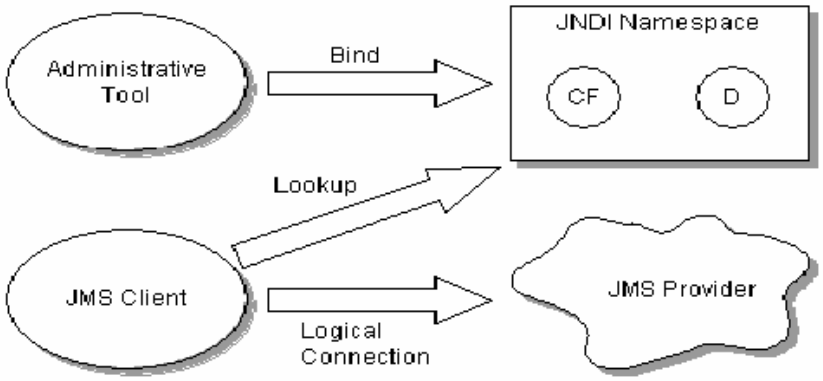
4.7.1 Múltiple concurrencia Administrada

El uso de colas de mensajes es aprovechado considerablemente en entornos de alto desempeño, con gran cantidad de concurrencia, esto hace que el servidor siempre atienda un mensaje a la vez y no se esté saturado, una de las tecnologías que se usa para encolar mensajes es Java Message Services (JMS.

JMS define algunas reglas específicas que restrinjan el uso concurrente de sesiones. Puesto que requieren más conocimiento que los específicos de JMS.

El uso de JMS, es una buena opción para hacer aun más versátil al integrador de información, en la arquitectura diseñada se ve que es considerado, pero no se ha implementado, esto puede ser un tema de gran interés para futuras investigaciones.

Figura 4.3 Administración de JMS [42].



Capítulo V

Contribución Teórica sobre integración de Información

La construcción de un integrador de información necesita de la concepción de una arquitectura que pueda sostener toda la funcionalidad necesaria para cumplir con los objetivos del proyecto. Una metodología y el uso de un lenguaje unificado es importante para modelar y ordenar los conceptos de un producto software, que al final tendrá una base teórica para que pueda ser entendido y aplicado.

5.1 Visión Conceptual del Proyecto

Para poder realizar el presente trabajo de investigación, tiene como elemento importante la construcción del software que prueba todos los fundamentos teóricos y técnicos descritos hasta el momento, y este software como tal ha sido tratado como proyecto, es por ello que se contemplan aspectos propios de la construcción metodológica de un producto software.

Desde el punto de vista metodológico el proyecto ha sido desarrollado mediante la metodología RUP, es por ello que se presentan, casos de uso, diagramas de componentes, además en los anexos se podrán encontrar con más detalle la documentación que soporta esta contribución.

5.2 Objetivos

Desarrollar un software integrador de soluciones, que integre sistemas antiguos o heredados, desarrollados en herramientas como, Foxpro, Clipper, etc., y también bases de datos relacionales, con sistemas de última generación, y así lograr un mejor aprovechamiento de los datos; dado que los sistemas heredados son como islas de información que generan gran cantidad de ésta y por la falta de de integración, no puede ser explotada.

Se ha elegido J2EE para el desarrollar la herramienta de integración, porque es un estándar que puede ser desplegado en cualquier plataforma de sistema operativo, y se ha elegido la plataforma .Net en la parte cliente del sistema porque nos brinda mayor rapidez, seguridad, mantenimiento óptimo, e interfaces más versátiles.

Como parte de prueba de esta visión se dará el caso de estudio realizado a un sistema Pro-tour basado en visual Fox Pro la empresa CTM Tours del Grupo Empresarial Costamar.

El integrador de información no solo estará ligado a una sola solución, sino que será estándar y servirá para que diferentes sistemas legacy o heredados puedan ser integrados.

5.3 Alcance

El alcance del proyecto es el de realizar un integrador entre java y .Net para orígenes de datos como archivos planos DBF y SQL Server, y otros gestores de base de datos como PosgreeSQL, MySQL, y algunos otros que podrían requerirse en el futuro.

5.4 Declaración del Problema a Resolver

Tabla 5.1 Declaración del Problema a Resolver

El problema de	*Aislamiento de información*
Afecta	*A las organizaciones y las personas*
El impacto está	*En que demoran los procesos, surgimiento del desorden, personal realizando la misma labor, demora en las actividades diarias, malestar, incomodidad en los usuarios.*
Una solución adecuada sería	• *Tener un solo repositorio de datos correcto actualizado, y alimentado por orígenes antiguos.*

El problema de	*Información Errónea.*
Afecta	*A la empresa*
El impacto está	*En la distorsión de la información muchas veces cuando se transcriben datos de u sistema cerrado que solo genera datos para él.*
Una solución adecuada sería	• *Tener los datos en formatos correctos, y en forma ordenada.*

El problema de	*Duplicidad de Información*
Afecta	*A la empresa*
El impacto está	*En que se está manejado datos doblemente, entonces las oficinas y unidades muchas veces se convierten en islas, y por consiguiente no hay integridad de la información y hay confusiones y errores en el resultado final.*
Una solución adecuada sería	• *Empezar a realizar una migración de datos que con el tiempo evolucione en la realización de un nuevo aplicativo moderno.*

5.5 Requerimientos

En esta sección se listan y describen los requerimientos previos que se realizó antes de plantear la solución de integración de información, estos requerimientos que se plantean han sido obtenidos en base necesidades generales de integración de información de sistemas legacy o heredados, y en lo que se basa en la seguridad, integridad de la información, confiabilidad del servicio, alta disponibilidad y adaptabilidad.

Seguidamente se presenta la lista de requerimientos:

Tabla 5.2 Lista de Requerimientos

Nro	Requerimiento	Atributos						
		FURPS +	Prioridad (A,M,B)	Dificultad (A,M,B)	Visibilidad (V,O)	Riesgo (A,M,B)	Categoría (P,S,O)	Precedencia
R1	Validar ingreso de Consulta de Aplicación Cliente hacia la Aplicación Servidor restringiendo el servicio solo a usuarios autorizados.	F	A	A	O	A	P	NO
R2	Transformar la cadena de consulta al formato estándar XML, para ser transmitido al Servidor.	F	A	A	O	A	P	R2
R3	Establecer Conexión entre Aplicación Cliente y Servidor a través de puertos TCP/IP.	F	A	A	O	A	P	R14, R2
R4	El Servidor debe validar el Formato XML de la Consulta.	F	A	A	O	A	P	R3
R5	Enviar Consulta a Servidor.	F	A	A	O	A	P	R4
R6	Grabar archivo Log en Aplicación Cliente.	F	A	A	V	M	S	R5
R7	Valida resultado de Consulta a orígenes de datos.	F	A	A	O	A	P	R5
R8	Servidor Transforma resultado de consulta a formato final (XML).	F	A	A	O	A	P	R7
R9	Enviar Resultado de Consulta.	F	A	A	V	A	P	R8, R7
R10	Grabar archivo Log de Eventos en el Servidor.	F	A	A	V	M	S	R9
R11	Validar Resultados en Aplicación Cliente.	F	A	A	V	M	S	R9
R12	Administrar Cola de Mensajes.	F	A	A	O	A	P	R7
R13	Establecer el protocolo TCP/IP para la comunicación entre Sistemas.	F	A	A	O	A	P	NO

Nro	Requerimiento	Atributos						
		FURPS +	Prioridad (A,M,B)	Dificultad (A,M,B)	Visibilidad (V,O)	Riesgo (A,M,B)	Categoría (P,S,O)	Precedencia
R14	Administrar de Posibles Errores.	C	A	A	O	A	P	NO
R15	El Servidor debe estar disponible las 24 horas del día.	C	A	M	O	A	P	NO
R16	Debe ser posible la encriptación de los mensajes tanto en el cliente como en el Servidor.	F	A	M	O	A	P	NO
R17	El sistema debe permitir una comunicación por Web Services, Sockets y Servidores de colas.	F	M	A	O	A	P	R3
R18	El sistema debe poder ser configurado antes de su ejecución.	F	M	A	O	A	P	NO
R19	El sistema debe permitir el cambio de configuración cuando este ejecutándose.	F	A	M	O	A	P	R3
20	El sistema debe permitir una rápida construcción de la aplicación a migrar.	F	A	A	V	A	S	NO
R21	El sistema debe permitir generar las clases que se utilizarán para la integración.	F	A	M	V	M	S	R20
R22	El sistema debe permitir guardar las configuraciones para futuras ediciones.	F	B	M	V	M	S	R21
R23	La configuración de las operaciones debe ser automatizada.	F	A	A	V	A	S	R22
R24	El sistema debe permitir generar las sentencias SQL para la integración o migración.	F	A	M	V	A	S	R20
R25	La interfaz de configuración debe ser fácil de utilizar	U	A	M	V	A	P	NO
R26	La integración de información de un origen de datos debe construirse de forma rápida e intuitiva.	U	A	M	V	A	P	NO
R27	El paso de los mensajes entre cliente y servidor no debe ser en menos de 5 segundos	P	A	M	V	A	P	NO

Descripción de la Metodología FURPS para clasificar requerimientos:

F Funcionalidad
U Usabilidad
R Confiabilidad
P Performance
S Soporte

5.6 Análisis

5.6.1 Casos de Uso

En esta sección se lista los casos de uso del integrador divididos por los diferentes módulos que lo conforman.

También los cuadros siguientes muestran la interacción entre un caso de uso y un actor del sistema, se muestra una "D", si es que la interacción es directa, y una "I" si es que la interacción es indirecta, es decir que el actor puede beneficiarse del caso de uso pero no es el que inicia la acción para que se ejecute el caso de uso.

5.6.1.1 Módulos de Integración/Exportación y Servidor de Integración

Los Módulos de Integración/Exportación y el Servidor de Integración módulo de interacción/exportación están conformados por 2 partes llamadas: "Integrador JAVNET parte .NET" e "Integrador JAVNET parte Java", fueron llamados así, dado a la tecnología en la que están basados.

Desde el punto de vista de interacción del sistema, estos 2 módulos tienen dos actores que son:

- Aplicación Cliente

  La Aplicación cliente es toda aplicación que se sirve de los datos migrados o exportados desde la aplicación servidor. La aplicación cliente, puede hacer uso de estos datos para desarrollar nuevos procesos y operaciones, poner a disposición la información en un Website, o en algún dispositivo. La aplicación cliente es la que ordena la ejecución en una operación de tratamiento de datos al servidor.

- Aplicación Servidor

  La aplicación servidor es la que pone a disposición los datos desde un origen de datos específico. Primeramente este servidor recibe una orden de pedido de ejecución desde la aplicación cliente, este pedido se realiza a través del integrador JAVNET parte .NET.

  Una vez recibido el mensaje, este es analizado y re direccionado hacia un origen de datos especifico y finalmente se devuelve los resultados.

Tabla 5.3 Casos de Uso VS Actores del Sistema

Nro.	CASO DE USO DEL SISTEMA	ACTORES DEL SISTEMA	
		Aplicación Cliente	Aplicación Servidor
1	Validar ingreso de Consulta de Aplicación Cliente hacia la Aplicación Servidor restringiendo el servicio solo a usuarios autorizados.	D	
2	Transformar la cadena de consulta al formato estándar XML, para ser transmitido al Servidor.	D	
3	Establecer Conexión entre Aplicación Cliente y Servidor a través de puertos TCP/IP.	D	D
4	Validar el Formato XML de la Consulta.		D
5	Grabar archivo Log en Aplicación Cliente.	D	
6	Valida resultado de Consulta a orígenes de datos.		D
7	Transformar resultado de consulta a formato final (XML).		D
8	Grabar archivo Log e Servidor.		D
9	Validar Esquema XML de Resultados para la Aplicación Cliente.	D	
10	Manejar Cola de Mensajes.	I	D
11	Manejo de Posibles Errores.	I	I

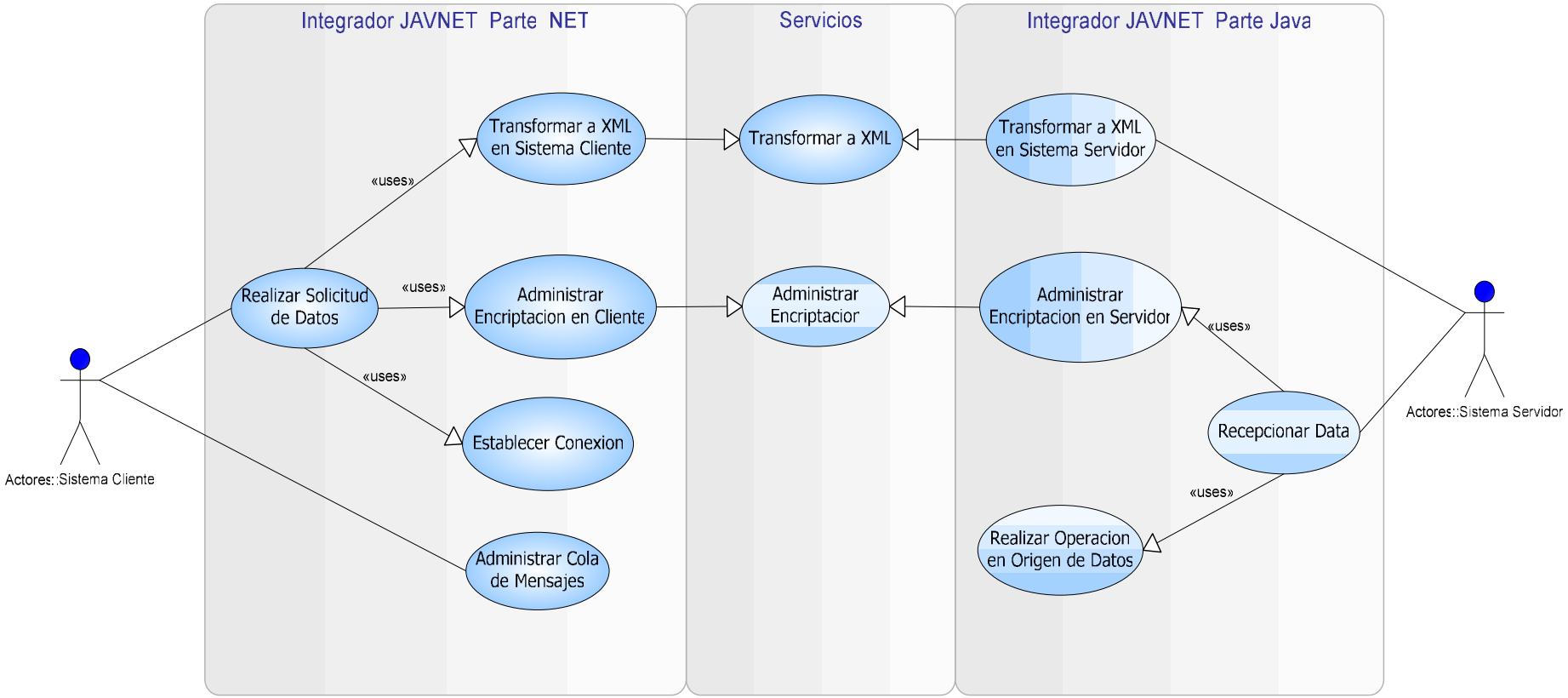
Descripción de la interacción de los Actores del Sistema

D Directo
I Indirecto

Tabla 5.4 Accesos a Casos de Uso VS Actores del Sistema

Nro.	CASO DE USO DEL SISTEMA	ACTORES DEL SISTEMA	
		Aplicación Cliente	Cliente Aplicación
1	Validar ingreso de Consulta de Aplicación Cliente hacia la Aplicación Servidor restringiendo el servicio solo a usuarios autorizados.	Con acceso	Sin acceso
2	Transformar la cadena de consulta al formato estándar XML, para ser transmitido al Servidor.	Con acceso	Sin acceso
3	Establecer Conexión entre Aplicación Cliente y Servidor a través de puertos TCP/IP.	Con acceso	Con acceso
4	Validar el Formato XML de la Consulta.	Sin acceso	Con acceso
5	Grabar archivo Log en Aplicación Cliente.	Con acceso	Sin acceso
6	Valida resultado de Consulta a orígenes de datos.	Sin acceso	Con acceso
7	Transformar resultado de consulta a formato final (XML).	Con acceso	Con acceso
8	Grabar archivo Log e Servidor.	Sin acceso	Con acceso
9	Validar Esquema XML de Resultados para la Aplicación Cliente.	Con acceso	Sin acceso
10	Manejar Cola de Mensajes.	Con acceso	Con acceso
11	Manejo de Posibles Errores.	Con acceso	Con acceso

Figura 5.1. Diagrama de Casos de Uso del Sistema de módulo de integración/exportación



5.6.1.2 Módulo de Configuración

El módulo de configuración llamado "Smart JavNet Configuration" permite a un usuario configurar aspectos técnicos para poner en marcha el integrador de información, ya que este debe poder adaptarse a una situación de negocio que tienen necesidad de integrar su información, con esto se gana generalidad, adaptabilidad y escalabilidad, ya que el integrador no está solo enfocado a una situación concreta sino diferentes escenarios de negocio.

Desde el punto de vista de interacción del sistema, el módulo de configuración tiene un único actor:

- Usuario Smart JavNet

  El actor de aplicación es una persona que se tiene la responsabilidad de configurar en integrador de información de acuerdo a situación que tenga que resolver, esto incluye las entidades manejadas en el negocio, el acceso a datos, la forma de comunicación y la seguridad de acceso de los usuarios.

Tabla 5.5 Casos de Uso VS Actores del Sistema

Nro.	CASO DE USO DEL SISTEMA	ACTORES DEL SISTEMA
		Usuario Smart JavNet
1	Configurar Integración	D
2	Configurar Entidades	D
3	Configurar Acceso a Datos	D
4	Generar Clases de XML	I
5	Configurar Comunicación	D
6	Registrar Usuarios	D
7	Consultar Entidad Generada	D
8	Consultar DataAccess Generado	D

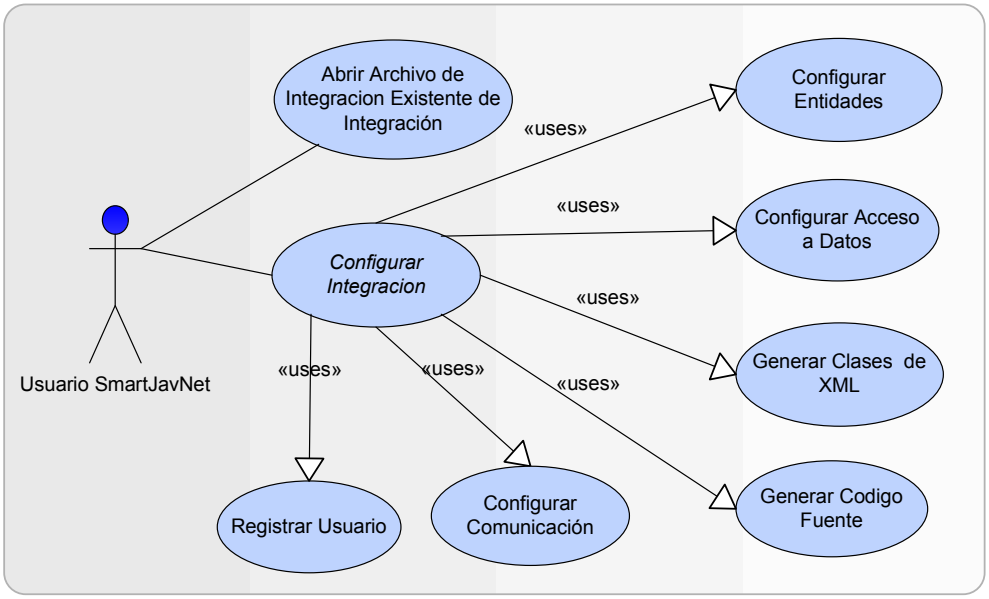
Descripción de la interacción de los Actores del Sistema

D Directo
I Indirecto

Tabla 5.6 Accesos a Casos de Uso VS Actores del Sistema

Nro.	CASO DE USO DEL SISTEMA	ACTORES DEL SISTEMA
		Usuario Smart JavNet
1	Configurar Integración	Con acceso
2	Configurar Entidades	Con acceso
3	Configurar Acceso a Datos	Con acceso
4	Generar Clases de XML	Con acceso
5	Configurar Comunicación	Con acceso
6	Registrar Usuarios	Con acceso
7	Consultar Entidad Generada	Con acceso
8	Consultar DataAccess Generado	Con acceso

Figura 5.2. Diagrama de Casos de Uso del Sistema de módulo de Configuración



5.6.2 Organización por Módulos

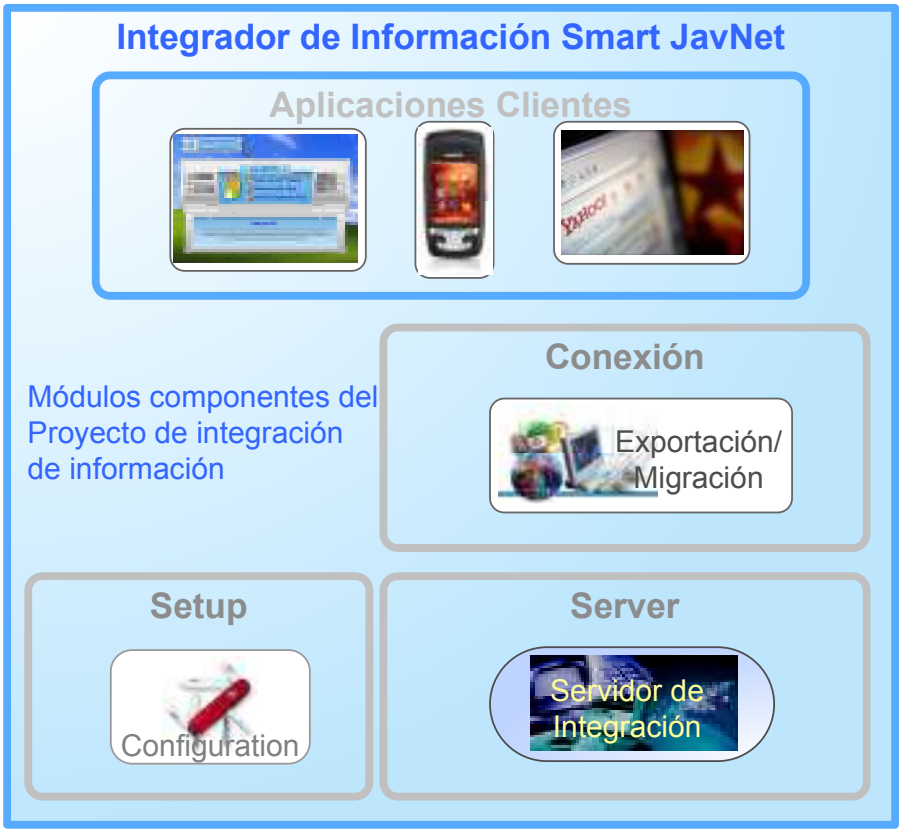
En el diagrama siguiente se muestra la organización por módulos del integrador de información, en ello se puede apreciar los módulos de:

- Configuración
- Servidor de Integración
- Exportación/Migración

Los 2 últimos módulos son los que deben ser puestos en producción para ejecutarse todo el tiempo, el módulo de configuración se ejecuta la primera vez para configurar el integrador de información.

Por otro parte están las aplicaciones clientes que pueden ser de diversos tipos (desktop, web, móvil, etc.), que se sirven del integrador para poder aprovechar la información.

Figura 5.3. Diagrama del integrador de información organizado por Módulos



5.7 Diseño

5.7.1 Arquitectura General de la Aplicación

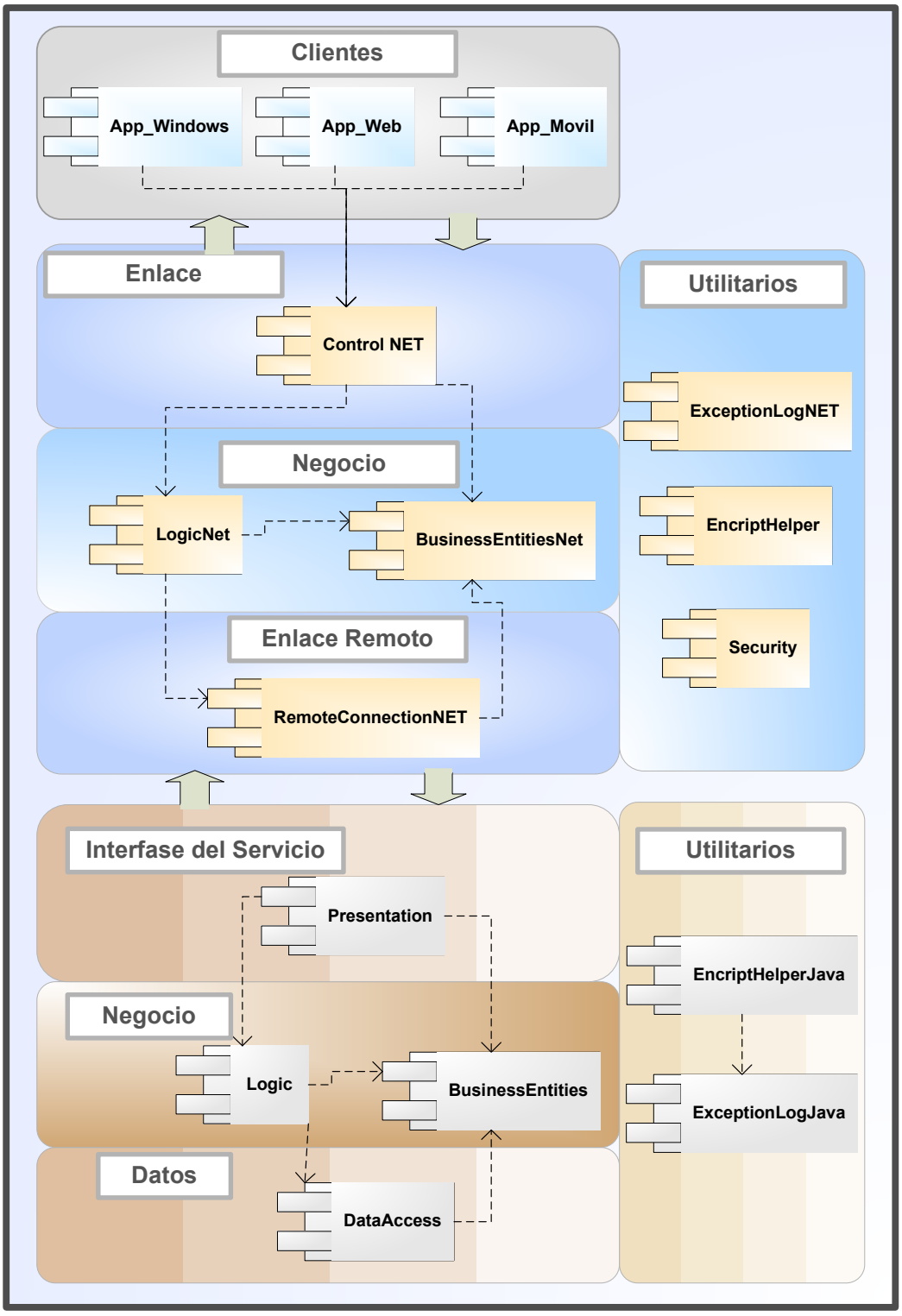
La arquitectura general de la aplicación muestra los diferentes componentes que forman parte de los diferentes módulos de la aplicación en su totalidad. La organización de la aplicación está separada por capas, es así que se divide en una capa de clientes, en la que existen tipos de aplicaciones que pueden ser desktop, web, móvil, etc. La siguiente capa es la de Enlace, que sirve como la única puerta de enlace entre en integrador de información y las aplicaciones clientes. Una vez ingresada en la capa existe otra capa intermedia que es la capa de Negocio, en ella se conserva cierta lógica del integrador de información, para poder hacer validaciones y gestión de configuración. La tercera capa es la de Enlace Remoto, que brinda toda la funcionalidad de comunicación con el servidor de integración, definida ya en la configuración previa.

Existen 2 capas más de Servicios que brindan servicios de encriptación y seguridad, y otra capa de configuración, que brinda toda una funcionalidad para poder configurar el integrador de información por primera vez y sea adaptado un escenario en concreto. Hasta aquí todas las capas vistas están basadas en tecnología .NET.

La siguiente etapa del integrador de información muestra capas de: Interface de Servicio que es la única puerta de enlace de recepción de las peticiones de operaciones enviadas desde los clientes, ya sean estas a través de servicios Web o sockets, existe también una capa media de lógica, que al igual que su predecesora del lado Exportación/integración, cumple con funcionalidad de validación y gestión de integración, luego tenemos a la tercera capa y tal vez la más importante para el integrador de información, es la capa de Datos, esta capa toma la data de los diversos orígenes de información configurados previamente, da formato, y realiza las operaciones directamente, enviando un resultado de la misma hacia la aplicación cliente. En esta parte también del Server de integración se tiene capas, una capa de Servicios que sirve para Encriptar y Desencriptar la información enviada o recibida. Todas estas capas de esta parte del integrador Server de integración están basadas bajo la tecnología J2EE.

Así se dio una vista rápida y en forma general de las capas de la arquitectura general del servidor de integración, esto está descrito en la figura siguiente, más adelante se podrá leer en mayor detalle la funcionalidad de cada componente de cada capa.

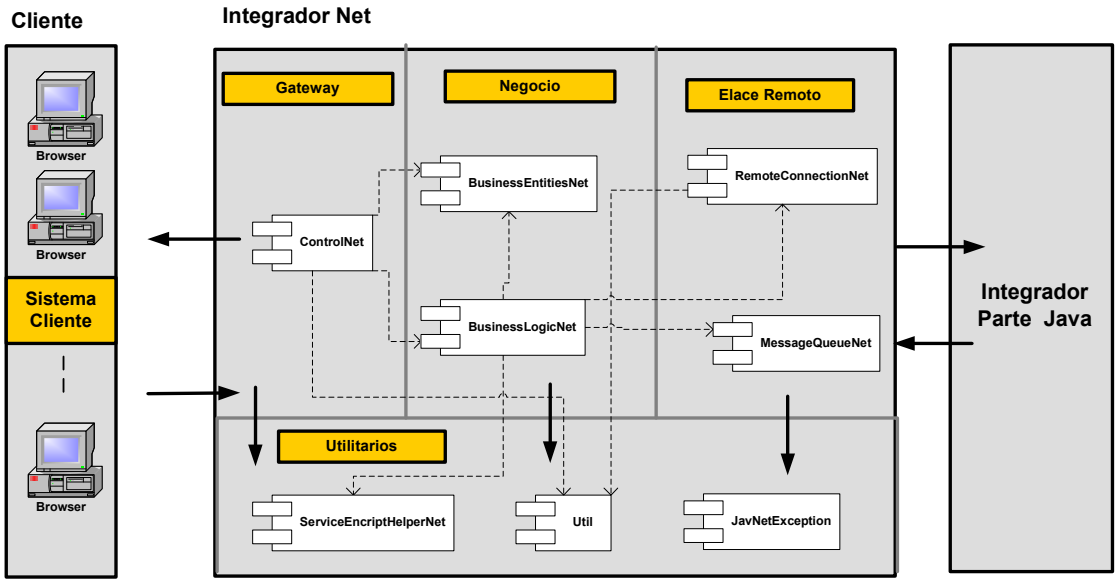
Figura 5.4. Arquitectura general y completa del integrador de información



5.7.2 Módulos de la Solución

5.7.2.1 Conector Cliente de Integración de Datos (.NET)

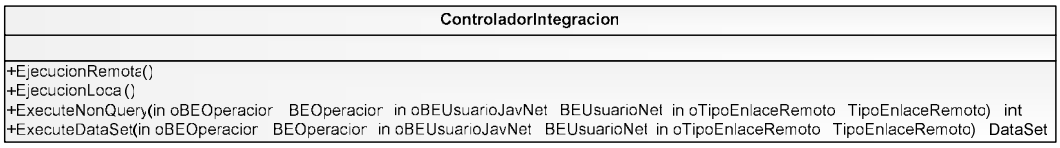
Figura 5.5 Componentes del Integrador en la parte Cliente .Net



5.7.2.1.1 Componentes

- **ControlNet**
  El cual tiene la tarea de controlar si una operación va a ser realizada remotamente o localmente, ya que el sistema cliente podría operar con una base de datos local además de la base de datos u origen remoto, esto facilitaría la migración y flexibilidad del software.

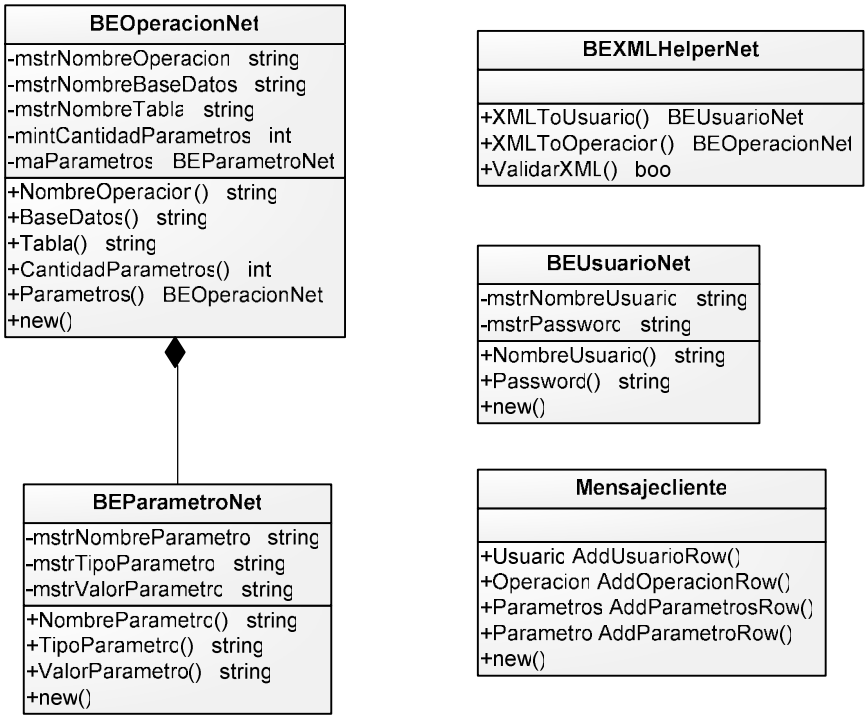
Figura 5.6 Diagrama de clases del componente Control.NET



- **BusinessEntitiesNET**
  Este componente está encargado de guardar la información acerca de la operación que se va a realizar en el origen datos remotos, es muy general de tal forma que sea cual fuera la operación ésta la soportará para

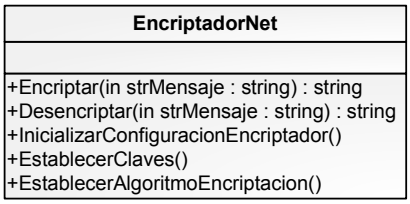
poder enviarla como XML al servidor y así realizar la operación.
Este componente BussinesEntitiesNet está basado en esquemas XML bajo el formato UTF 8, estos esquemas son representados por estructuras de datos como son los Dataset con tipo (Typed Datasets), así esto puede generar los mensajes automáticamente; pero antes de generar el mensaje XML se le debe dar un cierto formato para que pueda ser correctamente recorrido en la plataforma receptora.

Figura 5.7 Diagrama de clases del componente Business EntitiesNet



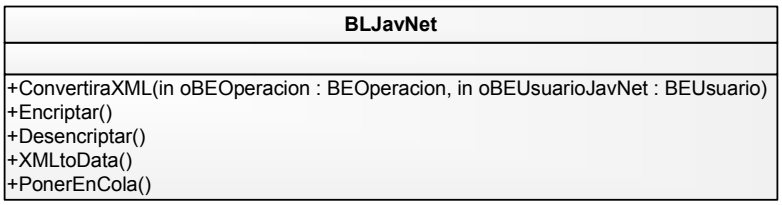
- **ServiceEcriptHelperNet**
  Componente encargado de encriptar y desencriptar la información pasada entre las múltiples plataformas, ya que manejan la misma llave pública, y están basados en el mismo algoritmo de encriptación.

Figura 5.8 Diagrama de clases del componente ServiceEncriptHelperNet



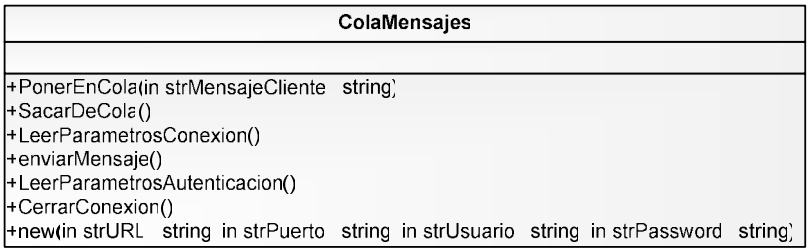
- **BusinessLogicNet**
  Este componente sirve de componente medio que orquesta las llamadas al servidor, aísla a la aplicación cliente del componente de conexión remota, brindando simplicidad y bajo acoplamiento entre las capas.

Figura 5.9 Diagrama de clases del componente BusinessLogicNet



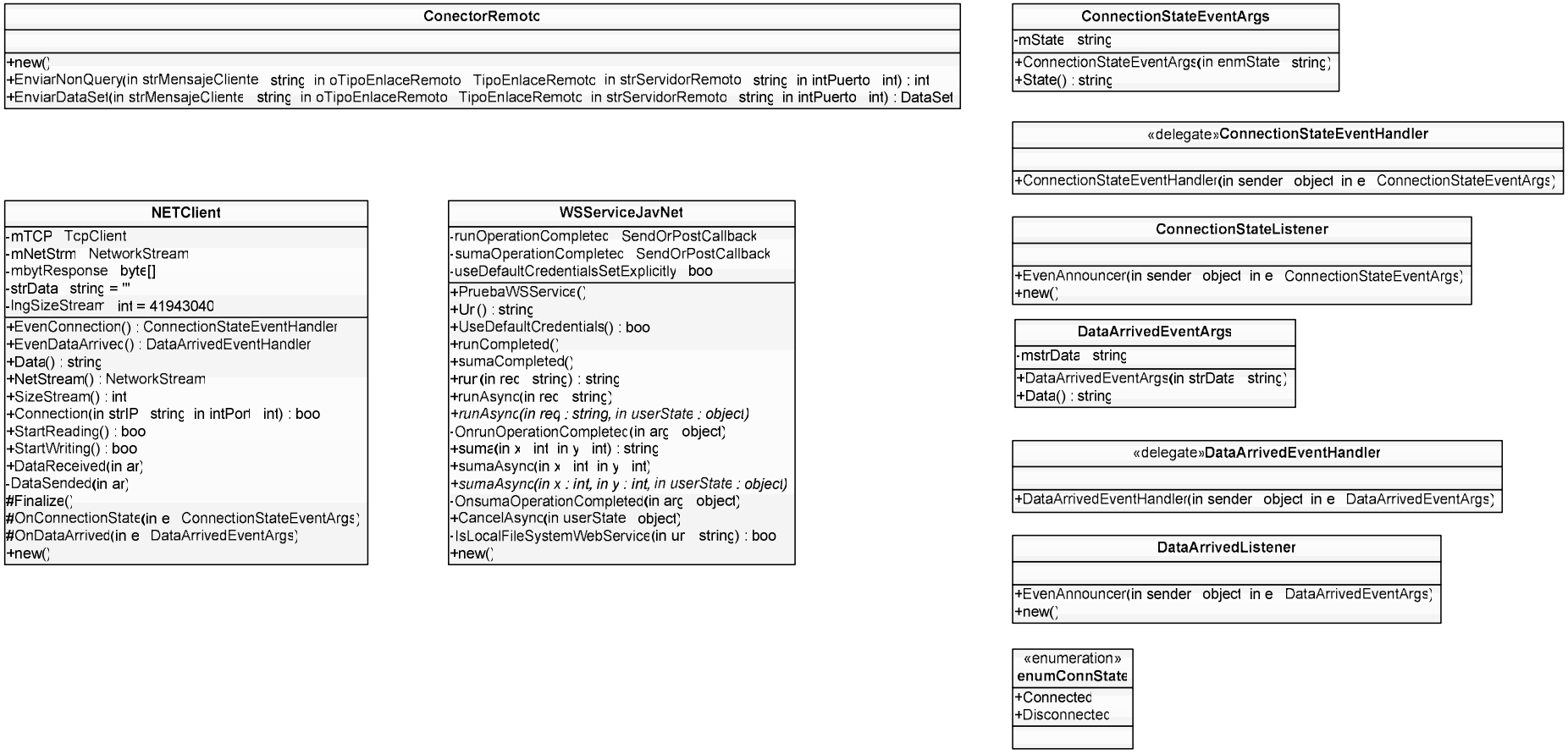
- **MessageQueueNet**
  Este componente está basado en OpenJMS, y recepciona los mensajes de manera que sean enviados solo uno a la vez y así poder prevenir el colapso del servidor por múltiple concurrencia en ambientes de alta transferencia, así la aplicación es escalable y podría soportar múltiples usuarios conectados. Este componente no ha sido implementado, pero se coloca en la arquitectura para que pueda ser analizado en futuras investigaciones.

Figura 5.10 Diagrama de clases del componente MessageQueueNet



- **RemoteConnection**
  Este componente recoge el mensaje de la operación a realizar en formato XML, y la envía al Servidor de integración, en este componente aquí se manejan la configuración de puertos de comunicación cuando la se podría realizar a través de Sockets o Web services o incluso Colas.

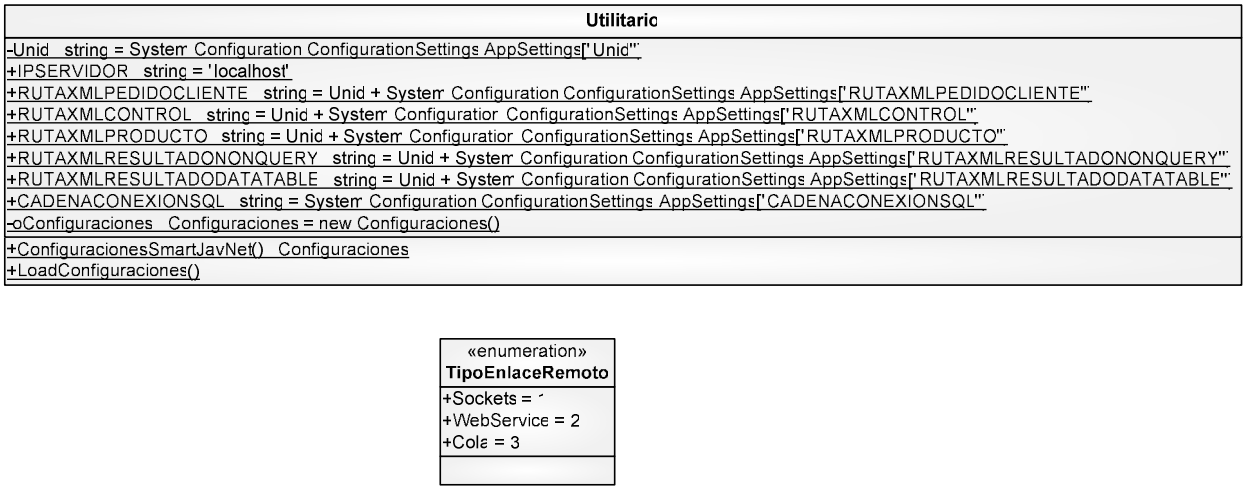
Figura 5.11 Diagrama de clases del componente RemoteConnection



- **Util**
  Este componente contiene a las clases Utilitario y un enumerador TipoEnlaceRemoto, el primero guarda estáticamente en memoria las rutas de archivos de configuración y la ruta de la cadena de conexión a base de datos local.

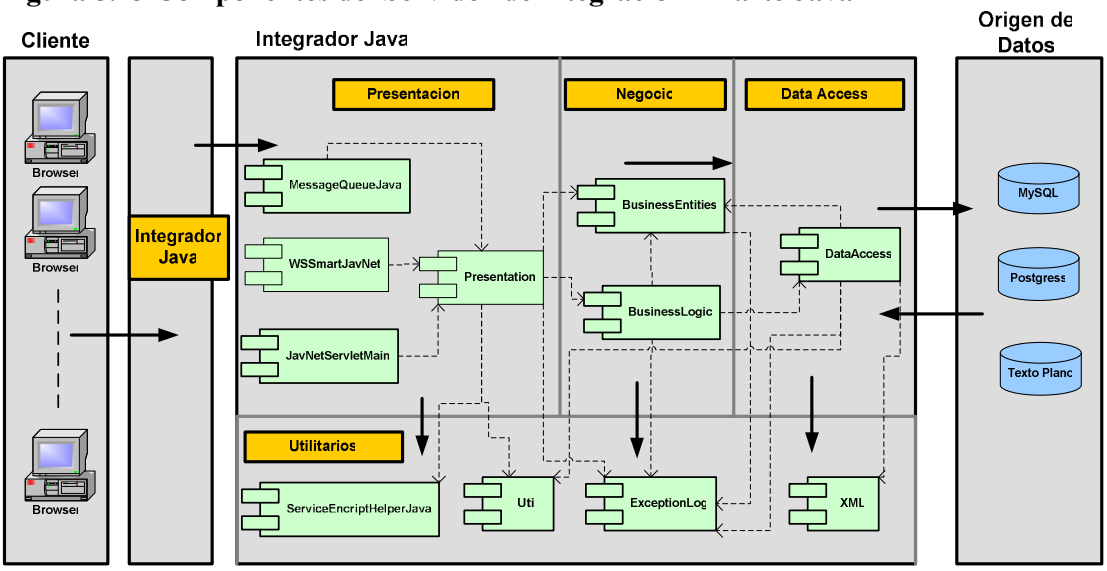
  De esta forma se puede tener en memoria estos valores que pueden ser utilizados por los distintos componentes y clases durante la ejecución de la aplicación.

Figura 5.12 Diagrama de clases del componente Util



5.7.2.2 Servidor de Integración de Datos

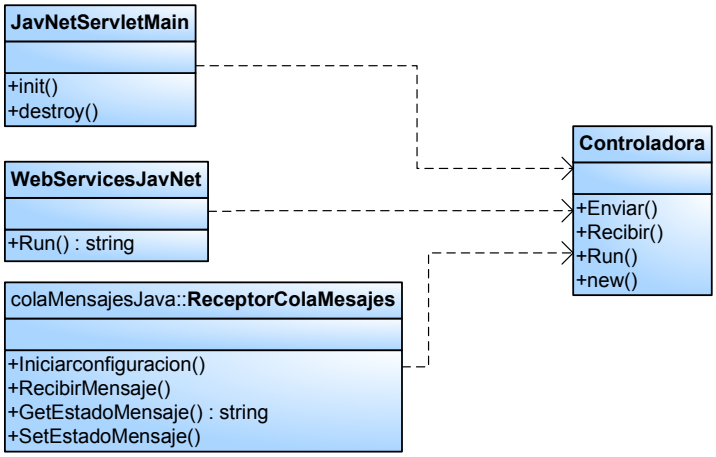
Figura 5.13 Componentes del Servidor de integración – Parte Java



5.7.2.2.1 Componentes

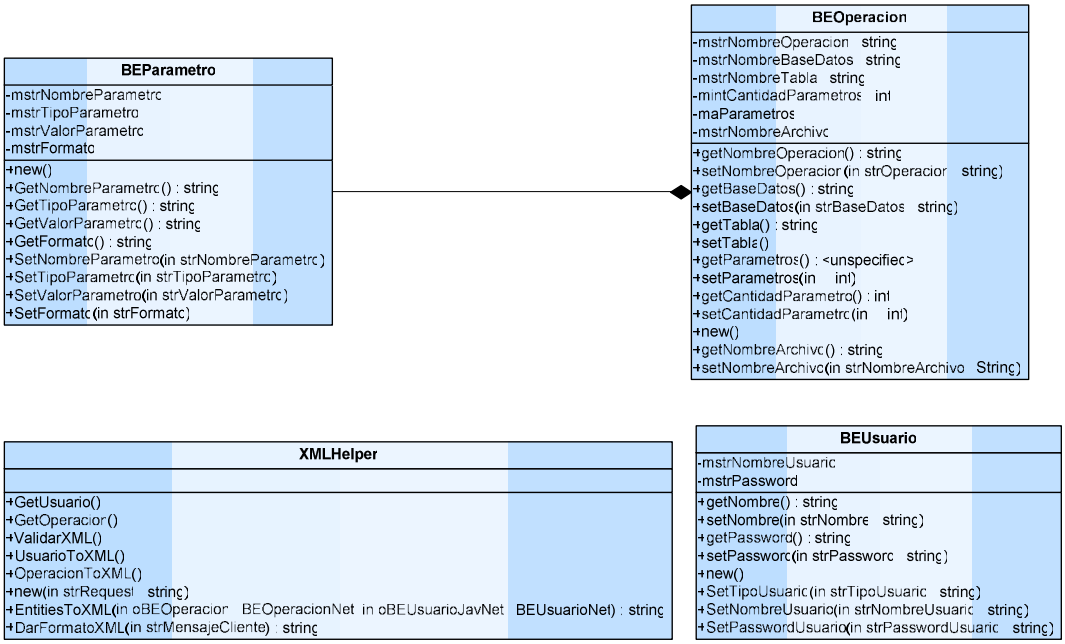
- **Presentation**
  Tiene la tarea de recibir y comunicarse con el cliente, este componente lee la información en XML, la recorre jerárquicamente, y luego crea entidades de negocio (Business Entities) a partir de ella.

Figura 5.14 Diagrama de clases del componente Presentation



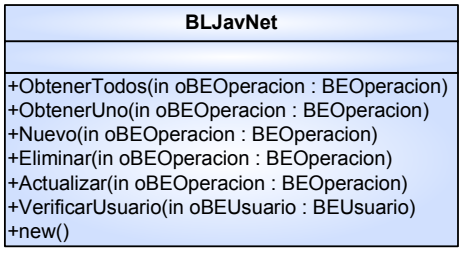
- **BusinessEntities**
  Las clases de este componente son las encargadas de almacenar en ellas las operaciones que se ejecutarán en los orígenes de orígenes de datos.

Figura 5.15 Diagrama de clases del componente BusinessEntities



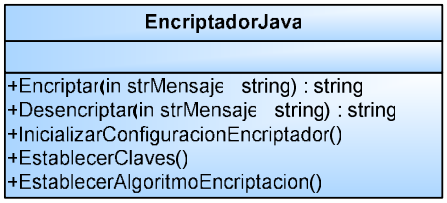
- **BusinessLogic**
  Este componente contiene clases de validación y además orquesta las operaciones que serán enviadas a ejecutarse contra los orígenes de datos, dado que en esta parte se ejecuta la orden de en cuál de los orígenes de datos se llevara a cabo las consultas y operaciones, esto puede ser dinámicamente ordenado desde el servidor.

Figura 5.16 Diagrama de clases del componente BusinessLogic



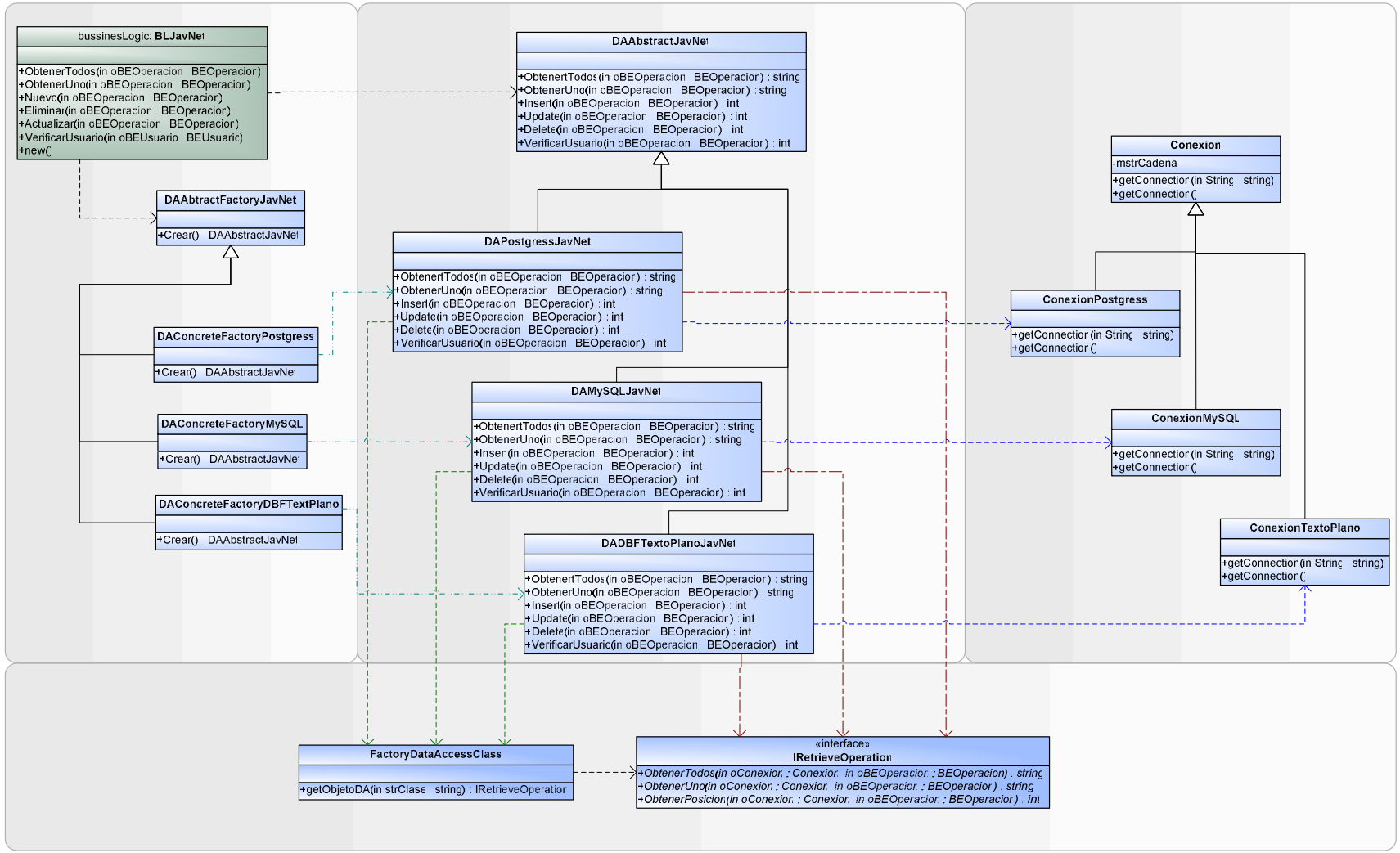
- **ServiceEcriptHelperJava**
  Componente encargado de encriptar y desencriptar la información pasada entre las múltiples plataformas.

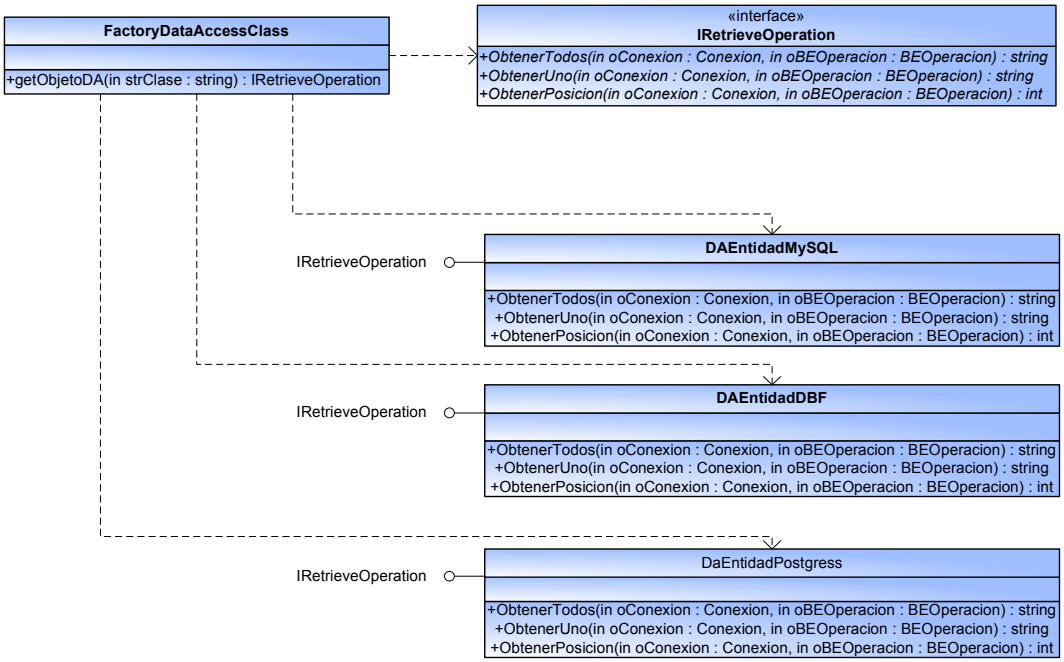
Figura 5.17 Diagrama de clases del componente ServiceEcriptHelperJava



- **DataAccess**
  Este componente contiene las clases para acceder a los diferentes orígenes de datos, aquí entra a tallar la utilización de una Fabrica Abstracta de Objetos (Patrón Abstract Factory), que nos ayuda a organizar las clases por especialización de operaciones hacia un determinado origen de datos, y si se desea cambiar por otro, solo bastaría con realizar un ligero cambio en la aplicación, y no necesariamente acondicionar toda la aplicación.

Figura 5.18 Diagrama de clases del componente DataAccess





El diagrama anterior es la continuación del Diagrama de clases del componente DataAccess, que muestra, como es que se relacionan las clases de las entidades de un origen de datos Específico, con la interface IRetrieveOperation.

Este origen de datos específico podría ser una clase MySQLDAProducto, PostgreSQLDAProducto o DBFDAPersona por dar unos ejemplo, lo que se presenta aquí es que cada de una de estas clases especializan el tratamiento de los métodos de la interface IRetrieveOperation. Pero estas a su vez están encapsuladas de tal forma que las clases padres, solo hagan referencia a ellas al momento de la ejecución a través de la característica de Reflection, con esto si se desea agregar más clases de orígenes de datos, no significa un cambio en la aplicación integradora.

El código de la operación de getObjetoDA de la clase FactoryDataAccessClass es el siguiente:

```
public static  IRetrieveOperation getObjetoDA(String strClase)
{  // Clase Class, para instanciar una clase por su nombre
   Class clazz = Class.forName(strClase);
   // Clase Constructor, para una llamada a un constructor genérico
   Constructor ct  = clazz.getConstructor(null);
   // Clase Object, para Obtener una instancia del constructor
   Object retobj = ct.newInstance(null);
   // Se Realiza casting del objeto a IRetrieveOperation
   IRetrieveOperation ObjetoDA = (IRetrieveOperation) retobj;

   return ObjetoDA;
}
```

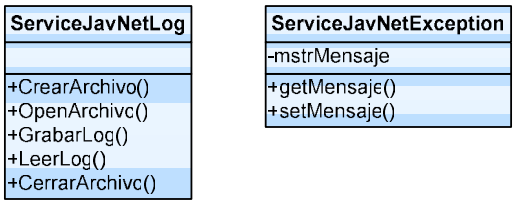
Y el código que ejecuta al método anterior en las clases: DADBFTextoPlanoJavNet, DAMySQLJavNet y DAPostgreSQLJavNet es el que se muestra a continuación, en este caso se tomo el de la clase DADBFTextoPlanoJavNet, pero es similar al de las otras 2 clases, solo difiere en la cadena que da parte nombre de la clase, vista en la primera línea de código del método, resaltada de otro color, estas líneas serían:

- javNet.dataAccess.textoPlanoDBF.DBFDA
- javNet.dataAccess.postgreSQL.PostgreSQLDA
- javNet.dataAccess.mySQL.MySQLDA

```
public String ObtenerTodos(BEOperacion oBEOperacion)
{
  IRetrieveOperation oDA =
  FactoryDataAccessClass.getObjetoDA("javNet.dataAccess.textoPl
  anoDBF.DBFDA" + oBEOperacion.GetNombreTabla());

  return  oDA.ObtenerTodos(oConexion,oBEOperacion);

}

public String ObtenerUno(BEOperacion oBEOperacion)
{

  IRetrieveOperation oDA =
  FactoryDataAccessClass.getObjetoDA("javNet.dataAccess.textoPl
  anoDBF.DBFDA" + oBEOperacion.GetNombreTabla());

  return  oDA.ObtenerUno(oConexion,oBEOperacion);

}
```

- **ExceptionLog**
  Maneja todas las excepciones y errores que ocurren en el sistema, además de guardar los eventos en archivos históricos (Logs) de ocurrencias.

Figura 5.19 Diagrama de clases del componente ExceptionLog



- **MessageQueueJava**
  Este componente contiene a la clase ReceptorColamensajes, que estaría basado en la tecnología JMS (Java Message Service) con se encargaría de recibir los pedidos puestos en cola por la aplicación cliente, esta clase seria manejada por el servidor de aplicaciones. Se podría usar OPEN JMS para poder implementarlo, esta clase dependería de los demás componentes para poder resolver las respuestas de operaciones contra los orígenes de datos.

Figura 5.20 Diagrama de clases del componente MessageQueueJava

5.7.2.3 Realizaciones de los Módulos de Integración/Exportación y Servidor de Integración

La siguiente sección describe las realizaciones de cada caso de uso modelado de los módulos de Integración/Exportación y Servidor de Integración, y también se muestran los diagramas de secuencia correspondientes, de tal forma que se pueda apreciar el orden lógico de los procesos.

5.7.2.3.1 Realización del Caso de Uso Realizar Solicitud de Datos

1. El caso de uso comienza cuando el sistema cliente realiza un pedido de ejecución por medio de la capa su Acceso a Datos éste a su vez ejecuta el proceso ExecuteNonQuery si es que se trata de una inserción, actualización o borrado, o ExecuteDataSet, si es que se trata de una consulta de datos.

2. La clase ControlIntegracion instancia un objeto de la clase BEUsuarioNet, un objeto de BEOperacionNet y por último un arreglo de objetos de BEParametroNet.

3. Luego ejecuta el método ConvertiraXML de la clase BLJavNet pasándole como parámetros el objeto de BEOperacion que contiene el arreglo de Objetos de BEParametroNet, y el objeto BEUsuarioNet.

4. El Método ConvertiraXML de la Clase BLJavNet ejecuta el método EntitiesToXML de la clase XMLHelperNet, que lo que

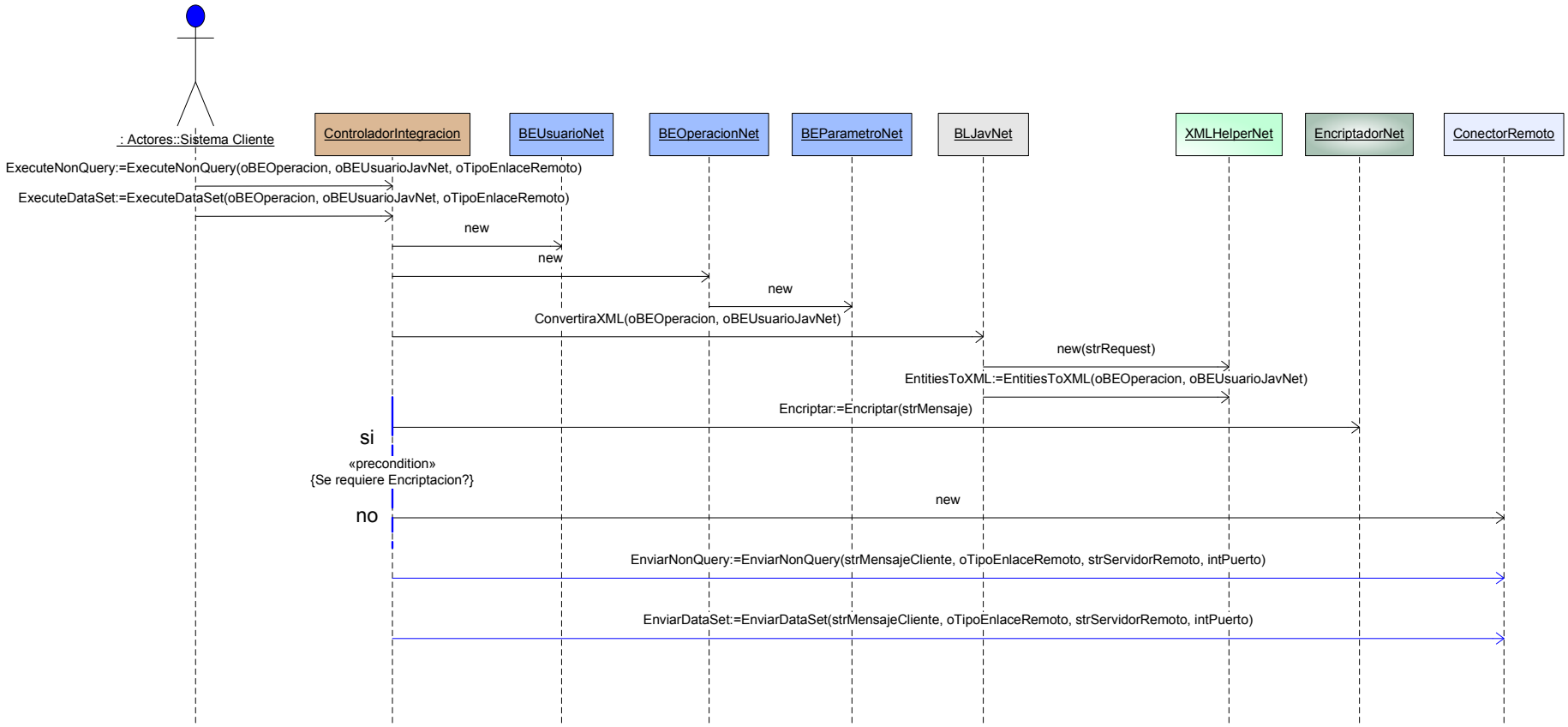
hace es generar XML a partir de los objetos BEOperacion y BEUsuarioNet.

5. El Mensaje generado es retornado a la clase ControladorIntegracion, si leyendo el archivo de configuración, indica que el mensaje debe ser encriptado, entonces se ejecuta el método Encriptar de la clase EncriptadorNet, sino el mensaje pasa en XML legible.

6. Luego la clase ControladorIntegracion instancia a la clase ConectorRemoto ejecuta los métodos EnviarNonQuery o EnviarDataSet según sea el caso.

7. En ambos casos anteriores se les pasa como parámetros el MensajeCliente, el Tipo de Enlace Remoto (Sockets, Web Services, Colas), y el puerto en el que está activo el servidor de integración, y hasta esta parte finaliza el caso de uso.

Figura 5.21 Diagrama de Secuencia del Caso de Uso Realizar solicitud de Datos



5.7.2.3.2 Realización del Caso de Uso Establecer Conexión

1. El caso de uso comienza cuando se ejecuta los métodos ExecuteNonQuery o ExecuteDataSet.

2. Si se trata de una conexión por Sockets entonces se instancia la clase NetClient, se le asigna a su propiedad Data el MensajeCliente como texto, y también a su método Connection se le pasa como parámetros la dirección IP del servidor y el puerto en el cual está atendiendo pedidos.

3. Luego la clase ConectorRemoto instancia las clases ConnectionStateListener y DataArrivedListener.

4. Las instancias de las clases instanciadas anteriormente tendrán la tarea de disparar los eventos cuando se produzcan:

   a. EvenConnection
   b. EvenDataArrived
   c. StartReading

5. Una vez que se tenga respuesta del lado del servidor este empezara a recibir la información por la propiedad Data de la clase NetClient.

6. Recibida toda la información, el resultado de la operación es devuelto hacia la clase ControladorIntegracion.

7. Si se trata de una conexión por Web Services entonces se instancia la clase WebService, y se le asigna el URL donde se encuentra el servidor de integración.

8. Luego se ejecuta la operación RUN, que devuelve un texto, este resultado es devuelto hacia la clase ControladorIntegracion.

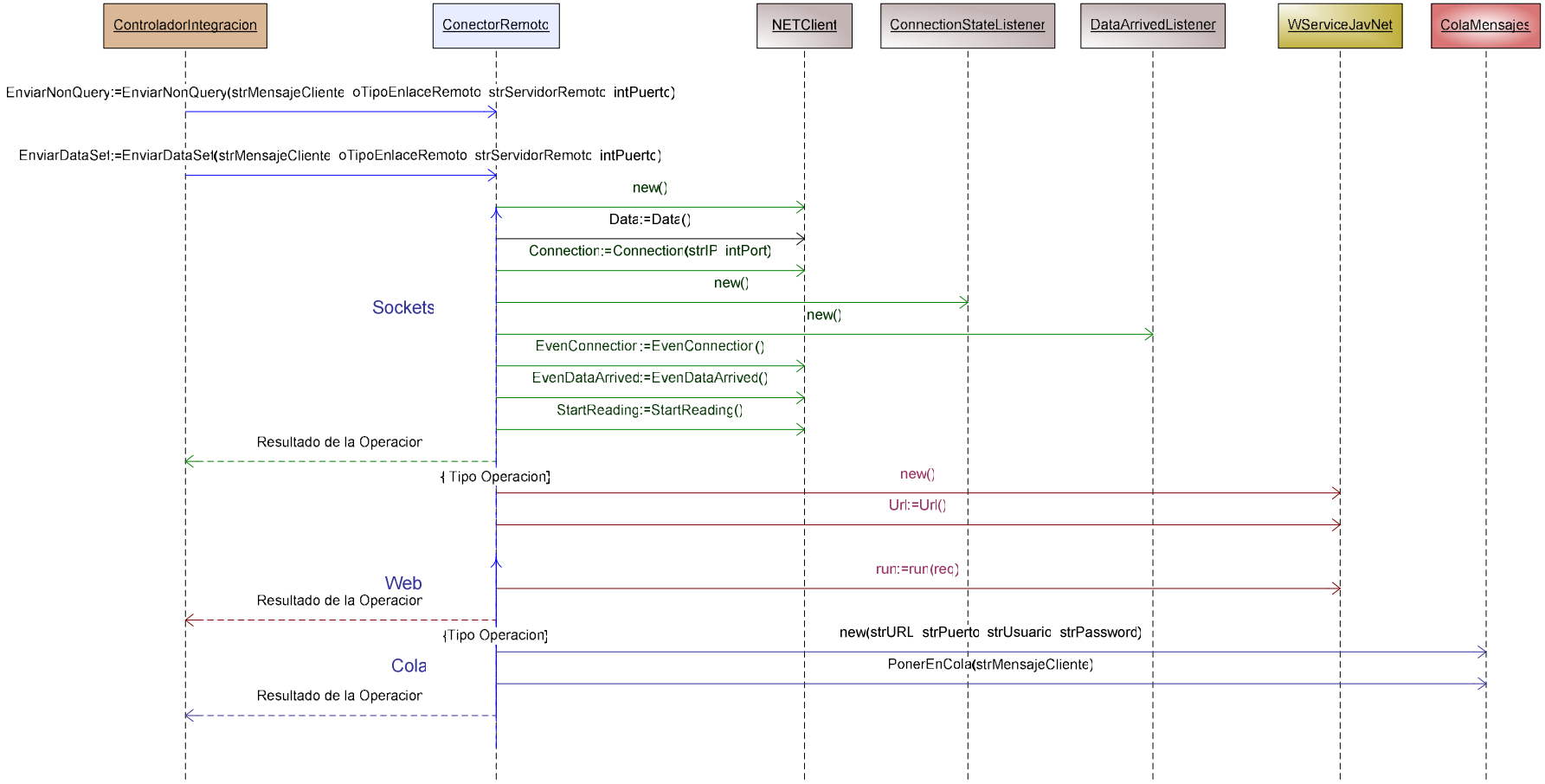
9. Si se trata de una conexión por Colas entonces se instancia la clase ColaMensajes, con los parámetros:

   a. URL del Servidor
   b. Nombre del Usuario
   c. Password

10. Luego se ejecuta el método PonerEnCola de la clase ColaMensajes, y se le pasa como parámetro el Mensaje Cliente.

11. Finalmente se envía el resultado de la operación hacia la clase ControladorIntegracion.

Figura 5.22 Diagrama de Secuencia del Caso de Uso Establecer Conexión



5.7.2.3.3 Realización del Caso de Uso Recepcionar Data

1. El caso de uso comienza cuando el servidor recibe una petición de ejecución en el método Run de la clase controladora, sea cual fuere el tipo de conexión.

2. Una vez recibido el mensaje, si se indica mediante archivo de configuración que necesita ser desencriptado entonces se llama al método Desencriptar de la clase EncriptadorJava, y se le pasa como parámetro el mensaje recibido, para que sea desencriptado.

3. Luego la clase Controladora instancia un objeto de la clase XMLHelper, y luego se ejecuta sus métodos GetUsuario y GetOperacion, para así obtener a partir del mensaje XML recibido los objetos BEUsuario y BEOperación que a vez instancia un arreglo del BEParametro.

4. Después de haber obtenido los objetos BEUsuario y BEOperacion entonces la clase controladora instancia un objeto de la clase BLJavNet dentro de este método se instancia por la característica de "Reflection" que fabrica de Acceso a datos se construirá en tiempo de Ejecución, esta puede ser para MySQL, PossgreeSQL o DBF.

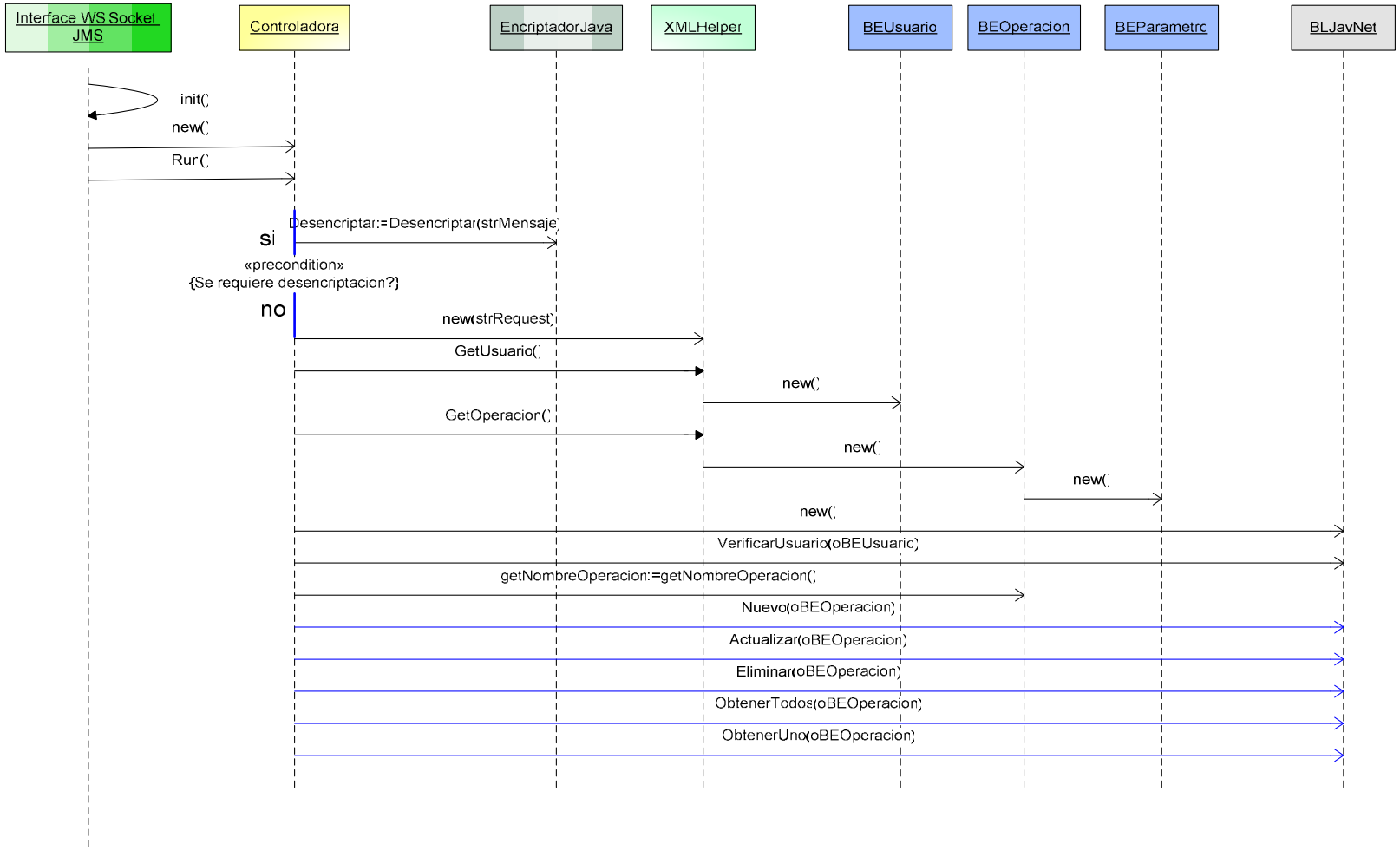
5. Se ejecuta el método ValidarUsuario y se le pasa como parámetro el objeto de BEUsuario.

6. Si es un usuario valido entonces se ejecuta el método GetNombreOperacion, para que según sea el nombre se ejecute la operación indicada que puede ser:

   a. Nuevo
   b. Actualizar
   c. Eliminar
   d. ObtenerTodos
   e. ObtenerUno

7. Se ejecuta la operación indicada y el resultado es devuelto a la clase Controladora.

Figura 5.23 Diagrama de Secuencia del Caso de Uso Recepcionar Data



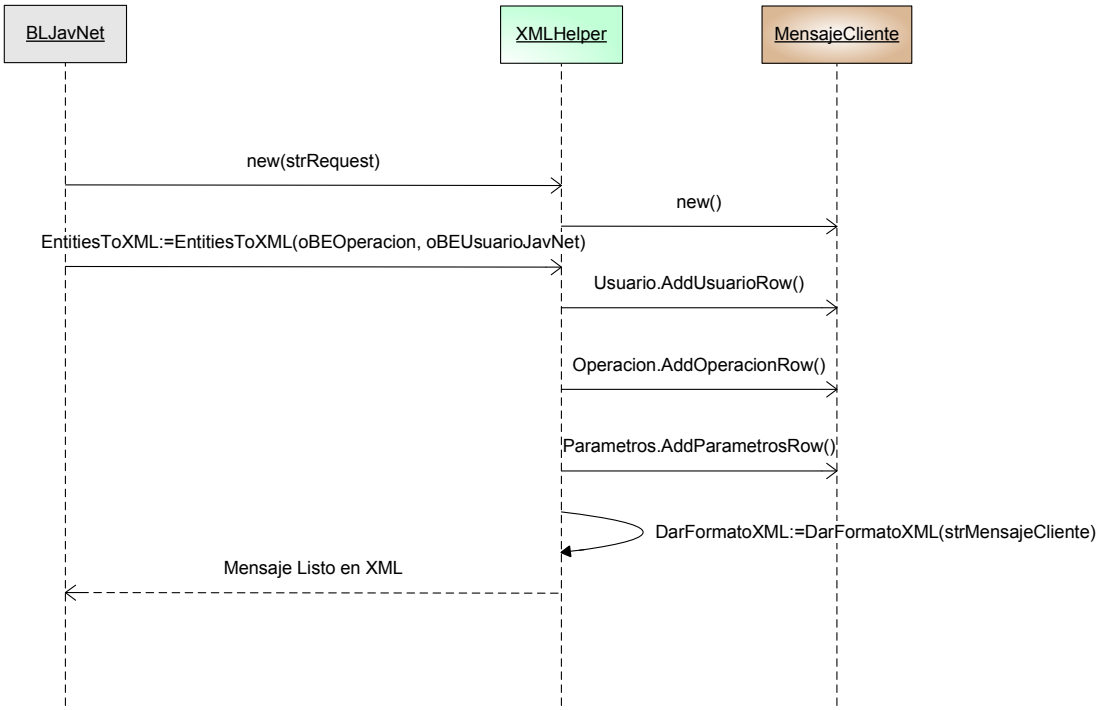
5.7.2.3.4 Realización del Caso de Uso Transformar a XML en Sistema Cliente

1. El caso de uso comienza cuando la clase BLJavNet instancia la clase XMLHelper y este a su vez instancia un DataSet con tipo MensajeCliente.

2. Luego la clase BLJavNet ejecuta el método EntitiesToXML, pasándole como parámetro los objetos de BEOperacion y BEUsuarioJavNet.

3. Dentro del Método EntitiesToXML se agrega una filas a la tablas Usuario, Operación y Parámetros (este último por cada uno de los parámetros que existan), luego se llama al método DarFormatoXML que dejara listo al mensaje XML generado, para después regresar el mensaje listo para enviárselo al servidor de integración.

Figura 5.24 Diagrama de Secuencia del Caso de Uso Transformar XML en Sistema Cliente



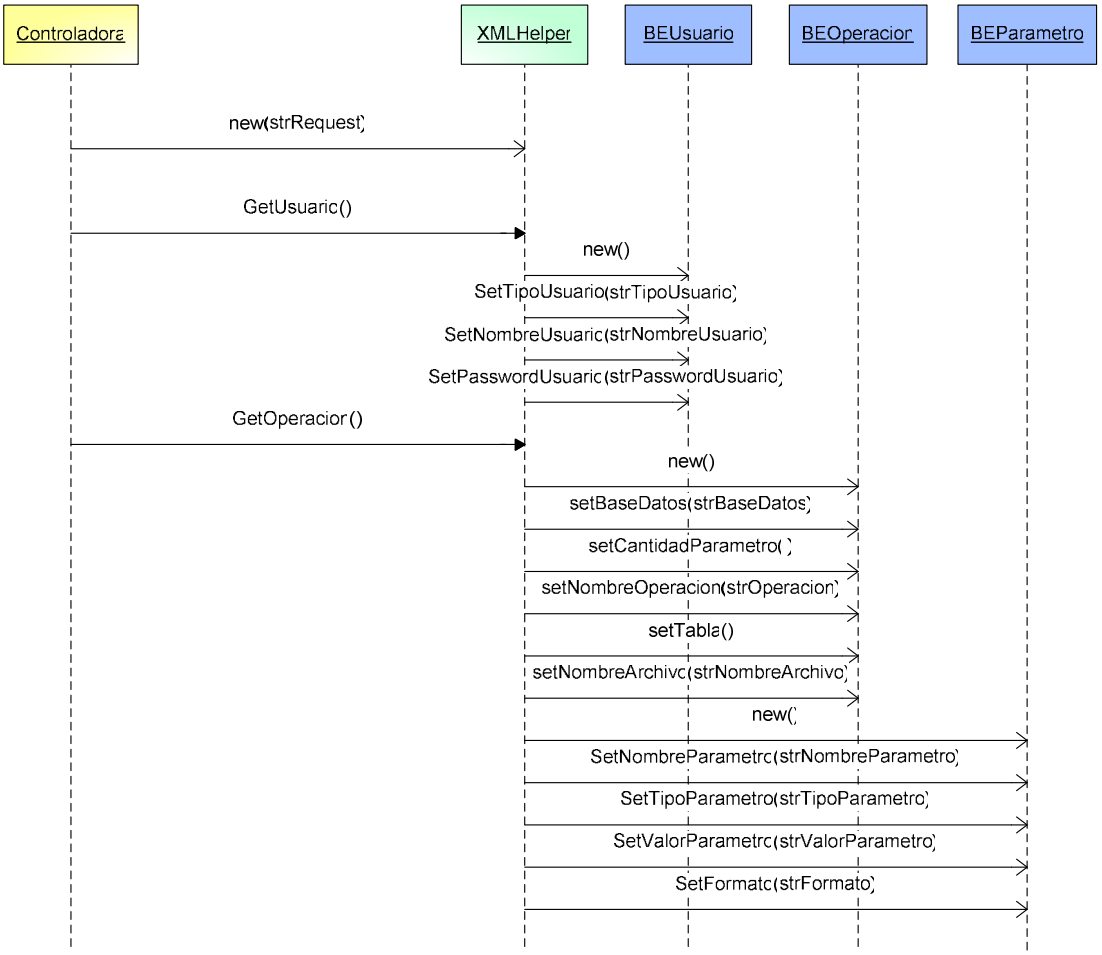
5.7.2.3.5 Realización del Caso de Uso Transformar a XML en Sistema Servidor

1. El caso de uso comienza cuando la clase Controladora instancia la clase XMLHelper y se le pasa como parámetro del constructor el MensajeCliente en XML.

2. Luego la clase Controladora ejecuta el método GetUsuario, que por medio de Dom4J, extrae el Tipo del Usuario, Nombre, y Password y se los asigna por propiedad al objeto BEUsuario, esto devuelve este objeto a la controladora.

3. Por último la clase Controladora ejecuta el método GetOperacion, que también por medio de Dom4J, extrae el Nombre de la Base de Datos, Cantidad de Parámetros, Nombre de la Operación, Nombre de la Tabla y Nombre del Archivo y se los asigna por propiedad al objeto BEOperacion. Luego por cada parámetro también se instancia la clase BEParametro, y se le asigna por propiedad el Nombre del Parámetro, el Tipo, Valor, y el Formato. Y son retornados dentro del objeto Operación a la clase Controladora.

Figura 5.25 Diagrama de Secuencia del Caso de Uso Transformar XML en Sistema Servidor



5.7.2.3.6 Realización del Caso de Uso Administrar Cola de Mensajes

1. El caso de uso comienza cuando la clase ConectorRemoto instancia la clase ColaMensajes, y se le pasa como parámetro del constructor el URL del servidor, el Número de Puerto, Usuario y Password.

2. Luego se ejecuta el método PonerEnCola con el parámetro srMensajeCliente, luego se ejecuta los métodos LeerParametrosConexion, LeerParametrosAutenticacion.

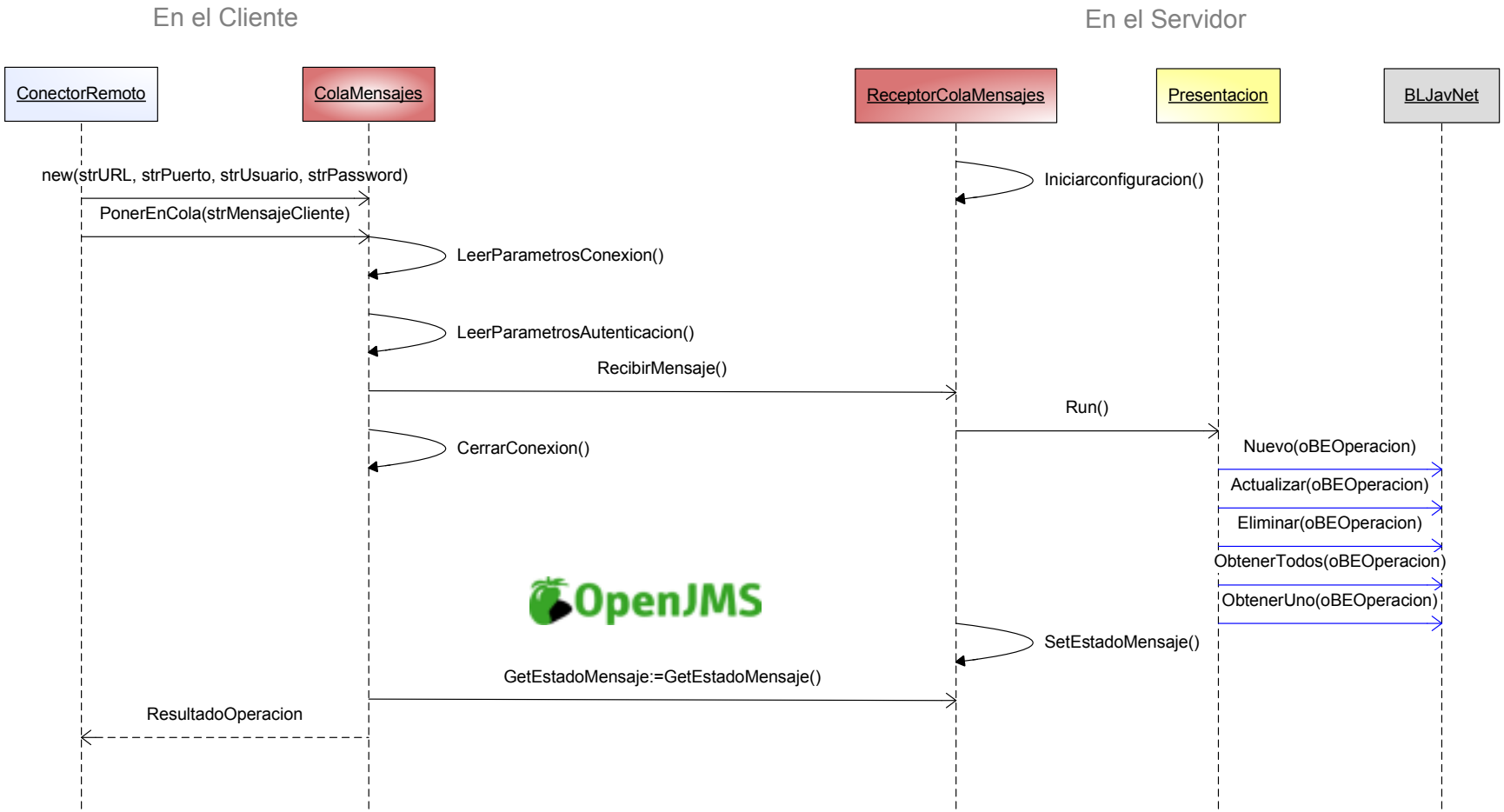
3. Una vez que se tiene configurada la cola con sus parámetros se ejecuta remotamente EjecutarMensaje, que es un método que a través de tecnologías como Grasshopper.

4. Luego se cierra la conexión.

5. Asíncronamente, desde la clase ReceptorColaMensajes, basada en Open JMS se ejecuta el método Run que realiza la ejecución de las operaciones Nuevo, Actualizar, Eliminar, ObtenerTodos y ObtenerUno.

6. Luego se ejecuta el método SetEstadoMensaje, cuando esto pasa un evento es lanzado en el cliente, que hace que se ejecute el método GetEstadoMensaje, y este trae el Resultado de la Operación, que es devuelta a la clase ConectorRemoto.

Figura 5.26 Diagrama de Secuencia del Caso de Uso Administrar Cola de Mensajes



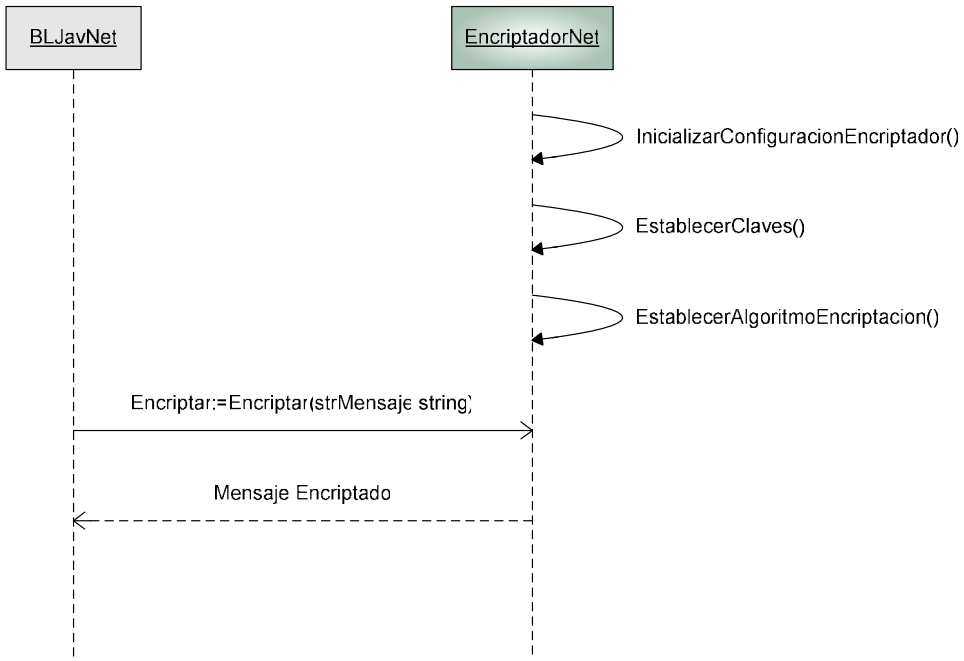
5.7.2.3.7 Realización del Caso de Uso Administrar Encriptación en Cliente

1. El caso de uso comienza cuando la clase BLJavNet ejecuta el método Encriptar de la clase EncriptadorNet.

2. Previamente la clase EncriptadorNet, debe haber ejecutado conjuntamente cuando se inicie el sistema los métodos InicializarConfiguracionesencriptador, EstablecerClaves, y EstablecerAlgoritmoEncriptacion.

3. El Mensaje encriptado es devuelto a la Clase BLJavNet.

Figura 5.27 Diagrama de Secuencia del Caso de Uso Administrar Encriptación en el Cliente



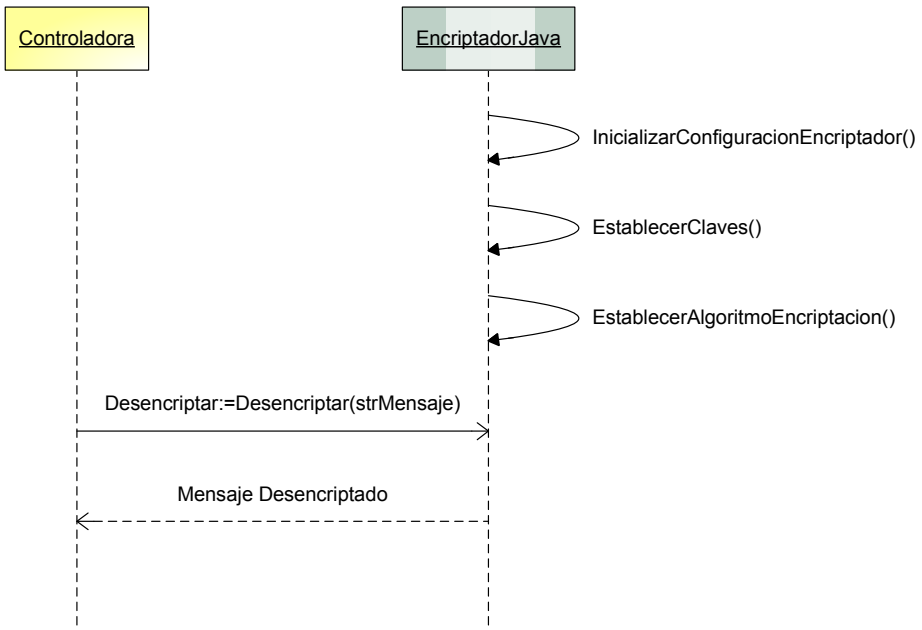
5.7.2.3.8 Realización del Caso de Uso Administrar Encriptación en el Servidor

1. El caso de uso comienza cuando la clase Controladora ejecuta el método Desencriptar de la clase EncriptadorJava.

2. Previamente la clase EncriptadorJava, debe haber ejecutado conjuntamente cuando se inicie el servidor los métodos InicializarConfiguracionesencriptador, EstablecerClaves, y EstablecerAlgoritmoEncriptacion.

3. El Mensaje desencriptado es devuelto a la Clase Controladora.

Figura 5.28 Diagrama de Secuencia del Caso de Uso Administrar Encriptación en el Servidor



5.7.2.3.9 Realización del Caso de Uso Realizar Operación en Base en Datos

1. El caso de uso comienza cuando la clase BLJavNet ejecuta el método CrearJavNetData de la clase DAAbstractFactoryJavNet, pero asignado una objeto de la ConcretFactory determinado que podría ser:
    a. DAConcrectFactoryTextoPlano
    b. DAConcrectFactoryMySQL
    c. DAConcrectFactoryPossgreSQL

2. Entonces esto devuelve un objeto que puede ser de tipo:
    a. DADBFTextoPlanoJavNet
    b. DAMySQLJavNet
    c. DAPostgreSQLJavNet

3. Y según esto por medio de la característica "Reflection", se instancia el objeto con el cual se desea hacer las operaciones

```
DAAbtractFactoryJavNet oFactory;

Class clazz = Class.forName(oBEOperacion.GetNombreBaseDatos());

Constructor ct  = clazz.getConstructor(null);

Object retobj = ct.newInstance(null);

oFactory = (DAAbtractFactoryJavNet)retobj;

oAccesoDatos = oFactory.Crear();
```

4. Una vez obtenido el objeto entonces ya se puede hacer las operaciones que se pidieron:

```
oAccesoDatos.ObtenerTodos(oBEOperacion);

oAccesoDatos.ObtenerUno(oBEOperacion);

oAccesoDatos.Insert(oBEOperacion);

oAccesoDatos.UpDate(oBEOperacion);

oAccesoDatos.Delete(oBEOperacion);
```

5. En el caso de las operaciones ObtenerTodos y ObtenerUno, se tiene que construir dinámicamente la clase que personaliza dichas operaciones como por ejemplo, si se desea traer todos los Productos, entonces se tendría que construir por medio de la Interface IRetrieveOperation para generalizar las llanadas desde un DA Genérico y la clase FactoryDataAccessClass que devuelva el objeto específico construido, en el ejemplo

DAProducto, todo utilizando la característica de "Reflection" esto mediante el código:

```
//Construyendo clase
Class clazz = Class.forName(strClase);

Constructor ct  = clazz.getConstructor(null);

Object retobj = ct.newInstance(null);

IRetrieveOperation ObjetoDA = (IRetrieveOperation) retobj;

return ObjetoDA;
```

Figura 5.29 Diagrama de Secuencia del Caso de Uso Realizar Operación en Origen de Datos (DBF)

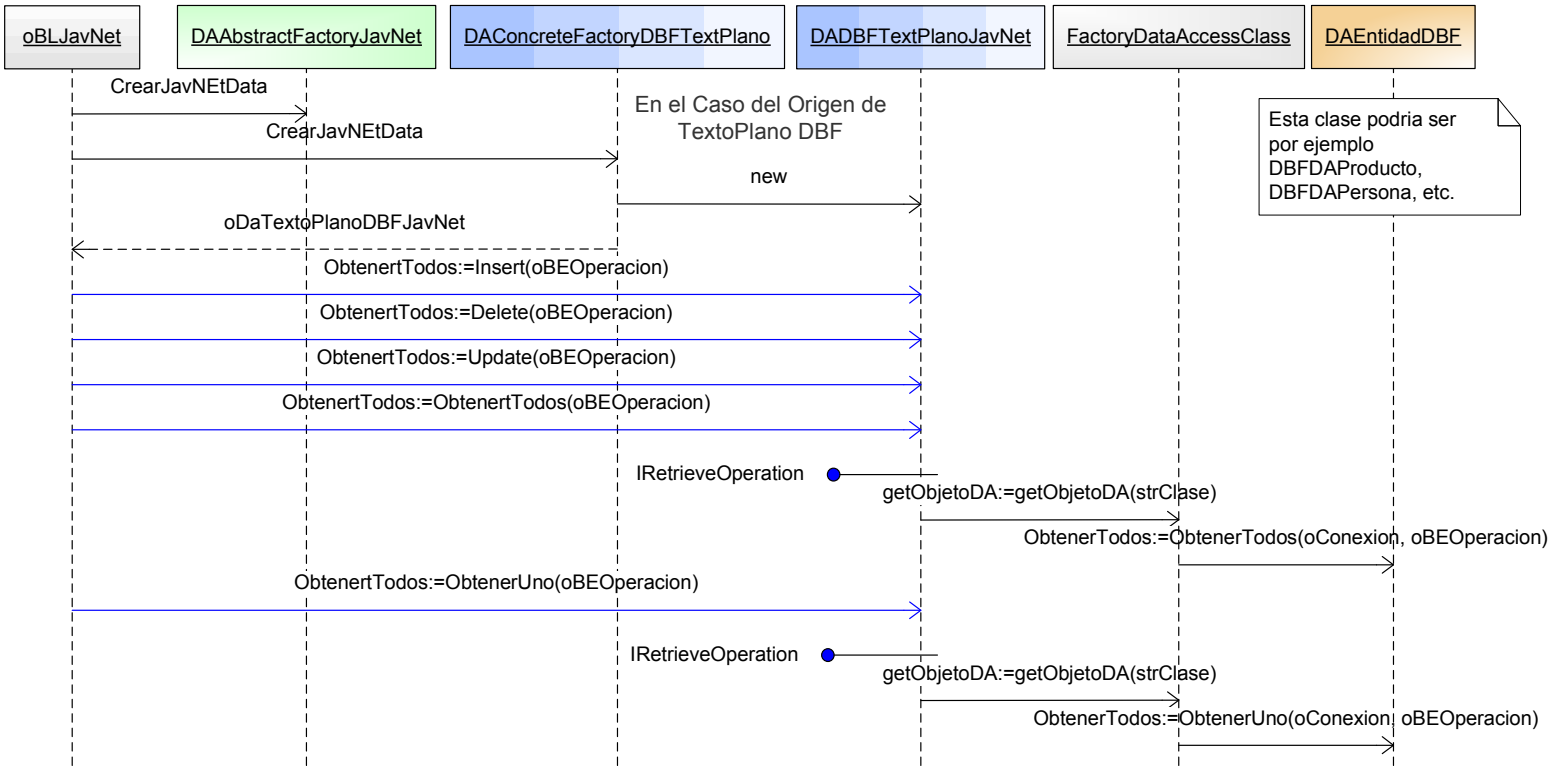


Figura 5.30 Diagrama de Secuencia del Caso de Uso Realizar Operación en Origen de Datos (MySQL)

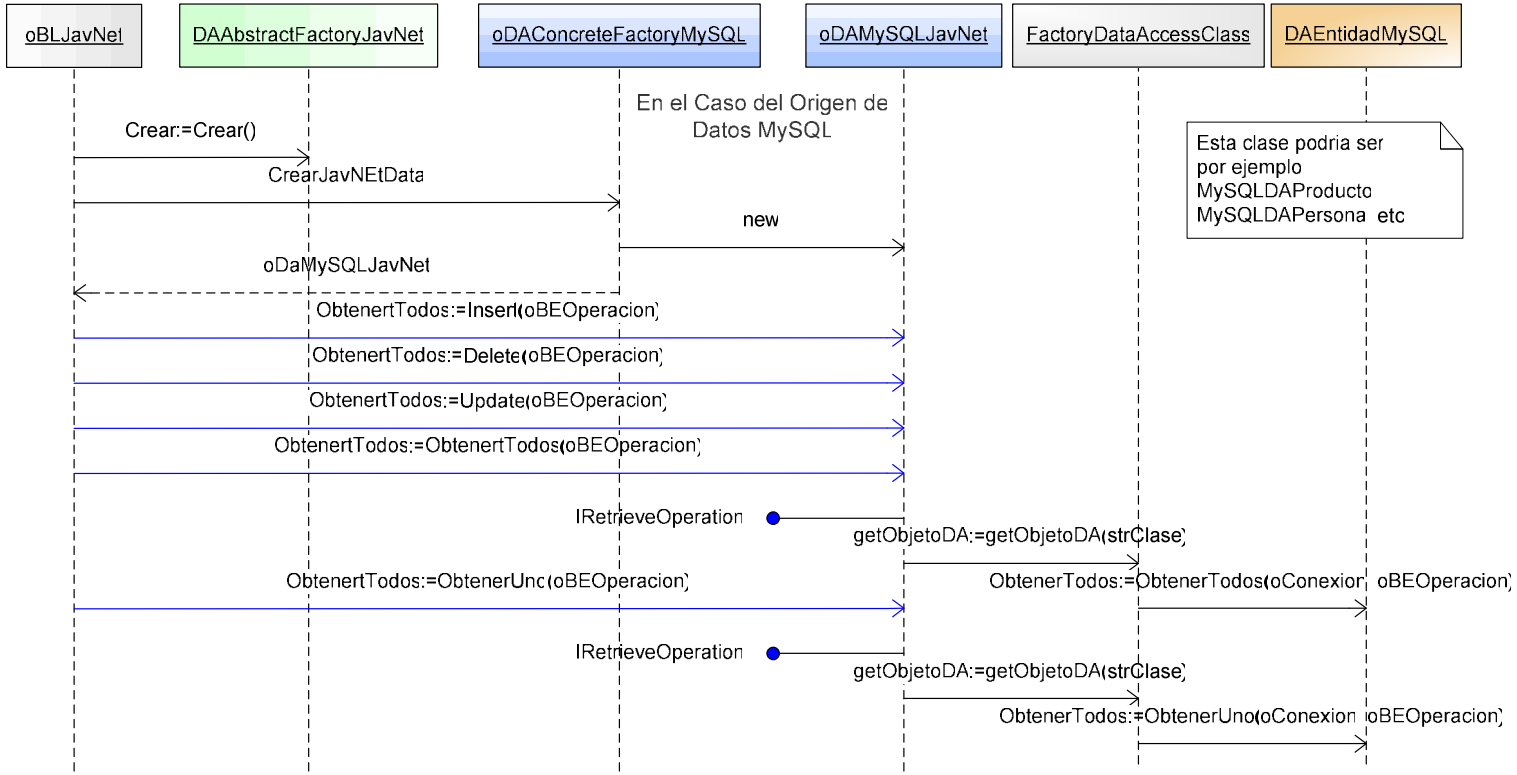
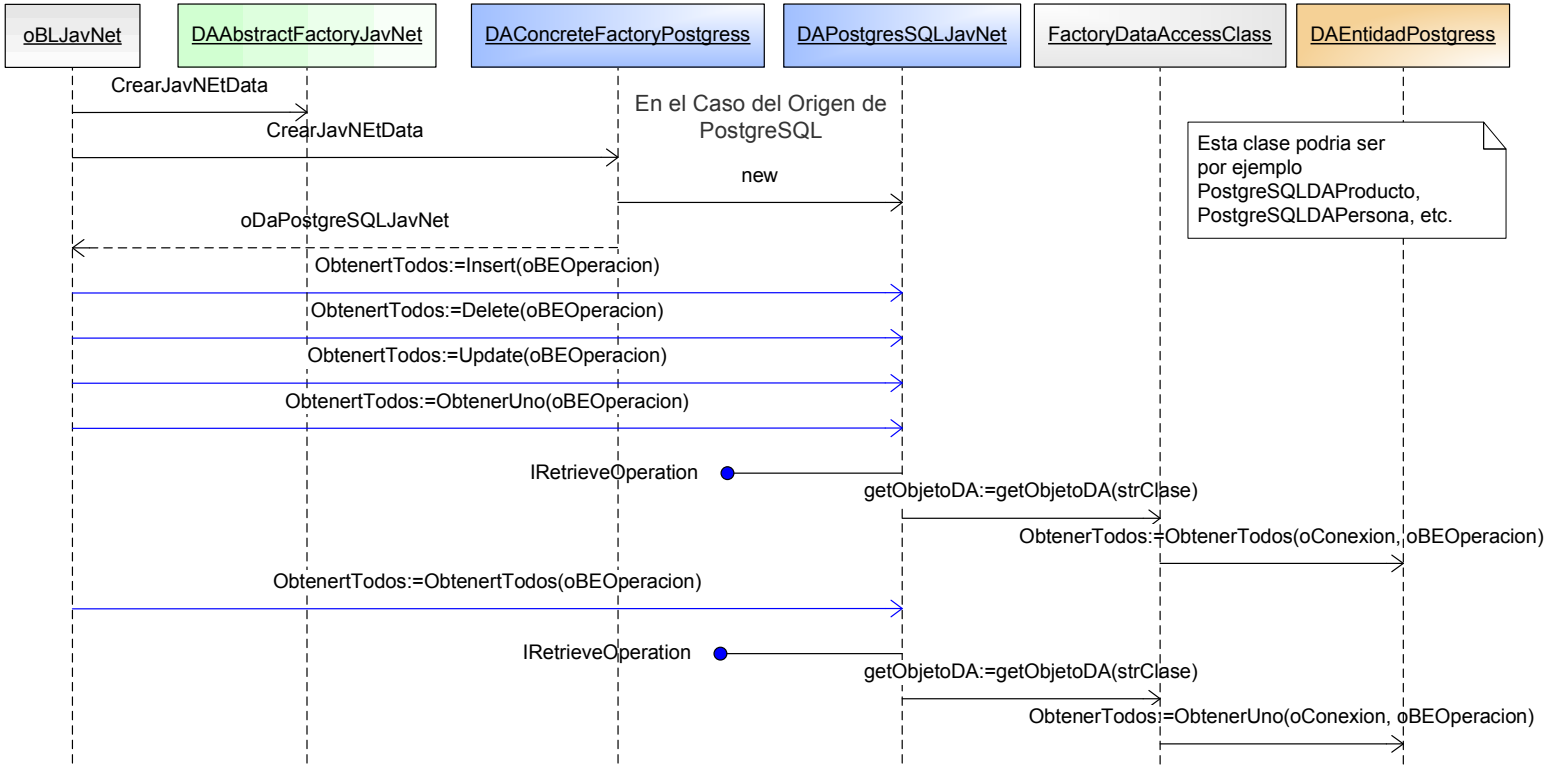


Figura 5.31 Diagrama de Secuencia del Caso de Uso Realizar Operación en Origen de Datos (PostgreSQL)

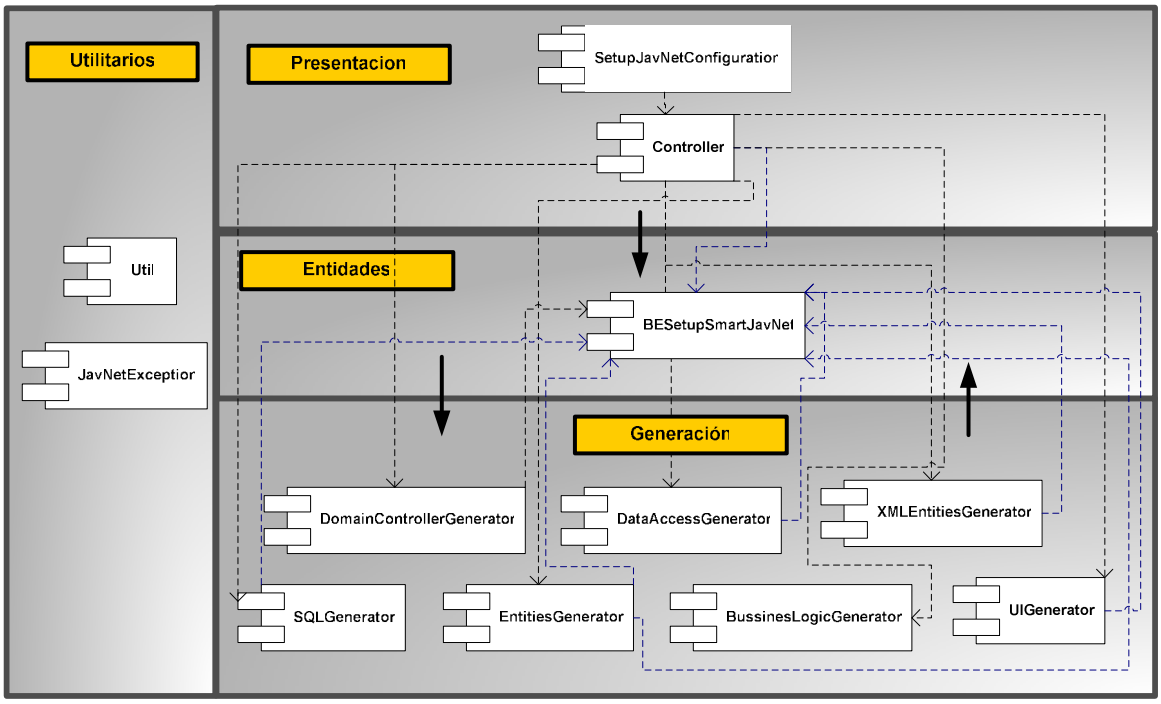


5.7.2.4 Configurador del Integrador de Información

El configurador de integración es una aplicación cuyo objetivo es el de definir los datos y las clases y adaptarlas a las necesidades de un dominio especifico de negocio.

A continuación se puede apreciar cómo están organizados los componentes del configurador de integración:

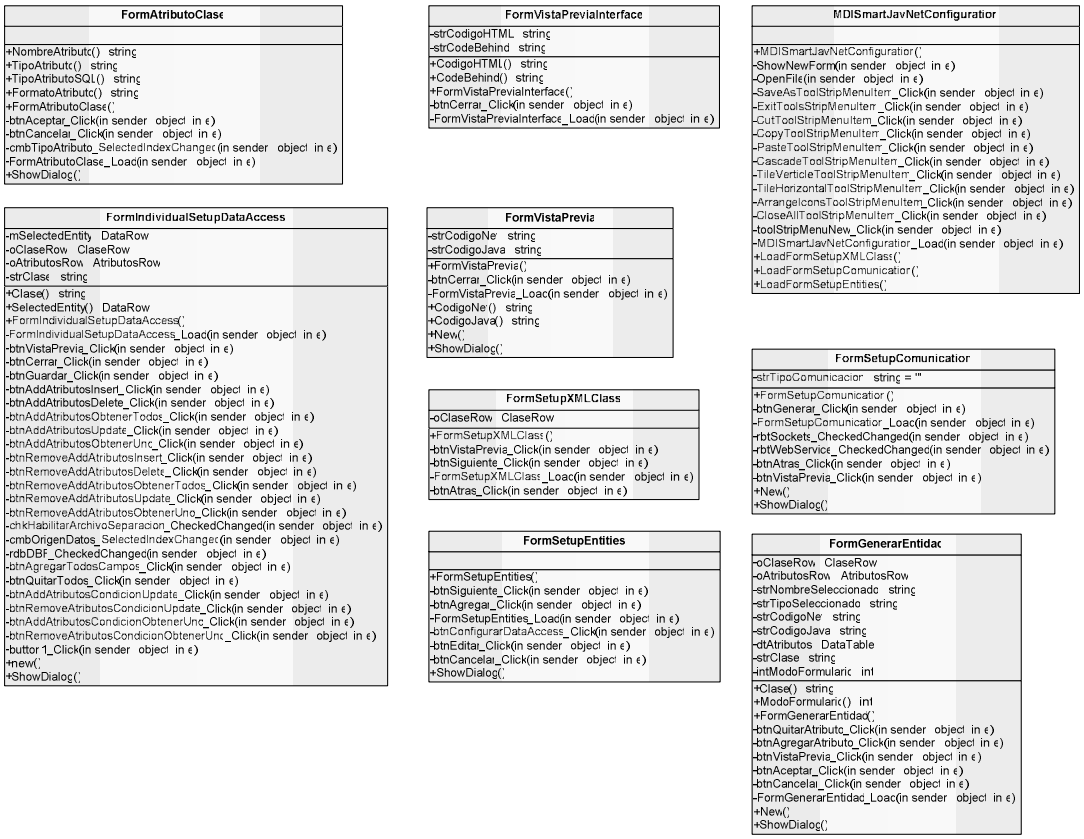
Figura 5.32 Componentes del Configurador del integrador de información



5.7.2.4.1 Componentes

- **SetupJavNetConfiguration,**
  Este componente es una aplicación de escritorio que tiene las interfaces graficas para poder realizar la configuración del integrador de información.

Figura 5.33 Diagrama de Clases del componente SetupJavNetConfiguration



- **BESetupSmartJavNet**
  Este componente contiene agrupa al conjunto de clases entidades que contienen la información de configuración, que luego serán guardadas en archivos XML, para su posterior utilización o modificación.

Figura 5.34 Diagrama de Clases del componente BESetupSmartJavNet

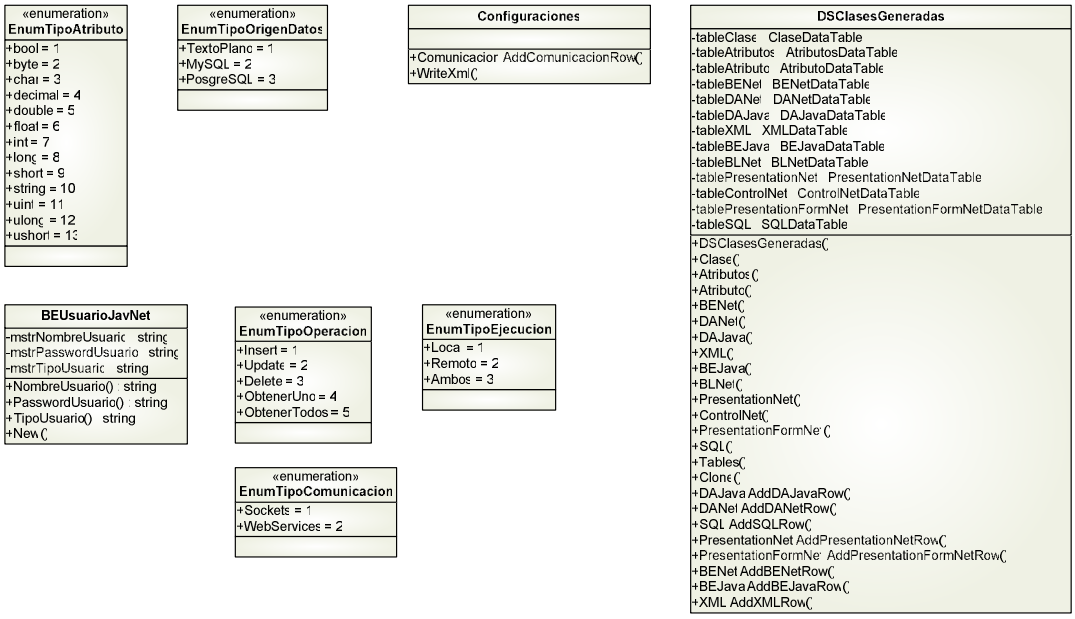
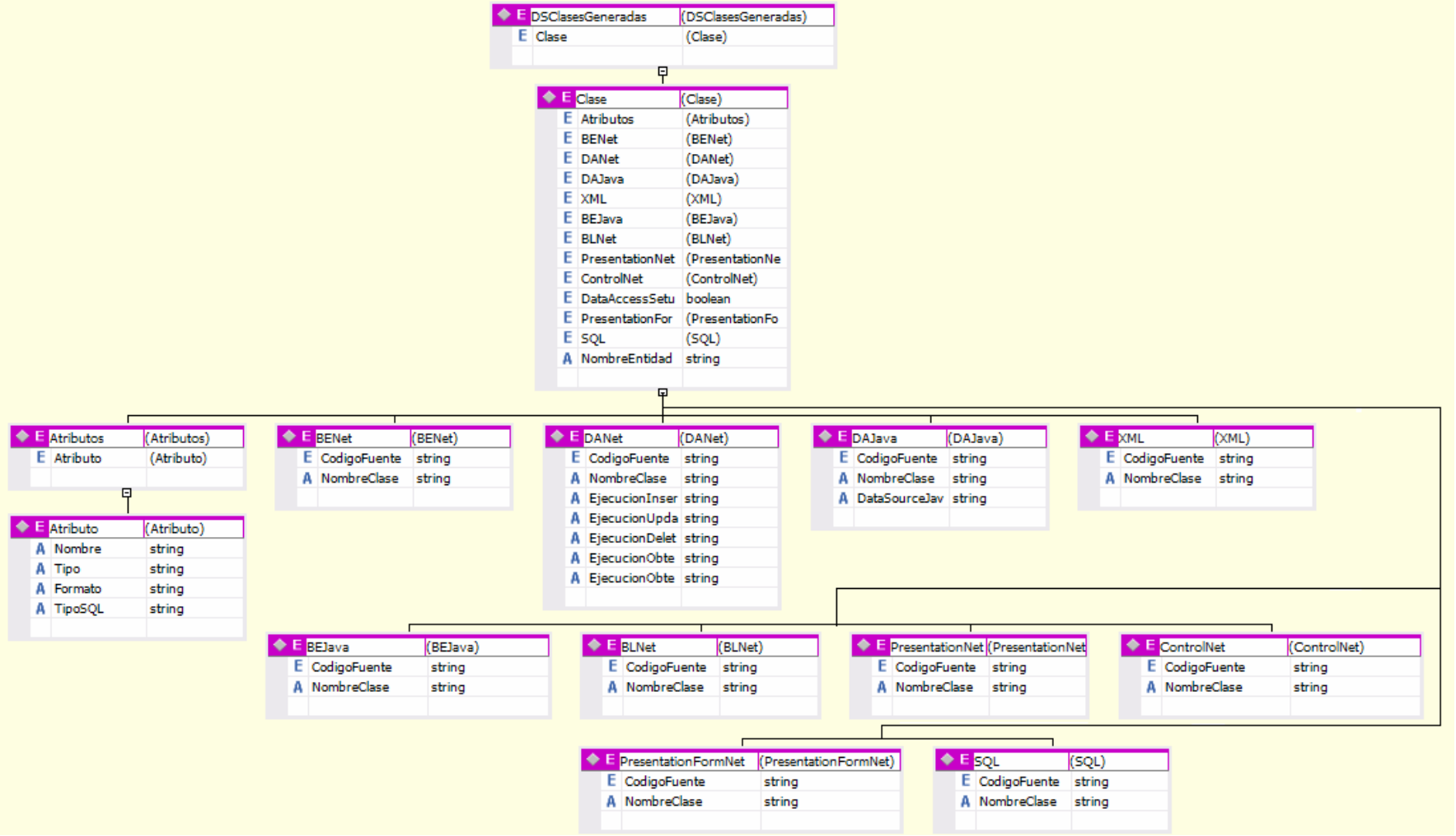
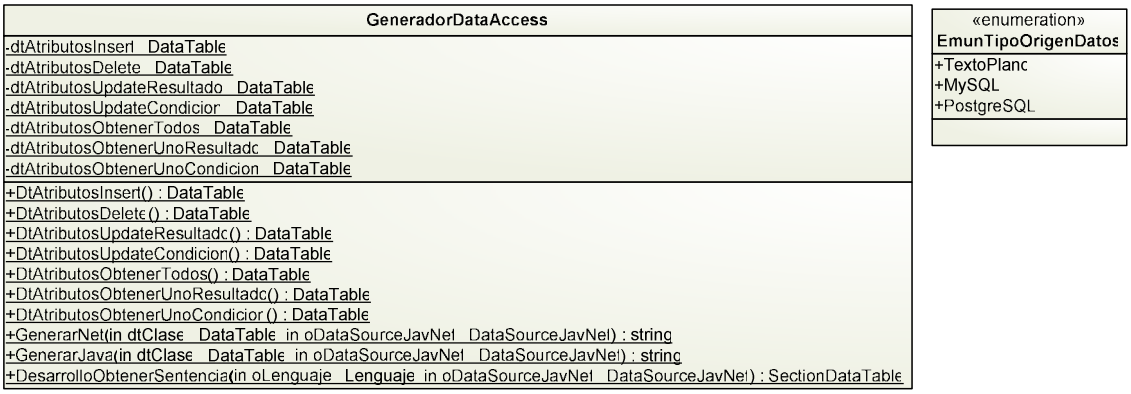


Figura 5.35 Diagrama del Esquema XML DataSet con Tipo del componente BESetupSmartJavNet



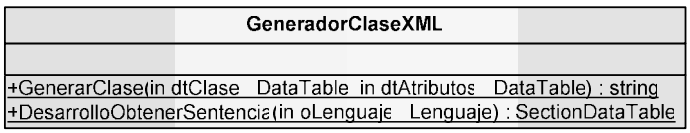
- **DataAccessGenerator**
  Este componente contiene a las clases generadoras de código para generar clases de Acceso a datos en lenguaje Java, de acuerdo a las especificaciones que el usuario desee, como son: tipo de origen de datos, si es texto plano el origen, se configura también la delimitación de los archivos, si es por carácter o por longitud. Las clases generadas son guardadas como archivos con extensión .java, listas para que sean compiladas y ejecutadas en el servidor de integración.

Figura 5.36 Diagrama de Clases del componente ataAccessGenerator



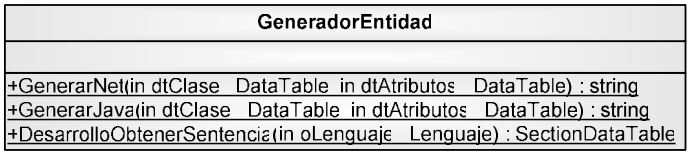
- **XMLEntitiesGenerator**
  Este componente contiene a las clases generadoras de código para generar las clases que manejaran los mensajes en XML a través de los Parsers o Intérpretes, estas clases también están en lenguaje java.
  Las clases generadas tienen que generar lenguaje XML, para que puedan ser utilizadas cuando se devuelva una consulta de datos, y también deben ser capaces de pasar estos mensajes a las entidades del servidor de integración para que pueda ejecutar las instrucciones debidas.

Figura 5.37 Diagrama de Clases del componente XMLEntitiesGenerator



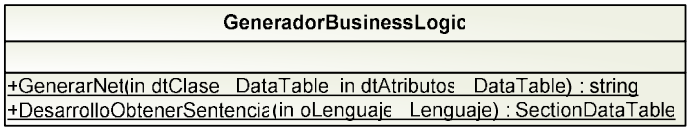
- **EntitiesGenerator**
  Este componente genera las clases entidades, tanto en lenguaje Java como en C#, es decir, desde ambas plataformas, para que cuando se den consultas, estas tengan ya un repositorio de datos en el cual guardar la información de acuerdo a cada modelo de negocio, que el usuario desee implementar. Esto facilita la configuración del integrador de información dado que puede adaptarse a diversos escenarios de negocios.

Figura 5.38 Diagrama de Clases del componente EntitiesGenerator



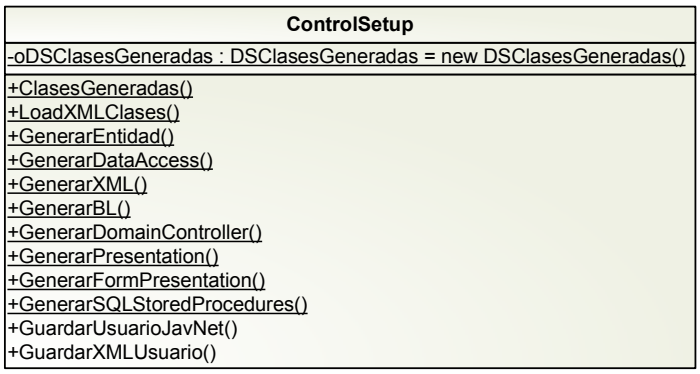
- **BLGenerator**
  Este componente genera las clases de lógica del negocio (Bussines Logic) en lenguaje en C#, la clase generada implementa los métodos Nuevo, Actualizar, Borrar, ObtenerTodos y ObtenerUno. Este tipo de clases aíslan a la capa de acceso a datos, y da la posibilidad de ingresar lógica propia del negocio andes de realizar una operación de acceso a datos.

Figura 5.39 Diagrama de Clases del componente BLGenerator



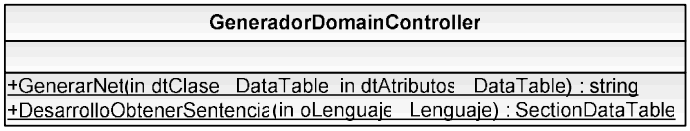
- **Controller**
  Este componente contiene a la clase ControlSetup, que se encarga de orquestar los procesos de configuración y generación de código.

Figura 5.40 Diagrama de Clases del componente Controller



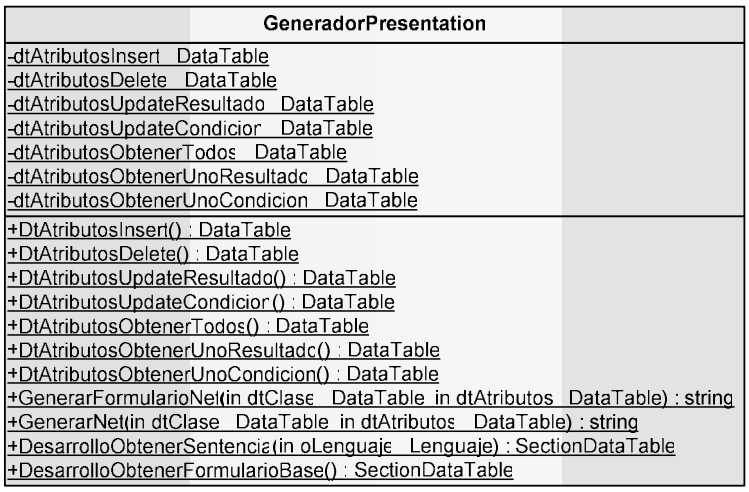
- **DomainControllerGenerator**
  Este componente contiene a la clase DomainSetupJavNet que genera la clase Controladora (DomainController) llamada así porque ella agrupara todas las llamadas desde capa de presentación, y esta interactuara con la capa de negocio y acceso a datos.
  Por cada entidad generada se genera el método que agrupa todos sus métodos de acceso a datos.

Figura 5.41 Diagrama de Clases del componente DomainControllerGenerator



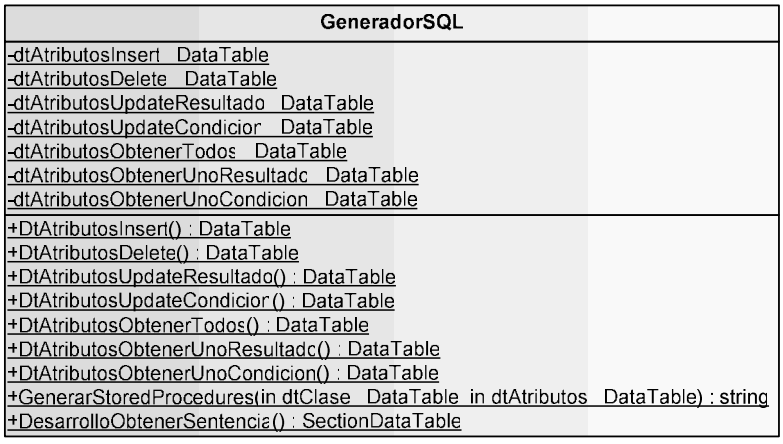
- **UIGenerator**
  Este componente contiene a la clase GeneradorPresentacion que genera el formulario en ASPX y la clase que contiene al código que ejecuta los eventos del formulario.
  Básicamente este formulario es un mantenimiento, de la entidad, de acuerdo a la configuración y condiciones que se establecieron para cada entidad en su configuración de acceso a datos.

Figura 5.42 Diagrama de Clases del componente UIGenerator



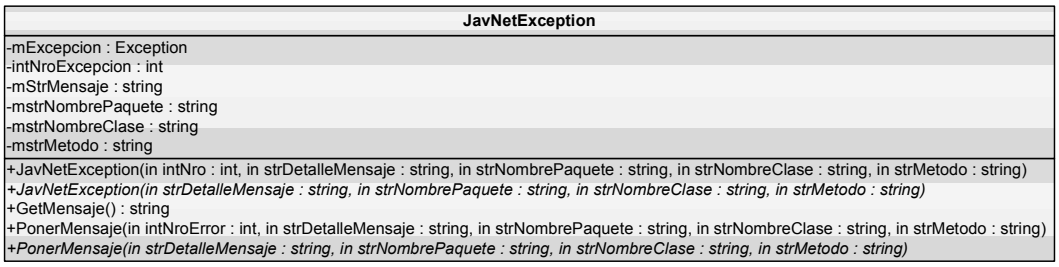
- **SQLGenerator**
  Este componente contiene a la clase GeneradorSQL que genera un archivo SQL, que contiene la tabla a ser creada y los procedimientos almacenados para una probable migración del origen de datos.
  Esta generación de código está de acuerdo a la configuración y condiciones que se establecieron para cada entidad en su configuración de acceso a datos.

Figura 5.43 Diagrama de Clases del componente SQLGenerator



- **JavNetException**
  Este componente contiene a la clase JavNetException que hereda de la clase System.Exception, y nos sirve para personalizar el manejo de las excepciones que pudieran ocurrir en la ejecución del sistema Configurador.

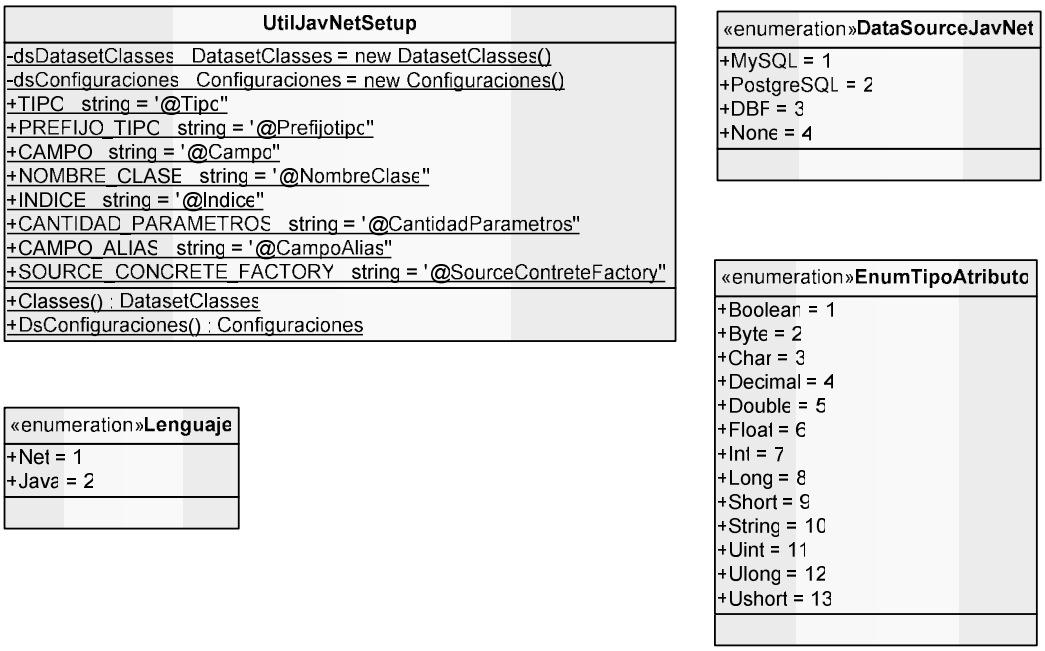
Figura 5.44 Diagrama de Clases del componente JavNetException



- **Util**
  Este componente contiene a las clases: UtilJavNet que contiene atributos estáticos que guardan el Dataset que contiene las clases generadas, el Dataset que contiene la configuración de comunicación y tipos de ejecuciones para cada entidad; también por otro lado contiene constantes que sirven para buscar patrones en las plantillas de las clases a generar.

  El componente también contiene varios tipos enumeradores, que sirven para listar elementos como el tipo de origen de datos (DataSourceJavNet), los tipos de atributos que contendrán las entidades (EnumTipoAtributo), y los tipos de lenguajes en los que serán generadas las clases (Lenguaje).

Figura 5.45 Diagrama de Clases del componente Util



5.7.2.5 Realizaciones de los Casos de Uso del Módulo Configurador

La siguiente sección describe las realizaciones de cada caso de uso modelado del módulo de configuración, y también se muestran los diagramas de secuencia correspondientes, de tal forma que se pueda apreciar la secuencia lógica de los procesos.

5.7.2.5.1 Realización del Caso de Uso Abrir Archivo de Integración Existente

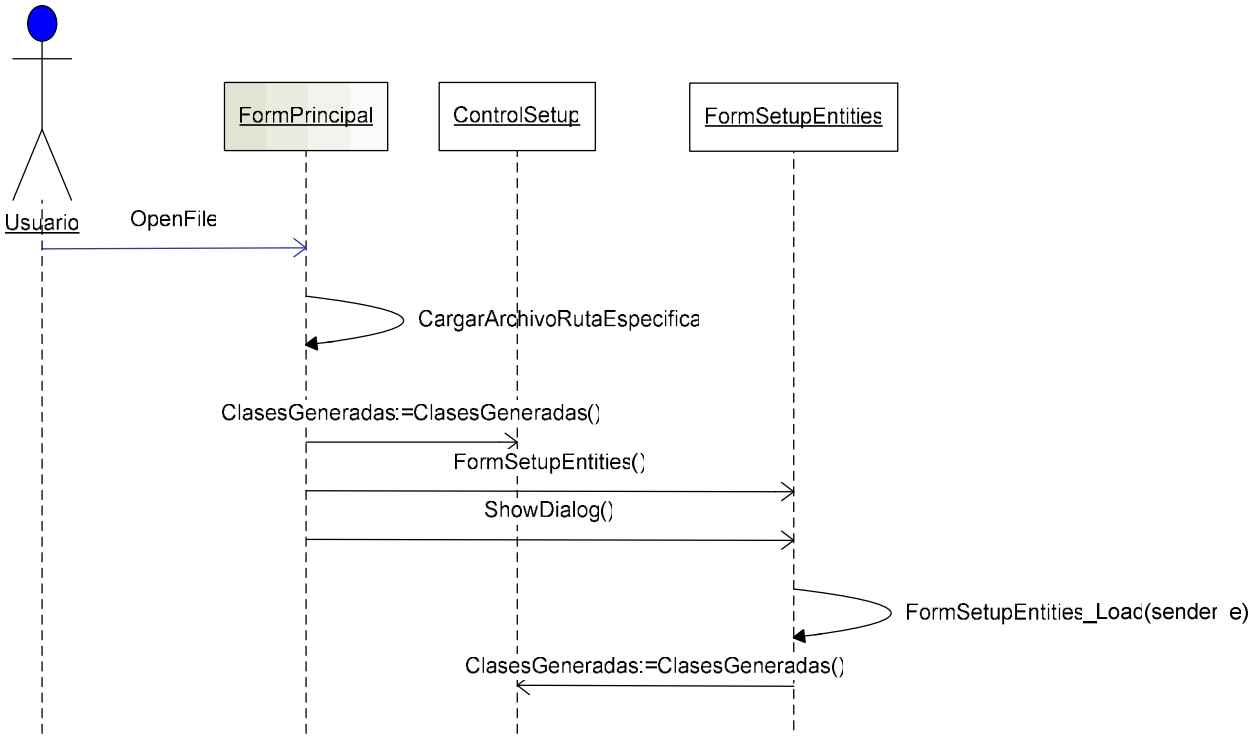
1. El caso de uso comienza cuando el usuario accede al menú del Formulario Principal y elige la opción File y luego Open, y se le muestra una ventana de dialogo que le indica elegir la ruta y nombre del archivo que desea abrir, este tiene que tener la extensión SJN.

2. El usuario elige la opción aceptar en la ventana de dialogo de abrir archivo.

3. El formulario principal accede a la clase ControlSetup, a la propiedad estática Clases Generadas, y le asigna la lectura del archivo XML, con extensión SJN.

4. La Clase ControlSetup instancia al formulario SetupEntities e invoca a su método ShowDialog, éste formulario a su vez invoca su método FormSetupEntities_Load, que carga en la grilla las entidades del archivo abierto, listo para una Edición.

Figura 5.46 Diagrama de Secuencia del Caso de Uso Abrir Archivo de Integración Existente



5.7.2.5.2 Realización del Caso de Uso Configurar Entidades

1. El caso de uso comienza cuando el usuario accede al formulario FormSetupEntities y presiona el botón agregar, enseguida se crea una nueva instancia del formulario FormGenerarEntidad, y se invoca a su método ShowDialog, y se muestra el formulario FormGenerarEntidad.

2. El Usuario ingresa el nombre de la Entidad, y luego presiona el botón de agregar atributos, y se instancia un objeto del formulario FormAtributoClase.

3. El Usuario ingresa el nombre del atributo, así como también el tipo para C# y Java y el tipo para SQL, el usuario presiona agregar, y el formulario FormAtributoClase se cierra, y el atributo queda mostrado en la grilla del formulario FormGenerarEntidad. Este proceso es repetitivo según cuantos atributos se desea ingresar.

4. El usuario presiona el botón de vista previa en el formulario FormGenerarEntidad, y se invoca al método GenerarEntidad de la clase Controlsetup, este a su vez ejecuta el método GenerarNet y GenerarJava de la clase GeneradorEntidad, que generan las entidades en C# y Java respectivamente, cada uno de estos métodos ejecuta el método de la misma clase DesarrolloObtenerSentencia, que busca la plantilla en XML específica para generar la clase. Los métodos GenerarJava y GenerarNet generan el código según sus atributos señalados.

5. En el formulario FormGenerarEntidad, una vez que tenemos los códigos fuente de la clases generadas, se instancia un objeto del formulario FormVistaPrevia y se le asigna por propiedad los códigos fuentes de las clases en texto, luego se ejecuta el método ShowDialog para poder visualizar el formulario, el formulario ejecuta su método FormVistaPrevia_Load, y carga los códigos fuente como texto en sus controles para ser visualizados.

6. Una vez que el usuario está seguro que su configuración es la correcta entonces, da click sobre el botón cerrar en el formulario FormGenerarEntidad, invoca al método GenerarEntidad de la clase Controlsetup, este a su vez ejecuta el método GenerarNet y GenerarJava de la clase GeneradorEntidad, que generan las entidades en C# y Java respectivamente, cada uno de estos métodos ejecuta el método de la misma clase DesarrolloObtenerSentencia, que busca la plantilla en XML específica para generar la clase. Los métodos GenerarJava y GenerarNet generan el código según sus atributos señalados.

7. Listos los códigos fuentes generados, el formulario FormGenerarEntidad llama a la propiedad ClasesGeneradas, que se encargan de guardar en memoria las clases generadas en un DataSet de tipo DSClasesGeneradas, a las tablas de este DataSet BENet y BEJava se le agregan un nuevas Filas que guardaran el código fuente generado hasta que se guarden en archivos .java y .cs.

8. De esta forma quedan configuradas las entidades, este proceso puede ser repetido por cada entidad que se desee configurar.

Figura 5.47 Diagrama de Secuencia del Caso de Uso Configurar Entidades – Parte 1

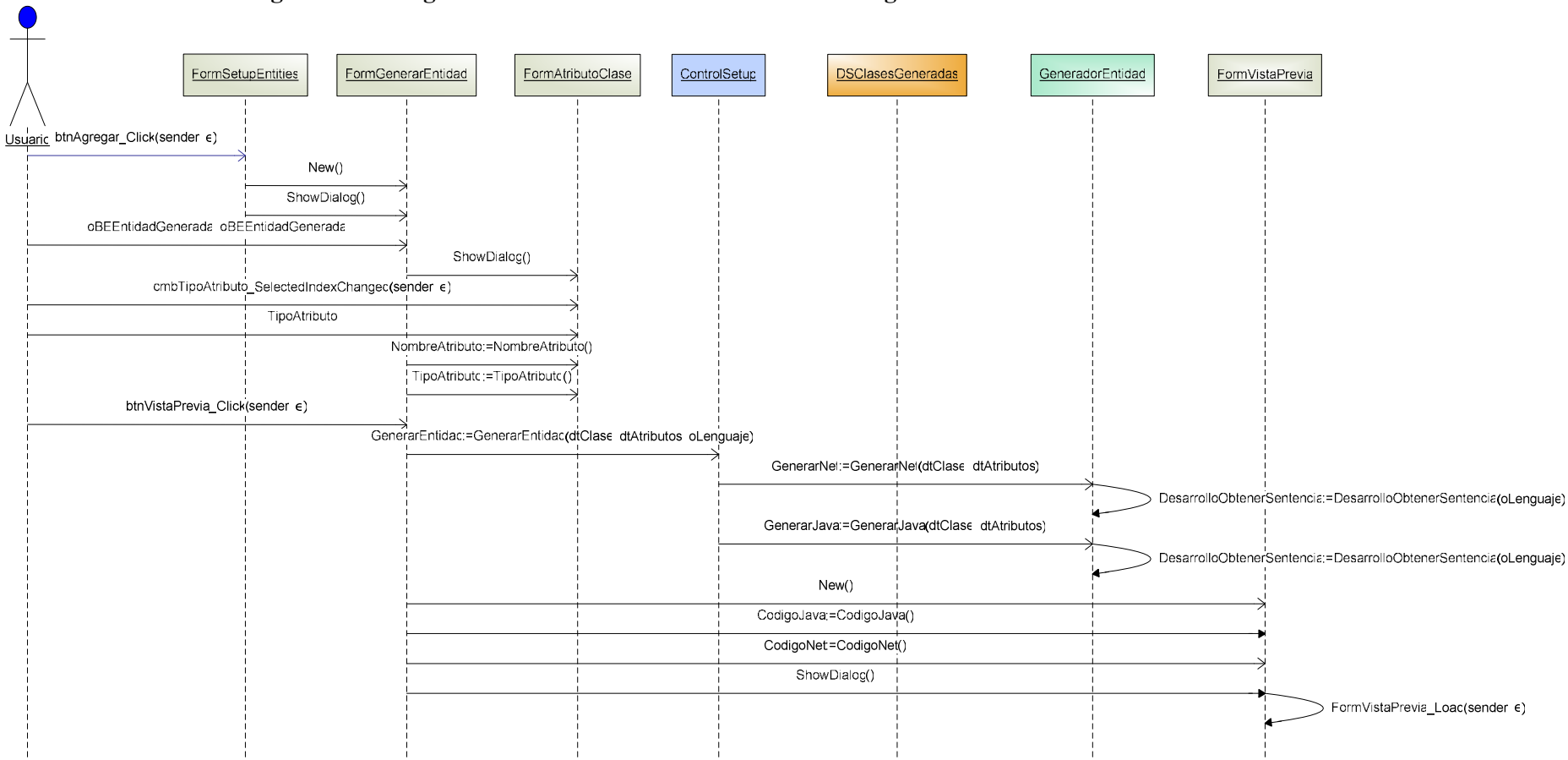
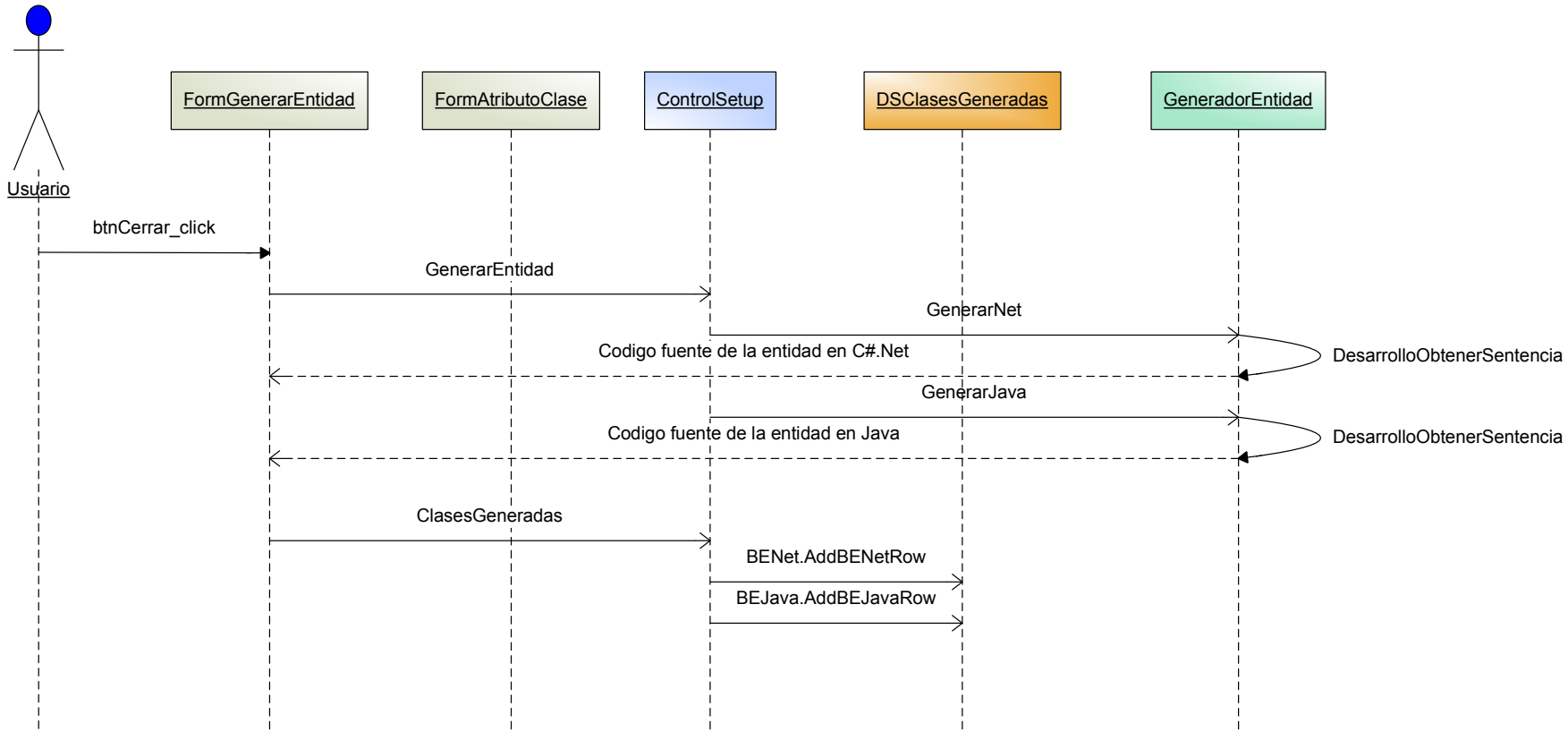


Figura 5.48 Diagrama de Secuencia del Caso de Uso Configurar Entidades – Parte 2



5.7.2.5.3 Realización del Caso de Uso Configurar Acceso Datos – Vista Previa

1. El caso de uso comienza cuando el usuario accede al formulario FormSetupEntities y presiona el botón Configurar Data Access, enseguida se crea una nueva instancia del formulario FormIndividualSetupDataAccess, y se invoca a su método ShowDialog, y se muestra el formulario FormIndividualSetupDataAccess.

2. El Usuario puede elegir las configuraciones que desee, como tipo de origen de datos y los atributos que irán en cada operación de acceso a datos, y presiona el botón Vista Previa.

3. El formulario FormIndividualSetupDataAccess ejecuta el método GenerarDataAccess de la clase ControlSetup, esta clase a su vez, asigna los DataTables correspondientes a la clase GeneradorDataAccess, estos datatables son :

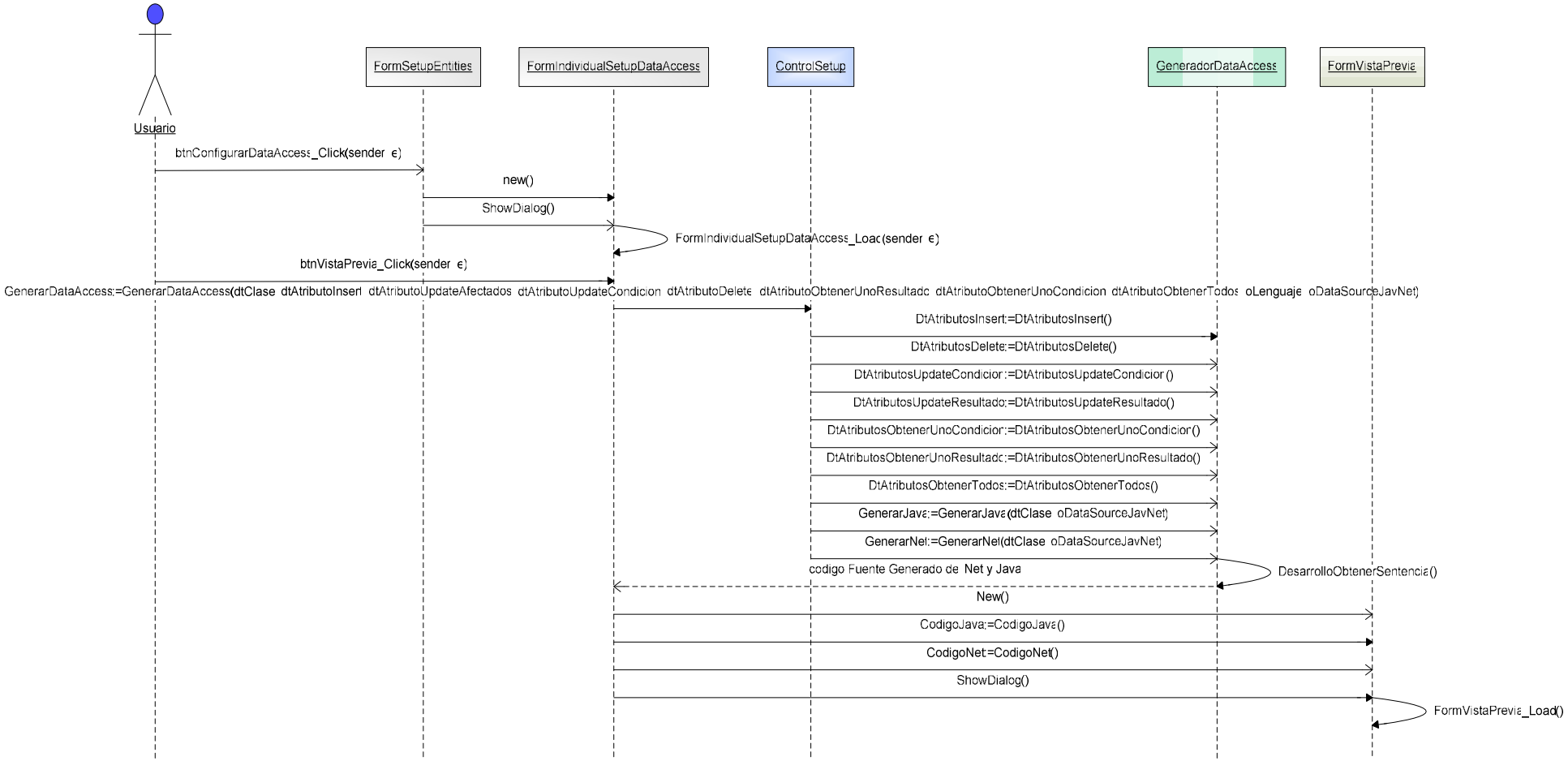
   a. DtAtributosInsert
   b. DtAtributosDelete
   c. DtAtributosUpdateCondicion
   d. DtAtributosUpdateResultado
   e. DtAtributosObtenerUnoCondicion
   f. DtAtributosObtenerUnoResultado
   g. DtAtributosObtenerTodos

4. Luego el método GenerarDataAccess de la clase ControlSetup ejecuta los métodos GenerarJava y GenerarNet, que generan las clases de acceso a datos tanto para java como para C#.

5. Los métodos GenerarJava y GenerarNet, ejecutan el método DesarrolloObtenerSentencia que obtiene la plantilla correspondiente para generar el código fuente de las clases, una vez generadas las clases de acuerdo a las especificaciones definidas por el usuario para cada operación, y el tipo de origen de datos, el códigos fuente en Java y C#, son devueltos al formulario FormIndividualSetupDataAccess.

6. En el formulario FormIndividualSetupDataAccess, una vez que tenemos los códigos fuente de la clases generadas, se instancia un objeto del formulario FormVistaPrevia y se le asigna por propiedad los códigos fuentes de las clases en texto, luego se ejecuta el método ShowDialog para poder visualizar el formulario, el formulario ejecuta su método FormVistaPrevia_Load, y carga los códigos fuente como texto en sus controles para ser visualizados.

Figura 5.49 Diagrama de Secuencia del Caso de Uso Configurar Acceso a Datos – Vista Previa



5.7.2.5.4 Realización del Caso de Uso Configurar Acceso Datos – Generación de Código

1. El caso de uso comienza cuando el Usuario puede elegir las configuraciones que desee, como tipo de origen de datos y los atributos que irán en cada operación de acceso a datos, y presiona el botón Guardar.

2. El formulario FormIndividualSetupDataAccess ejecuta el método GenerarDataAccess de la clase ControlSetup, esta clase a su vez, asigna los datatables correspondientes a la clase GeneradorDataAccess, estos datatables son :

   a. DtAtributosInsert
   b. DtAtributosDelete
   c. DtAtributosUpdateCondicion
   d. DtAtributosUpdateResultado
   e. DtAtributosObtenerUnoCondicion
   f. DtAtributosObtenerUnoResultado
   g. DtAtributosObtenerTodos

3. Luego el método GenerarDataAccess de la clase ControlSetup ejecuta los métodos GenerarJava y GenerarNet, que generan las clases de acceso a datos tanto para java como para C#.

4. Los métodos GenerarJava y GenerarNet, ejecutan el método DesarrolloObtenerSentencia que obtiene la plantilla correspondiente para generar el código fuente de las clases, una vez generadas las clases de acuerdo a las especificaciones definidas por el usuario para cada operación, y el tipo de origen de datos, el códigos fuente en Java y C#, son devueltos al formulario FormIndividualSetupDataAccess.

5. Luego formulario FormIndividualSetupDataAccess ejecuta el método GenerarPresentacion de la clase ControlSetup, esta clase a su vez, asigna los datatables correspondientes a la clase GeneradorPresentation, estos datatables son :

   a. DtAtributosInsert
   b. DtAtributosDelete
   c. DtAtributosUpdateCondicion
   d. DtAtributosUpdateResultado
   e. DtAtributosObtenerUnoCondicion
   f. DtAtributosObtenerUnoResultado
   g. DtAtributosObtenerTodos

6. Luego el método GenerarPresentacion de la clase ControlSetup ejecuta los métodos GenerarNet y GenerarFormularioNet, el primero de ellos genera el código

detrás del formulario, y el segundo genera el código HTML del formulario por cada entidad.

7. Los métodos GenerarNet y GenerarFormularioNet, ejecutan el método DesarrolloObtenerSentencia que obtiene la plantilla correspondiente para generar el código fuente de las clases, una vez generadas las clases de acuerdo a las especificaciones definidas por el usuario para cada operación, y el tipo de origen de datos, el códigos fuente en C# y HTML, son devueltos al formulario FormIndividualSetupDataAccess.

8. Luego formulario FormIndividualSetupDataAccess ejecuta el método GenerarSQLStoredProcedures de la clase ControlSetup, esta clase a su vez, asigna los datatables correspondientes a la clase GeneradorSQL, estos datatables son :

   a. DtAtributosInsert
   b. DtAtributosDelete
   c. DtAtributosUpdateCondicion
   d. DtAtributosUpdateResultado
   e. DtAtributosObtenerUnoCondicion
   f. DtAtributosObtenerUnoResultado
   g. DtAtributosObtenerTodos

9. Luego el método GenerarSQLStoredProcedures de la clase ControlSetup ejecuta el método GenerarStoredProcedures, que se encarga de generar los Stored Procedures, Tabla y sentencias SQL, para cada entidad.

10. El método GenerarStoredProcedures, ejecuta el método DesarrolloObtenerSentencia que obtiene la plantilla correspondiente para generar el código SQL, una vez generado el código SQL de acuerdo a las especificaciones definidas por el usuario para cada operación, y el tipo de origen de datos, el códigos fuente en SQL, es devuelto al formulario FormIndividualSetupDataAccess.

11. Una vez completado los pasos de Generación de código fuente en C#, java, SQL y HTML, el formulario FormIndividualSetupDataAccess comienza por guardar la configuración en XML de las ejecuciones remota y local.

12. Finalmente el formulario FormIndividualSetupDataAccess llama a la propiedad ClasesGeneradas, que se encargan de guardar en memoria las clases generadas en un DataSet de tipo DSClasesGeneradas, a las tablas de este DataSet: DAJava, DANet, SQL, PresentationNet y PresentationFormNet se le agregan un nuevas Filas que guardaran el código fuente

generado hasta que se guarden en archivos con extensiones java, cs, aspx y sql.

13. De esta forma quedan configuradas los accesos a datos, este proceso puede ser repetido por cada entidad que se desee configurar.

Figura 5.50 Diagrama de Secuencia del Caso de Uso Configurar Acceso a Datos – Generación de Código – Parte 1

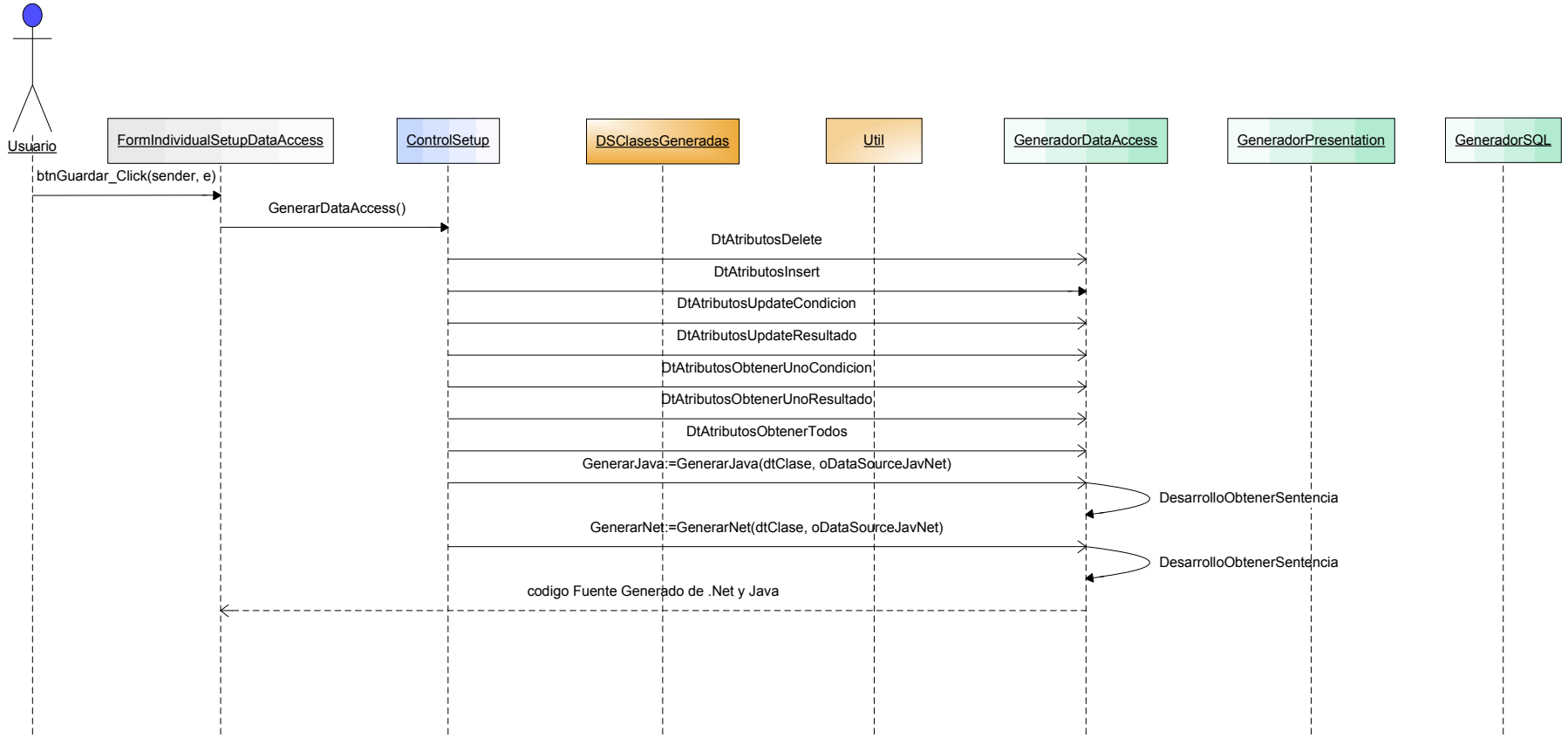


Figura 5.51 Diagrama de Secuencia del Caso de Uso Configurar Acceso a Datos – Generación de Código – Parte 2

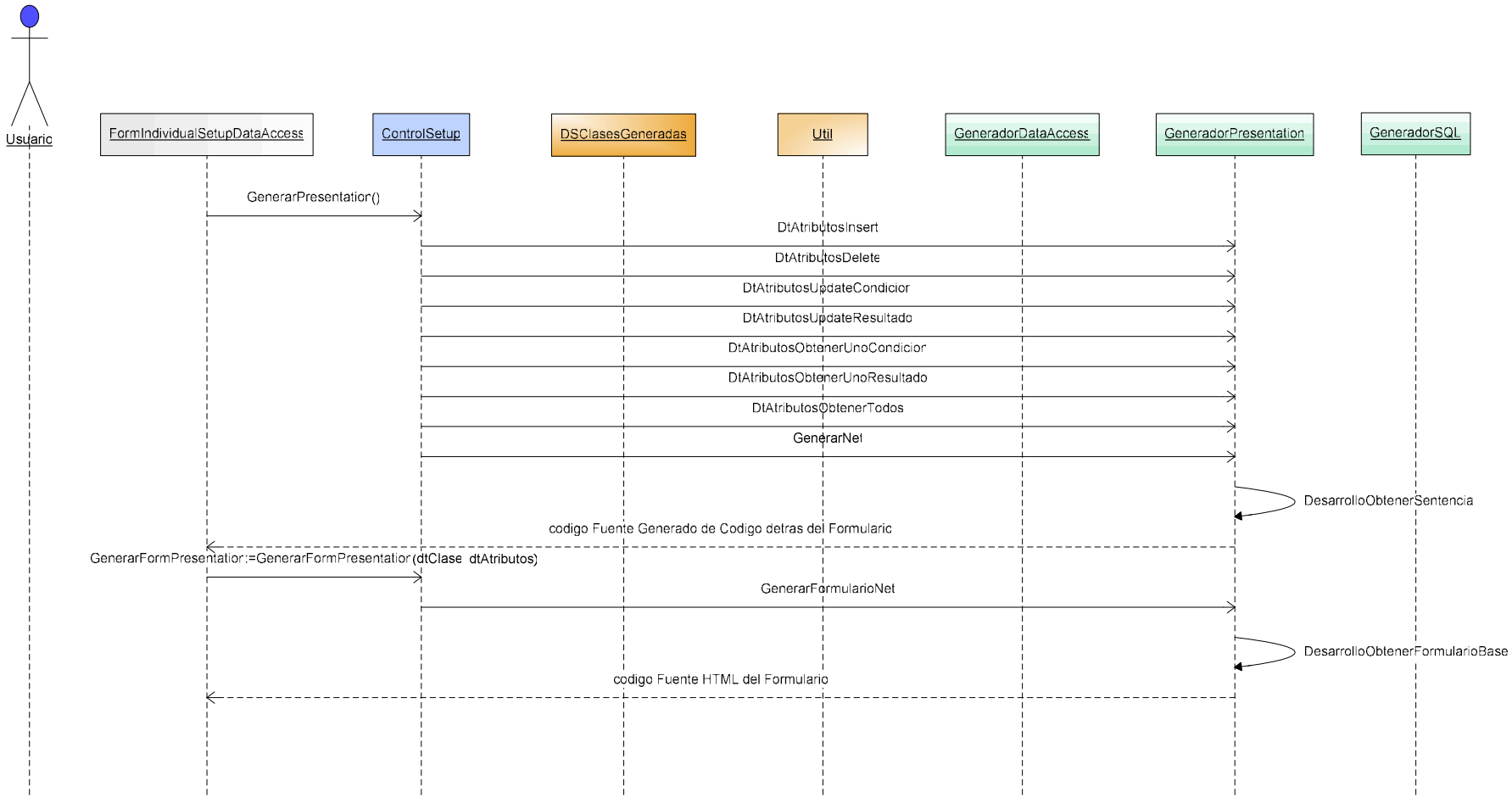
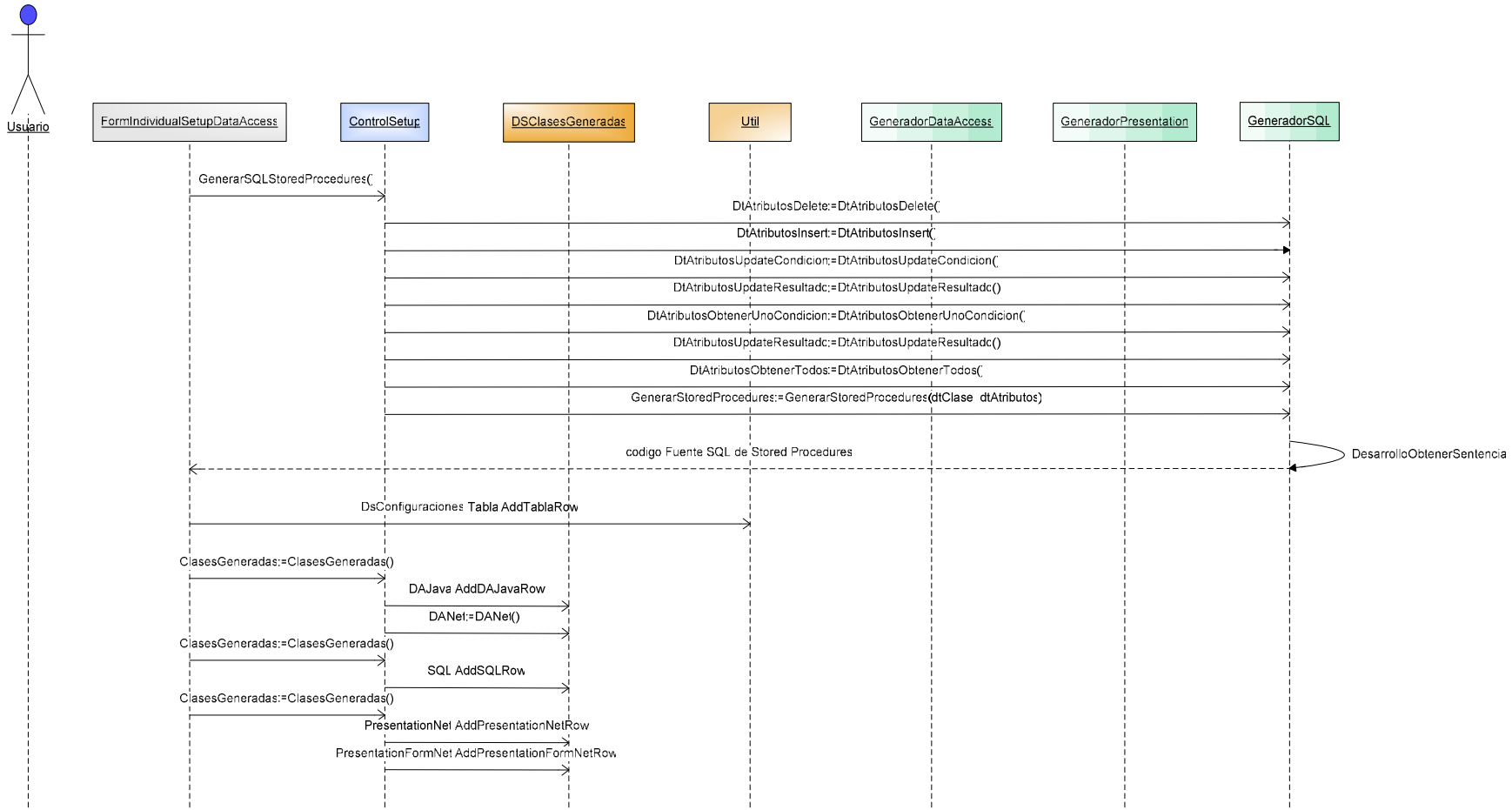


Figura 5.52 Diagrama de Secuencia del Caso de Uso Configurar Acceso a Datos – Generación de Código – Parte 3



5.7.2.5.5 Realización del Caso de Uso Generar Clases de XML

1. El caso de uso comienza cuando el usuario presiona el botón de vista previa en el formulario FormSetupXMLClass, y se invoca al método GenerarXML de la clase Controlsetup, este a su vez ejecuta el método GenerarClase de la clase GeneradorClaseXML, que genera la clase en Java, este método a su vez ejecuta el método de la misma clase DesarrolloObtenerSentencia, que busca la plantilla en XML específica para generar la clase, la clase generada servirá para generar XML por cada operación de consulta que se pida desde el cliente en .NET.

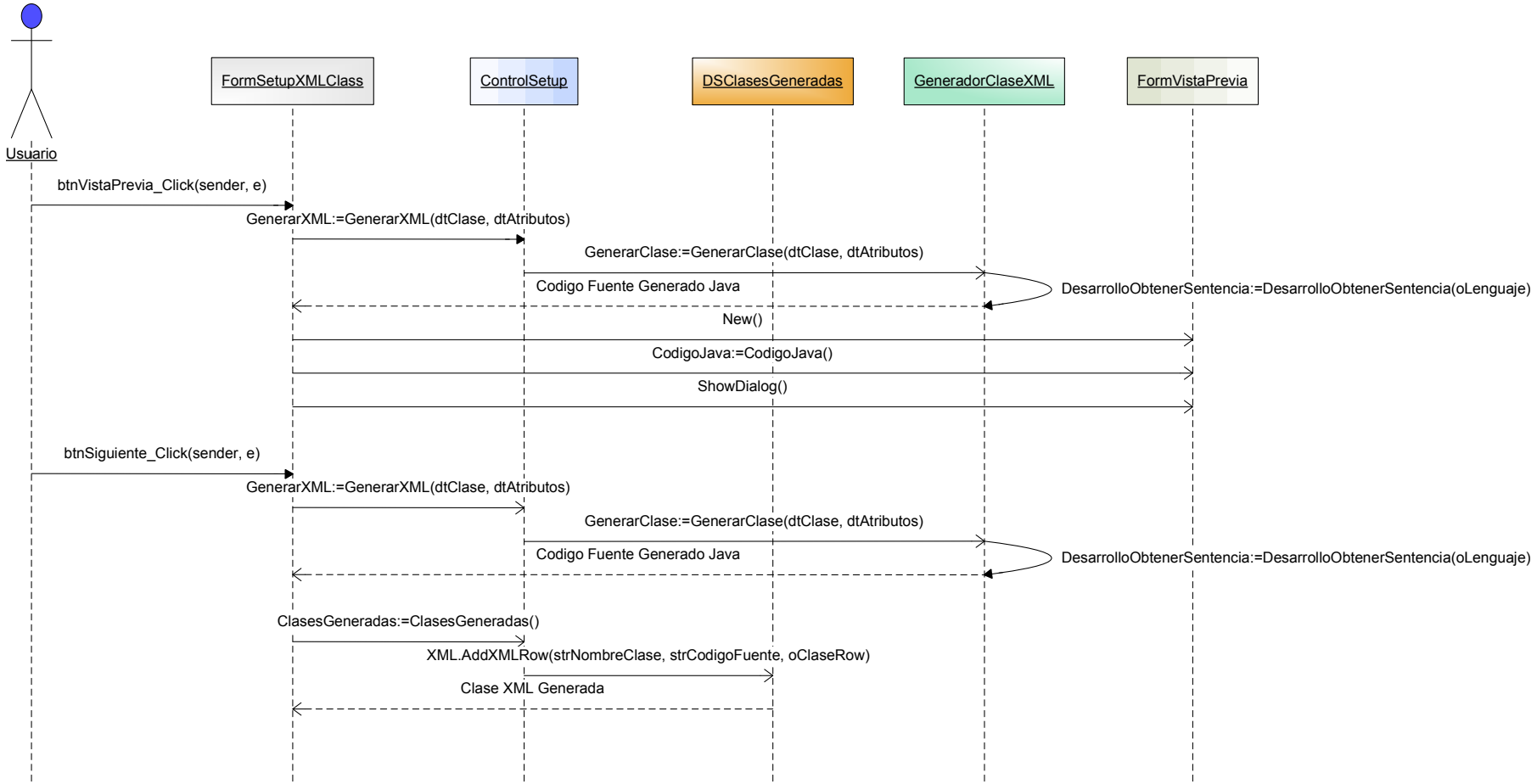
2. En el formulario FormSetupXMLClass, una vez que tenemos los códigos fuente de la clases generadas, se instancia un objeto del formulario FormVistaPrevia y se le asigna por propiedad el código fuente de las clase seleccionada en texto, luego se ejecuta el método ShowDialog para poder visualizar el formulario, el formulario ejecuta su método FormVistaPrevia_Load, y carga el códigos fuente en Java como texto en su controles para ser visualizado, solo se visualiza en Java por que no existe generación en C#.

3. Una vez que el usuario está seguro que su configuración es la correcta entonces, da click sobre el botón Siguiente en el formulario FormSetupXMLClass, invoca al método GenerarXML de la clase Controlsetup, este a su vez ejecuta el método GenerarClase de la clase GeneradorClaseXML, que genera la clase en Java, este método a su vez ejecuta el método de la misma clase DesarrolloObtenerSentencia, que busca la plantilla en XML específica para generar la clase.

4. Listos los códigos fuentes generados, el formulario FormSetupXMLClass llama a la propiedad ClasesGeneradas, que se encargan de guardar en memoria las clases generadas en un DataSet de tipo DSClasesGeneradas, a la tabla de este DataSet llamado XML y se le agregan un nuevas Filas que guardaran el código fuente generado hasta que se guarden en archivos .java.

5. De esta forma quedan configuradas las clases que generan XML en lenguaje Java, este proceso puede es repetido por cada entidad que se tiene configurada.

Figura 5.53 Diagrama de Secuencia del Caso de Uso Generar Clases de XML



5.7.2.5.6 Realización del Caso de Uso Configurar Comunicación

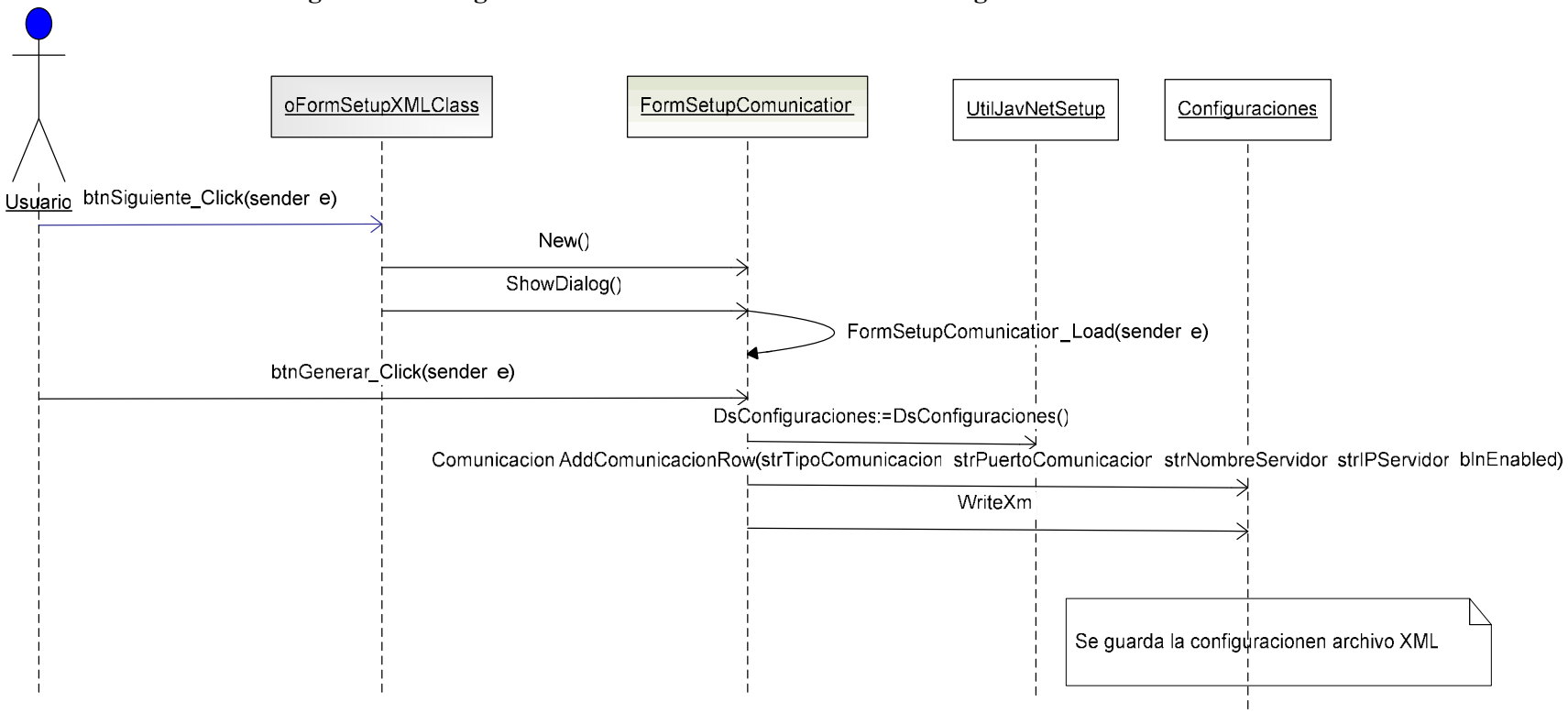
1. El caso de Uso comienza cuando el usuario da click en el botón Siguiente en el formulario FormSetupXMLClass, y después de generar su clase XMl, instancia un nuevo objeto del formulario FormSetupComunication y se invoca a su método ShowDialog, y el formulario FormSetupComunication es mostrado en pantalla.

2. Luego el usuario da click en el botón Generar y el formulario llama a la propiedad pública DsConfiguraciones de la clase UtilJavNetSetup que sirve como clase utilitaria para guardar información en memoria de todos los procesos del sistema.

3. La propiedad DsConfiguraciones es del tipo DataSet Configuraciones, que genera la configuración de la comunicación que tendrá la integración, ya sea por Web Services, o sockets, que número de puerto será utilizado y cuál es el nombre y dirección IP del servidor de integración.

4. El formulario FormSetupComunication agrega una nueva fila para la comunicación, luego ejecuta la operación WriteXML del DataSet Configuraciones para que se genere en XML, la configuración a ser leída por el sistema integrador.

Figura 5.54 Diagrama de Secuencia del Caso de Uso Configurar Comunicación

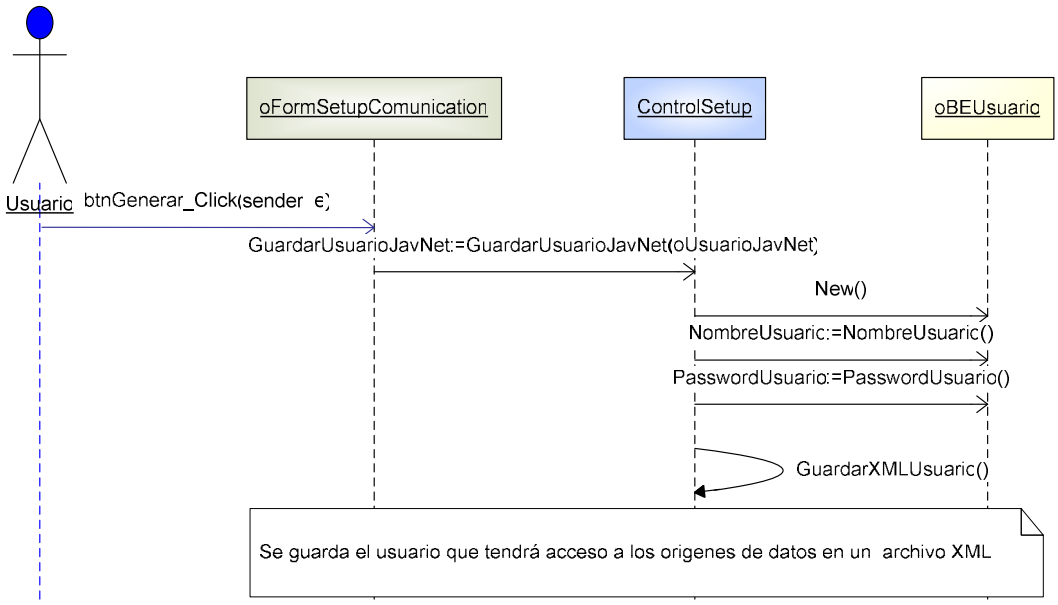


5.7.2.5.7 Realización del Caso de Uso Registrar Usuario

1. El caso de Uso comienza cuando el usuario da click en el botón Generar del formulario y se instancia un objeto de la clase BEUsuario, se llena las propiedades NombreUsuario y PasswordUsuario y luego se ejecuta el método GuardarUsuarioJavNet recibiendo como parámetro el objeto creado de BEUsuario.

2. La clase ControlSetup ejecuta su propio método GuardarXMLUsuario de comunicación pueda validar al usuario que realiza la operación desde el cliente.

Figura 5.55 Diagrama de Secuencia del Caso de Uso Registrar Usuario



5.7.2.5.8 Realización del Caso de Uso Generar Código Fuente

1. El caso de Uso comienza cuando el usuario da click en el botón Generar del formulario y luego se da una sentencia repetitiva por cada entidad que se haya configurado a lo largo de todo el proceso.

2. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable BENet y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.CS, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase generada de Bussines Entities de .Net, y se termina ejecutando el método Close del objeto StreamWriter.

3. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable BEJava y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.JAVA, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase generada de Bussines Entities de Java, y se termina ejecutando el método Close del objeto StreamWriter.

4. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable DANet y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con DA + NombreClase.CS, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase generada de Data Access de .NET, y se termina ejecutando el método Close del objeto StreamWriter.

5. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable DAJava y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del

archivo, que también contiene el nombre de la Clase que se genera con NombreClase.JAVA, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase generada de Data Access de Java, y se termina ejecutando el método Close del objeto StreamWriter.

6. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable SQL y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.SQL, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase generado el Archivo SQL que contiene las sentencias que crea la tabla y los procedimientos almacenados, y se termina ejecutando el método Close del objeto StreamWriter.

7. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable XML y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.JAVA, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase generada de generación de XML desde el servidor de integración, y se termina ejecutando el método Close del objeto StreamWriter.

8. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable ControlNet y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.CS, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase Controller que se ejecuta en el Cliente, luego se termina ejecutando el método Close del objeto StreamWriter.

9. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable BLNet y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se

genera con NombreClase.CS, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase Bussines Logic que se ejecuta en el Cliente, luego se termina ejecutando el método Close del objeto StreamWriter.

10. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable PresentationNet y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.CS, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene la clase de Presentacion que tiene el código detrás del diseño del formulario web que se ejecuta en el Cliente, luego se termina ejecutando el método Close del objeto StreamWriter.

11. El formulario FormSetupComunication hace referencia a la propiedad Clases Generadas del tipo DSClasesGeneradas, a través de este DataSet se obtiene el DataTable PresentationFormNet y a sus propiedades NombreClase y CodigoFuente, luego de esto se instancia un objeto de la clase StreamWriter con la ruta del archivo, que también contiene el nombre de la Clase que se genera con NombreClase.ASPX, y se ejecuta el método Write y con el parámetro CodigoFuente, de esta manera se genera el archivo que contiene el código HTML del diseño del formulario web que se ejecuta en el Cliente, luego se termina ejecutando el método Close del objeto StreamWriter.

12. Luego de generadas los archivos con extensión CS, JAVA, SQL y ASPX, se informa al usuario que se termino de generar, el usuario pulsa Aceptar y termina todo el proceso.

Figura 5.56 Diagrama de Secuencia del Caso de Uso Generar Código Fuente – Parte 1

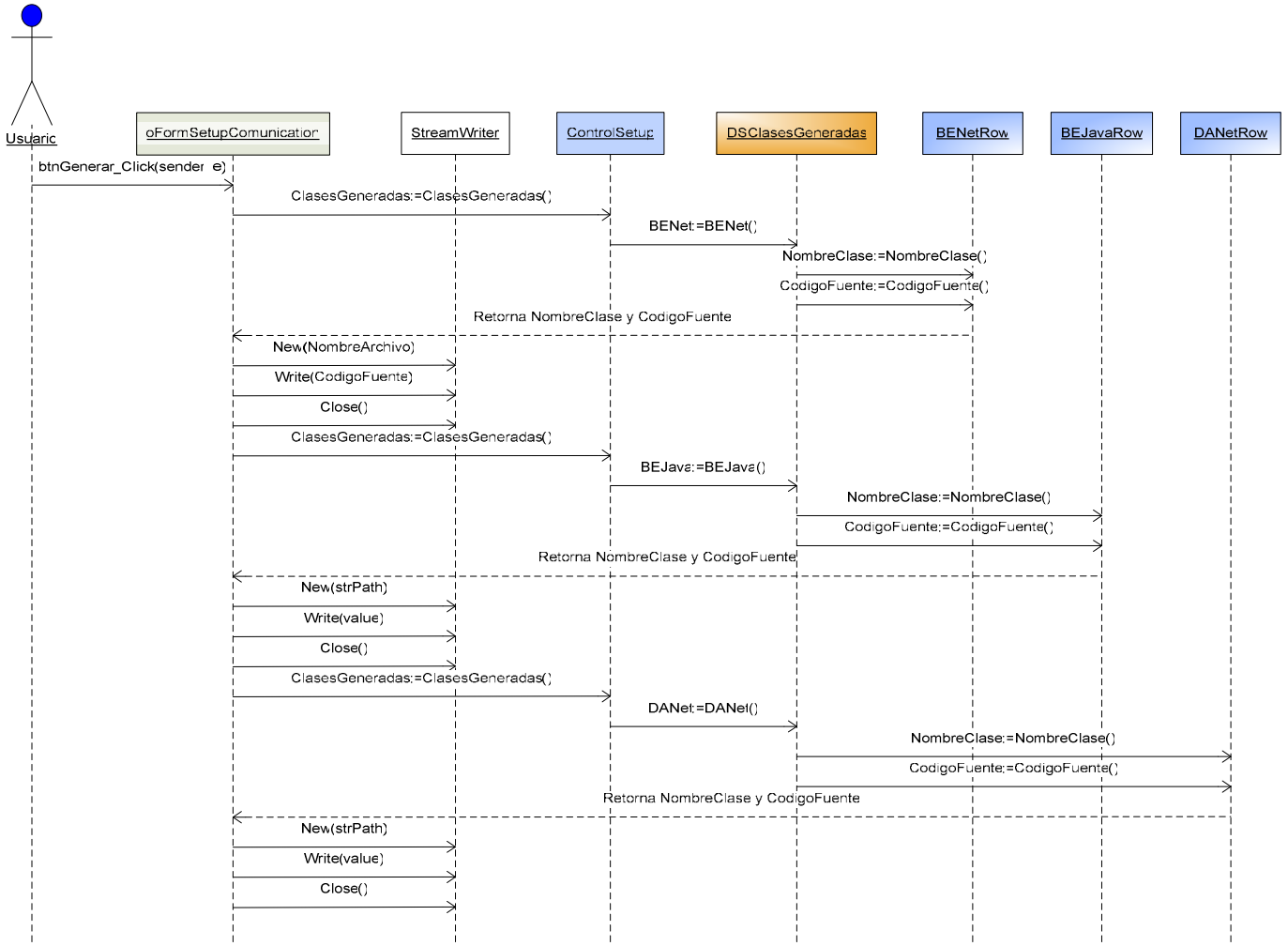


Figura 5.57 Diagrama de Secuencia del Caso de Uso Generar Código Fuente – Parte 2

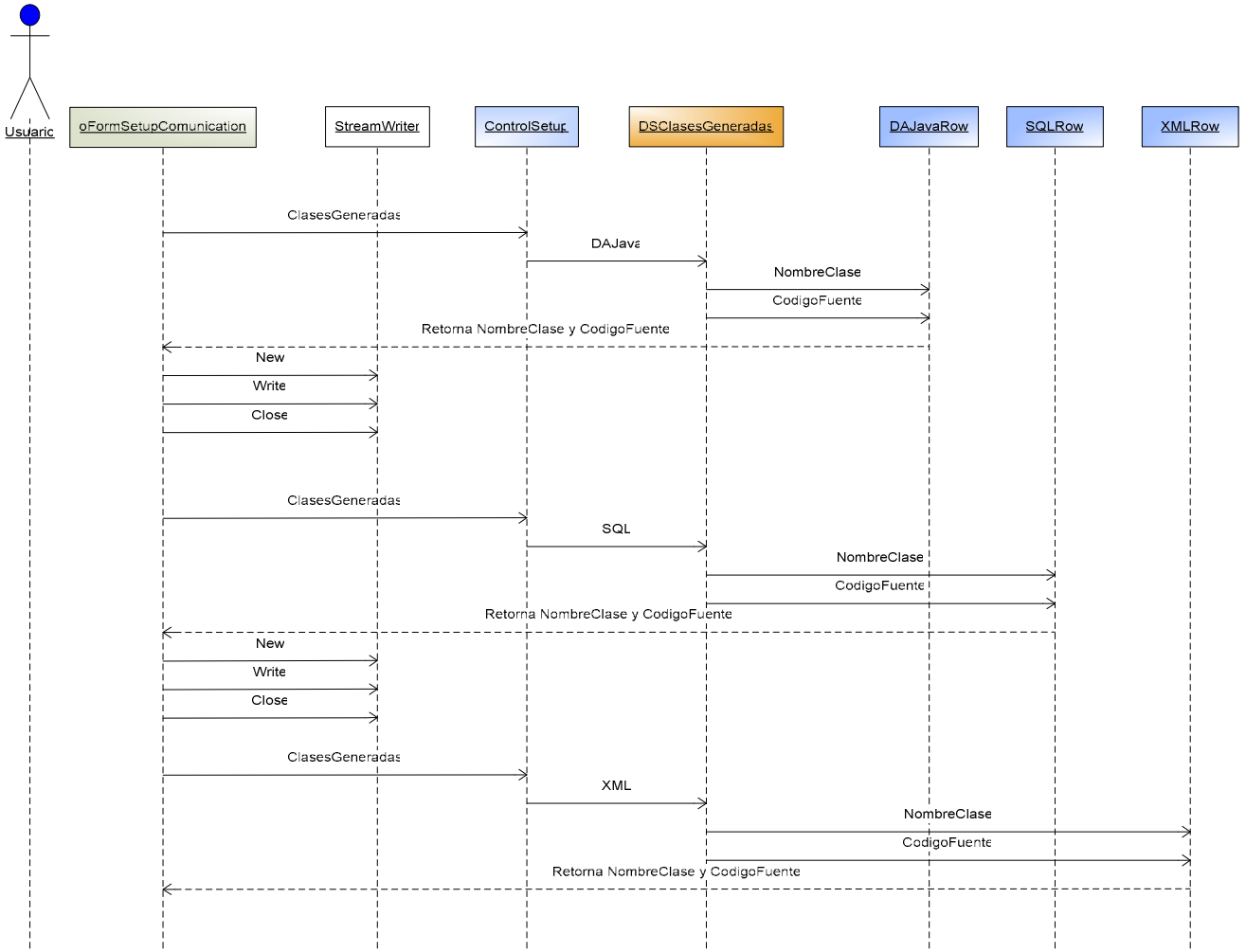


Figura 5.58 Diagrama de Secuencia del Caso de Uso Generar Código Fuente – Parte 3

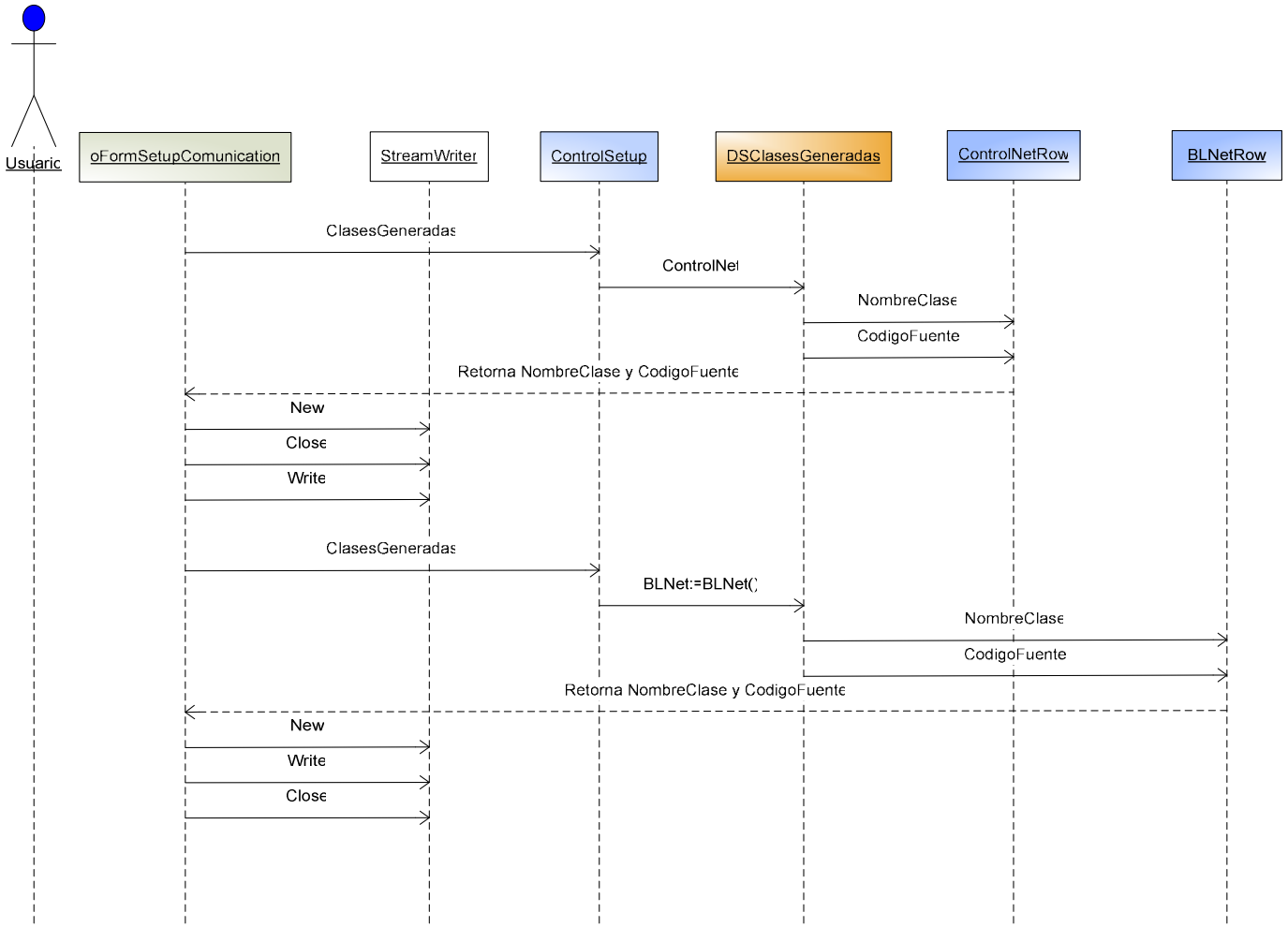
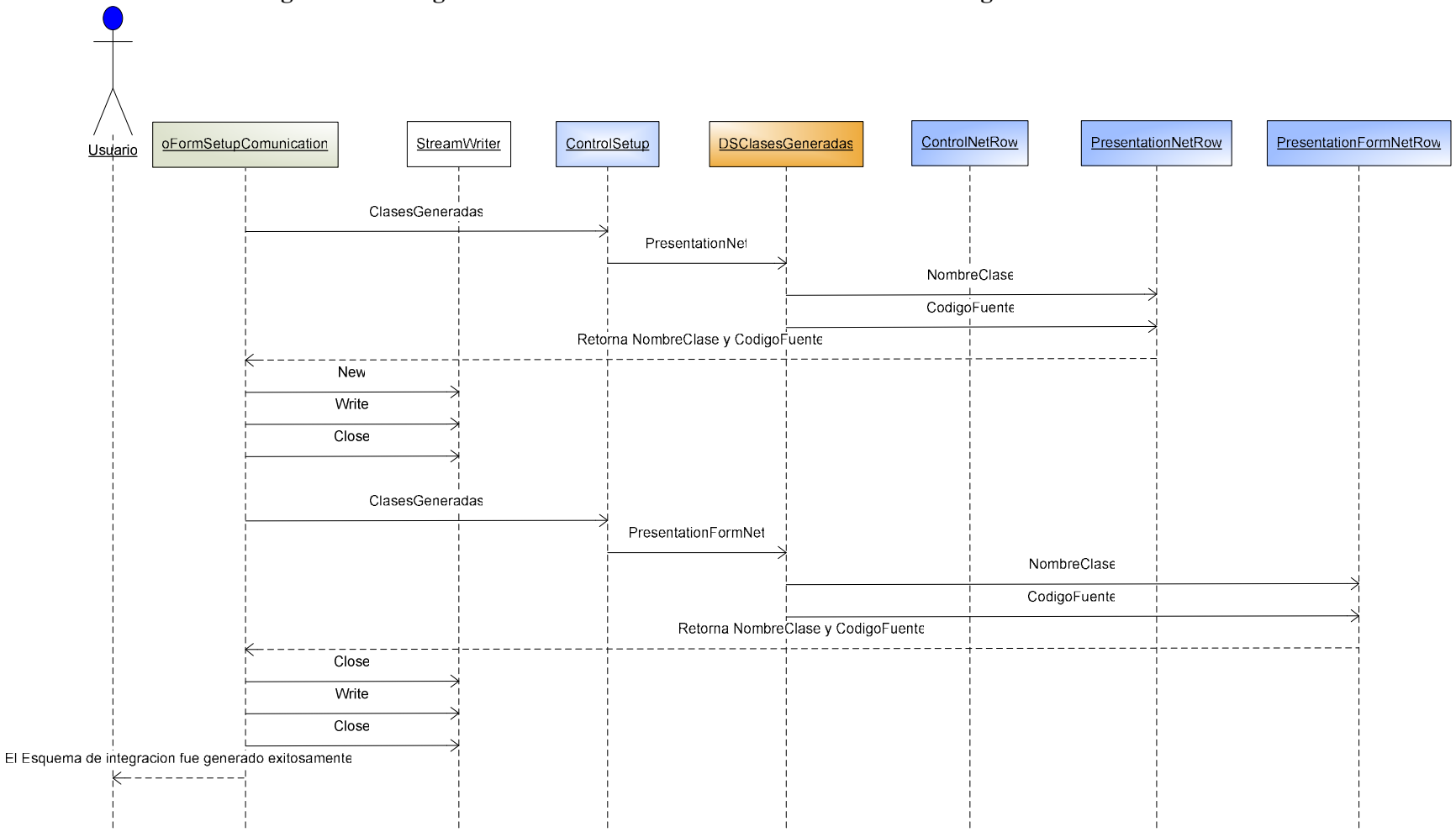


Figura 5.59 Diagrama de Secuencia del Caso de Uso Generar Código Fuente – Parte 4

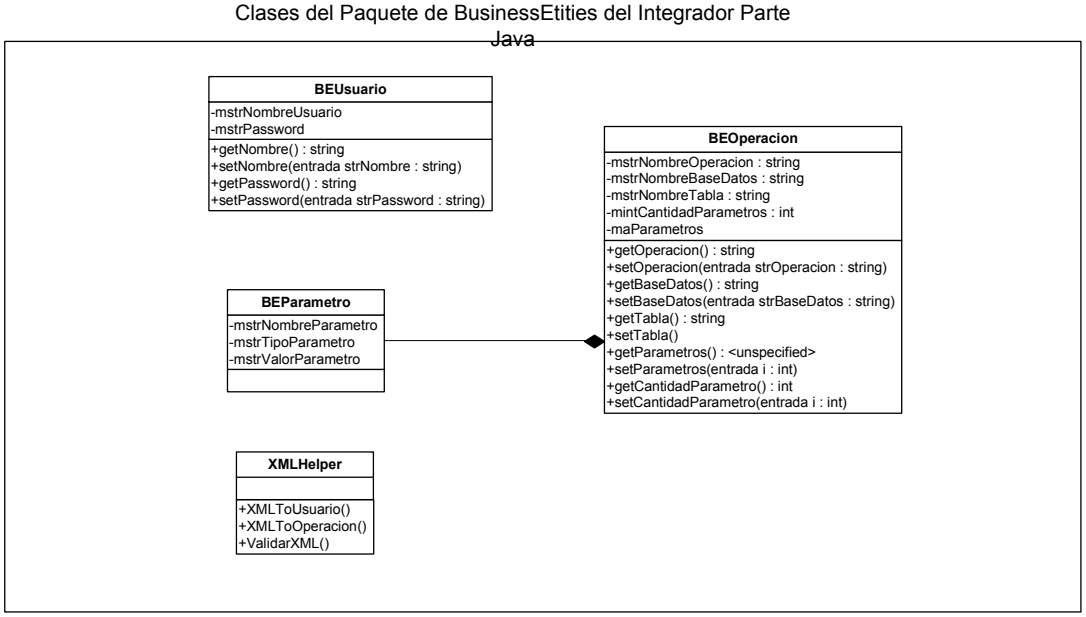


5.7.3 Clases y Patrones Implementados

Entre las principales características de la solución de integración se encuentran las entidades 'de integración, que hacen que se pueda replicar una estructura de datos en ambas plataformas. En .Net esta fue desarrollada mediante DataSets con Tipo, es decir, estructuras que nacieron a partir de esquemas XML y Bussines Entibies.

Por la parte del servidor de integración estas clases fueron replicadas mediante el uso de Java Beans, en ambas plataformas se implementó el patrón Indexer, para poder realizar composición de datos y asegurar de que estos repositorios de datos sean escalables y genéricos.

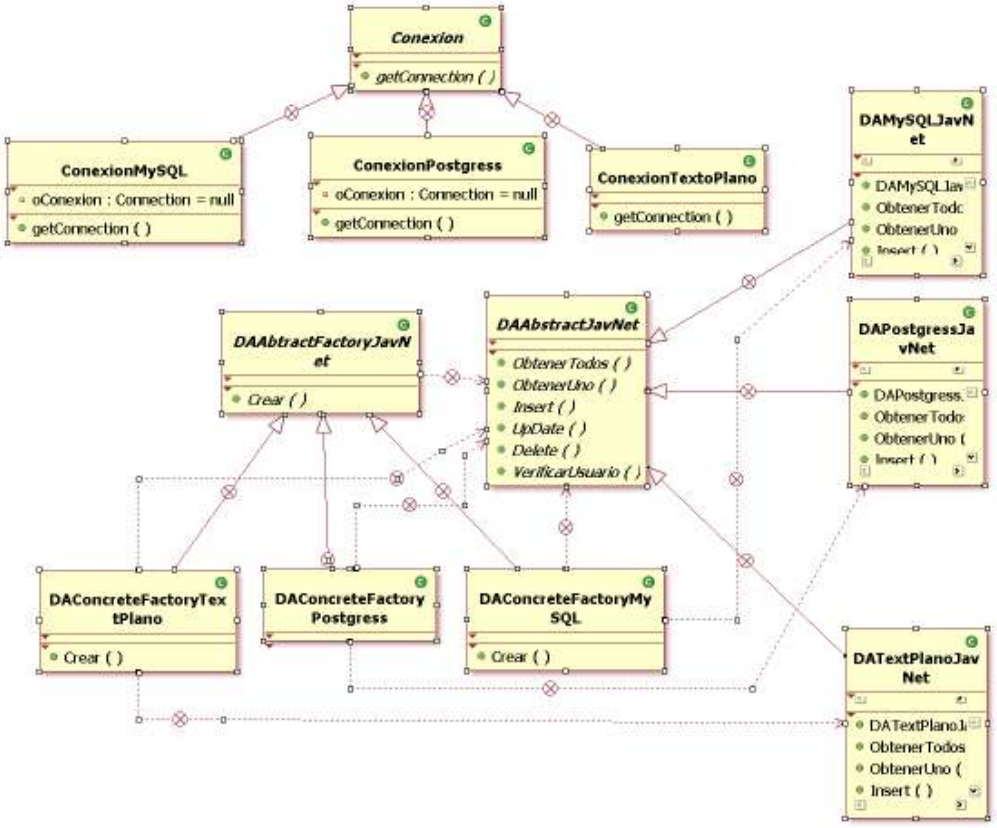
Figura 5.60 Entidades Replicadas en ambas plataformas



Por otro lado, se utilizó el patrón Abstract Factory, el cual es un patrón de diseño de Creación de objetos, consiste en abstraer las responsabilidades de crear objetos mediante una fábrica abstracta que crea objetos según como el cliente le pida. Esta clase fabrica abstracta redirecciona la responsabilidad a las clases fabricas concretas que heredan de ella, y así cada una de estas fábricas crean un tipo de producto determinando.

Para el caso del sistema integrador, se aprovechó las bondades de la implementación de este patrón para poder lograr que el sistema acepte múltiples orígenes de datos, de acuerdo a las necesidades de la empresa u organización, en este caso se realizaron pruebas con Bases de Datos como MySQL, PosgreSQL y Archivos de Texto Plano, como Fox Pro.

Figura 5.61 Patrón Abstract Factory implementado en componente DataAccess



5.8 Desarrollo Transición

5.8.1 Herramientas de Desarrollo

En la elección de herramientas a utilizar, como se pudo ver en la parte teórica y en los capítulos anteriores se utilizaron las siguientes herramientas y tecnologías

5.8.1.1 Visual Studio .Net

Para realizar la parte cliente del sistema, por su versatilidad, rendimiento, manejo seguro del entorno de ejecución a través del Net FrameWork, además que puede que la aplicación escale a diversos entornos de escritorio, web, móvil, Web Services, etc.

5.8.1.2 IBM Websphere

La potencia, versatilidad, agilidad y facilidad de uso hicieron que la herramienta de desarrollo IBM Websphere sea la indicada para desarrollar la aplicación en la parte Java.

5.8.1.3 Esquemas XML en el Módulo Cliente

Se utilizó esquemas para representar las entidades en la parte cliente y así construir el mensaje de datos que viaja a la parte servidor.

Dado que estos esquemas se construyeron en visual Studio .Net, la herramienta nos permite obtener objetos de tipo DataSet de un tipo en particular, esto hace más adecuado el desarrollo y la generación de código XML a partir de los esquemas para construir los mensajes que irán al servidor de datos.

5.8.1.4 Flujo de Bytes por medio de Sockets

Los sockets son un sistema de comunicación entre procesos de diferentes computadoras en una red. Más exactamente, un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información.

Se utiliza una serie de primitivas para establecer un punto de comunicación, para conectarse a una maquina remota, en un determinado puerto que esté disponible para escuchar en el, para leer, o para escribir y publicar información en él, y finalmente, desconectarse.

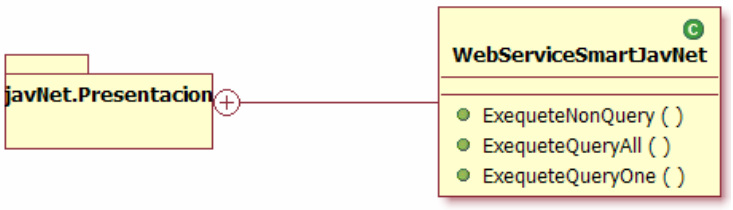
Para la construcción del integrador de información se utilizó este tipo de información entre .NET y Java, como es comunicación a bajo nivel, es bastante rápido, y es ideal para redes de área local.

5.8.1.5 Servicios Web entre Tecnologías

Se eligió a los servicios web para tener otra alternativa de comunicación, ya que es ágil y de fácil uso en escenarios de redes amplias utilizando Internet, esto abarata los costos, ya que está basada en información que es transportada por el protocolo HTTP, como una aplicación Web estándar.

Para el integrador de información se implementó un servicio Web operaciones genéricas, que recibe las peticiones de operaciones del "Conector", todas las peticiones de información, sea cual fuere el negocio que se esté integrando, no necesitan métodos particulares ya que el retorno de los métodos siempre será texto XML.

Figura 5.62 Clase del Servicio Web del integrador de información



5.8.1.6 Tratamiento de XML

La utilización de esquemas XML en ambos lados, tanto en el Conector, como en el servidor de integración facilita tener un único tipo de representación de los mensajes, se considera que una operación cualquiera que fuere (inserción, modificación, borrado, consulta) se contiene un conjunto de parámetros, cada parámetro un valor, nombre del parámetro y su tipo correspondiente.

También con las operaciones se puede enviar información del procedimiento, tabla o archivo a la que afectara la información, y nombre de la base de datos u origen de información.

En las figuras siguientes se pueden apreciar el esquema XML diseñado y también un ejemplo de un mensaje de ejecución de operación enviado desde el Conector hacia el Servidor de Integración.

Figura 5.63 Esquema XML de Representación de Mensajes

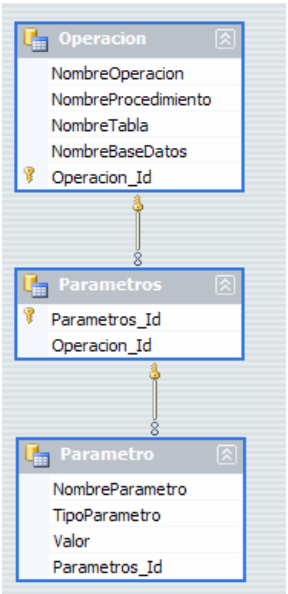


Figura 5.64 Mensaje XML enviado al servidor de integración

```
<?xml version="1.0" standalone="yes"?>
<MensajeCliente xmlns="http://tempuri.org/XMLPedidoCliente.xsd">
  <Usuario TipoUsuario="Administrador">
    <NombreUsuario>EdisonMuñoz</NombreUsuario>
    <PasswordUsuario>123456789</PasswordUsuario>
  </Usuario>
  <Operacion NombreOperacion="Insert" NombreProcedimiento="sp_insertarProducto"
             NombreTabla="Producto" NombreBaseDatos="BDJAVNET">
    <Parametros>
      <Parametro NombreParametro="PresentacionProducto" TipoParametro="String" Valor="caja Grande" />
      <Parametro NombreParametro="PrecioProduto" TipoParametro="String" Valor="10000" />
      <Parametro NombreParametro="StockMinimo" TipoParametro="int" Valor="150" />
      <Parametro NombreParametro="StockDisponible" TipoParametro="int" Valor="100" />
      <Parametro NombreParametro="NumeroSerie" TipoParametro="String" Valor="001" />
      <Parametro NombreParametro="IdLote" TipoParametro="String" Valor="1" />
      <Parametro NombreParametro="Estado" TipoParametro="String" Valor="1" />
    </Parametros>
  </Operacion>
</MensajeCliente>
```

5.8.1.7 Servidores de aplicaciones Websphere Application Server, JBOSS y Sun Java System Application Server

WAS se utilizó en un primer momento para realizar pruebas del sistema mientras estaba en desarrollo, pero el despliegue final fue realizado en JBoss 4.2 GA, y también en Sun Java System Application Server 9.0, ambos son servidores de software libres, con opción de licenciamiento con costo, potentes y portables.

Figura 5.65 Servidor de Aplicaciones J2EE JBoss

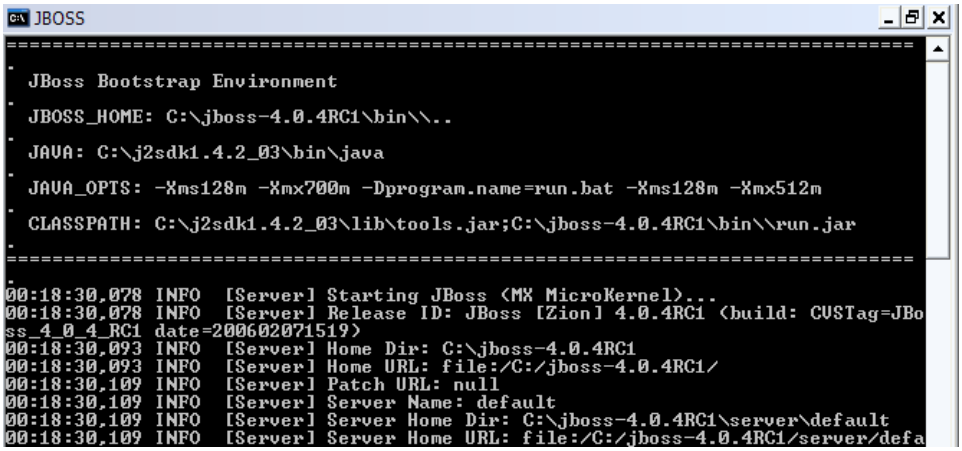
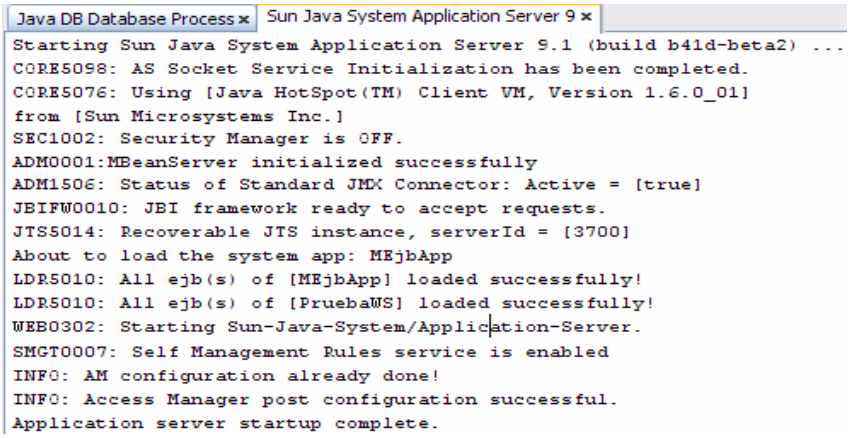


Figura 5.66 Servidor de Aplicaciones Sun Java System Application Server.



5.8.1.8 Conexiones a Base de Datos

Para los diversos orígenes de datos se descargaron los diferentes archivos JAR que sirven para conectarse y obtener data, menos en el caso de Texto plano que se tuvo que implementar una clase completa.

- mysql-connector-java-2.0.14-bin.jar
- postgresql.jar

5.8.1.9 Lectura y Escritura de Archivos Planos

- **Archivos de Separado fijo**

  La lectura de éstos archivos planos se realiza en forma secuencial, registro a registro, cuando se realiza una búsqueda éste recorrido se realiza hasta que se encuentre el registro requerido sabiendo de antemano como están estructurados los archivos (pueden estar separados sus campos por número de caracteres o carácter identificador de campo).

  En cuanto a la escritura mucho depende del archivo al que se accediendo, si este es un archivo de Fox Pro o Clipper la inserción es al final del archivo dejando marcado el fin del archivo.

  Estas operaciones son realizadas utilizando procedimientos y primitivas en java, que son los más adecuados.

  También se tiene que tener en cuenta de que se le debe dar acceso de lectura y escritura al repositorio donde se encuentren estos archivos, así como también los permisos de lectura y escritura.

- **Archivos DBF**

  Estos tipos de archivos no pueden ser leídos línea por línea, debido a que no tienen un fin de línea especifico, es necesario valerse de librerías especializadas como javdb.jar, esta librería tienen clases de acceso a archivos DBF bien formados, por medio del cual se pueden leer secuencialmente en memoria los registros del archivo, esta librería tiene licencia GNU, su distribución y uso es gratuita, se puede y se puede descargar del sitio http://sarovar.org/projects/javadbf/.

  Para la escritura también se realiza mediante la librería javdb.jar este agrega un registro al último y guarda automáticamente el fin del archivo.

  - javadbf-0[1].4.0.jar

5.9 Implementación

La implementación de la aplicación de integración tiene las siguientes etapas:

En primer lugar se tiene que estudiar el negocio, es decir, el escenario al cual se le dará solución, esto implica, saber que tipos de orígenes de data se desea integrar y qué tipo de comunicación se adecúa más a sus necesidades.

El aspecto más importante es saber qué archivos y tablas serán los que se desean integrar/migrar con la aplicación, esto involucra conocer también que sistemas alimentarán la información de estas entidades (tablas/archivos), así se puede completar totalmente el proceso de integración.

5.9.1 Generación de Código

El hecho de que el integrador se adapte a diversos escenarios de negocio, significa que se puede representar en las entidades con las que trabajan los sistemas heredados, así se puede generar código fuente tanto en un lenguaje bajo la plataforma .net como en lenguaje java y los lenguajes C# de .Net.

La siguiente secuencia de Figuras, muestra el funcionamiento intuitivo de la herramienta de configuración que guía paso a paso al usuario para que pueda configurar y generar el código apropiado para su integración de información.

Figura 5.67 Pantalla Principal del Módulo de Configuración

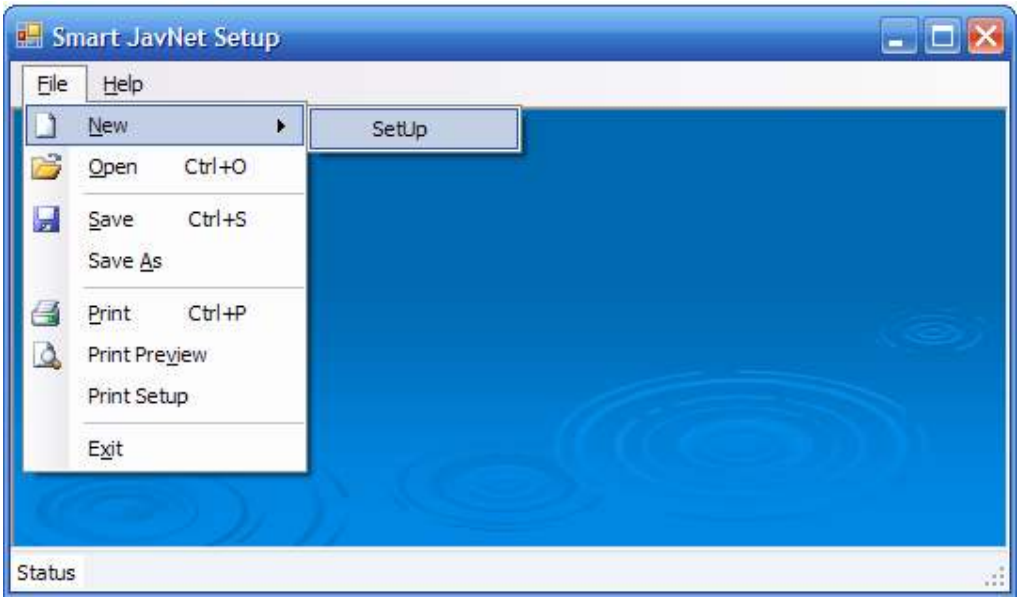


Figura 5.68 Pantalla de configuración de Entidades

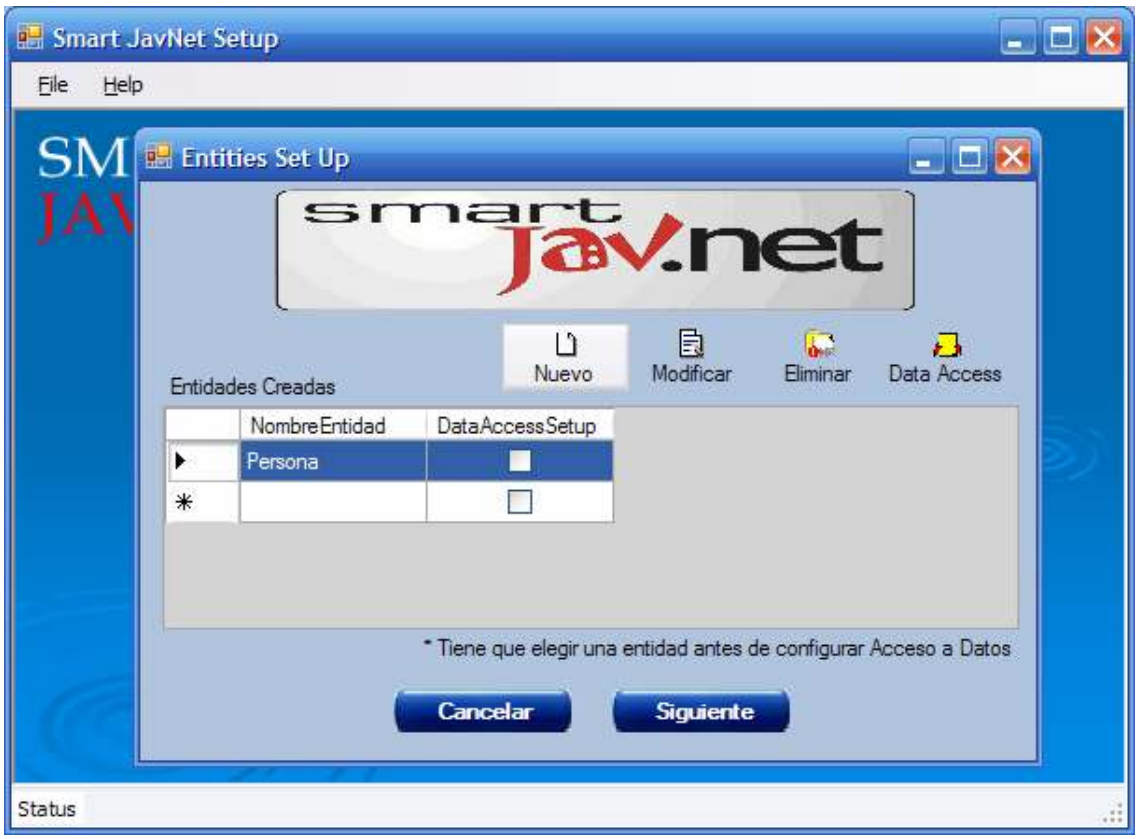


Figura 5.69 Pantalla para configurar una Entidad

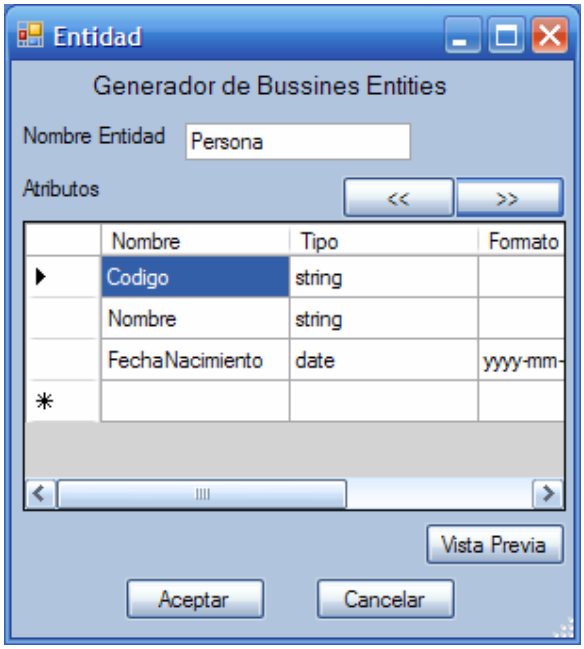


Figura 5.70 Pantalla para configurar un Atributo de Entidad

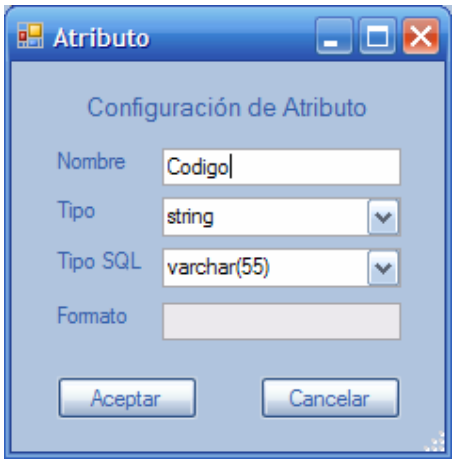


Figura 5.71 Vista Previa del Código en C#.NET

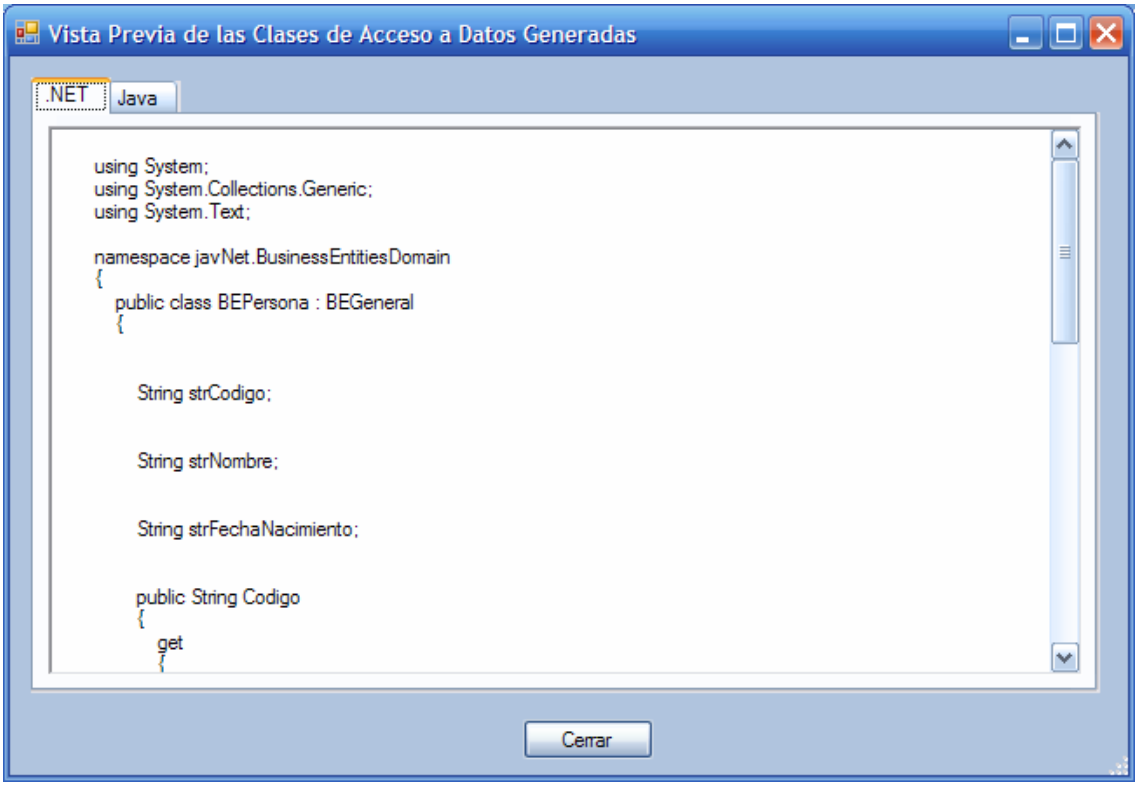


Figura 5.72 Vista Previa del Código en Java

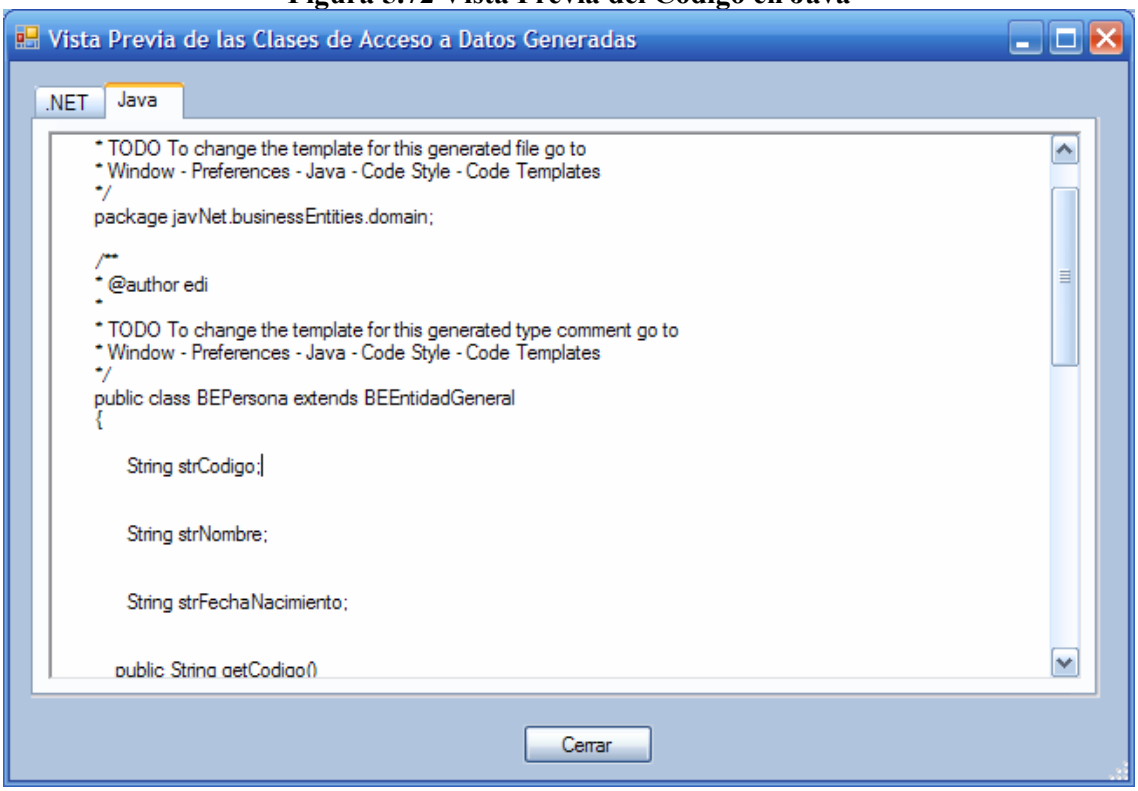


Figura 5.73 Pantalla para configurar el acceso a datos en General

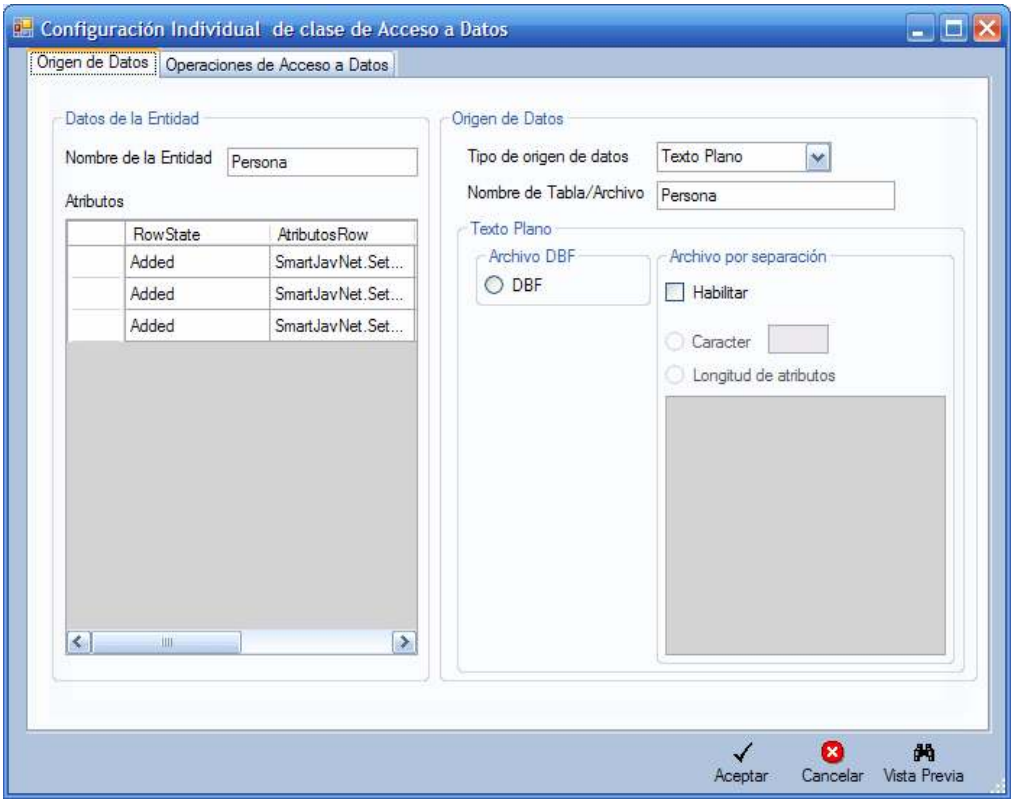


Figura 5.74 Pantalla para configurar los Atributos de las operaciones

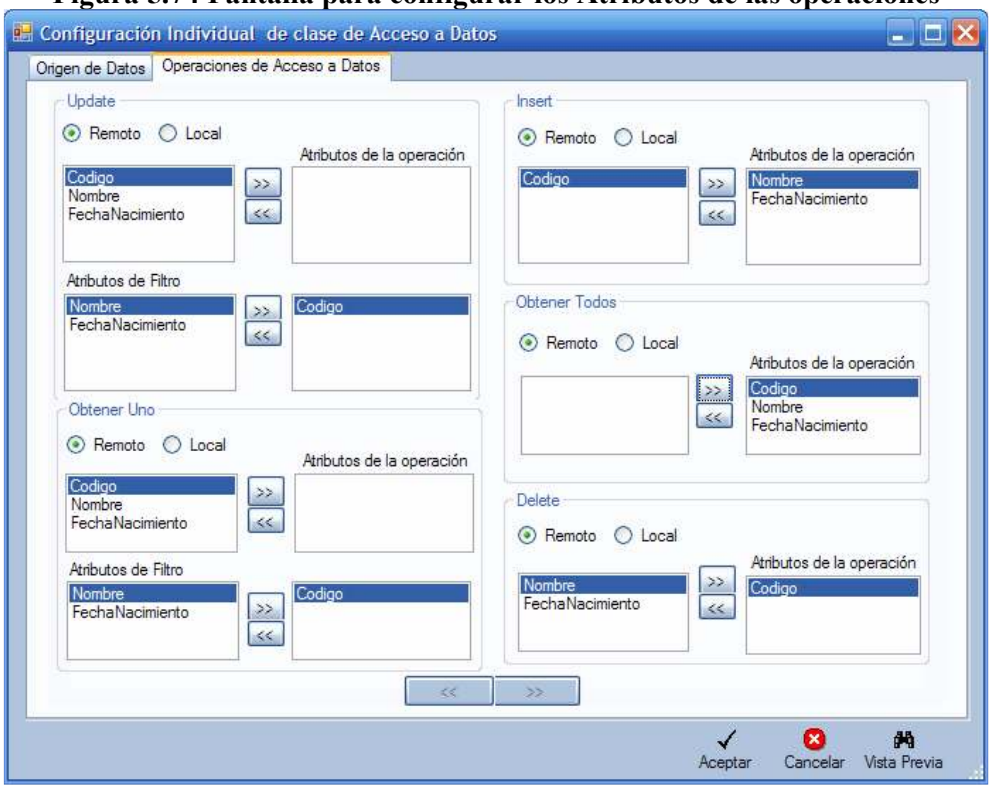


Figura 5.75 Pantalla de Generación de la Clase que Genera XML

Figura 5.76 Pantalla de Vista previa de Clase que genera XML en Java

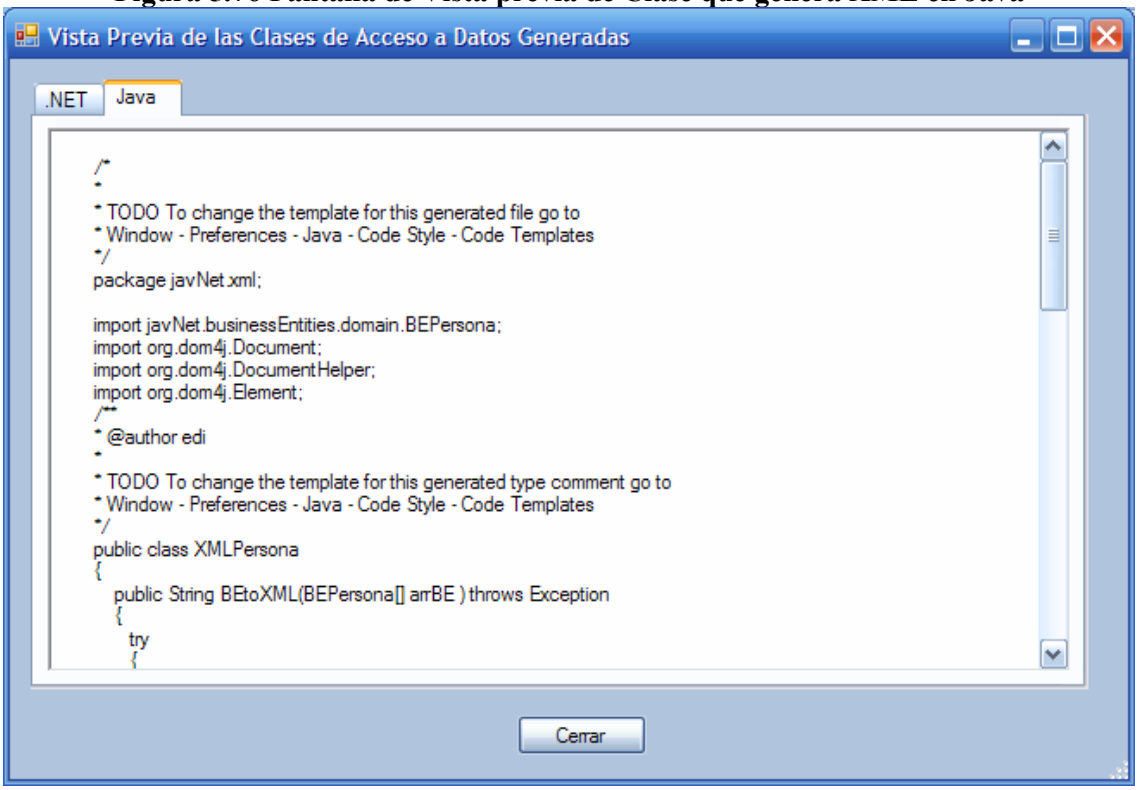


Figura 5.77 Pantalla Resumen de la Configuración de las Clases Generadas

Figura 5.78 Ventana de Dialogo de Generación de Clases

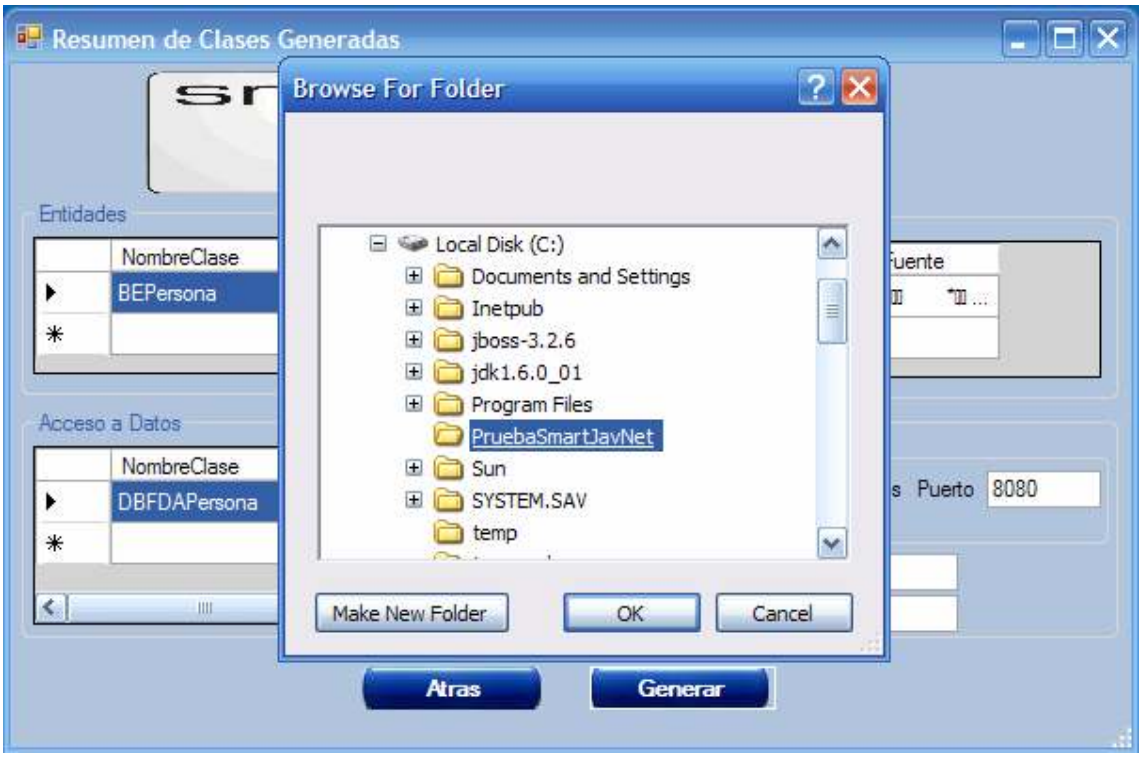


Figura 5.79 Aviso de Generación del Esquema de integración

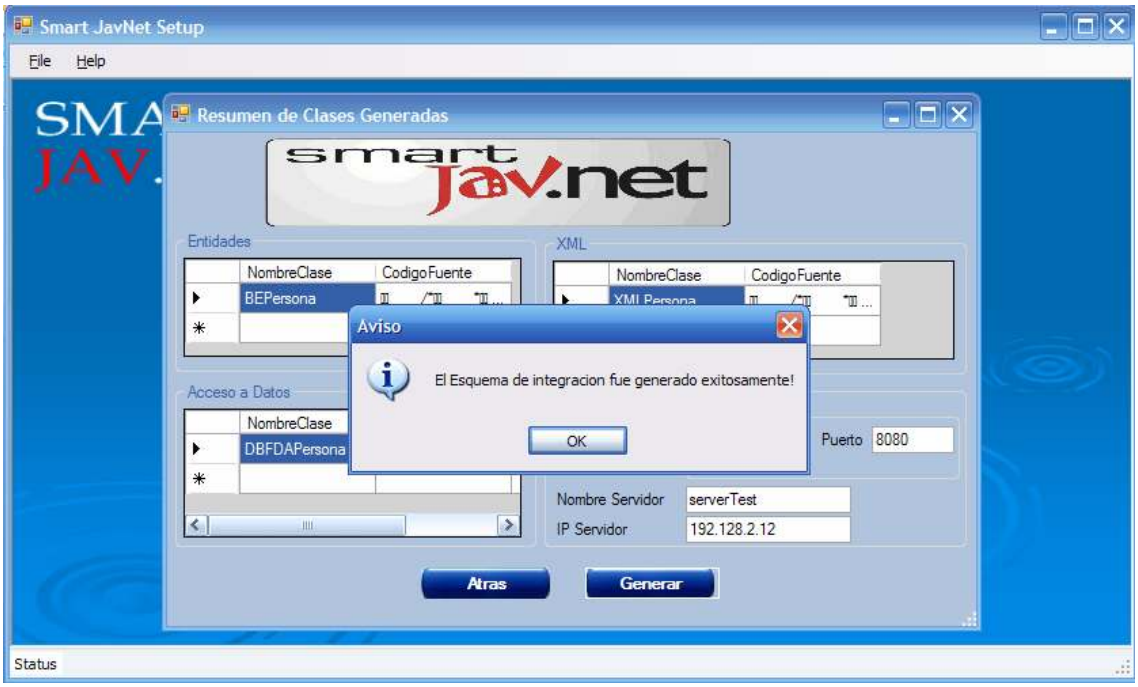


Figura 5.80 Lista de Archivos Generados

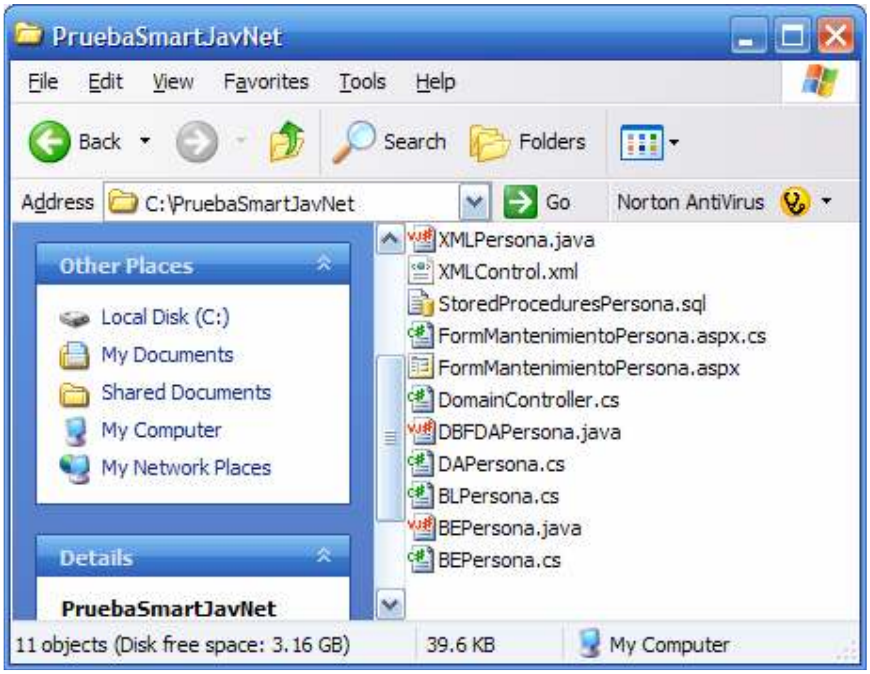


Figura 5.81 Clase Generadora de XML Generada

```
package javNet.xml;

import javNet.businessEntities.domain.BEPersona;
import org.dom4j.Document;
import org.dom4j.DocumentHelper;
import org.dom4j.Element;
/**
* @author edi
*
* TODO To change the template for this generated type comment go to
* Window - Preferences - Java - Code Style - Code Templates
*/
public class XMLPersona
{
    public String BEtoXML(BEPersona[] arrBE ) throws Exception
    {
        try
        {
            Document document = DocumentHelper.createDocument();
            Element root = document.addElement( "DSPersona" );

            for ( int i =0 ; i <  arrBE.length;i++)
            {
                Element oElement = root.addElement( "Persona" );
```

Figura 5.82 Archivo XMLControl.xml generado

```
<?xml version="1.0" standalone="yes"?>
<Configuraciones xmlns="http://tempuri.org/Configuraciones.xsd">
  <Tabla nombreTabla="Persona">
    <Insert>Remoto</Insert>
    <UpDate>Remoto</UpDate>
    <Delete>Remoto</Delete>
    <ObtenerTodos>Remoto</ObtenerTodos>
    <ObtenerUno>Remoto</ObtenerUno>
  </Tabla>
  <Comunicacion>
    <Tipocomunicacion>WebService</Tipocomunicacion>
    <Puerto>8080</Puerto>
    <NombreServidor>serverTest</NombreServidor>
    <IPServidor>192.128.2.12</IPServidor>
    <Enable>true</Enable>
  </Comunicacion>
</Configuraciones>
```

Figura 5.83 Clase Entidad Generada en C#

```
using System;
using System.Collections.Generic;
using System.Text;

namespace javNet.BusinessEntitiesDomain
{
    public class BEPersona : BEGeneral
    {
        String strCodigo;

        String strNombre;

        String strFechaNacimiento;

        public String Codigo
        {
            get
            {
              return strCodigo;
            }
```

Figura 5.84 Clase Entidad Generada en Java

```
package javNet.businessEntities.domain;
/**
 * @author edi
 *
 * TODO To change the template for this generated ty
 * Window - Preferences - Java - Code Style - Code T
 */
public class BEPersona extends BEEntidadGeneral
{
      String strCodigo;

      String strNombre;

      String strFechaNacimiento;

    public String getCodigo()
    {
        return strCodigo;
    }
    public String getNombre()
    {
        return strNombre;
    }
```

Figura 5.85 Clase de Acceso a Datos Generada en Java

```
package javNet.dataAccess.textoPlanoDBF;
import ...

public class DBFDAPersona implements IRetrieveOperation
{
    public String  ObtenerTodos(Conexion oConexion , BEOperacion oBEOperacion)
    {
        try
        {
            InputStream inputStream =   oConexion.getConnection( oBEOperacion.Get

            DBFReader reader = new DBFReader( inputStream);
            System.out.println("DBFReader reader = new DBFReader( inputStream)");

            ArrayList arrlist = new ArrayList();

            Object []rowObjects;
            BEPersona oBE;

            int intContAux=0;

            try
            {
                FormatoFecha oFormatoFecha;
```

Figura 5.86 Clase de Acceso a Datos Generada en C#.NET

```
using ...
namespace javNet.DataAccessDomain
{
 public class DAPersona : DAGeneral
 {

    public DAPersona()
    {
    }

    public override int Insert(BEGeneral oBEGeneral, BEUsuarioJavNet oBEUsuarioJavNet)
    {
        BEPersona oBE = (BEPersona)oBEGeneral;

        int i = 0;

        if (!ControladorIntegracion.EjecucionRemota("Persona", "Insert"))
        {
              i = 1;
        }
        else
        {
              BEOperacion oBEOperacion = new BEOperacion();
```

Figura 5.87 Clase de Lógica de Negocio generada

```
using System;
using System.Collections.Generic;
using System.Text;
using javNet.BusinessEntitiesDomain;
using javNet.BusinessEntitiesNet;
using javNet.DataAccessDomain;

namespace javNet.BusinessLogicDomain
{
    public class BLPersona : BLGeneral
    {
        public BLPersona()
            : base()
        {
            oDAGeneral = new DAPersona();
        }
        public override int Nuevo(BEGeneral oBE, BEUsuarioJavNet oBEUsuarioJavNet)
        {
            try
            {
                return oDAGeneral.Insert(oBE, oBEUsuarioJavNet);
            }
            catch (Exception ex)
            {
```

Figura 5.88 Clase DomainController generada

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using javNet.BusinessEntitiesNet;
using javNet.BusinessEntitiesDomain;
using javNet.BusinessLogicDomain;
using javNet.DataAccessDomain;

namespace javNet.DomainController
{
  public class DomainController
  {
      public static int MantenerPersona(BEPersona oBE , BEUsuarioJavNet oBEUsuarioJavNet,
                                                        EnumTipoRegistro oEnumTipoRegistro  )
      {
          try
          {
              BLPersona oBGeneral = new  BLPersona();
                  switch( oEnumTipoRegistro)
                  {
                       case EnumTipoRegistro.Nuevo:
                          return oBGeneral.Nuevo(oBE,oBEUsuarioJavNet);

                      case EnumTipoRegistro.Modificar:
                          return oBGeneral.Actualizar(oBE,oBEUsuarioJavNet);

                      case EnumTipoRegistro.Eliminar:
                          return oBGeneral.Borrar(oBE,oBEUsuarioJavNet);
```

Figura 5.89 Formulario de Mantenimiento ASPX Generado

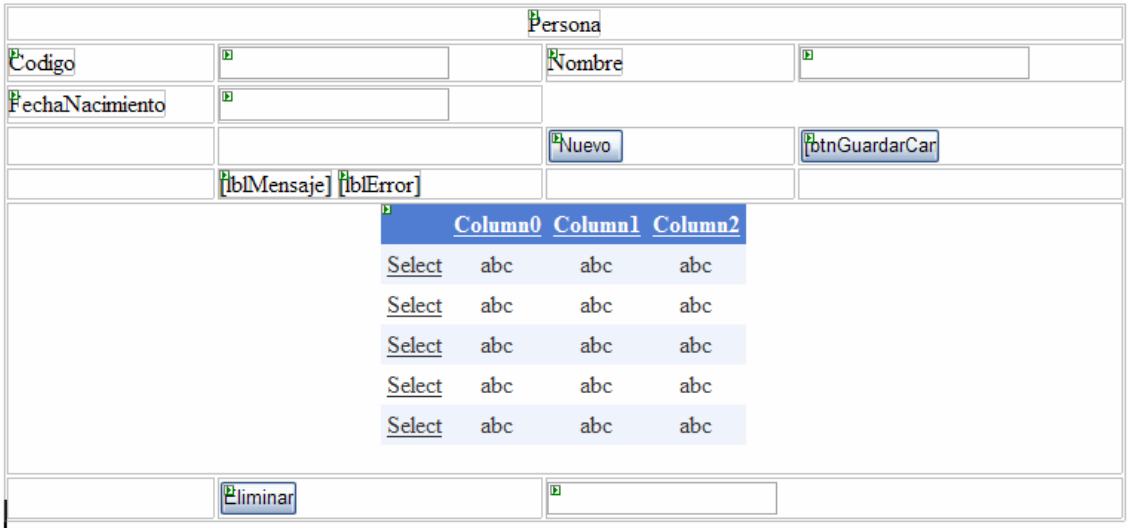


Figura 5.90 Código detrás del Formulario de Formulario de Mantenimiento Generado

```
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using javNet.BusinessEntitiesDomain;
using javNet.BusinessEntitiesNet;
using javNet.DomainController;

public partial class FormMantenimientoPersona : System.Web.UI.Page
{
    BEUsuarioJavNet oBEUsuarioJavNet = new BEUsuarioJavNet();

    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            if (!Page.IsPostBack)
            {
                oBEUsuarioJavNet.NombreUsuario = "Admin";
                oBEUsuarioJavNet.PasswordUsuario = "p@ssw0rd";
                oBEUsuarioJavNet.TipoUsuario = "Administrador";

                Session["oBEUsuarioJavNet"] = oBEUsuarioJavNet;

                dtgDatos.DataSource =
                    DomainController.ObtenerTodosPersonas(oBEUsuarioJavNet);

                dtgDatos.DataBind();
```

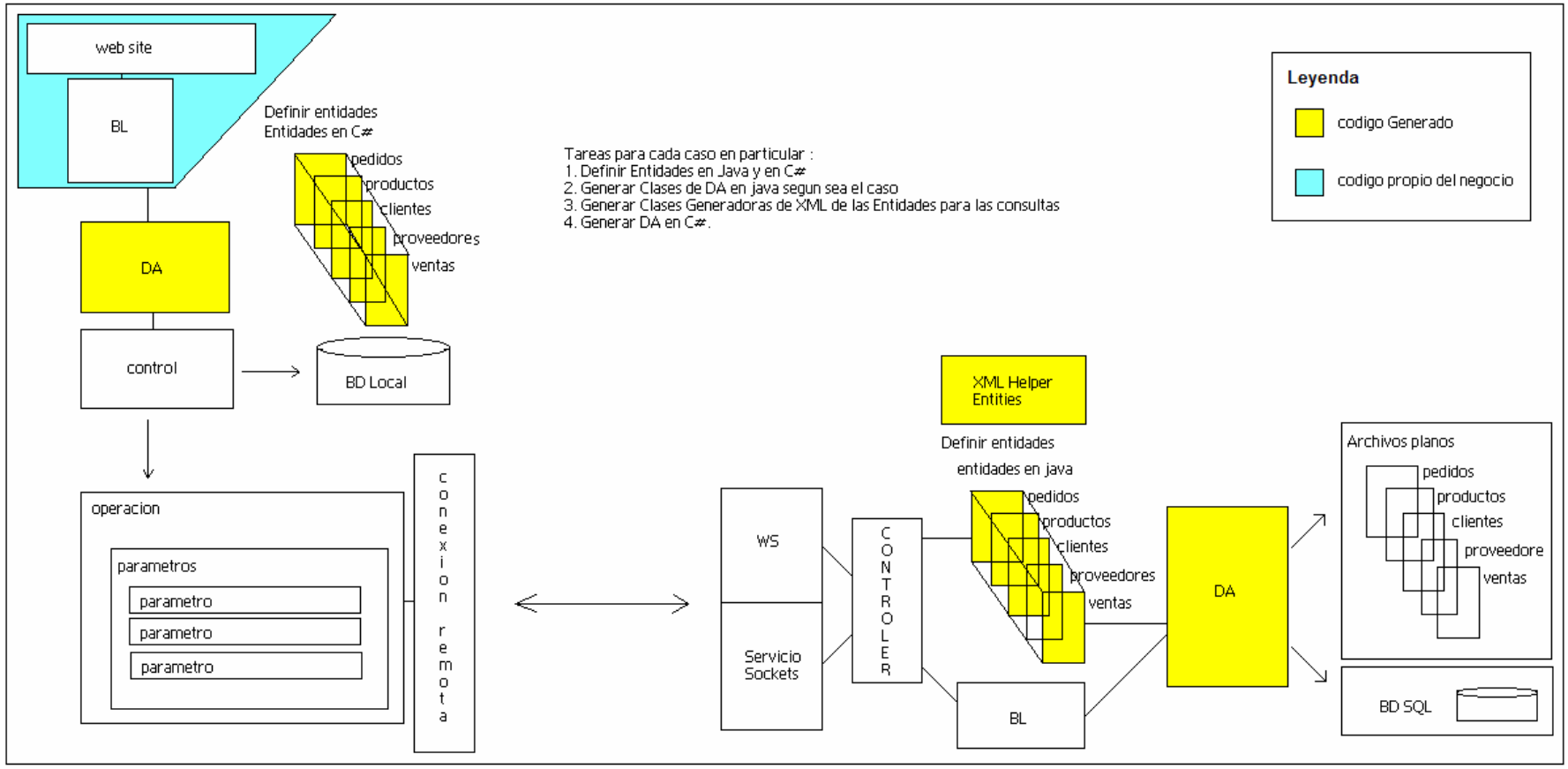
Figura 5.91 Tabla y Procedimientos Almacenados de Mantenimiento Generados

```
create table Persona
(
    Codigo varchar(55) ,
    Nombre varchar(55) ,
    FechaNacimiento varchar(55)
)

create procedure sp_deletePersona
      @Codigo varchar(55)
as
    Delete Persona  where
      Codigo = @Codigo
go
create procedure sp_insertPersona
      @Nombre varchar(55),
      @FechaNacimiento varchar(55)
as
    insert into  Persona (Nombre, FechaNacimiento)
    Values    (@Nombre,@FechaNacimiento)
go
create procedure sp_updatePersona
      @Codigo varchar(55),
      @Nombre varchar(55),
      @FechaNacimiento varchar(55)
as
    update  Persona set
                Codigo = @Codigo ,
                Nombre = @Nombre ,
                FechaNacimiento = @FechaNacimiento
    where
          Codigo = @Codigo
```

Figura 5.92 Procedimientos de Consulta y migración

```
create procedure sp_ObtenerTodosPersona
as
    select Codigo as  Codigo, Nombre as  Nombre, FechaNacimiento as  FechaNacimiento
    From  Persona
go

create procedure sp_ObtenerUnoPersona
        @Codigo varchar(55)
as
     select
        Codigo as  Codigo, Nombre as  Nombre, FechaNacimiento as  FechaNacimiento
    From  Persona
    where
          Codigo = @Codigo
go

create proc insertXMLPersona
      @docXML ntext
as
declare @Idocument int
Exec sp_xml_preparedocument @Idocument OUTPUT, @docXML

INSERT INTO  [Persona]
select *
from  OPENXML(@Idocument, '/DSPersona/Persona',1)
        WITH ( Codigo varchar(55) ,
               Nombre varchar(55) ,
               FechaNacimiento DateTime
              )
```

Figura 5.93 Ejemplo de un escenario para generación de código para integración de información



5.9.2 Compilación

Una vez generadas las clases, éstas deben ser compiladas para su despliegue y se generan las siguientes clases:

Tabla 5.7 Tablas de Generación y Compilación de código

	Tecnología	.NET	Entorno:	.Net Framework 2.0
Nro.	Componentes	Clases	Lenguaje	Descripción
1	Business Entities	Entidades de Negocio.	C#	Las entidades son clases que representan a unidades del origen de datos como son tablas o archivos
2	Data Access	Clases de Acceso a datos	C#	Clases que se encarga de interactuar con el integrador para realizar operaciones de acceso a datos.
3	Bussines Logic	Clases de Acceso a datos	C#	Clases que se encarga de interactuar con la capa de acceso a datos para realizar operaciones de acceso a datos.
4	DomainController	Clases de Acceso a datos	C#	Clases que agrupa y concentra las llamadas a las clases BL y DA.
5	UI –HTML UI –Code Behind	Clases de Acceso a datos	C#	Clases que se encarga de recibir la interacción directamente con el usuario.

	Tecnología	J2EE	Entorno:	Java Virtual Machine
Nro.	Componentes	Clases	Lenguaje	Descripción
1	XML	Clases generadoras e intérpretes de XML	Java	Estas clases se encargan de interpretar todos los mensajes que vienen del cliente y a partir de ellos crear objetos entidades.
2	Business Entities	Entidades de Negocio.	java	Clase similares a las de la tabla anterior que representan a unidades del origen de datos como son tablas o archivos, pero en el servidor de integración.
3	Data Access	Clases de Acceso a datos	java	Clases que se encarga de interactuar directamente con los orígenes de datos, según hayan sido configurados cada uno de ellos.

5.9.3 Generalización de las Consultas

Otro tipo de código generado también son las consultas para los orígenes de datos, estas estructuras pueden ser generadas en lenguaje SQL o estructuras propias para consultar archivos planos de acuerdo al tipo de archivo que este sea (Fox Pro, Clipper,etc).

Los tipos de consulta generados son del tipo:

- ObtenerUno

  Este tipo de consulta obtiene un registro de la tabla o archivo plano con una condición que es el campo un clave genérico.

- ObtenerTodos

  Este tipo de consulta obtiene todos los registros de la tabla o archivo plano sin ninguna condición.

5.10 Despliegue

5.10.1 Despliegue de Componentes

El despliegue de componentes de los diversos módulos y aplicaciones del integrador de información va a depender de la tecnología en que fueron desarrollados. A continuación se detallará cada uno con sus particularidades:

5.10.1.1 Conector Cliente de Integración de Datos (.NET)

Esta aplicación puede ser desplegada mediante un servicio de Windows que tenga interface hacia las aplicaciones clientes (Windows, móvil, web) mediante .Net Remoting o Web services dependiendo de las necesidades del usuario.

La otra alternativa es que sea parte de las aplicaciones clientes, y se haga uso de su funcionalidad en forma directa.

5.10.1.2 Servidor de Integración de Datos

Esta aplicación java, puede ser desplegada de dos formas:

Por medio de una interface de Web services, o por medio de una aplicación de comunicación, por medio de sockets, en ambos casos, los demás componentes que conforman esta aplicación serán desplegados como componentes .JAR en el servidor de aplicaciones (WAS, JBoss, Java Sun System Application Server).

5.10.2 Configuraciones Generales

5.10.2.1 Archivo de configuración en aplicación cliente

Dirección de Web Services
Configuración de la dirección IP donde está instalado el servidor del conector cliente de integración .Net.

5.10.2.2 Archivo de configuración en conector cliente de integración .Net

Dirección de comunicación
Configuración de la dirección IP y el puerto en con el cual se realizará la comunicación ya sea vía Sockets, Web Services, y Mensajería de colas.

Configuración por cada tabla y las operaciones
Por cada tabla o archivo plano se realizara las configuraciones de las los orígenes de datos, las operaciones, y los nombres de procedimientos si es que existieran.

5.10.2.3 Archivo de configuración en servidor de integración de Datos

Configuración de Orígenes de datos
Se configura si se hará de uso de acceso dinámico a diferentes orígenes de datos, o solo se realizará acceso a un solo origen.
También se configura el usuario y el password para validar cada operación contra el origen de datos.

Capítulo VI

Estudio de Caso de Aplicación del Integrador de Información Desarrollado

6.1 Organización elegida

La Organización elegida para el caso de aplicación, es la empresa CTM Tours del grupo Costamar, ubicada en la avenida Paz Soldán 225 B-14 en el distrito de San Isidro - Lima. Esta empresa transnacional privada tiene entre sus principales actividades, la venta de paquetes turísticos, boletos de cruceros, estadía en hoteles, alquiler de autos tanto para público en general así como para las agencias de viajes.

6.2 Problemática de la Organización Referente a Integración de Información

CTM Tours usa un sistema construido en Visual Fox Pro llamado Pro-Tour el cual se encarga de administrar los paquetes turísticos, llevar el control de stock de dichos paquetes, también lleva un registro de las agencias, restaurantes, hoteles y compañías de alquiler de autos, es decir, lleva el control de toda la información que es de importancia para la empresa.

El problema radica en que dicho sistema si bien fue muy útil en su tiempo, ahora ha quedado obsoleto, pero sigue siendo utilizado debido a su importancia para el negocio, pero su performance es bastante lento, sus consultas demoran bastante tiempo y por su arquitectura toda la información está basada en archivos planos DBF, y sus limitaciones son evidentes.

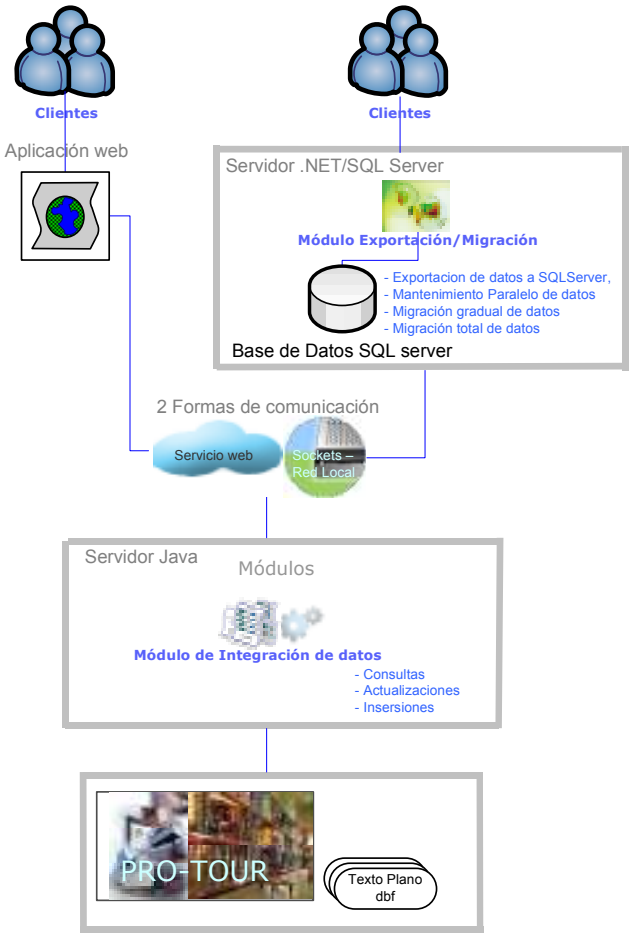
6.2.1 Aplicación del Integrador en el Caso de Estudio

La implementación del proyecto contempla los distintos aspectos necesarios para que la solución sirva como herramienta estratégica para integrar la información del Pro-Tour con otros entornos, además que debe cumplir las exigencias técnicas que le permitan convertirse en fuente de información, confiable, escalable y segura. Estos aspectos son:

1. Módulo de Integración de datos.
2. Módulo de Exportación de datos.
3. La arquitectura de comunicaciones necesarias para implantar la solución.

En la Figura siguiente se describe el modelo arquitectónico de la solución:

Figura 6.1 Modelo de Arquitectura Distribuida propuesta del caso de estudio



El modelo de arquitectura distribuida plantea la implantación de los módulos en 2 esquemas, uno para la integración de datos directamente del sistema Pro-tour, y el segundo, que es la exportación/migración de datos, en otros administradores de base de datos como SQL Server.

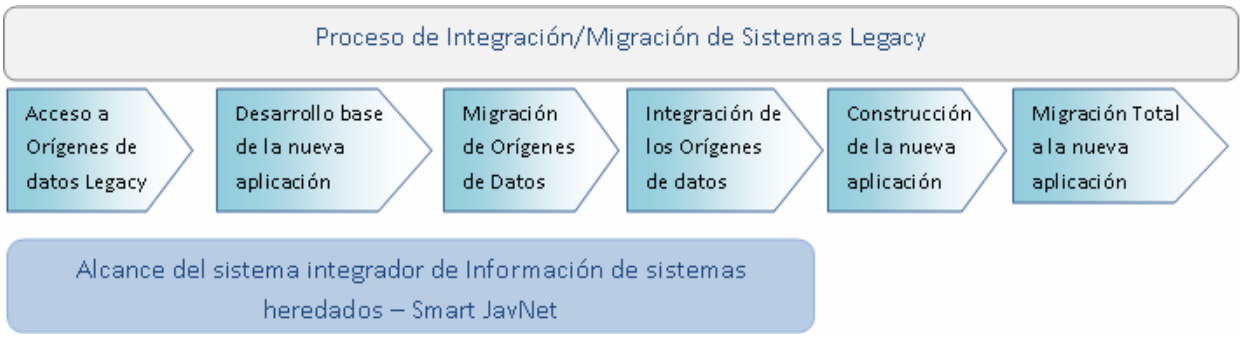
La integración de información del sistema pro tour con el sistema integrador propuesto, incluye los siguientes procesos:

- Acceso a los archivos DBF, mediante la construcción de clases de acceso (lectura y escritura,)
- Desarrollo base de la nueva aplicación en su primera etapa (Mantenimientos de los Archivos DBF)
- Migración de los Archivos DBF a una base de datos SQL Server
- Integración de los Archivos DBF mediante una interfaz Web, también generada automáticamente.

En base a estos elementos, y la arquitectura de integración propuesta se dejara lista a libre decisión del usuario, si desea seguir construyendo una nueva

aplicación para más adelante migrar definitivamente el sistema heredado o legacy, o sino mantener la integración y seguir usando el sistema heredado o legacy pero con la ventaja de poder explotar los archivos en un ambiente Web, en la figura siguiente se muestra, los procesos de integración de sistemas heredados o legacy, y cuál es el alcance que cubre el integrador de información propuesto en la presente tesis:

Figura 6.2 Alcance de la aplicación integradora, dentro del proceso de Integración de sistemas Heredados o Legacy



6.2.2 Despliegue de la aplicación

6.2.2.1 Módulo de integración de datos

El módulo de integración de datos debe utilizar un servidor Windows 2003 Server o en su defecto una maquina con Windows XP, con acceso al servidor de base datos que contiene a la aplicación del Pro-tour, este acceso es de escritura y lectura de los archivos de Texto Plano.

Se debe instalar un servidor de aplicaciones JBoss o Sun Application Server de Java, y en él se ejecutarán los archivos del integrador que accederán a los archivos de Texto Plano todo el tiempo que el servidor este ejecutándose.

6.2.2.2 Módulo de exportación/migración

El módulo de integración de datos también debe ser instalado en el mismo servidor Windows 2003 Server o en su defecto Windows XP, este módulo desarrollado íntegramente bajo la plataforma .NET necesita del .Net Framework 2.0, para que pueda ejecutarse, así que este necesita ser instalado. Este módulo tiene comunicación a través de un servicio Web con el módulo anterior, así que si más adelante se decide mudar este a otra estación o servidor la comunicación seria idéntica y no se alteraría.

6.2.3 Pruebas realizadas

Las pruebas de la aplicación del integrador se realizaron tomando como muestras los archivos de DBF del sistema Pro-tour, también es preciso mencionar que para efecto de estas pruebas se simulo el escenario de la distribución de la aplicación en una computadora portátil, instalando se software necesario.

Los resultados fueron los siguientes:

De los 295 Archivos que maneja el sistema Pro-tour se pudieron leer 266, y 29 no se pudieron leer debidos a que los archivos estaban mal formados, o tenían errores en su definición, este resultado se puede apreciar en las tablas, mostradas en el Anexo: "Tablas de Archivos de las Pruebas realizadas en el Caso de Estudio de la empresa CTM Tours".

6.2.4 Resultado de las Pruebas realizadas

Se leyeron desde el integrador sistema integrador parte Java leyendo uno a uno todos los archivos, para ver que si es factible la integración de estos archivos, luego se identifico un grupo de archivos muestra, para poder crear todo su proceso completo (creación de clases de acceso, capas de acceso a datos, lógica, controladora, capa de presentación, tabla en SQL Server, y procedimientos almacenados), y su visualización en un entorno web.

Respecto a los archivos que no pudieron ser leídos, no tienen importancia significativa por el cantidad que representan, sería necesario recomponer estos archivos desde el sistema Pro-Tour, ya que no están bien formados, estos archivos afectan al experimento en la medida de la importancia de dichos archivos para el negocio, otra solución factible con estos archivos seria leerlos manualmente, o proyectar una migración desde un archivo Excel hacia una base de datos SQL.

También es factible la escritura en dichos archivos, se realizo pruebas y fueron cumplidas satisfactoriamente, tanto como la actualización y la eliminación de registros.

La arquitectura propuesta en el caso de estudio deja lista la posibilidad para ser usada en una base de datos SQL server o cualquier otro sistema de gestión de base de datos, al mejor parecer del usuario de CTM para poder implantarlo, esta arquitectura queda como sugerencia para que sea implementada, y muestra como sería un despliegue en producción, mas no fue implantado.

Se probo también el módulo de configuración de la aplicación, este funciona correctamente, generando desde las clases de java, clases en .net formularios web, archivos XML, y sentencias SQL, la capacidad de este aplicativo nos permite adecuar el sistema integrador hacia cualquier realidad con un esfuerzo mínimo, ya que ayuda en gran parte a los usuarios, y el tiempo de programación se reduciría, por ejemplo de realizar todo un mantenimiento de una tabla que

demoraría sin esta herramienta entre 8 a 16 horas dependiendo del nivel del desarrollador y su experiencia, a 20 o 30 minutos con la herramienta.

6.2.5 Puesta en Producción

La puesta en marcha de la aplicación es factible según las pruebas realizadas, con los archivos DBF, para la puesta en producción se tendría que tener en cuenta algunos aspectos como:

- Que Archivos el usuario desea que se integren, es una elección autónoma de los administradores del dominio de la aplicación de negocio.
- Seguridad y acceso a los archivos de texto desde el servidor de integración.
- Configuración de usuarios que tendrán acceso a la aplicación.
- Configuración de puertos para comunicación entre servidores.
- Velocidad de la Red de área local.
- Mantener los archivos de configuración de sistema integrador.

Todos los aspectos mencionados anteriormente son básicos para un correcto funcionamiento de la aplicación en producción. Además, que se debe decidir qué es lo que se realizará con la data, si será mostrada, utilizada por otro sistema, migrada o exportada, esto también se puede configurar con el sistema integrador. Pero cualquiera que fuera la opción elegida será de mucha utilidad para darle un mejor aprovechamiento a la información del sistema Pro-tour de CTM del Grupo Costamar.

Capítulo VII
Conclusiones y Recomendaciones

7.1 Conclusiones

La construcción de una solución de integración de información para organizaciones que no desean invertir un alto presupuesto (entiéndase alto presupuesto desde US$ 20000 dólares americanos o más, estimado de acuerdo a la experiencia y precios en el mercado actual de sistemas de información y consultoría informática en el año 2007), sirve de soporte para que estas puedan resolver su problema de integración de información y así tener la oportunidad de darle un mejor uso desde sus sistemas legacy o heredados, y además tener una base construida para continuar con el desarrollo de un nuevo sistema que reemplace al sistema legacy o heredado, de esta forma quedaría migrado completamente.

El uso de el patrón Abstract Factory, herencia de Interfaces, y la característica de Reflection hacen posible que se realice una agregación de clases a los paquetes predefinidos, y con esto no es necesario compilar toda la aplicación cuando se quiera realizar la integración en un entorno de negocio diferente. De esta forma se gana adaptabilidad y facilidad de implantación; esto debido al diseño de la arquitectura que en la solución sirve como base sea cual fuere el origen de datos a integrar. Si se desearía integrar otro origen de datos, éste sería fácilmente añadido a la aplicación mediante sus librerías JAR y DLL, y mediante la generación de clases a través del módulo de configuración.

El tiempo de construcción de la base de un nuevo sistema de información que tenga que integrarse con información de sistemas legacy o heredados se hace de forma muy rápida, lo que antes podría tomar toda unas 5 horas en construir las operaciones básicas de las entidades y orígenes de datos para 1 tabla o archivo, ahora se pueden realizar en menos de 10 minutos, debido a que el integrador cuenta con el configurador que genera código fuente en lenguajes C# y Java, y deja establecida la arquitectura como base para seguir construyendo una aplicación ordenadamente.

Con la herramienta desarrollada es posible integrar información de diferentes plataformas, y a bajo costo, porque se basa en el uso de software libre para su construcción e implantación del servidor de integración, esto hace que se brinde una interface que puede ser accedida desde cualquier plataforma cliente, sin importar el sistema operativo en el que este la aplicación cliente.

Una de las opciones del sistema integrador Smart Javnet es integrar archivos DBF, pero está preparado para adaptarse a otros orígenes de datos, esto debido a su arquitectura, que comprende herencia del patrón Abstract Factory y la herencia de interfaces. El caso de estudio realizado se pudo verificar la factibilidad del sistema, se eligió archivos la integración de archivos DBF porque sistemas construidos en Fox Pro, Visual Fox Pro, Clipper, DBase y entre otros, trabajan con este tipo de archivo, y por esta razón la solución puede tener un alcance a varios tipos de sistemas legacy o heredados que fueron construidos bajo estos entornos de desarrollo.

Con respecto a las pruebas realizadas, se pudieron leer 266 archivos válidos del sistema Pro-Tour, y se pudo comprobar la eficacia al 100% del sistema integrador, luego se trabajó con una muestra de 10 archivos, pero se puede afirmar que la totalidad de los archivos pueden ser integrados y migrados hacia una base de datos SQL Server, a cada uno de estos archivos se les puede generar todo su proceso completo de desarrollo que consiste en: creación de clases de acceso a datos, clases de lógica de negocio, controladora, capa de presentación, tabla en SQL Server, procedimientos almacenados, y su formulario HTML con extensión ASPX en un entorno web.

7.2 Recomendaciones

Continuar la investigación sobre integración en sus diversas formas, que puede ser integración de información de sensores electrónicos o de imágenes, que permitirá que las diversas aplicaciones sean del tipo que fueran estén comunicadas y compartiendo información que será de gran utilidad las organizaciones y empresas.

Se recomienda implementar el uso de colas de mensajes para que la solución de integración maneje una de una mejor forma la concurrencia en ambientes de alta demanda, el capítulo IV de la tesis se plantea como seria la realización de este caso de uso en caso de usar cola de mensajes OPEN JMS, y según se pudo investigar es factible acceder a esta tecnología desde distintos entornos de desarrollo como .NET.

Se recomienda también implementar la encriptación de mensajes entre diferentes plataformas, para que se pueda lograr una mejor seguridad de la información, en la arquitectura quedó planteada los componentes de encriptación, también quedaron planteados los diagramas de secuencias; tanto .Net como J2EE, tienen librerías para el uso de encriptación, y lo que se tiene que hacer es usar el mismo algoritmo de encriptación, solo faltaría la implementación.

Se recomienda realizar futuras investigaciones, con archivos COBOL por ejemplo, y algunos otros de plataformas heredadas, ya que cada uno conlleva tener en cuenta ciertas particularidades, en la presente tesis se realizaron una prueba con archivos DBF que son manejados por Fox Pro, Clipper y DBase, entre otros, y también para bases de datos relacionales.

Con respecto al módulo de configuración puede ser usado como una fábrica de software adaptable, esto podría ser configurado mediante plantillas de clases XML y ser usado con distintos propósitos, bastaría con cambiar de plantilla, además también se puede agregar más adelante generadores de lógica y reglas de negocio particulares al dominio del negocio con el que se desea trabajar; también se pueden agregar configuradores de flujos de trabajos, elecciones de diseño de las pantallas, que podrían ser para entornos de escritorio y para dispositivos móviles.

Glosario

A

AXIS

Apache Axis es una implementación OpenSource que provee de un entorno de ejecución para Servicios Web implementados en Java. Proporciona un Entre otras cosas Axis proporciona:

- Un entorno de ejecución para Servicios Web Java (*.jws)
- Herramientas para crear WSDL desde clases java.
- Herramientas para crear clientes Java desde un WSDL.
- Herramientas para desplegar, probar y monitorizar Servicios Web.
- Integración con servidores de aplicaciones y contenedores de Servlets.

B

BizTalk Server

Servicio producido por Microsoft el cual provee las siguientes funciones: automatización de procesos de negocio, modelamiento de procesos de negocio, comunicación Business-to-business, integración de aplicaciones empresariales y administración de mensajes.

Este producto está dirigido a empresas de tamaño medio a grande. En un escenario común, BizTalk hace posible las compañías integrar y administrar procesos de negocio por flojo de documentos (por ejemplo, pagos, pedidos, ordenes remitidas y facturas) entre aplicaciones, dentro o atreves de los limites organizacionales.

C

C-Sharp (C#)

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (más notablemente de Delphi y Java). C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic.

C# significa, "do sostenido" (C corresponde a do en la terminología musical anglo-sajona). El símbolo # viene de sobreponer "++" sobre "++" y eliminar las separaciones, indicando así su descendencia de C++.

C#, como parte de la plataforma .NET, está normalizado por ECMA desde diciembre de 2001 (ECMA-334 "Especificación del Lenguaje C#"). El 7 de noviembre de 2005 acabó la beta y salió la versión 2.0 del lenguaje que incluye mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables. Ya existe la versión 3.0 de C# en fase de beta destacando los tipos implícitos y el LINQ (Language Integrated Query).

Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones; mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Aunque aún no existen, es posible implementar compiladores que no generen programas para dicha plataforma, sino para una plataforma diferente como Win32 o UNIX.

Capas de una aplicación

1.- Capa de presentación: es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

2.- Capa de negocio: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

3.- Capa de datos: es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único computador (no sería lo normal), si bien lo más usual es que haya una multitud de computadores donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más computadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del computador en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más computadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de computadores sobre los cuales corre la capa de datos, y otra serie de computadores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/ Lógica de Negocio/ Datos.

> En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

Una solución de tres capas (presentación, lógica, datos) que residen en un solo computador (Presentación+lógica+datos). Se dice, que la arquitectura de la solución es de tres capas y un nivel.

Una solución de tres capas (presentación, lógica, datos) que residen en dos computadores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.

Una solución de tres capas (presentación, lógica, datos) que residen en tres computadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

COM

Component Object Model (COM) es una plataforma de Microsoft para componentes de software introducida por dicha empresa en 1993. Esta plataforma es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología. El término COM es a menudo usado en el mundo del desarrollo de software como un término que abarca las tecnologías OLE, OLE Automation, ActiveX, COM+ y DCOM. Si bien COM fue introducido en 1993, Microsoft no hizo énfasis en el nombre COM hasta 1997.

Esencialmente COM es una manera de implementar objetos neutral con respecto al lenguaje, de manera que pueden ser usados en entornos distintos de aquel en que fueron creados, a través de fronteras entre maquinas. Para componentes bien creados, COM permite la reutilización de objetos sin conocimiento de su representación interna, porque fuerza a los implementadores de componentes a proveer interfaces bien definidos que son separados de la implementación. Las diferentes semánticas de reserva de memoria son acomodadas haciendo a los objetos responsables por su propia creación y destrucción por medio de la cuenta de referencias. Se puede hacer casting entre distintos interfaces de un objeto por medio de la función QueryInterface(). El método preferido de herencia en COM es la creación de subobjetos a los que se delegan las llamadas a métodos (llamado agregación).

COM+

Es una extensión de COM, COM+ introduce una mejora en la versión de SPM (la cual es llamada In-Memory Database). También provee administración asíncrona de los eventos, en múltiples clientes. Otra característica es la administración a través de MSMQ (servicio de colas). COM+ automatiza el balance de carga cuando existen múltiples servidores y una sola aplicación.

CORBA

En computación, CORBA (Common Object Request Broker Architecture — arquitectura común de intermediarios en peticiones a objetos), es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CORBA fue definido y está controlado por el Object Management Group (OMG) que define las APIs, el protocolo de comunicaciones y los mecanismos necesarios para permitir la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida. En un sentido general CORBA "envuelve" el código escrito en otro lenguaje en un paquete que contiene información adicional sobre las capacidades del código que contiene, y sobre cómo llamar a sus métodos. Los objetos que resultan pueden entonces ser invocados desde otro programa (u objeto CORBA) desde la red. En este sentido CORBA se puede considerar como un formato de documentación legible por la máquina, similar a un archivo de cabeceras pero con más información.

CORBA utiliza un lenguaje de definición de interfaces (IDL) para especificar los interfaces con los servicios que los objetos ofrecerán. CORBA puede especificar a partir de este IDL la interfaz a un lenguaje determinado, describiendo cómo los tipos de dato CORBA deben ser utilizados en las implementaciones del cliente y del servidor. Implementaciones estándar existen para Ada, C, C++, Smalltalk, Java y Python. Hay también implementaciones para Perl y TCL.

D

DOM

Es una forma de representar los elementos de un documento estructurado (tal como una página web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. El responsable del DOM es el World Wide Web Consortium (W3C).

En efecto, el DOM es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (Javascript).

E

EAR

Se empaqueta en un archivo EAR (Enterprise Archive), que es un archivo JAR estándar con una extensión .EAR. Para crear un archivo EAR, se combinan módulos EJB, Web y de aplicación con los descriptores de despliegue adecuados para cada uno de ellos.

ERP

Los sistemas de planificación de recursos empresariales (ERP) son sistemas de información gerenciales que integran y manejan muchas de de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios

H

HTTP

El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las

páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido

I

Internet Information Server

Internet Information Services (o Server), IIS, es una serie de servicios para los computadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS

J

JAR

JAR ("Java Archives") es un formato desarrollado por "Sun" que permite agrupar las clases diseñadas en el lenguaje Java, este formato es ampliamente utilizado en ambientes Java de todo tipo, esto se debe a que otorga un nivel de compresión y reduce la carga administrativa al distribuir clases en el lenguaje

Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros. JavaScript, un lenguaje interpretado, comparte un nombre similar y una sintaxis similar, pero no está directamente relacionado con Java.

Sun Microsystems proporciona una implementación GNU General Public License de un compilador Java y una máquina virtual Java, conforme a las especificaciones del Java Community Process, aunque la biblioteca de clases que se requiere para ejecutar los programas Java no es software libre.

JNDI

La Interfaz de Nombrado y Directorio Java (JNDI) es una Interfaz de Programación de Aplicaciones para servicios de directorio. Esto permite a los clientes descubrir y buscar objetos y nombres a través de un nombre y, como todas las APIs de Java que hacen de interfaz con sistemas host, es independiente de la implementación subyacente.

Adicionalmente, especifica una interfaz de proveedor de servicio (SPI) que permite que las implementaciones del servicio de directorio sean enchufadas en el framework. Las implementaciones pueden hacer uso de un servidor, un fichero, o una base de datos.

JNDI organiza sus nombres en una jerarquía (Namespace). Un nombre puede ser cualquier string tal como "com.mydomain.ejb.MyBean". Un nombre también puede ser un objeto que soporte la interfaz Name, sin embargo la forma más común de nombrar a un objeto es un string. Un nombre se asocia a un objeto en el directorio almacenando el objeto o una referencia JNDI al objeto en el servicio de directorio identificado por el nombre.

La API JNDI define un contexto que especifica donde buscar un objeto. El contexto inicial se usa normalmente como punto de partida.

En el caso más simple, un contexto inicial debe crearse usando la implementación específica y los parámetros extra requeridos por la implementación. El contexto inicial será usado para buscar un nombre

L

Legacy

Un sistema heredado (o sistema legacy) es un sistema informático (ordenador o aplicación) que continúa siendo utilizado por el usuario (típicamente una organización) y no quiere o puede ser reemplazado o actualizado. Habitualmente se utiliza este término para referirse a sistemas anticuados.

Los sistemas heredados son considerados potencialmente problemáticos por numerosos ingenieros de software por diversos motivos. Dichos sistemas a menudo operan en ordenadores obsoletos y lentos, cuyo mantenimiento tiene elevados costes y difíciles de actualizar por falta de componentes adecuados.

Logs

Son Registros planos de las ocurrencias exitosas y erróneas de un sistema informático.

M

Message Oriented Middleware (MOM)

Los mensajes enviados al cliente se recogen y se almacenan hasta que son solicitados, mientras el cliente continúa con otros procesos.

Metalografía

La metalografía es la ciencia que estudia las características estructurales o constitutivas de un metal o aleación relacionándolas con las propiedades físicas y mecánicas

Modelo NOLAN

El modelo de Nolan/Gibson trata de explicar las diferentes etapas de asimilación de las

nuevas tecnologías por las organizaciones.

El modelo de Nolan/Gibson considera que las empresas tienen una cartera de tecnologías de la información diferentes. Cada tecnología pasa a través de las siguientes fases: Inversión, Aprendizaje de la tecnología y adaptación, Racionalización/control de gestión. Madurez/difusión generalizada de la nueva tecnología e integración.

MySQL

MySQL es un sistema de gestión de base de datos, multihilo y multiusuario con más de seis millones de instalaciones1. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso.

O

Object Request Broker (ORB)

Este tipo de middleware permite que los clientes envíen objetos y soliciten servicios en un sistema orientado a objetos.

P

PDA

Personal Digital Assistant, (Ayudante personal digital) es un computador de mano originalmente diseñado como agenda electrónica (calendario, lista de contactos, bloc de notas y memos) con un sistema de reconocimiento de escritura. Hoy día se puede usar como una computadora doméstica (ver películas, crear documentos, juegos, correo electrónico, navegar por Internet, etc.).

PostgreSQL

Es un motor de base de datos, es servidor de base de datos relacional libre, liberado bajo la licencia BSD

Publish/subscribe

Este tipo de monitores middleware activan y entregan información relevante para los subscriptores.

R

Remote Procedure Call (RPCs)

El cliente realiza una llamada a procedimientos que están corriendo en máquinas remotas. Pueden ser síncronos o asíncronos.

S

SAX

SAX son las siglas de "Simple API for XML", originalmente, una API únicamente para el lenguaje de programación JAVA, que después se convirtió en la API estándar de facto para usar XML en JAVA. Existen versiones de SAX no sólo para JAVA, sino también para otros lenguajes de programación

SOAP

SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar creado por el W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

W

WAR

WARS o "Web Archive" es una especificación desarrollada por Sun que permite agrupar un conjunto de clases y documentos que conforman una aplicación Web en Java en un sentido un poco peculiar a la que se acostumbra en un sitio.

X

XDOCLET

XDoclet es un motor de Código abierto para el Lenguaje de programación Java, su función es la generación de código. Está asociado a la programación orientada a los atributos, es decir, usted puede lograr más funcionalidad agregándole metadata (atributos) a su código. Esto se lleva a cabo con tags JavaDoc

XML

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

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.
XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil

XML-RPC

XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar las llamadas y HTTP como mecanismo de transporte.

Es un protocolo muy simple ya que sólo define unos cuantos tipos de datos y comandos útiles, además de una descripción completa de corta extensión. La simplicidad del XML-RPC está en contraste con la mayoría de protocolos RPC que tiene una documentación extensa y requiere considerable soporte de software para su uso.

Fue creado por Dave Winer de la empresa UserLand Software en asociación con Microsoft en el año 1998. Al considerar Microsoft que era muy simple y adicionar funcionalidades y después de varias etapas de desarrollo el estándar dejó de ser sencillo y se convirtió en lo que es actualmente se conoce como SOAP.

BIBLIOGRAFIA

Fuentes de información

[1] MECELLA M.; PERNICI B. *Designing wrapper components for e-services in integrating heterogeneous systems*, Edited by F.Casati,M.-C.Shan,D.Georgakopoulos.Received:30 October 2000 /Accepted:9 January 2001 Publishedonline:28 June 2001 –c Springer-Verlag 2001.

Este Artículo trata acerca del desarrollo basado en componentes que son más populares para afianza el desarrollo basado en Internet. Los diferentes modelos de componentes pueden ser adaptados.
Para obtención de un determinado nivel de abstracción (conceptual y operacional). En este artículo se presenta una arquitectura basada en componentes como bloques construidos para el desarrollo de los e-services, donde estos están basados en servidores legacy. Discutimos sus características y su aplicabilidad en el desarrollo basado en aplicaciones para Internet.

[2] GIAGLIS George M., Focus Issue on Legacy Information Systems and Business Process Change: On The Integrated Design and Evaluation of Business Processes And Information Systems, Brunel University George.Giaglis@brunel.ac.uk 1999.

El rol de los sistemas de información de influenciar y permitir el diseño organizacional es ampliamente conocido. En este artículo se repasa disciplinas que hacen referencia a la ingeniería empresarial por lo que el diseño basado en los procesos de la organización, IS-develoment, y IS- evaluation.

[3] ENSINK Brian; HAVEMAN Kimberly; SHRESTHA Mochan; SCHAVEY Todd, XML based adaptation of the composite approach for database integration - April 1999.

[4] XJ: *integration of XML processing into java, Proceedings of the 13th international World Wide Web conference New York, NY, USA* , Year of Publication: 2004 ISBN:1-58113-912-8.

El artículo describe la importancia creciente de XML pues un formato universal de representación de datos ha conducido varios esfuerzos para permitir el desarrollo de aplicaciones que funcionen entendiendo datos desde XML. Estos esfuerzos se extienden de interfaces API runtime a lenguajes de programación basados en XML. El tema de este artículo es XJ, un lenguaje de investigación que propondrá mecanismos nuevos para la integración de XML como construcción de clases en Java. Los objetivos del diseño de XJ lo distinguen en la clave de integrar XML en lenguajes de programación; específicamente, el diseño de XJ adquiere al esquema y a los estándares Xpath de XML, y soporta actualizaciones sobre el tema de datos en XML de tal modo que guardan la naturaleza imprescindible de Java.

[5] BERTRAND Enrique, *Integración de información corporativa con Web Services* http://es.sun.com/infospain/eventos/javaexpo2004/presentaciones/tracks3/SoftwareAG_WSInformation Integration.pdf , – Software AG - Java Expo 2004.

[6] HASSELBRING Wilhelm, *Information system Integration* – Junio 2000 ACM- Information.

[7] SUTHERLAND Jeff; VAN DEN HEUVEL Willem-Jan *Enterprise Application Integration and complex adaptive systems communications*, ACM October 2002/Vol. 45, No. 10.

[8] FREMANTLE Paul; WEERAWARANA Sanjiva; KHALAF Rania, *Examining the emerging field of Web Services and how it is integrated into existing enterprise Infrastructures Enterprise Services*, COMMUNICATIONS OF THE ACM October 2002/Vol. 45, No. 10.

[9] NOFFSINGER W.B.; NIEDBALSKI Robert; BLANKS Michael; EMMART Niall *Legacy Object Modeling Speeds Software Integration,*

December 1998/Vol. 41, No. 12 COMMUNICATIONS OF THE ACM.

[10] AKOS Ledeczi; DAVIS James; NEEMA Sandeep; AGRAWAL Aditya
Modeling Methodology for Integrated and Simulation of Embedded Systems
Vanderbilt University - ACM Transactions on Modeling and Computer Simulation, Vol. 13, No. 1, January 2003, Pages 82–103.

[11] SCHULTZ David,
A Case Study In System Integration Using The Build Approach,
Computer Sciences Corporation - Falls Church, Virginia.

[12] YANDONG (Steven) Deng; MALY Wojciech,
System Integration: A Design Driven System Implementation Schema,
Department of Electrical & Computer Engineering Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213, USA {yangdon, maly}@ece.cmu.edu.

[13] ORMSET André; RIOS José Antonio; HOFBOER René; ZAMORANO Damián; GARCIA Bernard *Elementos de Programación*, E.T.S.I. Telecomunicación, 1ºC, 2004.

[14] RODRIGUEZ Alfredo; MARQUEZ Antonio; TORO Miguel,
Gestión de la evolución del software. El eterno problema de los legacy systems,
Departamento de Lenguajes y Sistemas Informáticos, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, Email:amarser@teleline.es.

[15] PEREZ J.; CARSI J.A.; RAMOS I.; ANAYA V.; SILVA J.,
Migración de datos automática a partir de la información de los esquemas conceptuales
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia - Camino de Vera s/n E-46071 Valencia – España jeperez | pcarsi | iramos | vanaya | jsilva}@dsic.upv.es.

[16] PRESSMAN Roger,
*Ingeniería del software – Un enfoque practico,*Cuarta Edición Mac Graw 1998.

[17] LARMAN; CRAIG,
UML y Patrones, Prentice Hall Tercera Edición 2003.

[18] SHORT Scott,
Building XML Web Services for the Microsoft .NET Platform, Microsoft Press 2002.

[19] STUART Madnick; WANG Richard.
A Framework of Composite Information Systems for Strategic Advantage. Proceedings of the 1988 Hawaii International Conference on System Sciences, enero 1988, págs. 35-43.

Plataformas y Herramientas

[20] GABRICK Kurt; WEISS David B,
J2EE and XML Development, 2002 Manning Publications Co., ISBN 1-930110-30-8.

[21] Course Number: 2124A Workbook,
Introduction to C# Programming for the Microsoft® .NET Platform
Microsoft (Prerelease) Course Number: 2124A , Part Number: X08-16666, Released: 03/2001.

[22] TAYLOR Luke; THE JBOSS GROUP,
Getting Started with Jboss, J2EE application on the Jboss 3.2.x Server.

[23] MCLAUGHLIN Brett,
Building Java™ Enterprise Applications Volume I: Architecture,
Publisher: O'Reilly First Edition March 2002 ISBN: 0-569-00123-1, 318 pages.

[24] GUEST Simon,
Microsoft .NET and J2EE Interoperability Toolkit,
Microsoft Press, ISBN 0-7356-1922-0.

[25] DEITEL H. M.; DEITEL P. J.; SANTRY S. E.,
Deitel & Associates, Inc
Java 2 Platform HOW TO PROGRAM

[26] BODOFF Stephanie; GREEN Dale, JENDROCK Eric, PAWLAN Monica, STEARNS Beth
The J2EE Tutorial,
Sun Microsystems, Inc. 2001

[27] BAMBAREN Jorge,
Visión General del e-Commerce, Comercio Electrónico - Sesión 1 Agosto 2004.

[28] HAPNER Mark; BURRIDGE Rich; SHARMA Rahul; FIALLI Joseph; KATE Stout,
Java Message Service, The JMS API is an API for accessing enterprise messaging systems from Java programs, Version 1.1 April 12, 2002, Sun Microsystems, Inc.

Enlaces Especializados

[29] http://www.escobol.com/modules.php?name=Sections&op=listarticles&secid=13
Sitio Web Relacionado al Lenguaje de Programación COBOL, Evolución, Manuales temáticos, Historia, Programas, Trucos, entre otros, Año 2007

[30] http://www.iespana.es/iabot/ciencia/software/historia_lenguajes_programacion.htm
Sitio Web que muestra la evolución de los lenguajes de programación desde los inicios con el lenguaje binario hasta los últimos como Java y Visual Basic.NET, Año 2007

[31] http://es.wikipedia.org/wiki/Basic
Sitio Web Relacionado a la 1 Historia, Antecedentes, Nacimiento y primeros años, Crecimiento, Madurez, Sintaxis, Procedimientos y Control de Flujo, Tipos de Datos, Disponibilidad y variantes del dialecto, Ejemplos, Dialectos, Estándares, etc. Año 2007

[32] MSDN Arquitectura**,**
Arquitectura de Aplicaciones de 3 capas
http://dotnetjunkies.com/WebLog/desarrollonet/archive/2004/06/17/16855.aspx,
Sitio web en donde se describe la Arquitectura en 3 capas de la plataforma de .NET, Año 2007

[33] http://www.microsoft.com/spain/enterprise/vision/net.asp
Sitio Web sobre la Tecnología .NET Visión .NET, Año 2007.

[34] http://www.grupoe.com/Web/edu_negocios_internet.asp
Sitio Web con información acerca de E – Business, Año 2007.

[35] SPROTT David; WILKES Lawrence,
Understanding Service-Oriented Architecture, CBDI Forum January 2004
http://msdn.microsoft.com/architecture/soa/default.aspx
Summary: Gives a concise explanation of service-oriented architecture, what it is, and how it affects what architects, CIOs, project managers, business analysts, and lead developers do. (13 printed pages)

[36] PARRA José David,
Hacia una Arquitectura Empresarial basada en Servicios,
http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp,
Año 2007

[37] Procesamiento de Imágenes Digitales

http://www.cedex.es/lceym/eq_labce/eq_cmima.html, Año 2007

[38] SYBASE, Herramientas de Integración de Información
http://www.sybase.es/productos.htm

[39] BIZTALK SERVER
http://www.microsoft.com/spain/biztalk/info/overview.asp.

[40] IBM DB2
http://www.microsoft.com/spain/biztalk/info/overview.asp.

[41] NOVATRONIC
http://www.novatronic.com

[42] OPENJMS
http://openjms.sourceforge.net/index.html

[43] Instituto Nacional de Estadística e Informática (INEI), Metodología para la elaboración de sistemas de información
http://www.inei.gob.pe/web/metodologias/attach/lib606/n00.htm, Año 1997.

[44] CARNEGIE MELLON, SOFTWARE ENGINEERING INSTITUTE, Middleware,
http://www.sei.cmu.edu/str/descriptions/middleware.html, Año 2007.

[45] SUN, Java Sun Application Server,
http://www.sun.com/software/products/appsrvr/index.xml, Año 2007.

Anexos

Anexo 1
ANÁLISIS DE FACTIBILIDAD

Proyecto: ***Integrador de información de sistemas legacy***

1. Situación actual

En la actualidad las empresas cuyos datos mayormente son manejados por sistemas antiguos en plataformas que no pueden integrarse fácilmente pero son útiles debido a su rapidez y facilidad de uso para los empleados de la empresa, pero que no pueden servir como tarjeta de presentación para los clientes, por ejemplo a partir de una página Web, es entonces que se hace necesario tener la forma de sacar esa data y poder dar una presentación más versátil y así dar a la empresa una oportunidad de competir al mismo nivel que sus competidores.

2. Alternativas

El proyecto tendrá un impacto primordial en la integración de sistemas heredados y en nuevos sistemas de información de hoy en día, los casos de uso que se dan son:

- Establecer Conexión
- Enviar Requerimiento
- Generar Respuesta desde sistema heredado

3. Factibilidad operacional

Debido a que en la mayoría de las empresas de este tipo el personal de la empresa tiene el mínimo conocimiento de computación, se necesitará dar capacitación que permita el manejo del sistema Integrador Smart JavNet.

Se requerirá de una persona con conocimientos de base de datos y programación que le permitan administrar el sistema y realizar el mantenimiento de la base de datos entre otras cosas.

4. Factibilidad tecnológica

Los requerimientos de HW, SW y comunicaciones se especifican en el documento de Análisis de Requerimientos.

Costos de Desarrollo:

- Licencia de Microsoft Windows XP profesional (Plataforma de desarrollo del sistema).
- Licencia de Microsoft SQL Server 2000-2005, para la implementación de la base de datos.
- Licencia de Microsoft Visual Studio .Net, para la programación del sistema.

- Licencia de IBM WebSphere – Eclipse, para la programación del sistema, o en su caso el entorno de software libre de Eclipse
- Licencia de Microsoft Visio, para el modelamiento del sistema.
- Licencia de Microsoft Erwin 4.0, para el modelamiento de la base de datos.
- Licencia de Microsoft Office XP (Word), para la documentación.
- Licencia de Microsoft Project 2000, para el desarrollo del Cronograma de Actividades.

Costos de Producción:

En caso de que la empresa no cuente con ningún tipo de licencia de los principales software, se necesitaran las siguientes para la instalación del sistema:

- Licencia de Microsoft Windows XP profesional.

5. Factibilidad Económica-Financiera

Costo estimado:

La duración de desarrollo del aplicativo fue aproximadamente de 6 meses, a 8 horas diarias, suponiendo un costo de 15 Dólares la hora, un costo aproximado seria estimado en:

TOTAL $ 14,400

Este es el costo del proyecto integral, del software que se construyo, cabe mencionar que este software es genérico, y está hecho para adaptarse a cualquier entorno de negocio.

Esta suma no involucra esto, la licencia de un servidor SQL estándar, es $ 5000 por procesador, y el software adicional de Integration Services de SQL server cuesta $30000 por procesador, además que es de solo lectura, no permite la opción de escritura en el archivo.

Tabla A1.1 Inversiones Tecnológicas para la implantación

Escenario	Procesador necesario	Procesador recomendado	RAM necesaria	RAM recomendada
1 Servidor	Pentium IV	Pentium IV Dual Core.	512 MB*	1 GB
1 Usuarios o mas	Pentium IV	Pentium IV Dual Core.	512 MB*	1 GB

Estación	Costo ($ Dólares Americanos)
Servidor	Entre 550 y 800
Cliente	Entre 500 y 800

Requerimientos de Software:

Los requerimientos de software, necesarios para instalar la aplicación son los siguientes, se pueden elegir entre las alternativas de sistemas operativos o bien Windows 2003 server (recomendable) o Windows XP:

Tabla A1.2 Requerimientos de Software

Resumido	Nro de Licencias	Costo licencia OPEN	Costo licencia OPEN + SA
Microsoft Windows 2003 Server	1	2159.5	3239.24
Windows XP Professional	1	48.19	48.19
Microsoft SQL Server 2005	1	22122.65	33183.98
JBoss Application Server	1	0	0
Sun Application Server 9.0	1	0	0

A diferencia de otras plataformas:

Tabla A1.3- Comparativa con otras plataformas

Resumido	Nro de Licencias	Costo licencia OPEN	Costo licencia OPEN + SA
Bistalk Server 2006 Standar *	1	8500	9600
Bistalk Server 2006 Professional *	1	24000	9600
Microsoft SQL Integration Services	1	30530	32450

***Precios Referenciales, es necesario un análisis del negocio, incluyendo La consultoría de Microsoft**

Precio estimado:

El precio estimado por licencia de uso de la aplicación seria de:

Solo Integrador:

TOTAL $ 3,000 + IGV (Por procesador)
(Incluye el servicio de configuración, y adecuación al negocio)

Solo Integrador + Configurador:

TOTAL $ 5,000

6. Beneficios Tangibles E Intangibles Esperados

Con la implementación del Sistema Integrador Smart Javnet, las empresas obtendrán los siguientes beneficios:

Tangibles:

- Ayuda en la toma de decisiones.
- Tener una Base de Datos actualizada.
- Seguridad y confidencialidad de los datos.

Intangibles:

- Prestigio.
- Rapidez en el servicio.
- Mejor atención al cliente.
- Obtener nuevos clientes, y por lo tanto mayores ganancias.

Anexo 2
LENGUAJES DE PROGRAMACIÓN EN QUE SE DESARROLLARON LOS SISTEMAS HEREDADOS

FORTRAN

Fortran es lenguaje de propósito general, principalmente orientado a la computación matemática, por ejemplo en ingeniería. Fortran acrónimo de FORmula TRANslator, y originalmente fue escrito con mayúsculas como FORTRAN. Sin embargo la tendencia es poner sólo la primera letra con mayúscula, por lo que se escribe actualmente como Fortran. Fortran fue el primer lenguaje de programación de alto nivel. El desarrollo de Fortran inicio en la década de 1950 en IBM y ha habido muchas versiones desde entonces. Por convención, una versión de Fortran es acompañada con los últimos dos dígitos del año en que se propuso la estandarización. Por lo que se tiene:

- Fortran 66
- Fortran 77
- Fortran 90 (95)

La versión más común de Fortran actualmente es todavía Fortran 77, sin embargo Fortran 90 esta creciendo en popularidad. Fortran 95 es una versión revisada de Fortran 90 la cual fue aprobada por ANSI en 1996. Hay también varias versiones de Fortran para computadoras paralelas. La más importante es HPF (High Performance Fortran), la cual es de hecho el estándar.

COBOL

El lenguaje COBOL (Common Business Oriented Language) ha sido, el más utilizado a nivel de gestión, y aún hoy, hay variantes del mismo que siguen funcionando. El volumen de programas, especialmente en las grandes empresas como la banca, ha sido tan grande que la sustitución no ha sido un paso fácil ni barato, y más teniendo en cuenta que prácticamente la totalidad de los grandes fabricantes de hardware tenían compiladores para él, incluyendo las computadoras personales microordenadores, aunque estuviese orientado a equipos de mayor tamaño.

Su creación fue un intento exitoso de unificar los entornos de programación que estaban funcionando hacia los años 50. Así, en el 1.959, todos los implicados en la materia se reunieron en Washington creando lo que se denominó la Codasyl (Conference on data system languages) y como resultado, en el siguiente año, surgió la primera versión del nuevo lenguaje: el Cobol-60. Al que siguieron las versiones del 63 y del 65, hasta llegar al estándar absoluto, el ANS COBOL, que se aprobó en el 1974.

Se dice que era una mezcla entre estructurado y anárquico. Así eran obligatorias cuatro "divisiones", aunque el código que se generase en la cuarta, podía ser de lo más anárquico y, en contra de lo que se suele decir, ininteligible, debido a la facilidad del lenguaje en el uso del GOTO sin ningún orden.

La totalidad de las líneas de código iban numeradas, aunque generalmente (como ocurría en todos estos lenguajes) el intervalo estándar fuese de 10 en 10, por evitar la

renumeración si es que había que incluir líneas nuevas, lo que en algunos computadores era automático.

No existían las estructuras como hoy se entienden, ni las bases de datos relaciónales, se trabajaban en el mejor de los casos con bases de datos jerárquicas. Hay que tener en cuenta que una parte de las computadoras que manejaban este lenguaje se alimentaban de información a través de lectoras de tarjetas perforadas.

Tomado del sitio web

http://www.escobol.com/modules.php?name=Sections&op=listarticles&secid=13

BASIC

El lenguaje BASIC original fue inventado en 1964 por John George Kemeny (1926-1993) y Thomas Eugene Kurtz (1928) en el Dartmouth College. En los años subsiguientes, mientras que otros dialectos de BASIC aparecían, el BASIC original de Kemeny y Kurtz era conocido como *BASIC Dartmouth*.

BASIC fue diseñado para permitir a los estudiantes escribir programas usando terminales de computador de tiempo compartido. BASIC estaba intencionado para facilitar los problemas de complejidad de los lenguajes anteriores, con un nuevo lenguaje diseñado específicamente para la clase de usuarios que los sistemas de tiempo compartido permitían: un usuario más sencillo, a quien no le interesaba tanto la velocidad, sino el hecho de ser capaz de usar la máquina. Los diseñadores del lenguaje también querían que permaneciera en el dominio público, lo que contribuyó a que se diseminara.'1

Los ocho principios de diseño de BASIC fueron:

1. Ser fácil de usar para los principiantes.
2. Ser un lenguaje de propósito general.
3. Permitir que los expertos añadieran características avanzadas, mientras que el lenguaje permanecía simple para los principiantes.
4. Ser interactivo.
5. Proveer mensajes de error claro y amigable.
6. Responder rápido a los programas pequeños.
7. No requerir un conocimiento del hardware de la computadora.
8. Proteger al usuario del sistema operativo.

El lenguaje fue en parte basado en FORTRAN II y en parte en Algol 60, con adiciones para hacerlo apropiado para tiempo compartido y aritmética de matrices, BASIC fue implementado por primera vez en la mainframe GE-2653, que soportaba múltiples terminales. Contrario a la creencia popular, era un lenguaje compilado al momento de su introducción. Casi inmediatamente después de su lanzamiento, los profesionales de computación comenzaron a alegar que BASIC era muy lento y simple. Tal argumento es un tema recurrente en la industria de los computadores.

Aún así, BASIC se expandió hacia muchas máquinas, y se popularizó moderadamente en las minicomputadores como la serie DEC PDP y la Data General Nova. En estos casos, el

lenguaje era implementado como un intérprete, en vez de un compilador, o alternativamente, de ambas formas.

Sin embargo, fue con la introducción de la Microcomputadora Altair 8800 en 1975 que BASIC se diseminó ampliamente. La mayoría de lenguajes de programación eran demasiado grandes para caber en la pequeña memoria que la mayoría de usuarios podía pagar para sus máquinas, y con el lento almacenamiento que era la cinta de papel, y más tarde la cinta de audio (los discos magnéticos aún no existían), y la falta de editores de texto adecuados, un lenguaje pequeño como BASIC era una buena opción. Uno de los primeros en aparecer fue Tiny BASIC, una implementación simple de BASIC escrita originalmente por el Dr. Li-Chen Wang, y portada más tarde a la Altair por Dennis Allison, a petición de Bob Albrecht (quien después fundó el *Dr. Dobbs Journal*). El diseño de Tiny BASIC y el código fuente completo fue publicado en 1976 en DDJ.

En 1977 Microsoft (Gates y Allen) lanzó Altair BASIC. Luego comenzaron a aparecer bajo licencia versiones para otras plataformas, y millones de copias y variantes pronto estarían en uso. Se convirtió en uno de los lenguajes estándar en la Apple II. Para 1979 Microsoft estaba negociando con varios vendedores de microcomputadores, incluyendo a IBM, para licenciar un intérprete de BASIC para sus computadores. Una versión se incluyó en los chips ROM de las PCs IBM, para PCs sin discos, y era iniciado automáticamente al encender el computador.

Mientras que las nuevas compañías intentaban seguir los pasos del éxito de Altair, IMSAI, North Star, y Apple, creando la revolución de la computador casera. BASIC se convirtió en una característica estándar para casi todos los computadores caseros; la mayoría venía con un intérprete de BASIC en ROM (algo hecho por primera vez por la Commodore PET en 1977). Pronto había muchos millones de computadores corriendo BASIC alrededor del mundo, un número mucho más grande que el de todos los usuarios de otros lenguajes juntos. Muchos programas, especialmente los de la Apple II e IBM PC, dependían de la presencia del intérprete de BASIC de Microsoft y no correrían sin éste; por lo que Microsoft usó la licencia de copyright en los intérpretes de BASIC para influenciar en las negociaciones con los vendedores de computadores.

El BASIC fue también el lenguaje prefijado en los computadores caseros europeos de la década de los 80 como el ZX Spectrum, MSX o el Commodore 64, haciendo muchas veces la función de intérprete y sistema operativo primitivo ya que venían implementados en ROM.

En su periodo de madurez se crearon versiones de BASIC nuevas y más poderosas. Microsoft vendió varias versiones de BASIC para MSDOS/PCDOS, incluyendo BASICA, GW-BASIC (una versión compatible con BASICA que no necesitaba la ROM de IBM), y Quick BASIC. El publicante de Turbo Pascal, Borland, publicó Turbo BASIC 1.0 en 1985 (versiones sucesoras aún se venden bajo el nombre de PowerBASIC por otra compañía). Aparecieron varias extensiones de BASIC para computadores caseras, típicamente con gráficos, sonido, y comandos DOS, así como facilidades para Programación estructurada. Otros lenguajes usan la sintaxis de BASIC como base para otros sistemas totalmente diferentes, como por ejemplo GRASS.

Sin embargo a finales de la década de 1980 los computadores nuevos eran mucho más complejos, e incluían características (como la Interfaz gráfica de usuario) que hacían a

BASIC menos apropiado para programarlas. Al mismo tiempo los computadores progresaban de ser un interés para aficionados a herramientas usadas principalmente para ejecutar aplicaciones escritas por otros, y la programación en sí se fue haciendo menos importante para la creciente mayoría de usuarios. BASIC comenzó a desvanecerse, aunque numerosas versiones aún estaban disponibles.

La suerte de BASIC dio un giro nuevamente con la introducción de Visual Basic de Microsoft. Aunque es algo difícil considerar este lenguaje como BASIC (a pesar de que usa muchas palabras clave conocidas de BASIC) se ha convertido en uno de los lenguajes más usados en la plataforma Windows, y se dice que representa del 70 al 80% del desarrollo comercial de aplicaciones. Visual Basic for Applications (VBA) fue añadido a Microsoft Excel 5.0 en 1993 y al resto de la línea de productos de Microsoft Office en 1997. Windows 98 incluyó un intérprete de VBScript. La versión más reciente de Visual Basic es llamada VB.NET. La suite OpenOffice.org incluye una variante de BASIC menos poderosa que su contraparte de Microsoft.

Tomado del Sitio web

http://www.iespana.es/iabot/ciencia/software/historia_lenguajes_programacion.htm

Lenguaje C

EL lenguaje C es el resultado de un proceso de desarrollo que inició con un lenguaje denominado BCPL. Este influenció a otro llamado B (inventado por Ken Thompson). En los años 70; éste lenguaje llevó a la aparición del C.

Con la popularidad de las microcomputadoras muchas compañías comenzaron a implementar su propio C por lo cual surgieron discrepancias entre sí.

Por esta razón ANSI (American National Standars Institute, por sus siglas en inglés), estableció un comité en 1983 para crear una definición no ambigua del lenguaje C e independiente de la máquina que pudiera utilizarse en todos los tipos de C.

Algunos de las C existentes son:

Quick C, C++ , Turbo C , Turbo C ++ , Borland C , Borland C++ , Microsoft C

C es un lenguaje de programación de nivel medio ya que combina los elementos del lenguaje de alto nivel con la funcionalidad del ensamblador.

Su característica principal es ser portable, es decir, es posible adaptar los programas escritos para un tipo de computadora en otra.

Otra de sus características principales es el ser estructurado, es decir, el programa se divide en módulos (funciones) independientes entre sí.

El lenguaje C inicialmente fue creado para la programación de sistemas operativos, intérpretes, editores, ensambladores, compiladores, administradores de bases de datos.

Actualmente, debido a sus características, puede ser utilizado para todo tipo de programas.

Tomado del Paper: Elementos de Programación E.T.S.I. Telecomunicación, 1ºC , André Ormset,

Pascal

Para poder dar a conocer la historia de Pascal deberíamos primero conocer la historia del ALGOL, del cual Pascal es una evolución. La historia del ALGOL comenzó en 1958, cuando un comité de representantes del GAMM (una organización europea de científicos en informática) y ACM (su contrapartida en USA) se reunieron en Zurcí y produjo un informe preliminar sobre un "International Algebraic Language", o IAL. Este lenguaje, conocido más tarde como ALGOL 58, atrajo mucho interés y fue implementado sobre varias computadoras. Los representantes europeos y estadounidenses se reunieron de nuevo en París en 1960 para considerar una versión completamente nueva de este lenguaje, conocida como ALGOL 60. Durante este período, ALGOL fue extremadamente popular entre los científicos informáticos, y su definición rigurosa marcó nuevos estándares para el diseño e implementación de lenguajes. ALGOL se convirtió en un lenguaje universal para la definición de algoritmos publicados en revistas. Con el paso del tiempo, fueron apareciendo nuevas versiones revisadas de ALGOL 60, como ALGOL W (desarrollado por Niklaus Wirth) o ALGOL 68, que fue intencionadamente un lenguaje de propósito general con aplicaciones en un amplio rango de interés, aunque rápidamente se reconoció como un lenguaje demasiado ambicioso para ser práctico. De esta manera, Wirth diseñó un sucesor más reducido del ALGOL 60 y lo llamó PASCAL. Su primer compilador se implementó en 1.970 y una versión revisada fue definida e implementada en 1973. PASCAL fue claramente diseñado para servir como un lenguaje para enseñar diseño de algoritmos y metodología de programación.

Como el ALGOL, PASCAL ha jugado un papel único como el principal lenguaje usado para publicar algoritmos en las revistas y libros. A pesar de sus fuertes mejoras sobre ALGOL, -especialmente en el área de entrada-salida, archivos, registros, gestión dinámica de memoria y estructuras de control- PASCAL también fue cuestionado por sus deficiencias, y por ello se propusieron sucesores importantes.

Tomado del Paper: Elementos de Programación E.T.S.I. Telecomunicación, 1ºC, André Ormset.

Clipper

A principio de los años ochenta, DBASE II hizo su aparición de la mano de George Tate (1943-1984) y su empresa Ashton-Tate. Esta nueva herramienta se presentaba en el emergente mundo de los microordenadores con la intención de facilitar la gestión de las bases de datos.

Evidentemente, los sistemas de gestión de bases de datos existían desde mucho antes, sobre todo, desarrollados para grandes sistemas, pero la cuestión estaba en cubrir una carencia que más tarde o más temprano debía ser atendida por los ingenieros de software y que era esperada ansiosamente por el creciente número de usuarios de los ordenadores personales.

El sistema de gestión de bases de datos había que diseñarse no exclusivamente como un entorno de programación, semejante a otros entornos o lenguajes con capacidad de tratamiento de grandes masas de datos. Este debía posibilitar la ejecución interactiva de instrucciones, ser amigable, accesible por usuarios no programadores, y debía estar formado por unas instrucciones potentes y fáciles de memorizar.

También, a principio de los ochenta se comienza a utilizar entre los usuarios de micros una nueva terminología informática de bases de datos, ésta era más familiar en otros ambientes informáticos y definía con precisión los conceptos más básicos:

Una base de datos puede definirse como la agrupación útil y organizada de información.

CLIPPER es un dialecto creado como otros tantos con la intención de mejorar las prestaciones de DBASE. Su primera versión se creó en 1985 en los laboratorios de Natuncket. CLIPPER está escrito en lenguaje C y Ensamblador y se presentó como un lenguaje atrevido que ha dado muchos quebraderos de cabeza en Ashthon-Tate. En el primer contacto que se tiene con él es difícil encontrar muchas diferencias con respecto a DBASE, ya que CLIPPER es un lenguaje formado por un conjunto de comandos y funciones similares a las usadas con DBASE, incluso la mayoría con igual formato sintáctico.

http://www.eltutorial.com/articulos/visitar/?id=139 Sitio Web Relacionado a la Historia, y avances del lenguaje CLIPPER.

Dbase

DBASE II también contribuyó a la filosofía de la programación estructurada, mejoró sus prestaciones y evolucionó en varias versiones (DBASE III, DBASE III+ y DBASE IV).

George Tate fallecido tempranamente nunca pudo comprobar la revolución que ocasionaría este producto, aún en constante evolución.

El éxito obtenido entre los usuarios de micros, principalmente atraídos por su versatilidad y potencia, y los grandes beneficios producidos en su comercialización, hizo que muchas empresas de software se adhirieran a la idea de desarrollar nuevos productos análogos, una gama de dialectos que hoy se les agrupa con el sobrenombre de entorno xBase (Clipper, Quicksilver, Foxbase, etc.).

La difusión de estos productos ha desbancado a muchos lenguajes de programación, como al Cobol que aunque propicia una fácil lectura de sus fuentes, la programación resulta lenta y laboriosa.

En los ochenta, en pleno boom informático DBASE sustituye a muchos lenguajes por la potencia de sus órdenes y facilidad de uso. Por entonces, hubo que estar muy despierto a la hora de seleccionar una herramienta de trabajo con futuro.

Fox Pro

Nació en 1984 de la mano de Dave Fulton y Bill Ferguson bajo el nombre de FoxBase. FoxBase era un compilador de dBaseII que creaba ejecutables bastante rápidos y

eficientes. En Junio de 1986 salía al mercado FoxBASE+ y en 1988 Fox Software recibía una demanda de Ashton-Tate. Sin embargo su desarrollo no se paró, y en 1989 apareció FoxPro 1.0, la primera versión que abandonaba el DOS en favor de Windows. En Julio de 1991 y ya publicada la sentencia absolutoria, aparece FoxPro 2.0 cuyo mayor logro fue la inclusión de la tecnología Rushmore, un método de acceso a registros muy eficiente que aceleraba considerablemente los programas escritos con Fox. Además, se añadió el lenguaje de datos SQL.

En Junio de 1992 se produjo un hecho que pocos esperaban, Microsoft adquiere FoxPro por 173 millones de dólares (un hecho singular ya que en ese momento existía un proyecto llamado Cirrus que más tarde daría lugar a un SGBD llamado MS Access). Inmediatamente se publica MS FoxPro 2.0 cuya única diferencia consiste en el logotipo de Microsoft en la caja del producto. Microsoft distribuyó varias versiones más de FoxPro, la versión 2.5 aparece en Enero de 1993 y siguiendo la política de MS nuevos parches aparecen cada 2 o 3 meses para tapar los fallos, son la versiones 2.5a y 2.5b. En Marzo de 1994 llega FoxPro 2.6 que promulga una mayor compatibilidad con dBase e incluye por primera ver diversos Wizars. En 1995 se produce un hecho importante, Dave Fulton abandona Microsoft. Salen varios parches más hasta Julio 1999, momento en que MS anuncia que dejará de dar soporte a FoxPro 2.x, pasando a incluirlo en su lista de productos obsoletos y centrándose en Visual FoxPro como su sucesor natural hasta el momento actual.

Tomado de: http://software.alanit.com/articulos/20030124.htm

Delphi

En el año 1995 se crea el nuevo sucesor de Pascal, al que se llamó Delphi, siendo la primera herramienta con un entorno de desarrollo visual construida por Borland. Está caracterizado por ser un lenguaje orientado a eventos, es decir, que la ejecución del programa no es secuencial, sino que depende de los eventos que suceden durante la ejecución de la aplicación.

Delphi es una herramienta de Desarrollo Rápido de Aplicaciones (RAD). Los componentes que incorpora facilitan el acceso a bases de datos, comunicación a través de Internet, calidad en impresiones, desarrollo de aplicaciones multimedia, enlaces DDE, componentes OLE y VBX, etc. Borland ha introducido al mercado varias versiones de Delphi, aportando mejoras notables, entre las que cabe destacar el CodeInsight, un asistente que muestra automáticamente las listas de parámetros de procedimientos, métodos y eventos.

En el año 2001 Borland lanzó al mercado la versión de Delphi 6.0 que funciona bajo Windows y es compatible con todas las versiones anteriores. Junto con esta versión se introdujo en el mercado la primera versión Kylix, una versión de Delphi que funciona bajo Linux. La última versión disponible en el mercado es Delphi 7.0. Entre las nuevas características se incluye un nuevo compilador que permite construir aplicaciones basadas en la plataforma .NET

Tomado del Paper: Elementos de Programación E.T.S.I. Telecomunicación, 1ºC, André Ormset.

Anexo 3
BENCHMARKING DE SOFTWARE EAI
(ENTERPRISE APPLICATION INTEGRATION)

Recopilación de información de los diferentes productos de Integración de aplicaciones, tomados desde los diferentes Websites de los fabricantes:

Tabla A3.1 Características de Bistalk Server

Nro	Biztalk Server
1	Automatización de procesos y servicios de gestión.
2	Servicios que soportan la integración de aplicaciones y orígenes de datos en procesos de negocio racionalizados tanto internamente como entre organizaciones.
3	Servicios y adaptadores que dan soporte a la integración de aplicaciones y tecnologías dentro de los procesos de negocio.
4	Aceleradores de soluciones verticales y sectoriales.
5	Monitorización de la actividad de negocio.
6	Completo motor de reglas de negocio.
7	Servicios de transporte y enrutamiento de documentos.
8	Servicios de transformación de documentos. Servicios que posibilitan la transformación de datos entre los distintos formatos utilizados por diferentes aplicaciones de negocio y comunidades comerciales, como XML, EDI, RosettaNet y otros muchos.
9	Servicios de análisis de datos. Servicios para combinar, acceder y analizar datos operativos mediante herramientas gráficas y utilidades automatizadas de minería de datos para descubrir patrones y tendencias.
10	Servicios de configuración y administración. Permiten la monitorización, gestión y administración automatizadas de la solución de integración del proceso productivo.
11	Servicios de integración de la plataforma Microsoft Windows. Servicios básicos de plataforma que soportan escenarios de interoperabilidad con servicios de directorio, datos, seguridad y sistemas operativos para una infraestructura empresarial heterogénea.
12	Servicios de escalabilidad. Servicios que soportan la escalabilidad de la solución de integración en sentido vertical y horizontal para cubrir los volúmenes requeridos por las mayores empresas.
13	Servicios de integración de seguridad. Servicios que permiten la configuración directa de las credenciales de seguridad entre dominios empresariales heterogéneos.

Tabla A3.2 Características de WebSphere MQ

Nro	WebSphere MQ
1	Conectividad heterogénea, desde equipos de escritorio hasta macrocomputadoras (admite más de 35 plataformas distintas).
2	Dispone de una extensa familia de interfaces de programación de aplicaciones (API) diseñados con el objeto de facilitar la creación de código para tareas de mensajería.
3	Garantiza la entrega sin duplicaciones de los mensajes importantes
4	Soporte de la plataforma Linux para iSeries y Linux para pSeries POWER.
5	Interfaces de 64 bits en AIX, HP-UX y Solaris.
6	Soporte al protocolo de Internet versión 6 (IPv6).
7	Soporte integrado para los datos de Web Services en un transporte MQ fiable.
8	Prestaciones de publicación/suscripción integradas.
9	Una interfaz ampliable de usuario para la configuración basada en Eclipse en las plataformas Microsoft Windows y Linux x86.
10	El producto Quick Tour que incorpora los conceptos básicos de la conectividad de aplicaciones y describe las funciones y prestaciones de WebSphere MQ; el programa de utilidad File Transfer (en las plataformas Microsoft Windows y Linux x86) proporciona un método sencillo para enviar datos (mensajes) de un sistema MQ a otro.
11	Soporte para IBM Message Service Client for C/C++.
12	WebSphere MQ (anteriormente MQSeries), el punto central de la familia MQ, proporciona conectividad de aplicaciones. Puede utilizarse de forma autónoma o con otros miembros de la familia para ofrecer una solución de integraciones de negocio global.

Tabla A3.3 Características de WebMethods

Nro	WebMethods
1	Registro de servicios
2	Descubrimiento dinámico
3	Gestión
4	Recuperación automática de fallos
5	Seguridad distribuida
6	Procesado de mensajes XML
7	Monitorización de los procesos
8	Auditorías, avisos de errores, notificaciones de sucesos
9	Manejo de excepciones
10	Clustering y una consola de gestión gráfica en tiempo real basada en un navegador

Tabla A3.4 Características de TIBCO BusinessWorks

Nro	TIBCO BusinessWorks
1	Soporte para estándares B2B
2	Soporte Nativo para intercambio de documentos XML
3	Entrega de Mensajería a través de rango amplio o mecanismos de transporte
4	Conectividad a los principales mercados electrónicos
5	Soporte para autenticación, encriptación y no repudio de tecnologías
6	Integración transparente con aplicaciones empresariales usando software de integración de TIBCO
7	Auditoria complete y funcionalidad de no repudio a las auditorias.

Tabla A3.5 Características de TIBCO SAP NetWeaver

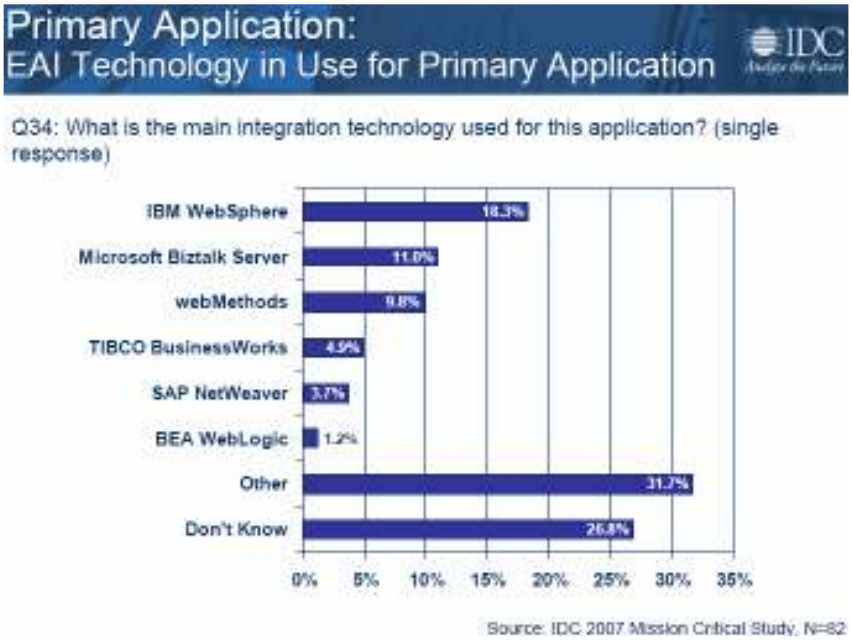
Nro	SAP NetWeaver
1	SAP NetWeaver, una completa plataforma de integración y aplicaciones que se integra con la infraestructura informática existente para hacer posible el cambio y su administración.
2	Con SAP NetWeaver, se puede diseñar, construir, implementar y ejecutar nuevas estrategias y procesos de negocio con flexibilidad y rapidez. También puede fomentar la innovación en toda su empresa combinando los sistemas existentes y manteniendo, al mismo tiempo, una estructura de costos sustentable.
3	SAP NetWeaver puede integrarse con estándares de Internet tales como HTTP, XML y servicios Web. Garantiza apertura e interoperabilidad con Microsoft .NET y Java 2 Platform Enterprise Edition (J2EE) y entornos como IBM WebSphere.

Tabla A3.6 Características de Bea WebLogic

Nro	Bea WebLogic
1	Proporciona un enfoque simplificado para asegurar las soluciones de integración y para aplicar los roles de seguridad sin necesidad de escribir código.
2	Los administradores pueden implementar fácilmente políticas dinámicas de seguridad durante el tiempo de ejecución.
3	Para el desarrollo, la infraestructura altamente disponible y con auto-corrección de BEA WebLogic Server, mantiene el estado de la aplicación incluso durante un fallo del sistema.
4	La innovadora arquitectura de replicación en memoria permite que los sistemas respondan automáticamente a los fallos sin que el éxito de su solución de integración se vea afectado.
5	La arquitectura J2EE de BEA empaqueta los recursos en controles: componentes de funcionalidad empresarial ligeros que conectan recursos empresariales a través de múltiples proyectos de integración, portal y desarrollo. Así, puede compartir y aprovechar trabajo previo, efectuar modificaciones rápidas y entregar proyectos de integración en mucho menos tiempo.

En la Figura siguiente se Muestra la Preferencia de las en una encuesta: 2007 Mission Critical North American- Application Platform Study realizada por IDC www.idc.com, en la cual se puede apreciar la fuerte acogida que tienen los productos de Microsoft (Biztalk server) e IBM (IBM Websphere), seguido por fabricantes independientes como WebMethods, TIBCO. Luego SAP para integrar aplicaciones a su producto SAP ERP, y BEA como servidor de Aplicaciones y de Integración.

Figura A3.1 Estadística del uso de tecnología de integración



Anexo 4

TABLAS DE ARCHIVOS DE LAS PRUEBAS REALIZADAS EN EL CASO DE ESTUDIO DE LA EMPRESA CTM TOURS

Tabla A4.1 Archivos que pudieron ser leídos con éxito

Nro	**Nombre archivo**	**NroCampos**	**NroRegistros**
1	Hotinv.dbf	14	676202
2	Optinv.dbf	12	127037
3	Reminder.dbf	7	50701
4	Hotprc.dbf	19	38648
5	Confirm.dbf	27	14725
6	R_price.dbf	33	10670
7	Voucher.DBF	44	9025
8	R_flag.dbf	6	8652
9	Accpay.dbf	28	7640
10	Je_detal.dbf	6	6331
11	P_date.dbf	6	4721
12	Rmtype.dbf	2	3348
13	Accrcv.dbf	21	3327
14	R_prog.dbf	8	3262
15	R_stop.dbf	12	3254
16	Je_entry.dbf	15	3238
17	Receive.dbf	28	3015
18	Cruprc.dbf	14	2914
19	Cruinv.dbf	14	2782
20	In_price.dbf	31	2463
21	Vprice.dbf	73	2047
22	Location.dbf	4	1894
23	Confirmd.dbf	10	1287
24	R_seqno.dbf	3	783
25	In_xnote.dbf	2	761
26	R_qpp.dbf	13	554
27	In_prog.dbf	8	479
28	In_stop.dbf	12	459
29	P_price.dbf	15	444
30	P_item.dbf	36	342
31	Rmdrint.dbf	7	333
32	R_xnotes.dbf	2	313
33	Bpjodetl.dbf	7	309
34	Modlist1.dbf	1	260
35	Batchrcv.dbf	15	187
36	Ptdep.dbf	3	174
37	Bpaypmt.dbf	52	155
38	Payment.dbf	20	155

Nro	Nombre archivo	NroCampos	NroRegistros
39	Bpay.dbf	6	145
40	Account.dbf	10	135
41	Ship_info.dbf	15	120
42	R_accm.DBF	38	103
43	Accm.dbf	7	86
44	Ar_note.dbf	5	78
45	Clpass.dbf	14	75
46	P_stop.dbf	15	68
47	User.dbf	19	57
48	Config.DBF	12	51
49	Cartype.dbf	1	45
50	Catagory.dbf	2	43
51	Sabfkdef.dbf	2	41
52	Ap_note.dbf	5	40
53	Depmnt0.dbf	5	35
54	Id3.dbf	5	35
55	Dbupd.dbf	6	29
56	Prgcms.dbf	3	26
57	Optinvls.dbf	4	23
58	Vconfirm.dbf	3	22
59	Deck.dbf	2	21
60	Funcname.dbf	2	19
61	Opcode.dbf	1	15
62	Asource.dbf	1	14
63	Hotcls.dbf	2	14
64	Glclass.dbf	1	11
65	R_pers.dbf	17	10
66	Bkupdir.dbf	1	9
67	Card.dbf	6	9
68	Pr_wdesc.DBF	3	9
69	Tax.dbf	6	9
70	Sspswd.dbf	5	8
71	Zoneprc.dbf	29	8
72	Bchcfg.dbf	7	7
73	Contact.dbf	7	7
74	In_qpp.dbf	13	7
75	Tacbuy.dbf	18	7
76	Taccm.dbf	6	7
77	Userx.dbf	9	7
78	Credname.dbf	1	6
79	Invhost.dbf	2	6
80	Oacbuy.dbf	18	6
81	Tmp_vnd.dbf	2	6

Nro	**Nombre archivo**	**NroCampos**	**NroRegistros**
82	Astatus.dbf	4	5
83	Batjour.dbf	8	5
84	Paymetho.dbf	1	5
85	Ccname.dbf	1	4
86	Cstatus.dbf	2	4
87	Curr.dbf	2	4
88	Inxnotes.dbf	2	4
89	Method.dbf	2	4
90	Billitem.dbf	13	3
91	Defacc.dbf	4	3
92	Or_item.dbf	54	3
93	Or_passa.dbf	12	3
94	Postlog.dbf	5	3
95	R_enpric.dbf	22	3
96	Zonetmp.dbf	44	3
97	Adfee.dbf	4	2
98	Carinv.dbf	12	2
99	Country.dbf	1	2
100	Dbver.dbf	5	2
101	Glrpt.dbf	5	2
102	Id.dbf	65	2
103	Insetup.dbf	70	2
104	Or_stop.dbf	12	2
105	P_fix.dbf	13	2
106	Packdir.dbf	3	2
107	Personal.dbf	40	2
108	Pgavchk.dbf	5	2
109	Qaclass.dbf	1	2
110	Statemnt.dbf	14	2
111	System.dbf	40	2
112	Accdate.dbf	3	1
113	Accheck.dbf	6	1
114	Agncms.dbf	21	1
115	Agtljm.dbf	31	1
116	Air2rout.dbf	25	1
117	Airinv.dbf	12	1
118	Airitem.dbf	10	1
119	Airitem2.dbf	10	1
120	Airline.dbf	54	1
121	Airprc.dbf	27	1
122	Airroute.dbf	24	1
123	Amanda.dbf	10	1
124	Ana.dbf	10	1

Nro	**Nombre archivo**	**NroCampos**	**NroRegistros**
125	Ana2.dbf	10	1
126	Aoife.dbf	5	1
127	Aticket.dbf	5	1
128	Audrey.dbf	5	1
129	Bchdat.dbf	9	1
130	Bchrqs.dbf	10	1
131	Brochure.dbf	2	1
132	Bus.DBF	40	1
133	Cardays.dbf	22	1
134	Carolyn.dbf	5	1
135	Catgory.dbf	3	1
136	Ccard.dbf	1	1
137	Cheryl.dbf	5	1
138	Chkbil.dbf	7	1
139	Chueca.dbf	10	1
140	Cob.dbf	5	1
141	Conor.dbf	5	1
142	Cp_date.dbf	5	1
143	Crdvrf.dbf	15	1
144	Credit.dbf	12	1
145	Crs.dbf	3	1
146	Cstjdetl.dbf	13	1
147	Cstrpt.dbf	2	1
148	Ctmfl.dbf	10	1
149	Ctmfll.dbf	10	1
150	Ctmhea.dbf	10	1
151	Ctmlax.dbf	10	1
152	Ctmnj.dbf	10	1
153	Ctmny.dbf	10	1
154	Ctmperu.dbf	10	1
155	Cur_dt.dbf	2	1
156	Currency.dbf	3	1
157	Dbpath.dbf	4	1
158	Debi.dbf	5	1
159	Depmnt.DBF	24	1
160	Dept.dbf	3	1
161	Doris.dbf	10	1
162	Dottie.dbf	5	1
163	Elaine.dbf	5	1
164	En_price.dbf	21	1
165	Fatima.dbf	10	1
166	Gaccpay.dbf	27	1
167	Gaccrcv.dbf	21	1

Nro	**Nombre archivo**	**NroCampos**	**NroRegistros**
168	Gbill.dbf	19	1
169	Gbillitm.dbf	11	1
170	General.dbf	6	1
171	Gpayment.dbf	21	1
172	Greceive.dbf	22	1
173	Group.dbf	3	1
174	Grp.dbf	14	1
175	Grpar.dbf	18	1
176	Grpfree.dbf	4	1
177	Grppay.dbf	27	1
178	Grprcv.dbf	22	1
179	Guide.dbf	23	1
180	Hotseat.dbf	10	1
181	I_enpric.dbf	22	1
182	I_enprice.dbf	22	1
183	I_enprice.dbf	22	1
184	Icvsubfn.dbf	4	1
185	Ifen.dbf	5	1
186	In_accm.dbf	37	1
187	In_aitn.DBF	26	1
188	In_apass.dbf	5	1
189	In_pers.dbf	17	1
190	In_rail.dbf	30	1
191	In_xnotes.dbf	2	1
192	Invoice.DBF	19	1
193	Je_auto.dbf	4	1
194	John.dbf	5	1
195	Kay.dbf	5	1
196	Keith.dbf	5	1
197	Kelly.dbf	5	1
198	Kkaren.dbf	10	1
199	L_price.dbf	13	1
200	Level3.dbf	5	1
201	Level4.dbf	5	1
202	Level5.dbf	5	1
203	Lisa.dbf	5	1
204	Login.dbf	4	1
205	Loginwww.dbf	4	1
206	Mark.dbf	5	1
207	Marty.dbf	5	1
208	Merchant.dbf	2	1
209	Michelle.dbf	5	1
210	Nuala.dbf	5	1

Nro	**Nombre archivo**	**NroCampos**	**NroRegistros**
211	Octavia.dbf	5	1
212	Odila.dbf	10	1
213	Op_qpp.dbf	10	1
214	Optlist.dbf	5	1
215	Orla.dbf	5	1
216	P_prog.DBF	4	1
217	P_seat.dbf	8	1
218	P_stmp.dbf	2	1
219	Phil.dbf	5	1
220	Prgseg.dbf	5	1
221	Printer.dbf	8	1
222	Protour.dbf	5	1
223	Pt.dbf	5	1
224	R_aitn.DBF	26	1
225	R_apass.dbf	5	1
226	R_enprice.dbf	22	1
227	R_notes.dbf	2	1
228	R_person.dbf	7	1
229	R_rail.dbf	30	1
230	Railcomm.dbf	5	1
231	Railpass.dbf	12	1
232	Railprc.dbf	11	1
233	Railtkt.dbf	46	1
234	Railval.dbf	1	1
235	Railzone.dbf	6	1
236	Ramirez.dbf	10	1
237	Recbatch.dbf	14	1
238	Refund.dbf	15	1
239	Revenue.dbf	7	1
240	Revjdetl.dbf	16	1
241	Revjenty.dbf	14	1
242	Ribeiro.dbf	10	1
243	Ronan.dbf	5	1
244	Rptlst.dbf	4	1
245	Ruth.dbf	10	1
246	Ruth.dbf	10	1
247	Saler.dbf	24	1
248	Sandy.dbf	5	1
249	Seat.dbf	10	1
250	Siobhan.dbf	5	1
251	Solange.dbf	10	1
252	Steve.dbf	5	1
253	Template.dbf	4	1

Nro	**Nombre archivo**	**NroCampos**	**NroRegistros**
254	Teresa.dbf	5	1
255	Thdinv.dbf	13	1
256	Thdprc.dbf	12	1
257	Thdprt.dbf	41	1
258	Ticket.dbf	10	1
259	Tracey.dbf	5	1
260	Tracy.dbf	5	1
261	Triona.dbf	5	1
262	Tritem.DBF	85	1
263	Vouchers.dbf	5	1
264	Xagent.dbf	12	1
265	Yarelis.dbf	10	1
266	Ztable.dbf	43	1

Tabla A4.2 Archivos que no pudieron ser leídos con éxito

Nro	**Nombre archivo**	**NroCampos**	**Descripción**
1	Reserv.dbf	68	Failed to parse Number: For input string: "Percentage" FIN RESERV.DBF
2	Res.dbf	63	Failed to parse Number: For input string: "0.0000A" FIN res.dbf
3	Tpgm.dbf	53	Failed to parse Number: For input string: "16:47:22UU" FIN tpgm.dbf
4	Tmpcln1.dbf	41	Failed to parse Number: For input string: "S" FIN tmpcln1.dbf
5	Zonegp.dbf	45	Failed to parse Number: For input string: "ZN1000" FINZONEGP.DBF
6	Vendor.dbf	45	Failed to parse Number: For input string: "St" FIN VENDOR.DBF
7	R_passan.dbf	32	Failed to parse Number: For input string: "F" FIN R_PASSAN.DBF
8	R_item.dbf	62	Failed to parse Number: For input string: "VE2" FIN R_ITEM.DBF
9	Quickadd.DBF	50	Failed to parse Number: multiple points FIN quickadd.DBF
10	Program.dbf	69	Failed to parse Number: For input string: "F" FIN PROGRAM.DBF
11	Option.dbf	55	Failed to parse Number: For input string: "." FIN OPTION.DBF
12	Agent-old.dbf	41	Failed to parse Number: For input string: "Percentage" FIN AGENT-OLD.DBF
13	Agent.dbf	41	Failed to parse Number: For input string: "Percentage" FIN AGENT.DBF
14	Agency.dbf	41	Failed to parse Number: For input string: "Percentage" FIN AGENCY.DBF
15	Allclt.dbf	54	Failed to parse Number: For input string: "N" FIN ALLCLT.DBF
16	In_passa.dbf	28	Failed to parse Number: For input string: "/P" FIN IN_PASSA.DBF
17	In_item.dbf	60	Failed to parse Number: For input string: "VE3" FIN IN_ITEM.DBF
18	Itn_air.dbf	21	Failed to parse Number: empty String FIN itn_air.dbf
19	Modlist2.dbf	55	Failed to parse Number: For input string: "2N" FIN modlist2.dbf
20	Inquery.dbf	66	Failed to parse Number: multiple points FIN INQUERY.DBF
21	Hotel.dbf	62	Failed to parse Number

Nro	**Nombre archivo**	**NroCampos**	**Descripción**
22	Bill.DBF	23	Failed to parse Number: For input string: "US$" FIN bill.DBF
23	Brcvdtel.dbf	62	Failed to parse Number: For input string: "30.00-" FIN BRCVDTEL.DBF
24	Carrent.dbf	56	Failed to parse Number: For input string: "CI" FIN CARRENT.DBF
25	Client.dbf	54	Failed to parse Number: For input string: "N" FIN CLIENT.DBF
26	Check.DBF	19	Failed to parse Number: For input string: "US$" FIN check.DBF
27	Cruise.dbf	55	Failed to parse Number: For input string: "-AIPL" FIN CRUISE.DBF
28	Foxuser.dbf	7	Failed to parse Number: For input string: "F4" FIN foxuser.dbf
29	Ditem.dbf	85	Failed to parse Number: For input string: "F" FIN ditem.dbf

Anexo 5

TENDENCIAS TECNOLOGICAS Y EMPRESARIALES EN EL DESARROLLO DE SISTEMAS DE INFORMACION

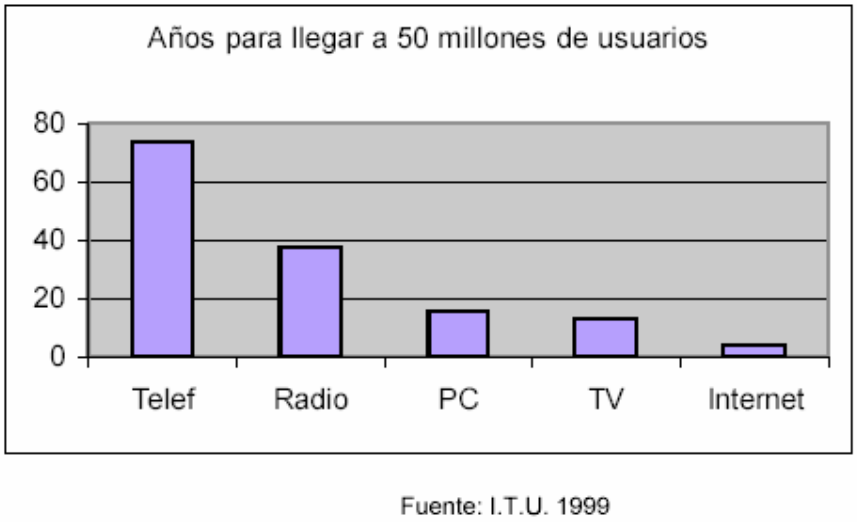
La tecnología de nuestros días cambia rápidamente la forma de realizar la gestión empresarial, y también el desarrollo de las organizaciones, y esto a su vez recae en la forma en que se plantean las soluciones informáticas; en el presente capitulo se muestran estas tendencias que rigen hoy en día.

1. Visión Empresarial

"Hoy en día en el mundo de los negocios la única constante son los cambios vertiginosos que tienen lugar. Los departamentos de Tecnología de la Información se ven presionados para encontrar soluciones flexibles que aprovechen al máximo las nuevas tecnologías manteniendo el valor de los sistemas tradicionales" [33].

La era en línea es aquella en la que estamos inmersos hoy en día, el Internet es un actor en el día a día de las personas y organizaciones, su crecimiento ha sido tal que no hay otro medio que lo iguale, ni el teléfono ni la televisión han logrado lo que Internet ha logrado en tan pocos años, como lo muestra el siguiente cuadro.

Figura A5.1 Comparación de Internet frente a otras Tecnologías



Fuente: I.T.U. 1999

Es por ello que el desarrollo de tecnologías para Internet, se está desarrollando cada vez con mayor fuerza, las redes ubicuas donde cada dispositivo pueda tener conexión a Internet, con nuevas y mejores funcionalidades, la idea es estar siempre conectado no importa en qué lugar uno se encuentre.

Los negocios actuales están obligados a al menos tener un sitio Web para que sus clientes al menos puedan ver sus productos y servicios.

2. E-Business

"Se refiere no solo a la compra y venta de productos o servicios, sino también al servicio a clientes, colaboración entre socios comerciales, transacciones intra-organizacionales o con el gobierno, la reingeniería de procesos y, los servicios comunitarios. Todo ello en una escala local, nacional, regional y/o global" [27].

Se debe diferenciar entre los negocios netamente electrónicos, y aquellos que detrás tienen toda una infraestructura montada años atrás, las primeras también llamadas empresas ".com" en un momento tuvieron un gran auge, en la década de los noventas, con una fuerte presencia en la bolsa de valores, pero luego sufrieron una caída estrepitosa que las hizo casi desaparecer porque no tenían sustento o soporte detrás del portal que mostraban a los clientes, las segundas se fortalecieron tras la caída de las primeras y se mantienen en la actualidad.

Algunos de los principales beneficios que las empresas que tienen participación en Internet están alcanzado, son los siguientes:

2.1 Mayor captación de clientes

Internet es un medio completamente globalizado que puede ser utilizado por las empresas para llegar potencialmente a millones de clientes. Esta herramienta habilita a las empresas para brindar atención a sus clientes actuales así como a clientes nuevos. El alcance de cualquier sitio Web depende de la estrategia online que la empresa lleve a cabo. Entre los principales factores de un sitio Web que influyen directamente para captar una mayor cantidad de visitantes se encuentran:

- Percepción de valor por parte de los visitantes
- Buen posicionamiento en buscadores
- Presencia en directorios (en categorías de acuerdo con el ramo de la empresa)
- Versiones en varios idiomas
- Mercadotecnia en línea
- Presencia de la empresa en el mercado

No hay que olvidar que un sitio Web compite con otros miles de sitios por la atención de los visitantes del Web, por lo tanto, es necesario ofrecer un valor diferenciador del resto para captar la atención de manera significativa.

2.2 Mejor comunicación interna

Una de las características más importante de Internet y el World Wide Web es la versatilidad con la que cuentan para establecer comunicaciones entre las personas. Esta propiedad puede ser aprovechada por las empresas para implementar soluciones que permitan establecer y organizar de manera más eficiente la comunicación entre colaboradores y directivos. Para este propósito, muchas de las empresas importantes crean *intranets* (conjunto de aplicaciones que son utilizados de manera interna) con una gran cantidad de servicios para los colaboradores y mediante los cuales, los directivos pueden comunicar y

recibir retroalimentación. Este tipo de herramientas enriquece mucho las relaciones interpersonales e interdepartamentales, así también como el trabajo en equipo y el involucramiento de todos los integrantes de la organización.

2.3 Ahorros en operación

Para nadie es un secreto la gran eficiencia de herramientas como el correo electrónico y los grandes ahorros económicos y de tiempo que ha brindado a las empresas desde su aparición. El uso de esta relativamente sencilla herramienta ha disminuido de manera dramática el tiempo en la entrega de mensajes entre personas distantes geográficamente, así mismo ha disminuido en gran proporción el uso del papel, líneas telefónicas y de fax. Otras herramientas como FTP (File Transfer Protocol / Protocolo de Transferencia de Archivos) que permiten transferir una gran cantidad de información de forma sumamente veloz entre computadoras distantes han sustituido en gran medida el envío de paquetes de documentos entre la empresa y sus sucursales, aliados, clientes y proveedores.

De manera similar la implantación de un Sitio Web puede traer consigo una gran cantidad de ahorros para las organizaciones. Esta herramienta permite a la empresa hacer publicidad de manera mucho más económica que con otros medios y le da la posibilidad de llegar a un mercado mucho más amplio. La empresa tiene también la posibilidad de sustituir un alto porcentaje de catálogos impresos por catálogos electrónicos incorporados al sitio Web que pueden ser mucho más funcionales y definitivamente más económicos. El proceso de atención de clientes en línea puede disminuir costos en llamadas telefónicas y en personal de soporte, además el horario de atención es de 24 horas al día. Así mismo, procesos como la elaboración y envío de cotizaciones, el contacto con el cliente y la venta de productos pueden ser realizados mediante el sitio Web de manera más eficiente.

2.4 Incremento en sus ventas

Una de las herramientas que puede generar un mayor y más rápido retorno de inversión a las empresas es el Comercio Electrónico. Vender en línea trae grandes beneficios tanto para la empresa como para sus clientes, debido al bajo costo por transacción, el alcance y la comodidad para los clientes. Las empresas que venden en línea aprovechan un nuevo canal de ventas alternativo a los tradicionales y dependiendo de la Industria, la proporción de ventas en línea respecto a las ventas por otros conductos puede llegar a ser muy significativa para algunas empresas.

2.5 Mejora en la imagen corporativa

Para cualquier empresa, contar con un sitio Web funcional significa una mejora en la percepción por parte de los clientes para la organización. Los consumidores gustan de las empresas que procuran utilizar nuevas estrategias y

herramientas para brindar un mejor servicio, además es un excelente medio para establecer contacto con la organización a cualquier hora y desde cualquier lugar del mundo.

2.6 Mejora en la atención a los clientes

Las posibilidades para brindar nuevos servicios en línea para los clientes son inagotables y un sitio Web puede incorporar una gran cantidad de herramientas encaminadas a mejorar la atención de los clientes. Algunas de estas son las siguientes:

- Formas de contacto
- Preguntas Frecuentes
- Herramientas para cotización
- Soporte en línea (live Chat)
- Bases de conocimiento
- Correo electrónico

2.7 Mayor conocimiento del Mercado

Una función muy importante de los sitios Web es la posibilidad de investigar su mercado, con el fin tener bases para la mejora continua de su contenido así como de los servicios que ofrece. Existen varias herramientas que se utilizan con gran éxito para llevar a cabo esta tarea [34]:

- Encuestas
- Preguntas rápidas
- Formas de registro de usuario
- Formas de contacto
- Grupos de discusión
- Chat
- Soporte en línea
- Personalización
- Estadísticas del servidor
- Órdenes de compra

3. E-Commerce

"Cualquier forma de transacción comercial donde las partes interactúan electrónicamente, en lugar del intercambio físico" [27].

El Comercio electrónico, es una forma avanzada de comercio a distancia, que utiliza para su funcionamiento Internet, sistemas computacionales y sistemas de Telecomunicaciones, que permiten que una transacción comercial se lleve a cabo sin que sea necesaria la presencia física en el mismo lugar del comparador y del vendedor.

Cualquier empresa que se integra al comercio electrónico pasa a ser parte de un mercado mundial, donde pueden hallar información, comprar y vender sin moverse de su computador o unidad de acceso a Internet.

Según Gartner Group. Estimó que en el año 2003, el e-commerce condujo a una reestructuración masiva de la industria, afectando la comercialización, el servicio de la venta y del cliente, y por supuesto las cosas que rodean al producto mismo, tales como la logística y el transporte por citar un ejemplo.

3.1 Beneficios [27]

Presencia global/elección global: Los límites del comercio electrónico no están definidos por fronteras geográficas o nacionales, sino por la cobertura de las redes de computadoras.

Aumento de la competitividad/calidad del servicio: El comercio electrónico permite a los proveedores aumentar la competitividad y estar más cerca de sus Clientes.

Adecuación generalizada/productos y servicios personalizados: Con la interacción electrónica, los proveedores pueden tener información detallada de las necesidades de cada cliente individual y automáticamente ajustar sus productos y servicios.

Cadenas de entrega más cortas o inexistentes/respuesta rápida a las necesidades: El comercio electrónico permite a menudo reducir de manera drástica las cadenas de entrega. El beneficio por parte del cliente es la posibilidad de obtener rápidamente el producto preciso que necesita, sin estar limitado a los niveles de inventario, actuales del distribuidor local.

Reducción de costos/reducción de precios: Una de las mayores contribuciones del comercio electrónico es la reducción de los costos de transacción. Los costos son menores, por cuanto no se necesita realizar gastos de mantenimiento, personal y suministros de una tienda real.

Nuevas oportunidades de negocio/nuevos productos y servicios: Además de la redefinición de mercados para productos y servicios existentes, el comercio electrónico también proporciona productos y servicios completamente nuevos.

Tipos de Comercio electrónico

Se puede distinguir los siguientes tipos:

- Comercio Electrónico Business to Consumer (B2C)
- Comercio Electrónico Business to Business (B2B)
- Comercio Electrónico Consumer to Consumer (C2C)
- Comercio Electrónico Business to Government (B2G)

4. Tendencias Tecnológicas

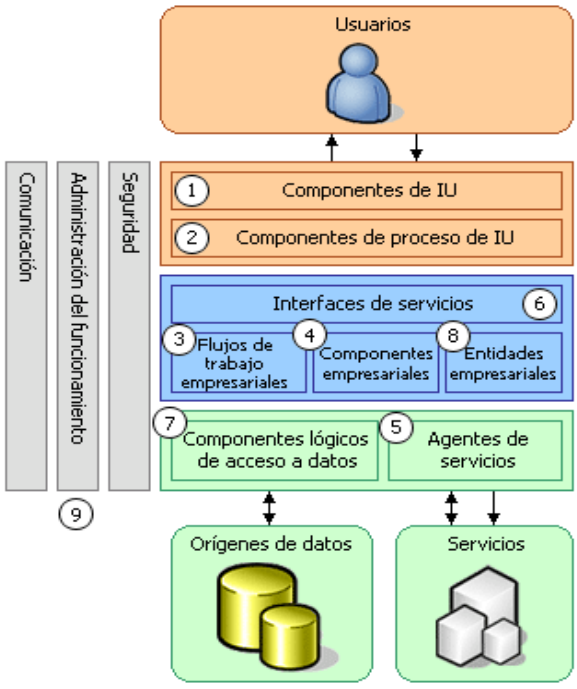
4.1 Arquitectura en N Capas

El diseño de la arquitectura en N capas esta fundamenta en la especialización y división de los componentes según funcionalidad y propósito, de lo que se trata es de no mezclar elementos que tienen diferentes objetivos en uno solo, es por eso que se denomina arquitectura por N capas que separa aspectos funcionales de los componentes en cada capa, así se logra un menor acoplamiento y una alta cohesión.

La más común es la arquitectura de 3 capas, que separa los aspectos de acceso a datos, lógica de negocio y las interfaces de usuario.

Podemos apreciar esta arquitectura en la siguiente figura:

Figura A5.2 Modelo de Arquitectura en 3 Capas [33]



4.1.1 Componentes de Interfaz de Usuario

La mayor parte de las soluciones necesitan ofrecer al usuario un modo de interactuar con la aplicación. En el ejemplo de aplicación comercial, un sitio Web permite al cliente ver productos y realizar pedidos, y una aplicación basada en el entorno de escritorio permite a los representantes de ventas por ejemplo escribir los datos de los pedidos de los clientes que han telefoneado a la empresa. Las interfaces de usuario pueden ser implementados utilizando formularios de Windows Forms, Web Forms ASP.NET, JSP etc., controles u otro tipo de tecnología que permita procesar y dar

formato a los datos de los usuarios, así como adquirir y validar los datos entrantes procedentes de éstos [33].

4.1.2 Componentes de Proceso de Usuario

En un gran número de casos, la interacción del usuario con el sistema se realiza de acuerdo a un proceso predecible. Por ejemplo, en la aplicación comercial, podríamos implementar un procedimiento que permita ver los datos del producto. De este modo, el usuario puede seleccionar una categoría de una lista de categorías de productos disponibles y, a continuación, elegir uno de los productos de la categoría seleccionada para ver los detalles correspondientes. Del mismo modo, cuando el usuario realiza una compra, la interacción sigue un proceso predecible de recolección de datos por parte del usuario, por el cual éste en primer lugar proporciona los detalles de los productos que desea adquirir, a continuación los detalles de pago y, por último, la información para el envío. Para facilitar la sincronización y organización de las interacciones con el usuario, resulta útil utilizar componentes de proceso de usuario individuales. De este modo, el flujo del proceso y la lógica de administración de estado no se incluyen en el código de los elementos de la interfaz de usuario, por lo que varias interfaces podrán utilizar el mismo "motor" de ínter actuación básica [33].

4.1.3 Componentes Empresariales

Independientemente de si el proceso empresarial consta de un único paso o de un flujo de trabajo organizado, la aplicación requerirá probablemente el uso de componentes que implementen reglas empresariales y realicen tareas empresariales. Por ejemplo, en la aplicación comercial, deberá implementar una funcionalidad que calcule el precio total del pedido y agregue el costo adicional correspondiente por el envío del mismo. Los componentes empresariales implementan la lógica empresarial de la aplicación [33].

4.1.4 Agentes de Servicios

Cuando un componente empresarial requiere el uso de la funcionalidad proporcionada por un servicio externo, tal vez sea necesario hacer uso de código para administrar la semántica de la comunicación con dicho servicio. Por ejemplo, los componentes empresariales de la aplicación comercial descrita anteriormente podrían utilizar un agente de servicios para administrar la comunicación con el servicio de autorización de tarjetas de crédito y utilizar un segundo agente de servicios para controlar las conversaciones con el servicio de mensajería. Los agentes de servicios permiten aislar las idiosincrasias de las llamadas a varios servicios desde la aplicación y pueden proporcionar servicios adicionales, como la asignación básica del formato de los datos que expone el servicio al formato que requiere la aplicación [33].

4.1.5 Interfaces de Servicios

Para exponer lógica empresarial como un servicio, es necesario crear interfaces de servicios que admitan los contratos de comunicación (comunicación basada en mensajes, formatos, protocolos, seguridad y excepciones, entre otros) que requieren los clientes. Por ejemplo, el servicio de autorización de tarjetas de crédito debe exponer una interfaz de servicios que describa la funcionalidad que ofrece el servicio, así como la semántica de comunicación requerida para llamar al mismo. Las interfaces de servicios también se denominan fachadas empresariales [33].

4.1.6 Componentes Lógicos de Acceso a Datos

La mayoría de las aplicaciones y servicios necesitan obtener acceso a un almacén de datos en un momento determinado del proceso empresarial. Por ejemplo, la aplicación empresarial necesita recuperar los datos de los productos de una base de datos para mostrar al usuario los detalles de los mismos, así como insertar dicha información en la base de datos cuando un usuario realiza un pedido. Por tanto, es razonable abstraer la lógica necesaria para obtener acceso a los datos en una capa independiente de componentes lógicos de acceso a datos, ya que de este modo se centraliza la funcionalidad de acceso a datos y se facilita la configuración y el mantenimiento de la misma [33].

4.1.7 Componentes de Entidad Empresarial

La mayoría de las aplicaciones requieren el paso de datos entre distintos componentes. Por ejemplo, en la aplicación comercial es necesario pasar una lista de productos de los componentes lógicos de acceso a datos a los componentes de la interfaz de usuario para que éste pueda visualizar dicha lista. Los datos se utilizan para representar entidades empresariales del mundo real, como productos o pedidos. Las entidades empresariales que se utilizan de forma interna en la aplicación suelen ser estructuras de datos, como conjuntos de datos, o secuencias de lenguaje de marcado extensible (XML), aunque también se pueden implementar utilizando clases orientadas a objetos personalizadas que representan entidades del mundo real necesarias para la aplicación, como productos o pedidos [33].

4.1.8 Componentes de Seguridad, Administración Operativa y Comunicación

La aplicación probablemente utilice también componentes para realizar la administración de excepciones, autorizar a los usuarios a que realicen tareas determinadas y comunicarse con otros servicios y aplicaciones [33].

4.2 XML y Servicios Web

Los servicios Web XML permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado las aplicaciones, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles los dispositivos utilizados para obtener acceso a ellas. Aunque los servicios Web XML son independientes entre sí, pueden vincularse y formar un grupo de colaboración para realizar una tarea determinada.

Bien, esta puede ser una primera definición de Servicio Web. Pero, quizá, lo que ahora más interese es saber ¿Para qué sirve un Servicio Web? La respuesta puede ser otra pregunta: ¿Para qué sirve en programación una rutina? Todos sabemos que una rutina es como una caja negra, que encierra cierto proceso o algoritmo, y que cumple una función clara. Muchas rutinas y un guión central componen un programa en lo que se llama "programación estructurada". Un Servicio Web viene a ser una rutina en Internet.

Pero, ¿por qué se llama "Servicio Web" y no "Rutina en Internet"? Los protocolos que soportan los servicios Web se comunican normalmente por el puerto 80, y basándose en HTTP, métodos GET y POST. Esto hace que podamos acceder a ellos al igual que lo hacemos en una página Web. La diferencia entre una página Web y un Servicio Web, es que la página la visita cualquier individuo interesado, mientras que el servicio sólo lo visitan programas que lo requieren.

4.2.1 Los Protocolos

Hay un convenio generalizado que nos da a entender que los Servicios Web se invocan en Internet por medio de protocolos estándar basados en XML. Hoy en día hay dos grandes tendencias: XML-RPC y SOAP. A la hora de programar un servicio Web, hay que decidir qué protocolo usar, porque un protocolo es incompatible con el otro. De modo que si programamos nuestro servicio Web con XML-RPC, no podremos invocarlo desde un lenguaje de programación que trabaje con SOAP, como por ejemplo .Net de Microsoft.

Tanto SOAP (Protocolo de acceso a objetos simple, Simple Object Access Protocol) como XML-RPC son lenguajes de mensajería basada en XML, estandarizados por el consorcio W3C, que especifican todas las reglas necesarias para ubicar servicios Web XML, integrarlos en aplicaciones y establecer la comunicación entre ellos.

La diferencia entre SOAP y XML-RPC es su complejidad. XML-RPC está diseñado para ser sencillo. SOAP por el contrario está creado con idea de dar un soporte completo y minucioso de todo tipo de servicios web. La curva de aprendizaje de XML-RPC es muy suave, por lo que un programador novato en este campo, puede conseguir resultados satisfactorios con poco esfuerzo. Con SOAP no pasa esto, pero a cambio se dispone de más potencia. Por ejemplo, con XML-RPC no se

puede elegir el conjunto de caracteres a utilizar en tus Servicios Web. En SOAP se puede elegir entre US-ASCII, UTF-8 y UTF-16.

4.3 Arquitectura Orientada al Servicio

"Entienda cómo una Arquitectura Orientada a Servicios (SOA) posibilitará la rápida creación de aplicaciones basadas en comunidades de servicios interoperables" [36].

En las últimas décadas, los departamentos de Sistemas y de Tecnologías de la Información en las empresas han venido construyendo una infraestructura que actualmente soporta en gran medida la operación de sus organizaciones y sus clientes. El camino para llegar hasta este punto no ha sido fácil. Se ha aprendido de los errores y aciertos de la industria. El resultado de este proceso, ha sido la creación y mantenimiento de un número considerable de aplicaciones al interior de las empresas, cada una responsable de sus propias tareas.

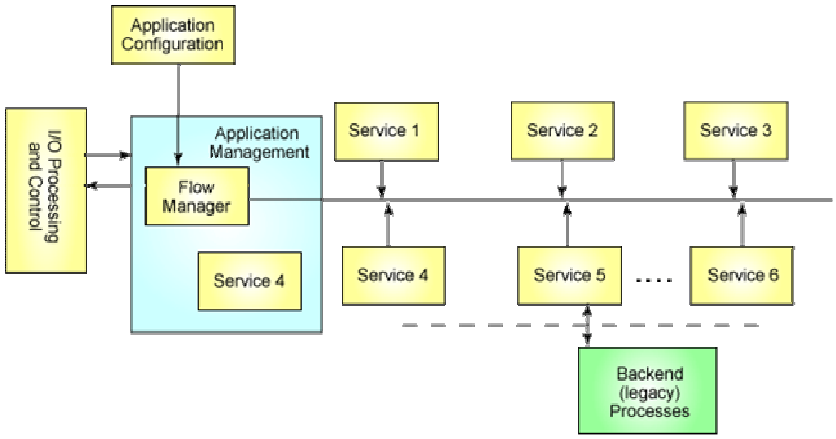
Ahora la idea de integración se hace más factible con una arquitectura orientada a los servicios ya que se puede reutilizar componentes, sistemas enteros o porciones de sistemas para que puedan interactuar con modernos sistemas con nuevas funcionalidades.

Pero no solo la idea de SOA es integrar funcionalidades antiguas y migrarlas, sino también hacer que componentes modernos presten servicios y a uno u otros componentes, mediante Web Services u otra tecnología que este estandarizada y pueda integrar distintas plataformas y formas de trabajo, claro está decir que XML es el elemento apropiado para basarse en una Arquitectura SOA.

"Toda función es definida como un servicio independiente, con interfaces bien definidas y que pueden ser invocadas en secuencias definidas para formar procesos de negocio" (Este es un ejemplo de una Arquitectura SOA) [36].

El siguiente grafico se muestra la interacción de los diferentes servicios todos comunicados con un bus, que maneja un estándar como es XML:

Figura A5.3 Modelo de Arquitectura orientada al servicio [36].

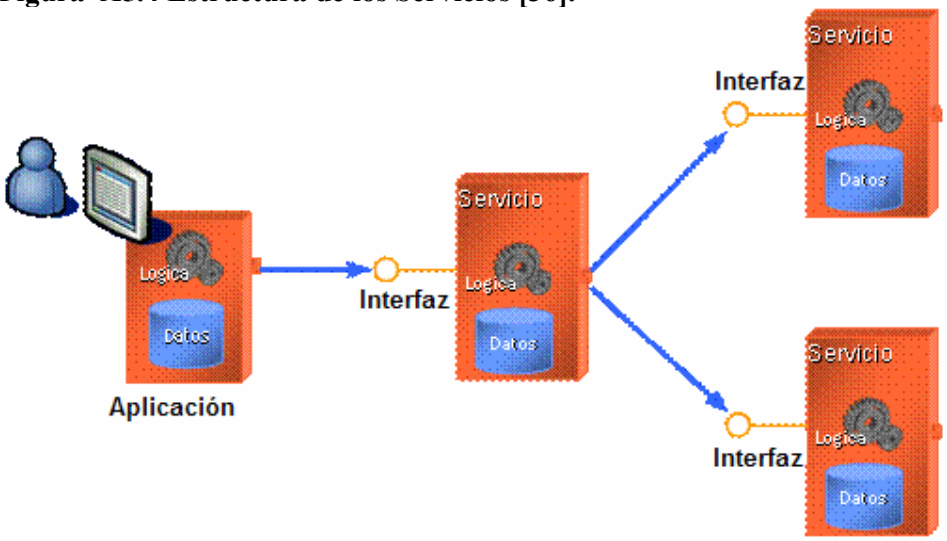


Para los próximos años la idea es de tener el Software como servicio, ya no comprar licencias sino pagar por recibir el servicio mediante un Browser por ejemplo, sin preocuparse por las actualizaciones, ni las fallas del software, todo estará basado en servicios.

> *"Para 2008, SOA será la práctica de ingeniería de software prevaleciente, acabando así 40 años de dominación de la arquitectura monolítica de software."*
>
> *Gartner LE-19-7652*

En el siguiente grafico se muestra que los servicios pueden tener lógica y orígenes de datos, los servicios pueden interactuar entre si para dar un resultado al usuario, que es transparente a los procesos que se realizan en los servicios.

Figura A5.4 Estructura de los Servicios [36].

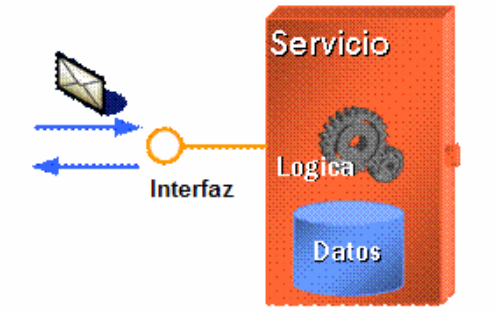


4.3.1 Visión Interna de los Servicios Web

Un servicio debe ser una aplicación completamente autónoma e independiente. A pesar de esto, no es una isla, porque expone una interfaz de llamado basada en mensajes, capaz de ser accedida a través de la red. Generalmente, los servicios incluyen tanto lógica de negocios como manejo de estado (datos), relevantes a la solución del problema para el cual fueron diseñados. La manipulación del estado es gobernada por las reglas de negocio [36].

En la siguiente figura se muestra que los servicios tienen una interfaz de ingreso y salida del mensaje (SOAP).

Figura A5.5 Vista de un Servicio [36].



4.4 Plataforma .NET

La plataforma .NET es el estándar de Microsoft para las nuevas aplicaciones de Tecnologías de la Información en la vida cotidiana de Personas y empresas, en la cual la conexión y el acceso a la información en todo momento, en todo lugar y en cualquier dispositivo, esa es la Visión .NET.

A partir de ahora, para poder hacer realidad esa visión es necesario tener tecnología innovadora y creativa que desarrolle el potencial de las personas, y soluciones sus problemas, de forma rápida y con interfaces amigables, sin importar el medio o el dispositivo de comunicación que las personas utilicen.

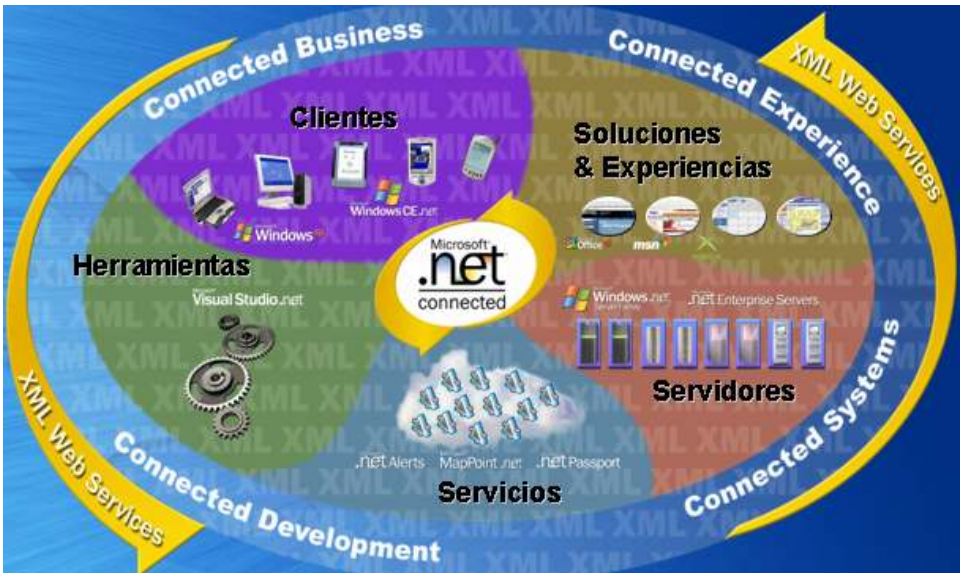
Entre las herramientas que Microsoft ha puesto como plataforma para .Net se encuentran:

- Servidores :
  - Microsoft Windows 2003 Server
  - Microsoft Windows XP
  - BizTalk Server
  - Microsoft SQL Server
  - Internet Information Server

- Herramientas de Desarrollo :
  - Visual Studio .NET

- Dispositivos :
  - Personal Computer
  - Pocket PC
  - Table PC
  - Smart Phone
  - PDA, etc.

Figura A5.6 Componentes que intervienen en la Plataforma .NET [36]



Como podemos ver en la Figura se aprecia todos los componentes, tanto software como Hardware, todos ellos interconectados a través de Servicios Web basados en XML, que en si es el nuevo estándar para el desarrollo de aplicaciones y la computación distribuida.

4.5 Plataforma J2EE

En el mundo de hoy, en que las necesidades del mercado de desarrollo de software, respecto a contar con medios y herramientas que permitan construir aplicaciones "empresariales", se diseñó la plataforma abierta y estándar de Java para este rubro, mejor conocida como Java 2 Enterprise Edition, (J2EE). Se le denomina plataforma porque proporciona especificaciones técnicas que describen el lenguaje pero, además, provee las herramientas para implementar aplicaciones basadas en dichas especificaciones.

J2EE ha sido diseñada para aplicaciones distribuidas que son construidas con base en componentes (unidades funcionales de software), los cuales interaccionan entre sí para formar parte de una aplicación J2EE. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea en la parte del servidor J2EE que le ofrece al componente ciertos servicios de bajo nivel y de sistema (tales como seguridad, manejo de concurrencia, persistencia y transacciones). J2EE no es sólo una tecnología, sino un estándar de desarrollo, construcción y despliegue de aplicaciones.