UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

INTEGRACIÓN DE SAP Y APLICACIONES LEGADAS A TRAVÉS DE SOA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
EN COMPUTACIÓN

ALFREDO ANDRES DÍAZ PUENTES

PROFESORA GUÍA:
CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
SERGIO OCHOA DELORENZI
CECILIA CASANOVA ARAYA

SANTIAGO DE CHILE
AGOSTO DE 2008

Resumen

El objetivo general del presente trabajo de título es un caso real de integración entre el ERP de SAP y los legacy del área de logística de una compañía distribuidora de combustible en Chile utilizando los principios que el patrón de diseño SOA ofrece y con la ayuda de herramientas comerciales que implementan y facilitan la integración entre sistemas computacionales heterogéneos. El área de logística de la compañía cuenta con una variedad de aplicaciones legacy Web del tipo J2EE que utilizan de manera independiente y que les permiten realizar la programación de los pedidos de combustible que sus clientes efectúan, asignación de los pedidos a los distintos camiones tanques, modificaciones y cancelaciones que se deban realizar a los pedidos, medición de eficiencia de los transportes y viajes efectuados, etc.

La integración se basó en los conceptos que el patrón SOA indica como mejor práctica con la ayuda de plataformas tecnológicas de la línea WebSphere de IBM tales como MessageBroker, MQ, Adaptadores del MessageBroker para comunicarse con el ERP y con el sistema operativo sobre el cual funciona que es Os/400 sobre iSeries, etc. Las principales decisiones tomadas durante el proyecto dicen relación con casos en los cuales se decidió hacer pasar el flujo de datos por la plataforma SOA instalada y cuando no hacerlo porque entregaba mejores prestaciones, tiempos de respuesta y seguridad. Durante todo el proyecto estos fueron temas de discusión y análisis dado que cada caso en la práctica trae un análisis individual que debe ser enfrentado bajo la mirada de conveniencia para la empresa, entendiéndose que estas conveniencias podrían ser tiempos de respuesta, puntos de falla, servicio e imagen al cliente.

En el detalle del trabajo se tomaron decisiones relacionadas con el tipo y formato de mensajes que transportan los datos entre las aplicaciones que participan del proceso de despacho de combustibles. Detalles como saber cuántos campos de datos son necesarios por mensaje, que separadores debían acordarse y ser utilizados, como serían tratados los encolamientos de los mensajes según si prioridad, los mensajes devuelta que deben ser enviados para cerrar los ciclos de cada proceso que se gatillen, etc. Cada una de estas decisiones en los distintos mensajes de datos que se gatillan en cualquier sentido dentro del proceso de despacho de combustible muestra los esfuerzos principales sobre los cuales se concentró el presente trabajo de título.

El resultado final fue un conjunto de aplicaciones del tipo Web y legacy que se comunican con el ERP de SAP para lograr de manera exitosa el despacho de combustible desde las plantas de la compañía; todo esto con la ayuda de una infraestructura conformada por filesystem y carpetas compartidas, plataformas de integración de IBM, flujo de mensajes que transmiten los datos, etc. Se concluye que los proyectos de integración si bien logran el objetivo principal sobre el cual se basan, carecen aún de la madurez necesaria en su diseño o bien en los productos comerciales que existen hoy en el mercado, dado que muchas decisiones se debieron tomar descartando lo que eran las mejores prácticas de SOA en beneficio de obtener una solución acorde a las necesidades de la compañía. La mirada de performance y eficiencia de los procesos es muy difícil dimensionarla y considerarla durante este tipo de proyectos y deben ser enfrentados, por lo general, posteriormente a su puesta en marcha, cuando los datos cuantitativos del funcionamiento entregan muchos más antecedentes que ayuden a encontrar las oportunidades de mejoras y los cuellos de botella que deben ser solucionados.

Dedicatoria

A mi madre María Cristina, por haber sacrificado muchas cosas para que yo pueda alcanzar las metas que me he propuesto y por entregarme de manera incondicional todo su apoyo y amor, sin los cuales no sería la persona que soy hoy.

A mi hermana Alexandra, por siempre creer en mí inclusive en los momentos más difíciles de nuestras vidas.

Ambas siempre han estado ahí y cada día me lo hacen saber con el amor que me entregan. A ambas les dedico el logro que más esfuerzo me ha tomado en mi vida y que sólo ellas pueden atesorar de la manera correcta.

Agradecimientos

Primero a mi madre, por entregarme las lecciones más valiosas: las de la vida.

A mi pololа Katy, por mostrarme que la vida sin abrirse al amor no es vida.

A mi hermana, quién me enseñó cómo la perseverancia es una herramienta poderosa para alcanzar los sueños.

A mi primo Roberto, por mostrarme que los obstáculos de la vida son sino incentivos para ser mejor persona cada día.

A mis parientes más cercanos, quienes me han enseñado con su amor de familia que nunca estoy solo.

A mis amigos de colegio y que me acompañaron en la Universidad, por mostrarme que la amistad es un camino largo por recorrer pero que juntos es imposible cansarse.

A los profesores de la carrera y administrativos, porque me enseñaron que las lecciones siempre son una mano dura para corregir pero que van acompañadas de una mano blanda para acoger. Gracias por todas las oportunidades que me dieron.

A Dios, por siempre entregarme y nunca quitarme.

Finalmente a mi padre, quién en algún momento de mi niñez supo enseñarme que siempre después de la tormenta viene la calma (Q.E.D.P.).

Contenido

Índice de Figuras

Capítulo 1 Introducción

Las mejoras tecnológicas del hardware y los nuevos desafíos de negocios han llevado a las empresas que requieren de un desarrollo a la medida migrar algunos de sus legacy[1] a lenguajes nuevos que vayan acorde con estas mejoras y con las nuevas necesidades de negocio. Sin embargo, para mucha de estas necesidades (varias de ellas claves en la producción) no existe un incentivo real para su migración o reemplazo dado los costos/beneficios que se obtendrían o simplemente porque no resulta estratégico su reemplazo o mejora.

Si a lo anterior se le suma la posibilidad de incorporar a las empresas software de clase mundial (worldclass) como son los ERP[2] y CRM[3] que existen hoy en el mercado, los cuales en su implementación absorben y reemplazan muchas de las funcionalidades de los legacy de la compañía pero dejan otras sin cubrir, estamos en presencia de un ambiente heterogéneo de estándares, lenguajes, protocolos, hardware, etc.

La gran mayoría del software dentro de las compañías requieren de la salida de datos de otros a modo de entrada, necesitan la ejecución secuencial para evitar inconsistencia de datos, requieren transformación de los datos, etc.; es decir, se requiere integrarlos para poder satisfacer las necesidades de negocio de las distintas áreas dentro de una compañía.

Esta integración no es fácil dentro de la infraestructura TI de una empresa donde coexisten variados y distintos sistemas, cada uno de ellos con alguna forma propietaria de comunicarse o dejar/tomar datos, sumándole inclusive la búsqueda para minimizar el esfuerzo en los desarrollos, ojalá reutilizando al máximo posible lo ya hecho.

A través del patrón SOA[4] el cual busca dividir los grandes componentes de software de las compañías en servicios utilizables por cualquier aplicación, dentro o fuera de la empresa, se puede lograr de forma más eficiente la buscada integración.

El presente trabajo de titulación aspira a mostrar la implementación que se realizó para satisfacer el flujo de datos que hacen realidad el despacho de combustibles debido a la introducción del ERP y del CRM de SAP AG [1].

1.1 Motivación

Sin el correcto diseño de una solución que logre realizar una integración adecuada y rápida entre las distintas aplicaciones, se corre el riesgo para la compañía de no entregar un buen nivel de servicio en el despacho de los combustibles a sus clientes y por tanto verse afectada la imagen y reputación de ella como así también la confianza y lealtad de todos sus clientes.

---

[1] Software legado o heredado

[2] Enterprise Resource Planning

[3] Customer Relationship Management

[4] Service Oriented Architecture

El vacío que dejó el antiguo sistema de intercambio de archivos y mensajes a través de colas debió ser superado por una solución que estuviese a la altura de la envergadura del problema. Es por eso que se debió analizar y buscar cual de todas era la solución más eficiente pero a la vez segura para la comunicación entre las distintas aplicaciones, llegándose a encontrar que el intercambio de archivos, como existe en el modelo actual, no presenta todas las garantías que la operación de la compañía requiere para funcionar.

Finalmente y utilizando los conceptos de integración que SOA promueve se diseñó un modelo tecnológico lo suficientemente eficiente para no descargar todo el trabajo de transformación y envío de datos a los legacy o al ERP. Utilizando de esa manera un bus de integración único junto con la infraestructura adquirida (hardware) para su funcionamiento se lograría concentrar el paso de los datos por un punto único permitiendo monitorear todo el traspaso de información y poder de dicha manera tener el control sobre posibles errores o pérdidas de datos.

Profesionalmente hablando, el interés que produce la problemática a resolver y la solución diseñada viene dado por lo atractivo que resulta el utilizar herramientas, aplicaciones y esquemas que están actualmente en constante desarrollo y actualización, como es todo lo relacionado con el SOA, además de poder entregar una solución real a implementaciones de SAP que día a día son mucho más comunes en las empresas, debido al valor agregado que le entregan como así también debido a la modernización que provee a los procesos de negocio.

1.2 Antecedentes

En la empresa en la cual se basa el presente trabajo se plantea la necesidad de implementar el ERP [2] de SAP como una solución tecnológica que satisfaga una gran cantidad de funciones de negocio. Una de ellas, la que motiva este estudio, dice relación con la logística necesaria para el despacho de combustibles.

El mapa tecnológico actual del área de logística de la compañía requirió la migración de algunas de sus aplicaciones de logística a versiones en J2EE [3] (mientras otras permanecieron siendo RPG[5] [4]) apoyándose en una infraestructura basada en SOA para lograr que las ventas realizadas en el CRM de SAP se traduzcan en asignación de carga de combustibles a determinados camiones tanques.

1.3 Objetivos del proyecto

El presente trabajo tiene como objetivo principal lograr la integración entre el ERP de SAP y los legacy del área de logística de una compañía distribuidora de combustible a través del

[5] Report Program Generator, lenguaje de programación de sistemas IBM para aplicaciones de negocio

diseño, modificación y/o construcción de aplicaciones del tipo J2EE y ABAP[6] tal que permitan realizar el proceso de despacho de combustibles.

El proceso de despacho de combustible trae un escenario complejo de abordar dada la cantidad de elementos tecnológicos distintos y por su importancia dentro de la compañía. Sin embargo con el presente trabajo se pretende alcanzar soluciones tales como: la exposición y comunicación de la lógica de negocio del ERP de SAP hacia software externos o legacy, transformación de datos originados ya sea en SAP o bien en los legacy, el ciclo completo del flujo de datos que comprende su origen desde el SAP con destino los legacy y la respuesta con origen en el legacy con destino el SAP.

1.4 Descripción del proyecto global

El enfoque del trabajo se hace en el proceso de despacho de combustible en sí, sin entrar en discusión la venta a los clientes o el cargo a facturar y cómo esa información es manejada dentro del SAP por ser ese flujo propio de la solución envasada del producto ERP.

En el caso del área de logística de la compañía sobre el cual se basa este trabajo, previo a la implementación del ERP y del CRM de SAP existían software hechos a la medida (por distintos proveedores) que entregaban distintos tipos de servicios tales como la disponibilidad de las flotas de camiones, el listado de pedidos existentes en un determinado momento y su fecha-hora solicitada de entrega, la asignación de un despacho a un camión determinado, la emisión de las guías de despacho para que puedan salir los camiones de las plantas, las facturas a entregar a los clientes, etc. Varias de estas funciones, en especial las relacionadas con la documentación y ventas, se encontraban implementadas en RPG sobre máquinas corriendo sistema operativo AS/400 y se comunicaban entre ellas vía archivos en directorios o bien mensajes en colas de MQ a modo de una integración económica.

Después de la implementación el escenario anterior se vio modificado ya que fue necesario integrar las ventas y generación de documentación (en el SAP) con las otras aplicaciones de asignación y programación de los despachos. Esta integración implica un flujo en dos sentidos entre el mundo SAP y el mundo no-SAP (desde ahora llamado legacy). Lo anterior significó modernizar los legacy que realizan el despacho de los combustibles junto con realizar una comunicación y traspaso de datos hacia y desde el SAP de manera rápida y sin pérdida de información.

La solución diseñada significó la generación de mensajes desde el ERP del SAP con los pedidos de venta de combustible destinados a despacho, su paso por una plataforma intermedia que transforme dicha información de una manera tal que las aplicaciones legacy los pudieran entender (con el objetivo de disminuir las modificaciones a software que se mantienen) y la carga de dicha información en aplicaciones del tipo Java a J2EE destinadas a la programación de las cargas y despachos de camiones. En el caso del software que controla la inyección del combustible dentro de los camiones tanques, FuelFacs, no sufrirá modificaciones por ser éste un software propietario de origen Canadiense y no tener sentido el modificarlo.

[6] Lenguaje de programación propietario de SAP

A nivel macro la solución propuesta contempló que una vez que se registre una venta (a través del CRM y dejando los datos de la venta en el ERP) se gatillará un pedido desde el ERP con un cliente como destinatario. Este pedido viajará a través de la infraestructura SOA (Message Broker como el ESB[7] de la integración elegido) desde el SAP hacia una aplicación legacy de tipo Web (J2EE) llamada **SCD** (Sistema Control de Despacho) la cual la recibirá y la mostrará en la pantalla a los operadores de logística encargados de los despachos. Estos operadores (o programadores como también se les conoce en la jerga de la compañía) visualizarán el pedido y realizarán la búsqueda del camión tanque que deberá llevar el pedido hasta el cliente. Esta búsqueda se basa en otra herramienta web llamada **SCR** (Sistema Control de Ruta) que es la encargada del control de flota de camiones. Esta búsqueda es a su vez alimentada por otra aplicación web, la **SCF** (Sistema Control de Flota), que es utilizada para ingresar de manera manual y a través de los registros de GPS instalados en los camiones la disponibilidad real de cada camión en tiempo real o programado con anticipación.

Una vez que los operadores tengan el pedido y tengan seleccionado el camión de la flota que se encargará del despacho del combustible, ingresarán a otra aplicación web llamada **BDP** [8] [5] que es la actual encargada de la programación de los despachos. Esta asignación-programación de los despachos a su vez generará un ticket de servicio a nombre del conductor del camión, el cual se acerca a las mangueras que inyectan el combustible controlado por un legacy propietario de origen canadiense llamada **FuelFacs**, ingresarán el código asignado a ellos y el software llenará los distintos estanques internos del camión con cada tipo de producto asignado para el viaje.

Se determinó que todo el proceso anterior tiene a su vez un ciclo de vuelta de los datos que deben finalmente llegar al ERP para que quede el registro de la carga del combustible y así poder emitir la guía de despacho para que el camión salga de la planta y emitir las facturas que deberán ser entregadas a los clientes.

1.5 Conceptos Básicos

El objetivo que busca SOA es dividir las funcionalidades macro de las grandes aplicaciones en funcionalidades genéricas y después componer otras más específicas en base a las anteriores, promoviendo así una reutilización del software ya hecho, mientras que se facilita y agiliza la respuesta a los requerimientos de las áreas de negocio de las empresas que siempre buscan maneras nuevas de hacer negocios.

Estas funcionalidades más pequeñas se llaman servicios [6], ya que están en forma pública ofrecidas para quién lo desee las pueda utilizar (un consumidor). Como se mencionó anteriormente, la composición de dos o más servicios puede generar un servicio nuevo más específico, el cual es ofrecido para ser consumido por distintas aplicaciones dentro o fuera de una compañía. El cómo se entera el resto de las aplicaciones de la existencia y la forma de

[7] Enterprise Service Bus

8 Manugistics Bulk Distribution Planning

utilizar estos servicios viene dado por una estandarización hasta el día de hoy en constante cambio y definición.

En la práctica se habla de registros que anuncian estos servicios para que otras aplicaciones las puedan dinámicamente encontrar. A través de formatos acordados (XML) las aplicaciones descubren los parámetros de entrada de los servicios como así también lo que retornan y van utilizando protocolos de comunicación en común para establecer el contacto y traspaso de datos (SOAP) [7].

Los elementos claves de los que se componen una arquitectura SOA incluyen:

1. **Aplicaciones:** inician y controlan todas las actividades de los sistemas empresariales.
2. **Servicios:** componente de software que encapsula una lógica de negocio y ofrece una interfaz pública de acceso.
3. **Repositorio de Servicios:** provee la facilidad de descubrir servicios y recolectar toda la información necesaria para utilizarlos.
4. **Bus de Servicios (ESB):** conecta a todos los participantes de un SOA (servicios, aplicaciones, etc.) entre ellos.

Por otro lado un ERP de clase mundial como es el R/3 [8] de SAP [2] es una familia de software de aplicaciones de origen alemán que tienen como objetivo integrar y alinear los procesos de negocio. Tradicionalmente los procesos que concentran son:

1. Finanzas
2. Recursos Humanos
3. Logística

Si bien en sus orígenes las aplicaciones de SAP únicamente se comunicaban entre sí lo que no permitía una fácil integración con otros software (excepto a través de la capa de datos), en sus últimas dos versiones del SAP Business Suite posee una plataforma tecnológica de integración llamada Netweaver [9] enmarcada dentro de los estándares de lo que es SOA.

Netweaver es esencialmente la base de integración de los productos tecnológicos de SAP. El SAP web application server (conocido como WebAS) es el ambiente de ejecución para las aplicaciones SAP, todas las soluciones de mySAP business suite corren sobre él (SRM, CRM, SCM, PLM, ERP).

Los principales productos que forman SAP Netweaver incluyen:

- SAP Web Application Server
- SAP Exchange Infrastructure (XI)
- SAP Enterprise Portal

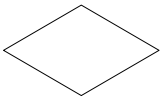
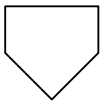
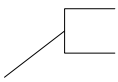
- SAP Master Data Management (MDM)
- SAP NetWeaver Mobile
- SAP Business Information Warehouse

Capítulo 2 Proceso de Negocio: “Programación y despachos de combustibles por camión”

Se describe a continuación el proceso de negocio que es base del presente trabajo a fin de entender de mejor manera como los distintos actores de este proceso se relacionan y la información fluye.

2.1 Nomenclatura técnica

La siguiente es la nomenclatura utilizada en la diagramación de los procesos de logística de combustibles:

	Actividad o tarea
	Decisión o alternativa de acción
	Inicio y fin de cada proceso
	Conexión o unión entre actividades
	Conexión de actividades dentro del mismo proceso
	Conexión de actividades a otro proceso
	Comentario

2.2 Alcance

Este procedimiento se aplica a las actividades realizadas en la programación y despacho de combustibles desde las plantas de la empresa. Éstas se originan por los pedidos de los clientes o por transferencias entre plantas.

2.3 Objetivos

Establecer procedimientos que normen los despachos, a fin de:

- Formalizar y estandarizar criterios y controles para procesos relacionados con la programación, despacho y documentación de los despachos.
- Asegurar la integridad y confiabilidad de la información relacionada con la programación, despacho y documentación de los despachos.
- Promover la actualización oportuna de la información referente a la programación, despacho y documentación de los despachos.

2.4 Responsables/Actores

En la siguiente tabla, se detallan las responsabilidades de los actores en las actividades descritas en el presente procedimiento.

Rol-cargo/área	Responsabilidad
Programador de despacho / Logística	Es responsable de programar el despacho.
Despachador –Facturero / Logística	Es responsable de: - Generar la documentación que involucra un despacho. - Cerrar el despacho al comprobar la entrega de los productos. - Es responsable de emitir los documentos de respaldo

Rol-cargo/área	Responsabilidad
	del despacho.
Conductor camión	Es responsable de: - Efectuar el traslado de los productos hasta el lugar de destino
Operario de patio/ Logística	Es responsable de revisar la carga.
Portero/ Logística	Es responsable de: - Revisar el vehículo a la entrada de la planta según la información contenida en el documento que presente el conductor. - Validar condición de estanque vacio cuando corresponda. - Revisar el vehículo a la salida de la planta, registrando incidencias. - Recibir la documentación que respalda la entrega de productos.
Recaudador/ Finanzas	Es responsable de recibir los valores que recolecta el transportista.

2.5 Propiedad

Se entenderá como dueño de este proceso al Gerente de Ventas de la compañía.

2.6 Mapa General del proceso "Programación y despachos de combustibles por camión"

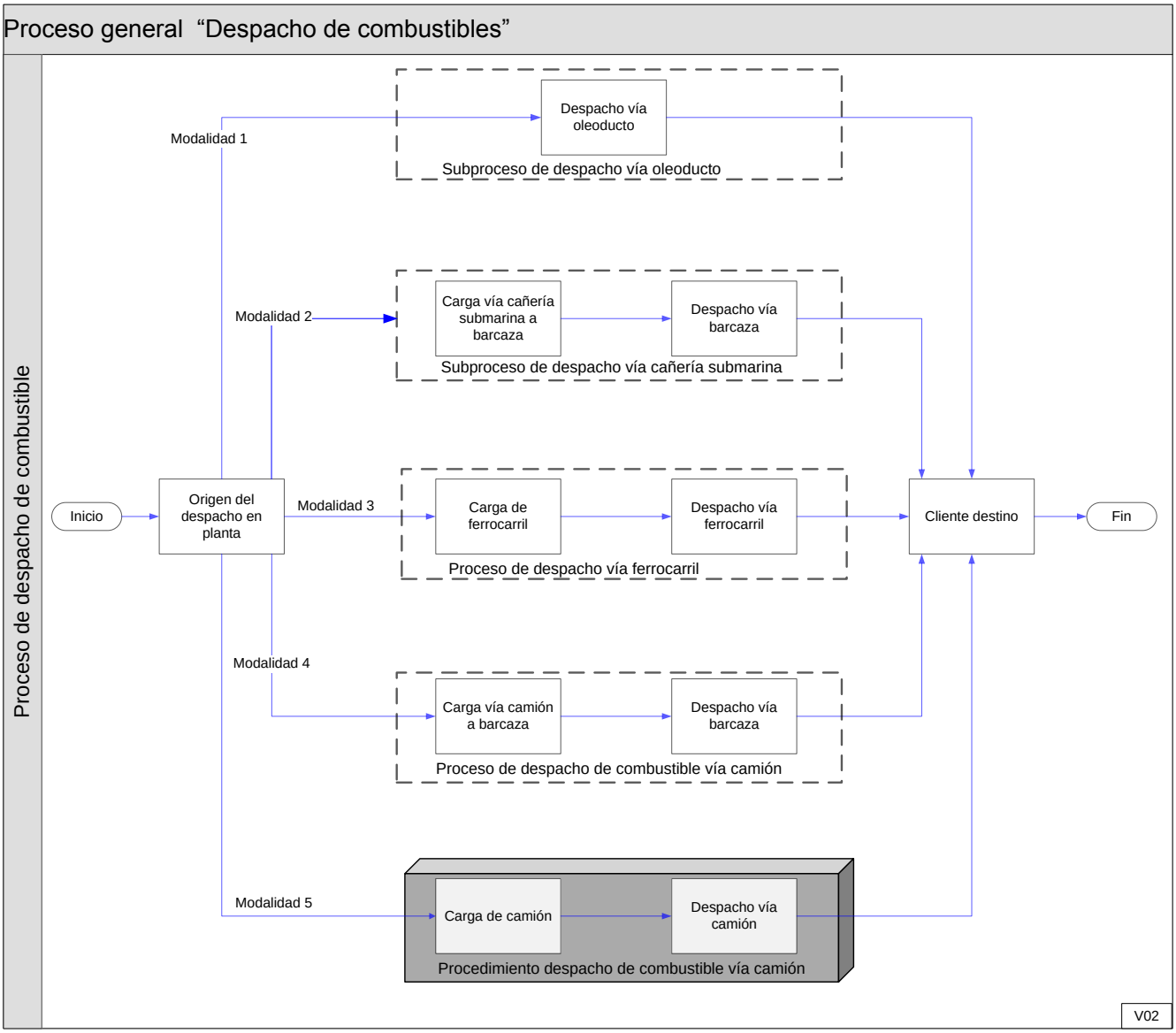


Figura 1-Proceso General

2.7 Mapa del proceso "Programación y despacho de combustibles por camión"

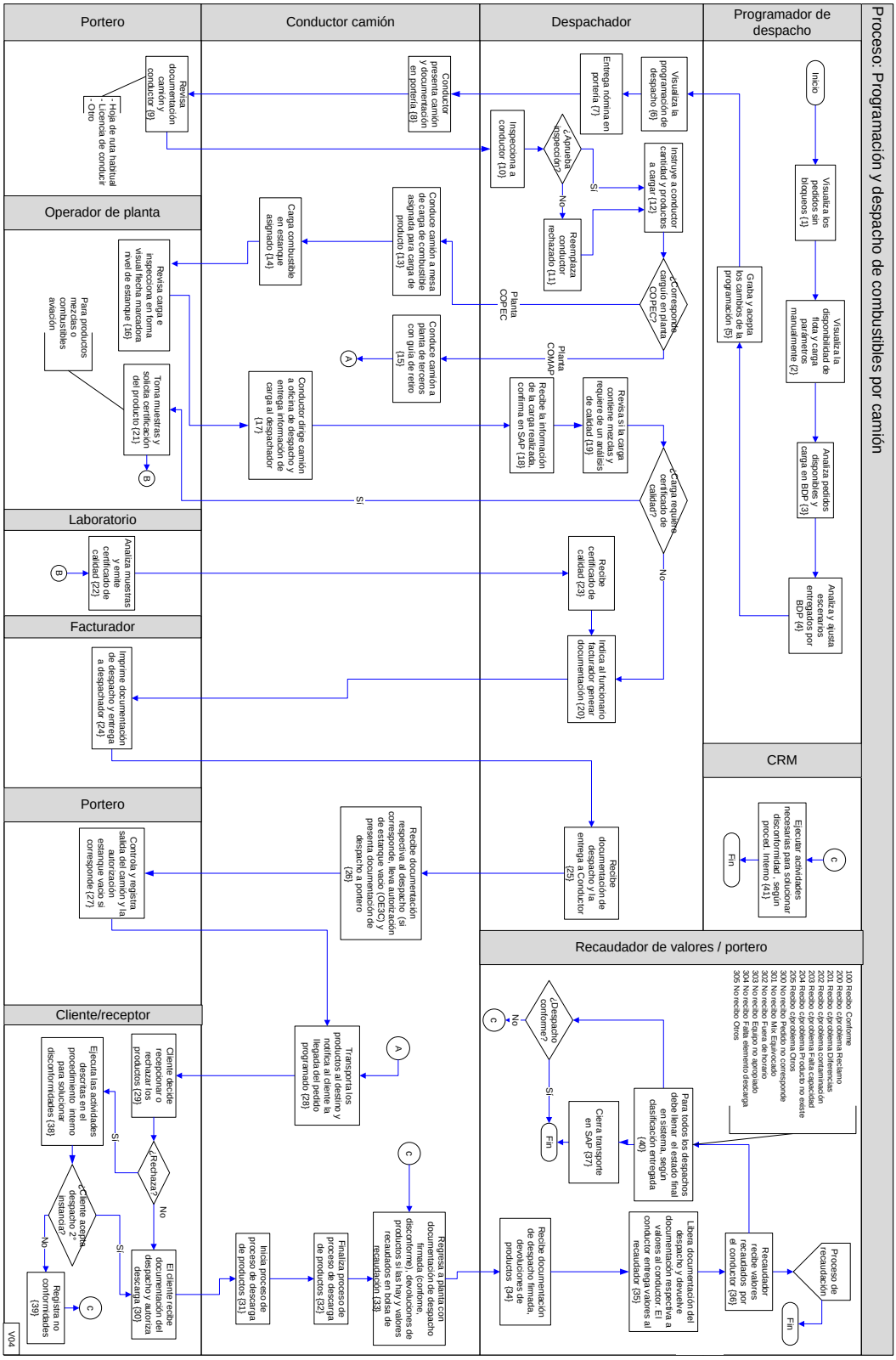


Figura 2- Proceso Detallado

2.8 Descripción del proceso "Programación y despacho de combustibles por camión"

2.8.1 Programación de despacho

Programador de despacho

{1}

1. El proceso se inicia cuando el programador de despacho visualiza en aplicación web, disponible en Portal, los pedidos en condición de ser programados (sin bloqueos).

{2}

2. Visualiza disponibilidad de flota en aplicación web, disponible en Portal.

   Nota: la disponibilidad diaria de flota es ingresada a Portal, o es enviada por correo electrónico por el administrador de flota (jefe de turno de planta) antes de las 17:00 horas.

3. Carga manualmente en BDP los parámetros de la disponibilidad de flota y habilita transporte y horario en Portal.

   Nota: esta actividad puede tener ajustes por cambios en la disponibilidad de flota ocurridos en la planta.

{3}

4. Analiza los pedidos en condición de ser despachados.

5. Carga en sistema BDP los pedidos en condición de ser despachados, en función de su fecha de entrega y de la disponibilidad de flota.

6. El sistema BDP filtra los pedidos por prioridad (desde P_1 a P_5) y programa los despachos de acuerdo a los parámetros de pedido.

7. El sistema BDP entrega una propuesta preliminar de escenarios de programación por segmentos (norte, centro, sur y Maipú) y asigna automáticamente un número único a la programación.

{4}

8. Analiza la programación entregada por BDP, corrigiendo los despachos no factibles.

Realiza estos ajustes o correcciones al programa, en el sistema BDP, con el fin de maximizar los despachos, modificando manualmente la programación según parámetros de flexibilidad de entrega a cada cliente y de acuerdo a la programación de cada pedido.

Nota: existe un programador especialista por segmento en Logística.

9. El sistema levanta alertas de incompatibilidad de productos/camión o ventanas horarias según requerimientos o restricciones del cliente.

10. Realiza ajustes o correcciones a la parametrización, o al programa manualmente, para corregir las alertas.

{5}

11. Graba los cambios en BDP.

12. Realiza la aceptación de la programación en sistema.

Nota: al aceptar la programación, procesos internos de sistema disponibilizan la programación para ejecución de las actividades siguientes. La actualización se realiza en SSD[9] y en SAP.

Despachador (Planta)

{6}

13. Visualiza la programación en SSD (Sistema de Seguimiento de Despacho).

14. Solamente en el caso que sea necesario se informa al conductor de transporte la programación asignada.

{7}

15. El despachador entrega la nómina escrita actualizada en portería. La nómina contiene un listado con los conductores autorizados a ingresar, tanto de la empresa como de otras compañías.

16. El despachador deberá modificarlas cada vez que se produzca un cambio.

[9] Sistema de Seguimiento de Despacho

Nota: el Jefe de turno colabora en las actividades de despacho en los casos de alta demanda.

Conductor camión

{8}

17. El conductor se presenta en la planta según las necesidades de cada planta y según la reglamentación vigente.

Portería Planta

{9}

18. En Portería se revisarán los siguientes antecedentes presentados por el conductor.

Para flota de la planta:

- Libro de hoja de Ruta por conductor habitual
- Hoja de Ruta y cédula de identidad, en caso de ser conductor de reemplazo.
- Licencia de conducir vigente y apta para la categoría del vehículo.

Para flota de otras compañías:

- Documento tributario (guía de retiro plantas de terceros) de la otra compañía, si es conductor habitual.
- Documento tributario de la otra compañía y cédula de identidad, si es conductor de reemplazo.
- Licencia de conducir vigente y apta para la categoría del vehículo.

19. Cuando se trate de camiones que no pertenecen a la flota éstos deben ser previamente autorizados. La persona responsable de la función de portería anotará en la bitácora de la Planta según procedimiento vigente:

- Hora de entrada
- Nombre del conductor
- Cédula de identidad

- Motivo del ingreso
- Nombre de Institución o Empresa a la que pertenece
- Patente del camión
- Licencia del conductor vigente y apta para la categoría del vehículo.

Nota: cada conductor habitual presentará su licencia, a lo menos, cada 15 días. En caso de detectar la existencia de una infracción de tránsito, la persona que ejerce la función de Portería deberá registrarlo en la bitácora e informarlo al Jefe de Planta verbalmente.

Despachador

{10}

20. Realiza inspección visual del conductor, fecha de vigencia de su documentación y que se encuentre en las condiciones adecuadas (físicas, profesionales y técnicas), para el transporte de los productos de acuerdo a las normas operacionales vigentes.

{11}

Si el conductor no está en las condiciones adecuadas para realizar el transporte, asigna otro vehículo y conductor.

{12}

Si el conductor está en las condiciones adecuadas para realizar el despacho asignado, aprueba al conductor para la ejecución del despacho.

21. Informa personalmente al conductor la cantidad y los productos a cargar, informa la mesa de carga asignada y entrega los sellos para los estanques del camión, con los colores respectivos para cada producto. Si no hay disponibilidad de este tipo de sellos, se reemplazan por sellos de colores neutros.

{15}

Si carga en planta de terceros (COMAP), continúa en punto 24.

Nota: si debe cargar producto en planta de terceros, completa manualmente la "Guía de retiro de productos en plantas de terceros", con la información de los productos que debe cargar.

Si carga en planta de la compañía, continúa en punto 22.

22. Si la planta dispone de tecnología para carga automática (Sistema Fuelfacs), el despachador entrega información de carga en un voucher y tarjeta magnética para accionar equipo de carga al conductor del camión.

23. Si la planta no dispone de tecnología para carga automática, el despachador le entrega ticket de carga (***OE3C*** - formulario 698) impreso con la información del destino y cantidades de carga.

Conductor camión

24. Realiza "estruje" o evacuación de los remanentes de productos anteriores del vehículo en el lugar indicado, con el fin de asegurar que no existan residuos en el estanque del camión.

25. Recibe la información de lugar de carga:

Si corresponde cargar combustible en planta de terceros (COMAP), es entregada la guía de retiro de productos o el documento convenido; el conductor se dirige a la planta de terceros, siendo el proceso de carga de combustible responsabilidad del personal de la planta de terceros.

Realiza comunicación telefónica con planta de tercero para notificar envío de camión para carguío de combustible.

El proceso continúa en el punto 42.

Nota: en el caso que el combustible requiera de aditivos, éstos se incorporan al producto en forma manual después del proceso de carga.

{13}

Si carga en planta de la compañía, conduce el camión a la mesa de carga asignada.

26. Conecta el camión a los caños de carga superior o inferior según lo especificado en la información de carga.

{14}

27. Inicia proceso de carga de combustible en mesa de carga según procedimiento operacional MOP N° 06-415 y N° 06-420.

Finaliza el proceso de carga.

Imprime ticket de carga (ticket printer ***OE3C***) para carga en plantas no automáticas, o voucher para plantas con tecnología de carguío de combustible automático.

Nota: en el caso que el combustible requiera de aditivos, éstos se incorporan al producto en forma automática durante el proceso de carga.

Operador de planta

{16}

28. Revisa la carga, inspeccionando en forma visual que la flecha de nivel del estanque del camión esté de acuerdo con lo indicado en la tarjeta de carga y cumpla con las tolerancias establecidas.

    En caso de existir diferencias por más de 1cm, debe informar al Jefe de Planta, el cual autoriza el relleno o instruye el retiro del producto del compartimento.

    Si los valores concuerdan con la lectura de la flecha, el proceso continúa en el punto 29.

Nota: todo camión que salga de la planta con uno o más compartimientos o estanques vacíos, debe ser autorizado por escrito por el Jefe de Turno en el documento ***OE3C***.

Conductor camión

{17}

29. Se dirige a oficina de despacho y entrega al despachador la información final de la carga realizada.

Despachador

{18}

30. Recibe la información entregada por el conductor (ticket o tarjeta con detalle de carga, y documentos adicionales), confirma la carga en SAP mediante la transacción [10] ***O4H1***, que registra la salida de mercancía automáticamente.

[10] Una transacción en SAP es un programa interactivo que es posible ser ejecutado por los usuarios.

{19}

31. Revisa si la carga contiene mezclas y requiere de un análisis de calidad del producto:

{20}

Si la carga no contiene mezclas, confirma la carga y solicita a funcionario facturero la emisión de la documentación de despacho que debe acompañar al transporte y continúa en el punto 32.

Operador de planta

{21}

Si la carga contiene mezclas, realiza proceso de toma y análisis de muestras según MOP N°10-310 “Muestreo de productos” y MOP N°10-400 “Control de calidad combustibles”. Se dirige al transporte y toma muestras de los productos cargados y entrega a analista de laboratorio para su análisis y certificación según procedimiento de control de calidad.

Laboratorio

{22}

Recibe las muestras de los productos y las analiza. Emite certificado de calidad del producto.

{23}

Entrega el certificado a despachador.

Despachador

Recibe certificado de laboratorio de los productos cargados.

Facturero

32. Visualiza en SAP mediante la transacción ***VF03*** y genera los documentos correspondientes al despacho mediante la transacción ***VF01***. (Factura o guía de despacho, según corresponda, de acuerdo a procedimiento “Facturación de combustibles y no core”).

{24}

33. Imprime la documentación de despacho:

Selecciona las entregas que van con guía de despacho y asocia las entregas con el número de folio de la guía de despacho e imprime la guía de despacho según procedimiento de “Facturación de combustibles y no core”.

Selecciona las entregas que van con la factura y asocia el número de folio de la factura con las entregas e imprime la factura según procedimiento de “Facturación de combustibles y no core”.

34. Entrega por mano al despachador la documentación de respaldo del despacho.

Nota: envía documento tributario para cobro a facturación.

Despachador

35. Para todos los despachos de producto, el despachador o el conductor, según instrucciones definidas por el Jefe de Planta, registra manualmente en el documento tributario el número de los sellos, ordenados por producto, sin diferenciar el de las tapas y los domos del camión. No se requerirá estampar timbre previo.

{25}

36. Entrega por mano al conductor del camión toda la documentación referida al despacho.

37. Guarda la copia número 2 correspondiente a la empresa emisora de la factura o la guía de despacho y envía la copia número 5 a Contabilidad.

38. El jefe de turno o despachador realiza una revisión selectiva de a lo menos un camión por turno.

Nota: todos los camiones que operan con la planta, deben ser revisados en forma completa, por lo menos una vez al mes, enfatizando especialmente la revisión de los sellos y de la flecha que marca el nivel del estanque del camión.

Conductor camión

{26}

39. Recibe documentación respectiva al despacho (si corresponde, lleva autorización de estanque vacío ***OE3C***) y presenta documentación de despacho al portero.

Portero

{27}

40. Controla la salida del camión, verifica visualmente el estado del camión y sus sellos, además de la documentación de despacho, incluyendo la autorización de estanque vacío cuando corresponda. Si el camión no cumple con la condición de verificación para autorizar la salida de la planta, debe solucionar inconformidades :

    Verifica los estanques vacíos del camión según lo indica documento de despacho "Orden de entrega y control de carguío de combustible" (***OE3C***).

41. Anota la salida del camión en "Libro de control" y en SAP mediante la transacción ***ZSD_OPTRA***.

Conductor camión

{28}

42. Realiza el transporte de los productos hacia el cliente destino.

43. Notifica al cliente de la llegada del pedido.

Cliente deudor / cliente solicitante

{29}

44. Cliente toma la decisión sobre la recepción del pedido:

{30}

Si acepta la recepción del pedido, recibe documentación asociada al despacho.

Si el cliente está conforme con la recepción, autoriza la descarga de los productos en su instalación.

Descarga en la instalación del cliente

Conductor camión

{31}

El proceso de descarga es responsabilidad del conductor del camión y debe pedir al cliente/receptor del producto que siga estrictamente el procedimiento de descarga en todos los aspectos operacionales, de seguridad y administrativos según los manuales de procedimientos operacionales MPO **N° 06-425**, **N° 06-430, N° 06-440.**

Si cliente/receptor no cumple el procedimiento, el conductor deja constancia del hecho en la guía de despacho, aún cuando la recepción final se dé por conforme. Dicha constancia debe señalar cada uno de los pasos no cumplidos.

En el caso que el cliente/receptor no conozca el procedimiento de descarga, o tenga dudas de algunos de sus pasos, el conductor mostrará la cartilla disponible en el camión con las instrucciones. El proceso continúa en el punto 45.

> **Nota:** cada camión debe disponer de una cartilla con instrucciones operacionales, de seguridad y administrativas relacionadas con la descarga de combustibles. La cartilla debe considerar la descarga de productos livianos, que normalmente se entregan en litros naturales, y la descarga de productos pesados, que se entregan en kilos.

Si existen aspectos especiales según el tipo de instalación, también deben estar especificados expresamente en la cartilla con instrucciones. Uno de los aspectos más importantes en este proceso es cumplir estrictamente con el procedimiento que debe seguir la recepción del producto.

Cliente deudor / cliente solicitante

{38}

Si no acepta recepcionar los productos notifica al conductor las causas.

Conductor camión

Comunica disconformidad por teléfono a la planta y solicita gestión adicional para obtener aceptación de los productos, si corresponde.

Programador de despacho / Call Center

Programador, en conjunto con Call Center, realizan gestiones adicionales para recepción del pedido en cliente o posible reprogramación del despacho.

Conductor camión

Recibe instrucciones del programador de despacho y procede según lo indicado:

Si el cliente aceptó recibir el despacho de productos, vuelve al punto 44.1.

Si el cliente no aceptó recibir el despacho de productos, cumple las instrucciones dadas por el despachador y se retira del lugar.

Cliente deudor / cliente solicitante

{32}

45. Una vez finalizada la descarga, el cliente firma en señal de conformidad los documentos indicados por el conductor del camión (factura o guía de despacho u otros).

Si el cliente considera que la recepción fue conforme, debe quedar anotada en el cuadriplicado de la guía de despacho o factura.

El cliente/receptor debe:

Estampar el timbre de la E/S, de la empresa o cliente.

Registrar el RUT de la persona que recibió el producto

Firma la recepción conforme.

{39}

.

Si el cliente considera que la recepción fue disconforme, debe quedar en el cuerpo central del documento o en su reverso, el detalle y los motivos de la disconformidad

El cliente/receptor debe:

Estampar el timbre de la E/S, de la empresa o cliente.

Registrar el RUT de la persona que recibió el producto.

Firmar la recepción disconforme.

Adicionalmente el cliente tiene la posibilidad de interponer un reclamo formal por cualquier causal que estime conveniente al número del Call Center 800200220.

Nota: el proceso de respuesta o solución a un reclamo queda sujeto a la clasificación que el ejecutivo de atención del Call Center asigne. Según las personas definidas es derivado a la unidad responsable para dar respuesta. Posteriormente se realiza el seguimiento del estado del reclamo.

Conductor camión

El conductor firmará en señal de "toma de conocimiento" de la recepción disconforme por parte del cliente sin importar la apreciación personal del hecho. El proceso continúa en el punto 58.2

Cliente deudor / cliente solicitante

46. Para los despachos a concesionarios o consignatarios con condición de venta contra pago, el cliente debe preocuparse que el cuadruplicado de la guías de despacho o factura quede en el bolsillo exterior de la bolsa recaudadora especialmente acondicionado para ello.

47. Entrega al conductor del camión la documentación

Conductor camión

{33}

48. Recibe la documentación firmada por el cliente deudor o solicitante.

49. El conductor retorna a la planta para completar la actividad de retorno de vuelta.

Nota: el conductor siempre debe siempre retornar a la planta para cerrar su vuelta, aún cuando ésta sea la última del día.

Portero planta

50. El conductor ingresa a portería de la planta para registrar el cierre de vuelta.

51. Anota en planilla de registro “Libro de control” ingreso del camión y en SAP mediante la transacción ***ZSD_OPTRA***.

52. Consulta al conductor si hay novedades para registrar cualquier detalle o incidente.

Conductor camión

53. La recepción de la vuelta del conductor después de un transporte debe efectuarla sólo la persona que está encargada del Control de Despachos.

54. El conductor entregará las bolsas y los documentos a quien esté cumpliendo la función en ese momento.

55. Entrega en la planta a encargado del control de despachos, la documentación de respaldo de la entrega, copia de factura (copia número 4) o guías de despacho correspondientes, donde el cliente debe haber indicado conformidad de recepción y la hora y fecha de recepción de los productos. Además debe traer valores (en bolsa sellada) que ha recaudado de los clientes concesionarios o consignatarios por ventas o entregas contra pago, los que una vez confirmados serán entregados al encargado de recaudación.

Portero o despachador

{34}

56. Recibe la documentación y el conductor procede a notificar estado de las entregas.

{35}

En caso que las entregas hayan sido efectuadas en forma completa y conformes, libera los cuadruplicados de los documentos. El proceso continúa con las siguientes actividades normalmente:

- Despachos a crédito o no cobrables, pasarán al responsable de archivo de documentación de despacho de la planta.
- Despachos contra pago, volverán al bolsillo exterior de la bolsa con valores.

En caso que las entregas hayan sido efectuadas en forma parcial y/o disconforme debe proceder:

Cuando las observaciones escritas en los documentos de despacho son muy extensas o tienen datos difíciles de transcribir, sacará fotocopias para adjuntar al formulario de disconformidad.

Cuando el conductor no entrega el reporte del estado completo del despacho, el encargado del control no entregará nuevos documentos para otro despacho e informará de los problemas al Jefe de Planta o Jefe de Servicio.

Nota: para ambos casos el conductor debe entregar los valores recaudados.

Conductor camión

57. El conductor entrega valores transportados al recaudador de valores.

Recaudador de valores

{36}

58. Revisa las bolsas de valores que le entrega el conductor del camión, verificando que estén debidamente selladas y recibe los valores enviados por los clientes.

Despachador

{40}

59. Recibe los documentos de recepción de la entrega y libera los cuadruplicados de los documentos de despacho, de igual manera que en el caso de recepción disconforme.

60. Para todos los casos de recepción de la entrega, ingresa en el sistema el estado correspondiente:

Código	Descripción
100	Recibo Conforme
200	Recibo c/problema Reclamo
201	Recibo c/problema Diferencias
202	Recibo c/problema Contaminación
203	Recibo c/problema Falta capacidad
204	Recibo c/problema Producto no existe
205	Recibo c/problema Otros
300	No recibo Pedido no corresponde
301	No recibo Mix Equivocado
302	No recibo Fuera de horario
303	No recibo Equipo no apropiado
304	No recibo Falta elemento descarga
305	No recibo Otros

Nota: el ingreso en el sistema de la disconformidad por parte del cliente, implica que debe realizarse un seguimiento e interacción con el cliente, de manera tal que el reclamo sea gestionado, aclarado y solucionado, según corresponda el caso. Estas actividades son administradas vía CRM, según lo descrito en el procedimiento de "Administración de postventa"

{37}

61. Cierra transporte ingresando en SAP, mediante la transacción ***ZSD_OPTRA*.**

62. La recepción conforme debe ser registrada mediante la transacción ***ZSD_OPTRA***.

63. Confirma la entrega, ingresando la fecha y hora, y si dicha entrega fue completa o incompleta, mediante la transacción ***ZSD_OPTRA*.**

Capítulo 3 Lógica de Componentes

3.1 Visión General Técnica de Componentes

El siguiente diagrama muestra una visión a alto nivel del modelo de componentes de la arquitectura de integración. Este modelo de componentes está basado en una Arquitectura Orientada a Servicios (SOA).

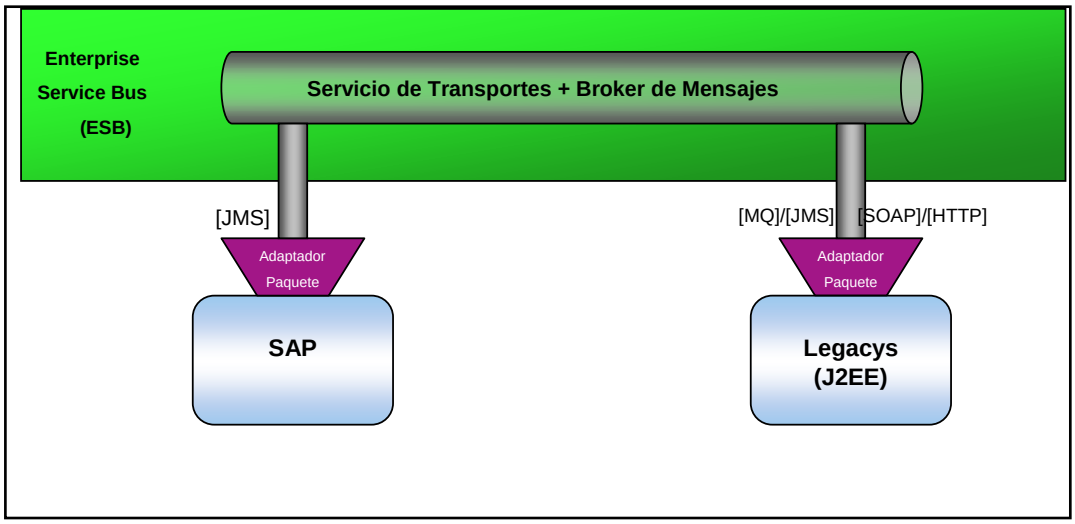


Figura 3-Arquitectura de Integración

Los componentes de la Arquitectura de Integración son los siguientes:

3.1.1 Enterprise Service Bus

Un Enterprise Service Bus (ESB) es una solución tecnológica que implementa SOA que oficia de intermediario de los mensajes de requerimientos y respuestas a invocaciones a servicios. Un ESB brinda la infraestructura común para la transmisión de los mensajes, así como servicios adicionales, como por ejemplo, la transformación de los mensajes.

Brinda principalmente 3 tipos de servicios:

1. Servicios de transporte de mensajes

Brinda las capacidades necesarias para trasmitir los mensajes, soportando distintas tecnologías de comunicación con las distintas partes. Las tecnologías soportadas por este componente para la comunicación son las siguientes:

- SOAP/HTTP
- WebSphere MQ
- J2EE JMS

2. Servicios de manejo de eventos

Brinda las capacidades para la publicación y recepción de eventos generados por las distintas aplicaciones, que disparan la ejecución de servicios provistos por el ESB.

3. Servicios de mediación de mensajes

Brinda las capacidades de trasformación de los mensajes, ruteo, seguridad, balanceo de carga, etc.

3.1.2 Servicios de Transporte

Los servicios de transporte de mensajes están soportados por WebSphere MQ [10] y por el soporte a Web services de los distintos componentes de integración involucrados.

3.1.3 Broker de Mensajes

Este componente brinda los servicios de manejo de eventos y mediación de mensajes. Este componente está implementado completamente por WebSphere Message Broker.

3.1.4 Adaptador SAP

Brinda la capacidad de conectar SAP al ESB, de forma que pueda consumir servicios provistos por el ESB o exponer servicios al ESB por medio de IDOC[11] y RFC[12].

Este componente traduce los mensajes intercambiados con el ESB a las IDOC de SAP.

El siguiente diagrama muestra cómo se mapean los distintos productos de la familia WebSphere para la implementación de SOA:

[11] Intermediate Document

[12] Remote Function Call

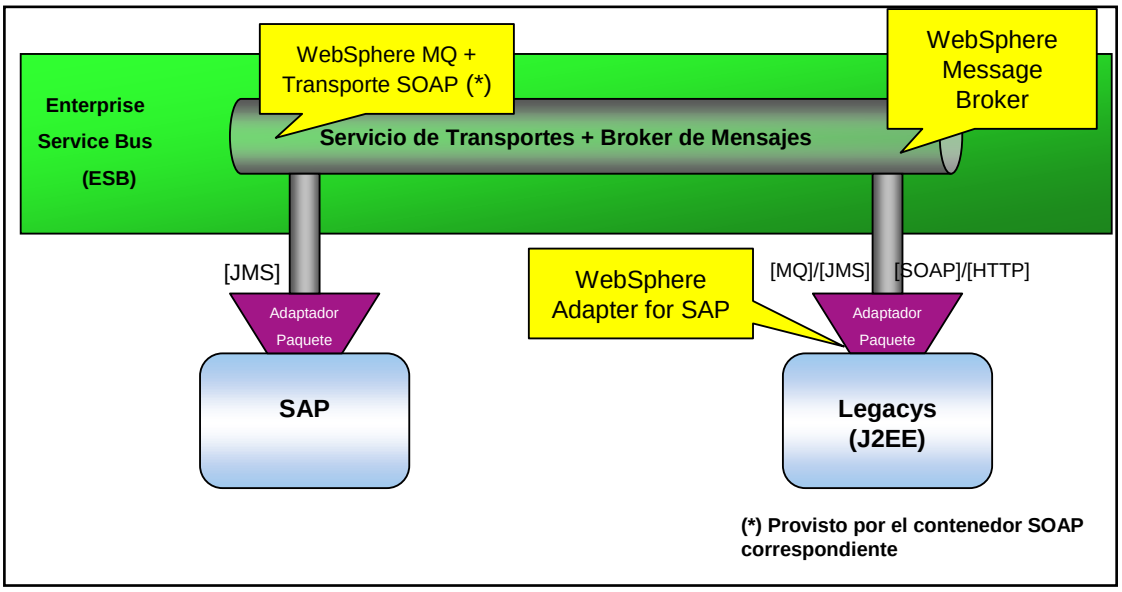


Figura 4-Integración con WebSphere

3.2 Implementación de los distintos componentes

Durante la etapa de desarrollo, cada vez que un escenario de integración necesitó la utilización de alguno de los componentes de integración, se debió generar y configurar distintos objetos. A continuación, se muestran los objetos generados y configurados para la implementación de cada componente de integración.

3.2.1 WebSphere MQ

- Definición de colas de entrada y salida
- Configuración del Provider JMS correspondiente

3.2.2 WebSphere Message Broker

- Creación de Message Sets (tipos de datos a intercambiar en los mensajes)
- Creación de flujos necesarios para implementar la lógica de mediación

3.2.3 WebSphere Aplication Server

- Información técnica de configuración
- Generación de Aplicaciones de Despliegue
- Configuración de recursos.

3.2.4 WebSphere Business Integration Adapter for SAP

- Generación de XSDs correspondientes a IDOC de SAP
- Configuración de SAP para mapear IDOC a XSDs
- Configuración correspondiente para mapear RFC.

3.3 Visión General Funcional de Componentes

El siguiente diagrama muestra una visión a alto nivel del modelo de componentes del SSD. Este modelo de componentes está basado en una Arquitectura Orientada a Servicios (SOA).

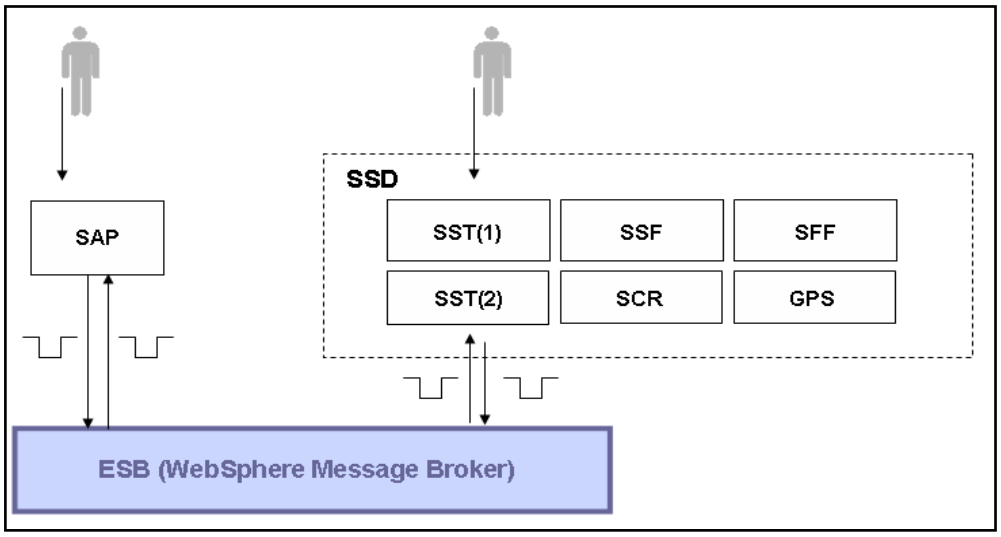


Figura 5-Componentes SSD

Sólo para hacer una diferencia de lo que implica la interfaz SSD funcionalmente la que se denominará SST parte uno y dos. En el resto del documento se definirá como SSD.

Los distintos componentes de las interfaces que componen el SSD (Sistema de Seguimiento de Despacho) son los siguientes:

- SST
  - SST(1) : SAP – BDP
  - SST(2) : BDP – SAP
- GPS
  - SAP – GPS
  - GPS – SAP
- SCR
  - SAP – SCR
- SSF
  - SAP – SCR2 (Disponibilidad de Flota)
- SFF
  - SAP – FuelFacs - SAP
  - FuelFacs – SAP

3.4 Implementación de los distintos componentes

La implementación de cada uno de estos componentes funcionales que dan el servicio de SSD, fue tratado en algunos casos como parte de un desarrollo J2EE y también como parte Integración (SOA). La intención de este documento no es explicar en detalle la implementación de cada uno de estos componentes, sino más bien dar un contexto general de lo que significa el desarrollo y cómo se compone el SSD, junto a cada una de sus interfaces.

- SST(1) : Es el encargado del registro de pedidos y la planificación proceso previo a la programación en un paquete externo BDP.

- SST(2) : Es el encargado de la programación y actualización de Pedidos y sus programaciones tanto en SSD como en SAP.

- GPS : Es el encargado de enviar información relevante respecto a las fechas y horas reales de despacho y llegada a cliente.

- SCR : Es el encargado de desplegar un graficador de rutas, seguimiento de la programación de pedidos, seguimiento planilla de pedidos.

- SSF : Para realizar la programación de los despachos en el BDP es necesario el envío de la disponibilidad de la flota de camiones desde SAP, en un proceso similar a la planificación.

- SFF : Determinación y creación del documento para inicio de carga en los Transportes. Impresión y generación del ticket de carga. Carga del combustible en el camión.

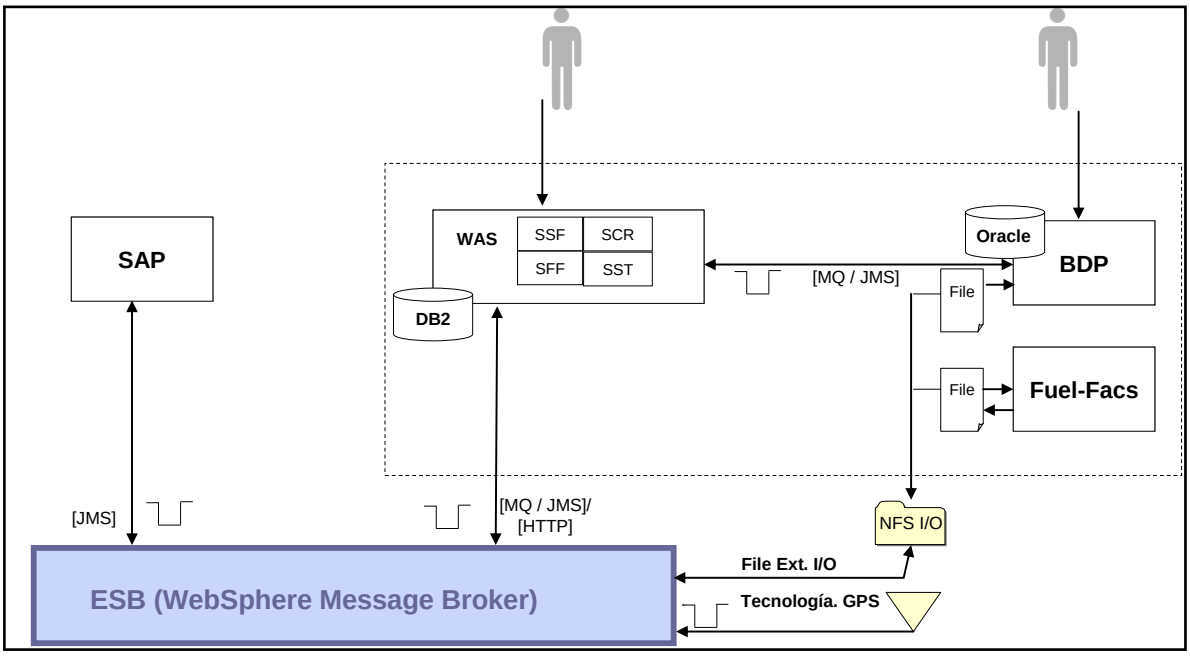


Figura 6-Implementación Componentes

En la implementación de todas estas interfaces, se aplicó el patrón SOA, que es la evolución de aplicaciones monolíticas y cerradas (como por ejemplo BDP y FuelFacs) a arquitecturas de servicios reutilizables en red distribuidas en capas.

Un **servicio** es una unidad de trabajo realizado por un proveedor de servicios para lograr los resultados finales de un servicio al consumidor. Ambos proveedor y el consumidor son las funciones que desempeñan los agentes de software en nombre de sus propietarios.

En la componente J2EE se encuentra el despliegue de más de una interfaz, como lo es SST, SCR, SSF, SFF. Por ende cada una de ellas utiliza el ESB (WebSphere Message Broker), para mediar la mensajería entre SAP y las distintas componentes expuestas como servicios. Adicionalmente el ESB es accedido en forma directa por algunos paquetes, ya sea por medio de MQ o de archivos planos.

No sólo el ESB es directamente accedido como un servicio por estos paquetes, sino que además los servicios expuestos en las componentes J2EE también son accedidos vía MQ, en forma directa sin la necesidad de una mediación por parte del ESB.

Hay que destacar que para aquellos servicios que son expuestos a través de la Web, hay que mencionar otra componente que es Portal de WebSpehere la que facilita el despliegue Web a través de una URL.

Los servidores de Portal proporcionan una estructura de presentación que tradicionalmente ha actuado como un consumidor de servicios Web, no obstante, los portales se han comenzado a activar también como proveedores de servicios.

Capítulo 4 Descripción de la implementación

En la implementación se utilizaron las siguientes herramientas:

- Rational Software Architect (RSA)
  - Permite transformar el proyecto de WebSphere Business Modeler a un modelo UML de los procesos de negocio, que sirven como punto de partida del modelado de la arquitectura y diseño de los componentes de software involucrados.
  - Adicionalmente, permite la generación de código "template" para la implementación de los servicios a partir del modelo UML.
- Rational Application Developer (RAD)
  - Permite la implementación de los servicios J2EE, así como también la generación de los Web services para que los mismos puedan ser utilizados por WebSphere Message Broker.
- WebSphere Message Broker Toolkit.
  - Utilizado para el desarrollo de los flujos necesarios para brindar la mediación entre la forma en que los servicios son expuestos por las aplicaciones y la forma en que los mismos esperan ser consumidos por parte del "cliente".

4.1 Definición de Ambiente de Desarrollo

Se utilizaron una combinación de:

- Estación de trabajo local
  - Corresponde al entorno local ("workspace"), en donde el desarrollador SOA realiza las tareas de modelado, programación, testeo unitario y generación de unidades de instalación. Utiliza los servicios del Servidor ClearCase para dar soporte al trabajo en equipo y control de versiones. Las pruebas unitarias las realiza el desarrollador en su propia estación de trabajo.
- Servidor de Desarrollo
  - Este ambiente recibe las unidades de instalación del servidor ClearCase y brinda el entorno de ejecución de los componentes del SOA (por ejemplo la ejecución de los flujos de integración) a los demás desarrolladores que participan en sus respectivos ambientes de desarrollo (por ejemplo, SAP, Sistema Legacy, etc.). Adicionalmente puede ser utilizado para realizar pruebas de integración durante el desarrollo, previo al traspaso de los elementos al ambiente de QA.

4.2 Implementación de componente Web

La implementación del componente Web, muestra típica de la aplicación de flujo de un navegador Web utilizando cualquiera de los clientes JDBC desde un servlet o de una solicitud EJB de acceso a bases de datos se muestra en la "Figura 7-Componente Web".

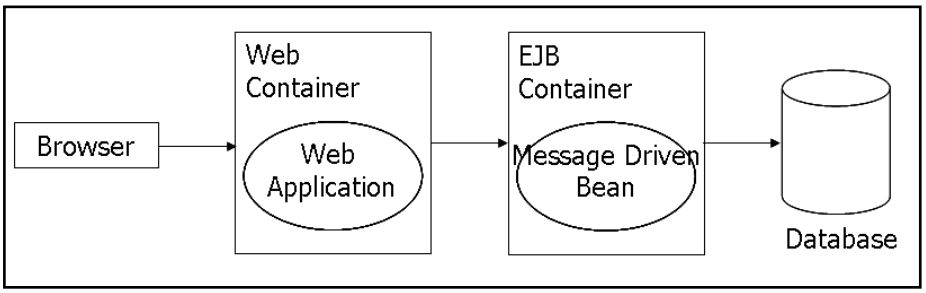


Figura 7-Componente Web

La implementación de esta interfaz consta de varias partes. En primer lugar es una arquitectura J2EE tres capas, Cliente Servidor. Se utilizó la forma de separar las preocupaciones en una aplicación de software y utilizar un patrón de arquitectura Modelo-View-Controller (MVC). El Modelo (M) representa a la empresa o la base de datos de código, la Vista (V) representa el diseño de páginas de código y el Contralor (C) representa el código de la navegación. Struts es el marco en el que está diseñado para ayudar a crear aplicaciones Web que utilizan una arquitectura MVC.

Para el acceso a los datos se utilizo el patrón DAO (Data Access Object) el cual consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

4.3 Descripción de los servicios Web

Web Services, como indica su propio nombre, son servicios ofertados vía Web. Representan una solución para la integración de aplicaciones en Internet y una buena y fácil alternativa para integrar aplicaciones sencillas (ej: intercambio de información) en una intranet. Los Servicios Web son componentes de aplicación que están diseñados para apoyar en forma ínter-operables máquina-a-máquina la interacción a través de una red. Esta interoperabilidad es adquirida a través de un conjunto de descriptores XML basados en estándares abiertos, como los Web Services Description Language (WSDL).

La idea es sencilla pero requiere:

- Un protocolo de intercambio de mensajes petición/respuesta sobre HTTP.
- Una forma de que clientes y proveedores puedan interactuar a través de los mensajes, es decir, un lenguaje de especificación de interfaces.

Se ha optado por utilizar SOAP (Simple Objet Access Protocol) como protocolo de intercambio de mensajes. Es un protocolo sencillo basado en XML y estandarizado por el W3C[13].

El lenguaje de especificación de interfaces utilizado en servicios Web es WSDL (Web Services Description Language). WSDL permite especificar en XML las operaciones y tipos de datos de un

[13] World Wide Web Consortium

servicio Web. Así, aunque el cliente y el servidor estén escritos en lenguajes distintos (por tanto con sintaxis y tipos de datos diferentes) pueden interactuar al utilizar un lenguaje neutral para comunicarse.

En un escenario típico de Web Services una aplicación de negocio envía una petición vía HTTP a un servicio situado en una URL. El servicio recibe la petición, la procesa y devuelve una respuesta también sobre HTTP.

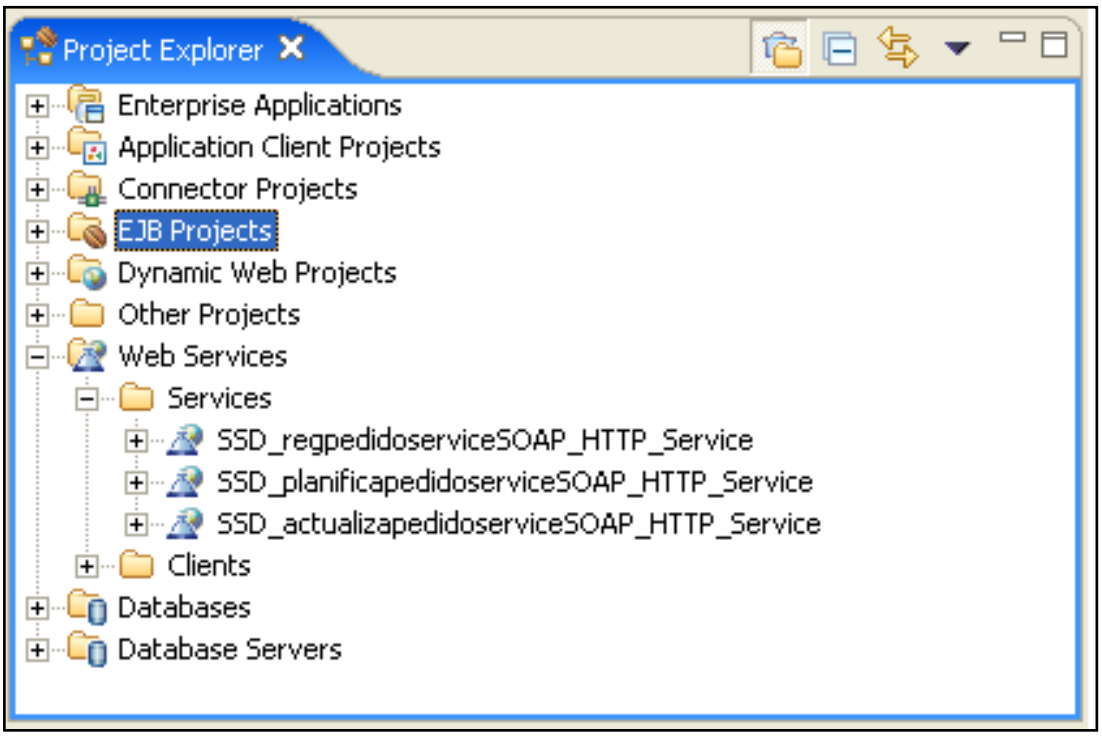


Figura 8-WebServices

Para crear estas tres componentes Web Services, se hizo a partir de una Definición de Servicios Web (WSDL).

4.3.1 La Creación de una definición del servicio Web desde un mensaje Set

Se describe cómo generar un documento del lenguaje descriptivo de los servicios del Web (WSDL) de un sistema de mensaje, eso especifica la interfaz para un servicio del Web.

Para terminar esta tarea se debe haber terminado ya las tareas siguientes:

- Crear un message set project
- Crear un message set
- Crear un message definition file
- Crear un message category file

Antes de comenzar esta tarea se debe observar los puntos siguientes:

- Utilizar el estilo del documento de WSDL siempre que sea posible.
- Donde sea necesario el estilo WSDL del RPC, utilizar la codificación del literal.
- Se debe sustituir cualquier construcción desaprobada antes de generar las representaciones de WSDL de sus modelos del mensaje.

4.3.2 Descripción de los servicios MDB

Son Componentes Beans Manejados por Mensajes (*Message-Driven Bean)* que describen procesos de negocio que son accedidos asíncronamente:

- Se subscriben y reaccionan a determinados eventos
- Facilitan la integración de sistemas ya existentes
- No se declaran interfaces ya que sólo reaccionan a un método: ***onMessage()***

La especificación Enterprise JavaBeans (EJB) versión 2.0 introduce un nuevo tipo de EJB [14] llamado "bean controlado por mensaje" (MDB). Mensajes asíncronos impulsados por el bean son mensajes de los consumidores que se ejecutan en el contexto de un contenedor de los servidores de aplicaciones EJB. Esto permite que el contenedor EJB para proporcionar servicios adicionales al Message-driven bean durante la tramitación de un mensaje tales como las transacciones XA, la seguridad, la concurrencia y el mensaje de acuse de recibo.

El contenedor EJB también es responsable de la gestión de la vida útil del mensaje impulsado y del Message-driven bean para invocar un nuevo mensaje. Cuando se recibe un mensaje de un determinado bean es el consumidor.

WebSphere Application Server V6 es totalmente compatible con la versión 1.4 de la especificación J2EE, que exige que los servidores de aplicaciones den apoyo a la versión 2.1 de la especificación EJB, que es la que se utilizó en la implementación del MDB.

XA DataSource en MDB

Una XA operación, en los términos más generales, es una "transacción global", que puede abarcar múltiples recursos. Una transacción XA no siempre implica un solo recurso.

Una transacción XA implica un gestor de la coordinación de las transacciones con una o más bases de datos (o de otros recursos como JMS) todos los que participan en una sola transacción a nivel mundial. Las No XA transacciones no tienen ningún coordinador de transacción y un único recurso hace todo el trabajo (esto se denomina a veces las transacciones locales) [11].

Transacciones XA provienen de la X/Open grupo de especificaciones de transacciones distribuidas y globales. JTA [15] incluye las especificaciones X/Open XA en forma modificada.

En esta implementación el MDB engloba una transacción XA que comprende desde los mensajes hasta los diversos recursos de acceso a los datos como lo son JMS, Oracle y DB2.

[14] Enterprise Java Bean

[15] Java Transaction API

Por ende si algunas de las operaciones de mensajería o de commit a la BD falla, como consecuencia de esto falla la transacción global y es invocado un rollback. Cuando esto ocurre el Message-driven bean devuelve el mensaje a la cola del oyente (Listener Port), para que se vuelva a procesar.

El oyente (Listener Port) recupera los mensajes de la cola de mensajes y los pasa a la Message-driven bean. Si hay un error en el procesamiento de un mensaje después de la Message-driven bean el mensaje del oyente (Listener Port) se devolverá a la cola MQ.

Desde luego una vez de vuelta en la cola este mensaje será capturado de nuevo por el oyente (Listener Port) y un ciclo sin fin de intercambio de mensajes será iniciado, lo que ocasiona la caída del oyente (Listener Port).

Hay algunos parámetros que pueden poner fin a esto además de la configuración de una cola de BKOUT, donde dejaremos los mensajes con problemas. Esta política será utilizada para todas aquellas colas que requieran manejar errores y donde se necesite respaldar el mensaje con problemas sin afectar el funcionamiento del proceso. De esta forma el mensaje ira a una cola de BKOUT sin ocasionar la caída del oyente (Listener Port).

El número de reintentos elegido fue mayor a uno y el umbral de la cola de bkout definida para el oyente (Listener Port) fue inferior en uno respecto a la propiedad de Maximun retries del oyente.

4.4 Apoyo en la transacción de WebSphere Application Server

Esta sección proporciona información conceptual sobre el apoyo a las operaciones de transacción proporcionado por el Servicio de WebSphere Application Server [12].

Una transacción es la unidad de actividad en el que múltiples actualizaciones de los recursos se pueden hacer atómicas (como una unidad indivisible de trabajo) de tal manera que todas o ninguna de las actualizaciones se hacen permanentes. Si la transacción se refiere a múltiples administradores de recursos, por ejemplo, múltiples gestores de base de datos, como es este caso (Oracle, DB2, JMS), un gestor de las transacciones externas se requiere para coordinar los distintos administradores de recursos. WebSphere Application Server es un gestor de la transacción que puede coordinar a nivel global las transacciones, puede ser un participante en una transacción recibido como global y puede también proporcionar un entorno en el que el gestor de los recursos locales puede ejecutar transacciones.

La forma en que las aplicaciones utilizan operaciones depende del tipo de componente de aplicación:

El Bean de Mensajes es gestionado por el contenedor el cual gestiona las transacciones del componente EJB. Adicionalmente el MDB contiene una lógica de negocios que maneja una transacción XA distribuida con los siguientes elementos Oracle, DB2, JMS:

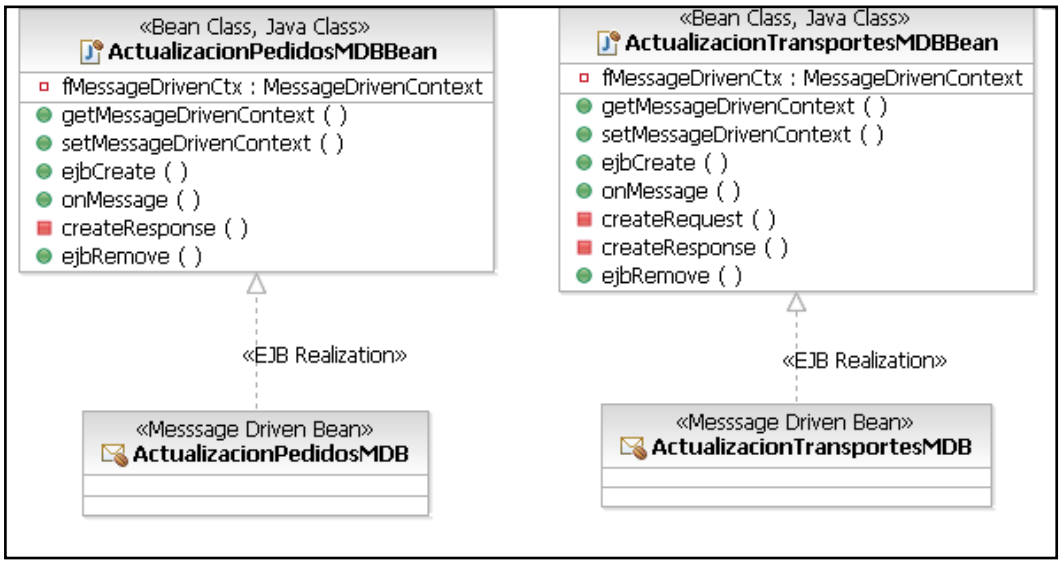


Figura 9-Lógica de Negocio con Beans

El Message-driven beans implementa la interfaz ***javax.jms.MessageListener*** que se puede utilizar para mensajería JMS, que va a ser usada por compatibilidad con WebSphere Application Server versión 6, como una configuración JMS message-driven beans en contra de un puerto oyente.

Los Componentes Web (servlets) y los componentes del cliente de aplicaciones de uso componentes administrador de transacciones. Tienen una implementación del componente Web ***SSDConsultasWeb*** durante la tramitación de una declaración SQL COMMIT, el gestor de bases de datos atómicamente comete múltiples sentencias SQL a una base de datos relacional (DB2 UDB iSeries).

También es importante recalcar que a pesar de que esta aplicación trabaja con dos motores de Base de Datos, solo realiza cambios en la base de datos de DB2, en la otra, Oracle 8i, en cambio solo realiza operaciones de consulta sin commit.

Capítulo 5 Descripción Componentes de Integración (SOA)

Esta sección describe las decisiones referidas a la arquitectura de cada uno de los componentes de integración de SOA. Los componentes que forman parte de la arquitectura de integración de SOA son:

- Enterprise Service Bus.
- Motor de Procesos.
- Adaptadores.

Los requerimientos de integración entre los distintos sistemas incluyen, entre otros, las siguientes necesidades:

- Desacoplamiento entre las partes involucradas
- Transformación entre distintos protocolos de transporte de mensajes
- Transformación entre distintos formatos de mensajes
- Incluir lógica de mediación entre la generación del mensaje y la recepción del mismo
- Implementar distintos patrones de integración de aplicaciones, como por ejemplo “point-to-point”, “ESB” (Enterprise Service Bus), etc.

Este es un patrón de arquitectura probado, generalmente utilizado como solución para este tipo de requerimientos. En particular, brinda:

- Gran flexibilidad
- Rehúso
- Desacoplamiento entre los participantes

En este componente de arquitectura será el único lugar donde se incluirá la lógica necesaria para:

- Aplicar transformaciones entre distintos protocolos de transporte de mensajes
- Aplicar transformaciones entre distintos formatos de mensajes
- Aplicar lógica de mediación entre la generación de mensajes y la recepción de los mismos.

5.1 Implementación del ESB

Este componente de la plataforma SOA se implementó utilizando los siguientes productos:

- WebSphere MQ V6.
- WebSphere Message Broker V6.
- WebSphere Message Broker File Extender.

5.1.1 Características

- Se cuenta con la posibilidad de distintos mecanismos de transporte de mensajes, principalmente:

    - Web services, mecanismo estándar y orientado a invocaciones sincrónicas
    - Colas de mensajes (WebSphere MQ), mecanismo propio de la plataforma IBM, orientado a invocaciones asincrónicas
- Permite utilizar adaptadores existentes, como por ejemplo:
    - Adaptadores para SAP
    - Adaptadores para archivos planos
- Es una solución probada, y con muchos casos de referencia
- Se utiliza la versión V6 de ambos productos por las siguientes razones:
    - Fecha de puesta en producción de la solución
    - Integración con otros productos utilizados (por ejemplo WebSphere Process Server)
    - Capacidades mejoradas en la utilización de Web Services (WebSphere Message Broker).

Las tecnologías a utilizar para el transporte de mensajes son las siguientes:

- HTTP (SOAP/HTTP), como mecanismo estándar y orientado a invocaciones sincrónicas
- JMS como mecanismo estándar dentro de la plataforma J2EE, orientado a invocaciones asincrónicas. Utilizado también por los adaptadores de la familia WebSphere.
- MQ como soporte para otras plataformas no J2EE, orientado a invocaciones asincrónicas.

Esta combinación de tecnologías y protocolos permite ofrecer una gran flexibilidad, en cuanto a la forma de invocación (sincrónica y asincrónica), formato de mensajes (SOAP, XML, otros) y plataformas (J2EE, otros lenguajes, paquetes y aplicaciones por medio de adaptadores).

5.1.2 Persistencia de Mensajes

WebSphere MQ ofrece la capacidad de ofrecer persistencia de los mensajes. Esto significa que frente a una caída del servicio de MQ, los mensajes no se pierden. HTTP, como protocolo de transporte de mensajes, no ofrece capacidades de persistencia de mensajes.

Por ende es necesario definir un lineamiento para asegurar una forma consistente de manejar las caídas del servicio de MQ por parte de las aplicaciones. Esto se logró definiendo la siguiente estrategia que consiste en definir unas propiedades como la *storage* a cada cola del MQ (queue) como sigue:

5.1.2.1 Configuración de Queues en WebSphere MQ

QUEUE	JNDI Prefix	Listener Port	TYPE	WAS I/O
BRKR.SSD.ACTUALIZA.PEDIDOS.SERV.IN	jms/	NO	LOCAL	O
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.IN	jms/	NO	LOCAL	O

BRKR.SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	jms/	NO	LOCAL	O
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.OUT	jms/	NO	LOCAL	O
BRKR.SSD.PLANIFICA.PEDIDOS.SERV.OUT	jms/	NO	LOCAL	O
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT	jms/	NO	LOCAL	I
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN	jms/	SI	REMOTE	I
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.IN	jms/	SI	LOCAL	I

QUEUE	TYPE	Nombre Remotas
BRKR.SSD.ACTUALIZA.PEDIDOS.SERV.IN	LOCAL	
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.IN	LOCAL	
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.OUT	LOCAL	
BRKR.SSD.ACTUALIZA.TRANSPORTE.REQ.SAP	LOCAL	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.IN	LOCAL	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.OUT	LOCAL	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.MAXRETRY	LOCAL	
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.IN	REMOTE	LS_UP_FILES_BDP.REQ
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT	LOCAL	
BRKR.SSD.PLANIFICA.PEDIDOS.SERV.OUT	LOCAL	
BRKR.SSD.REGISTRO.PEDIDOS.TRASLADO.REQ.SAP	LOCAL	
BRKR.SSD.REGISTRO.PEDIDOS.VENTA.REQ.SAP	LOCAL	
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN	REMOTE	SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	REMOTE	SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT
BRKR.SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	LOCAL	
AQUILA_DES	LOCAL	

QUEUE	PROFUNDIDAD	PERSISTENTE	BACKOUT
BRKR.SSD.ACTUALIZA.PEDIDOS.SERV.IN	5000	SI	SI
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.IN	5000	SI	SI
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.OUT	5000	NO	
BRKR.SSD.ACTUALIZA.TRANSPORTE.REQ.SAP	5000	SI	SI
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.IN	5000	SI	SI
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.OUT	5000	SI	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.MAXRETRY	5000	SI	
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.IN	5000		NO
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT	5000		NO
BRKR.SSD.PLANIFICA.PEDIDOS.SERV.OUT	5000	SI	SI
BRKR.SSD.REGISTRO.PEDIDOS.TRASLADO.REQ.SAP	5000	SI	SI
BRKR.SSD.REGISTRO.PEDIDOS.VENTA.REQ.SAP	5000	SI	SI
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN	5000		SI
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	5000		
BRKR.SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	5000	SI	SI
AQUILA_DES	5000	SI	

QUEUE	BACKOUT THRESHOLD	LARGO MAXIMO MENSAJE	REINTENTOS
BRKR.SSD.ACTUALIZA.PEDIDOS.SERV.IN	1	4MB	
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.IN	1	60MB	
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.OUT		4MB	
BRKR.SSD.ACTUALIZA.TRANSPORTE.REQ.SAP	1	60MB	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.IN	2	4MB	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.OUT		4MB	
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.MAXRETRY		4MB	
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.IN		4MB	NO
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT		4MB	NO
BRKR.SSD.PLANIFICA.PEDIDOS.SERV.OUT	1	4MB	
BRKR.SSD.REGISTRO.PEDIDOS.TRASLADO.REQ.SAP	1	4MB	
BRKR.SSD.REGISTRO.PEDIDOS.VENTA.REQ.SAP	1	4MB	
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN	2	4MB	NO
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT		4MB	NO
BRKR.SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	1	4MB	
AQUILA_DES		4MB	

QUEUE	DESCRIPCION
BRKR.SSD.ACTUALIZA.PEDIDOS.SERV.IN	put message SAP IDOC Transporte
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.IN	put message SAP RFC Transporte
BRKR.SSD.ACTUALIZA.PROGRAMADOS.SERV.OUT	get message SAP RFC Transporte (response)
BRKR.SSD.ACTUALIZA.TRANSPORTE.REQ.SAP	get message Transporte confirmación SAP
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.IN	Listener Port ActualizaPedidos Reintentos 10'
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.BDP.OUT	Timer espera Transportes sin respuesta de SAP
BRKR.SSD.ACTUALIZA.TRANSPORTES.MSG.MAXRETRY	put message excede numero intentos Timer de Espera
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.IN	put message Subida Archivos BDP (Mensaje Aviso BDP)
BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT	get message Subida Archivos BDP (Mensaje respuesta)
BRKR.SSD.PLANIFICA.PEDIDOS.SERV.OUT	put message Subida Archivos BDP Pedidos a Planificar SSD
BRKR.SSD.REGISTRO.PEDIDOS.TRASLADO.REQ.SAP	get message Registro Pedidos Traslados
BRKR.SSD.REGISTRO.PEDIDOS.VENTA.REQ.SAP	get message Registro Pedidos Ventas
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN	Listener Port Accepts
SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	put message response Accepts
BRKR.SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT	put message response Accepts (BROKER)
AQUILA_DES	TRANSMISION (SSD/AQUILA)

5.2 Implementación de WebSphere Message Broker

Las tecnologías a utilizar para el transporte de mensajes son las siguientes:

- HTTP (SOAP/HTTP)
- JMS
- MQ

El diseño de los flujos de integración se realiza de forma tal que la lógica de mediación de los mismos no dependa de la tecnología de transporte utilizada para el transporte de los mensajes de las aplicaciones con el ESB.

Cada servicio será expuesto a las aplicaciones que lo requirieren utilizando 2 flujos:

- Un flujo que expone el servicio a nivel del ESB para que sea utilizado por las aplicaciones que lo requieren ("Flujo Externo"). Este flujo considera las particularidades de cada tecnología de transporte de mensajes (por ejemplo, si se utiliza MQ, lee y consume los mensajes de las colas de mensajes).
- Un flujo que implementa la lógica de mediación necesaria e invoca al servicio que brinda el proveedor ("Flujo Interno"). Este flujo es consumido por los flujos externos como un "sub-flujo" (ver Figura 10-Arquitectura consumo Servicios).

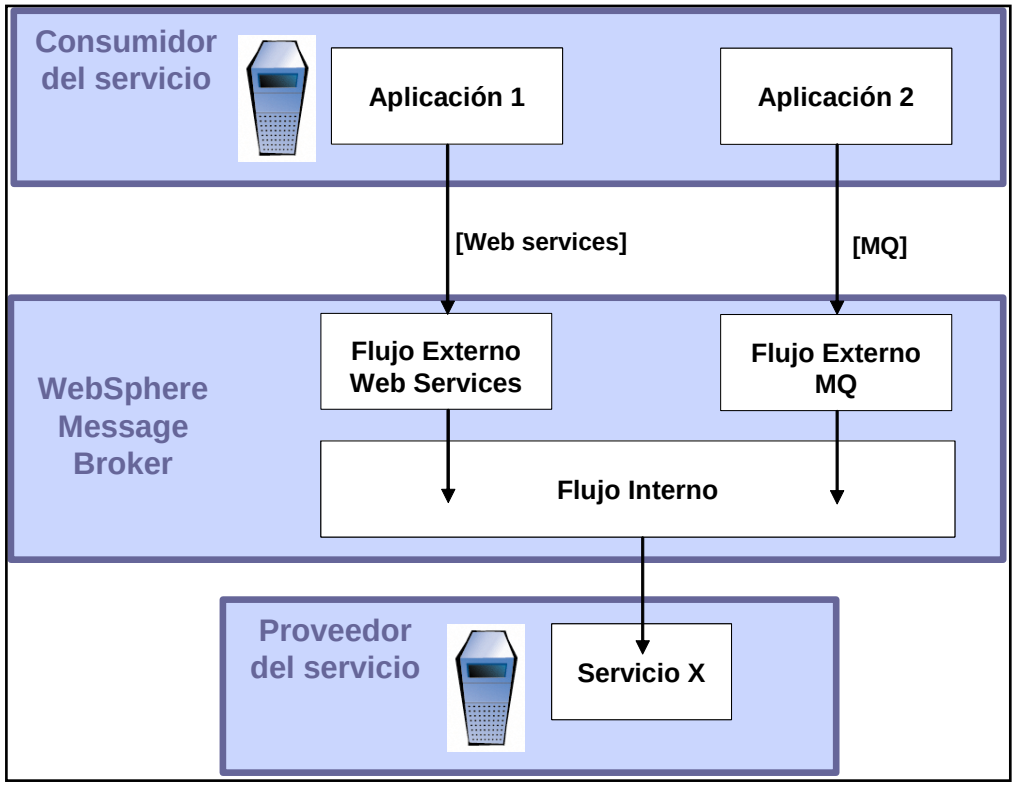


Figura 10-Arquitectura consumo Servicios

La implementación de los flujos no está restringida ni genera dependencias con ninguna tecnología de transporte de mensajes, dado que esto reduce la flexibilidad en la implementación del ESB y en la capacidad de integrar al mismo a distintas aplicaciones, desarrolladas con distintas tecnologías, ejecutando en distintas plataformas.

5.3 Estructura Estándar de intercambio de Datos (IDOC, RFC)

El IDoc [13] es un estándar de estructura de datos para el intercambio electrónico de datos (EDI[16]) entre programas de aplicaciones escritas para el popular sistema de empresa SAP o entre una aplicación de SAP y un programa externo (J2EE). Los IDocs sirven de vehículo para la transferencia de datos de SAP en la solicitud de habilitación Link (ALE[17]) [14]. Los IDocs se utilizan para las operaciones asíncronas: cada IDoc generado existe como un archivo de texto autónomo que puede ser transmitido a quién lo solicitó sin la necesidad de la conexión a la base de datos. Otro mecanismo de SAP, el Business Application Programming Interface (BAPI) se utiliza para transacciones sincrónicas.

Entre los IDoc hay tipos estándares para las distintas categorías de datos, tales como órdenes de compra o facturas, que pueden dividirse en categorías más específicas llamados tipos de mensajes. Una mayor especificidad de IDoc significa que un tipo es capaz de almacenar sólo los datos necesarios para una determinada operación, que aumenta la eficiencia y disminuye la demanda de recursos. En esta implementación se utilizaron los siguientes tipos de mensajes:

- OILTP50 (Pedido)
- ORDER05 (Reposición)
- OILTPI01 (Transporte)

Una RFC solo es una función que se puede llamar desde un sistema externo a SAP, el cual puede ser otro SAP u otro tipo de sistema [15]. Aquí se definen una RFC llamada ***"zvi_act_pedido_bdp2"*** (Modulo de acceso remoto). Las aplicaciones de una RFC son más extensas : permiten intercambiar datos entre SAP y el sistema llamante por lo que se pueden definir infinidad de cosas.

Esta función permite realizar operaciones en SAP, como desprogramar y re programar un transporte, enviando para tal efecto solo un mensaje. Esto es una forma rápida de solucionar la problemática de manejar IDOC para eliminar el transporte y volver a crear dichas programaciones.

No existe una estructura definida en SAP para manejar las reposiciones[18] (ORDER05) en los transportes, por ende cuando hay una combinación de casos de pedidos de venta y reposición, se utilizan las RFC.

[16] Electronic Data Interchange

[17] Application Linking and Enabling

[18] Reposiciones de combustible a los clientes

Además esta estructura que está definida para un grupo de casos, tiene una particularidad de que solo permite enviar datos de pedidos de venta y reposición, sin embargo no está definida para enviar datos de pedidos secundarios. Cuando este caso se presenta se utilizan las definiciones de IDOC y se envía una desprogramación seguida de una programación.

5.4 Solución de Arquitectura WebSphere Message Broker

Servicio Actualizar Pedidos IDOC: Este flujo es el encargado de actualizar las programaciones (Transportes) BDP - SAP utilizando para ello IDOC (OILSHI).

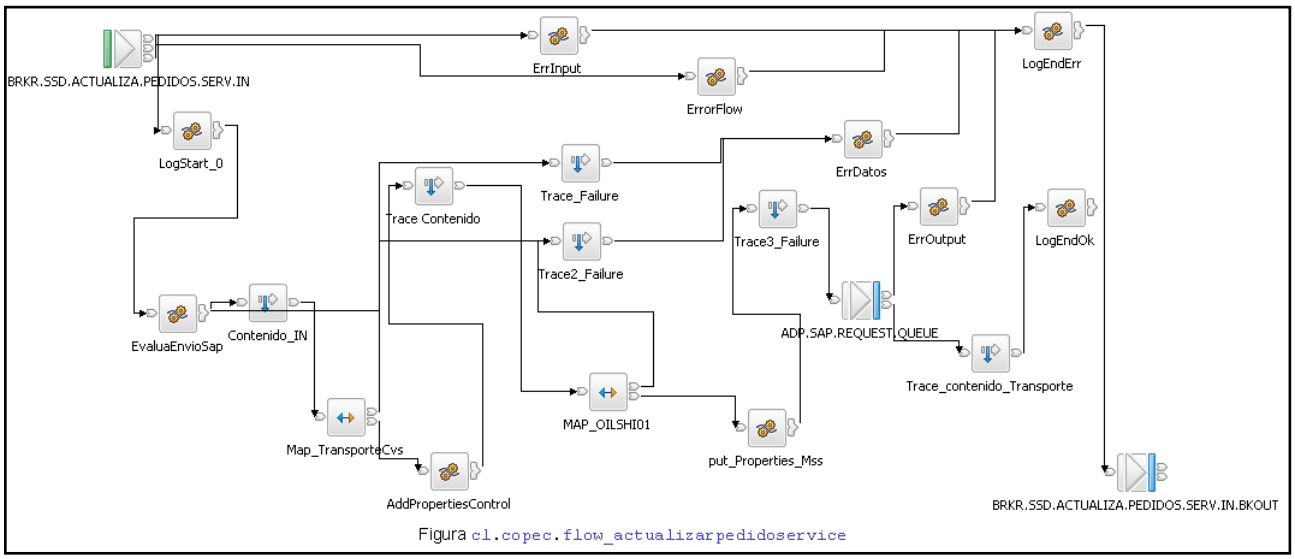


Figura 11-Flujo Actualizador Pedidos IDOC

Servicio Actualizar Pedidos RFC: Este flujo es el encargado de actualizar las programaciones (Transportes) BDP - SAP utilizando para ello llamadas a funciones RFC.

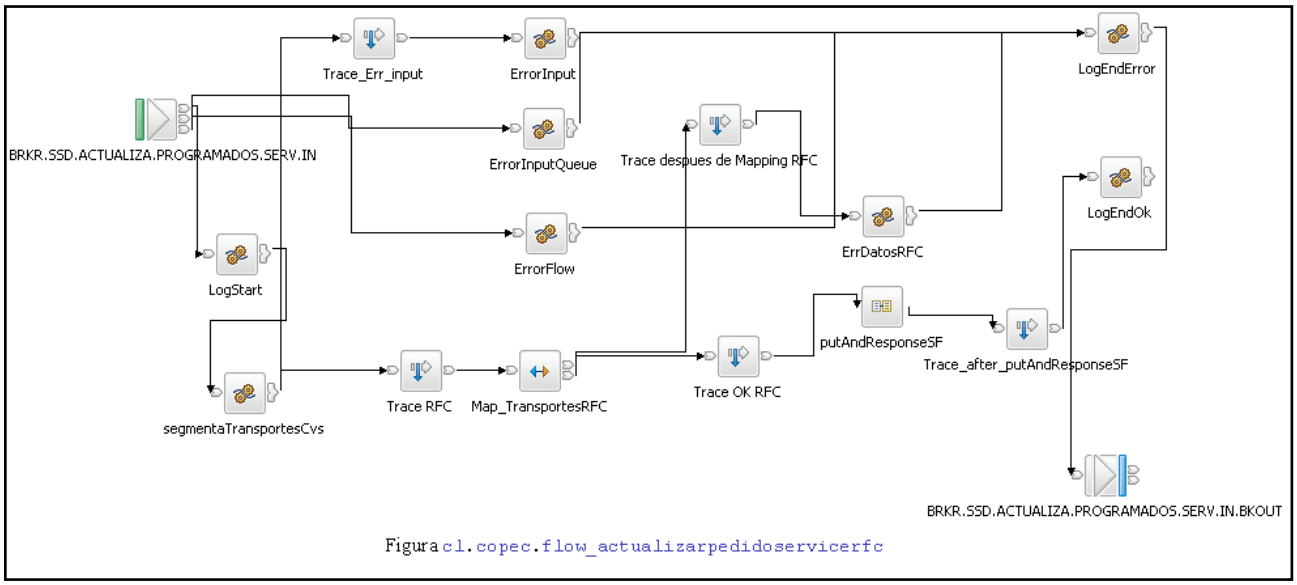


Figura 12-Flujo Actualizador Pedidos RFC

Servicio Actualizar Pedidos SAP: Este flujo es el encargado de actualizar/insertar las programaciones (Transportes) SAP – SSD. Los mensajes recibidos son IDOC de Transportes llamados OILSHI.

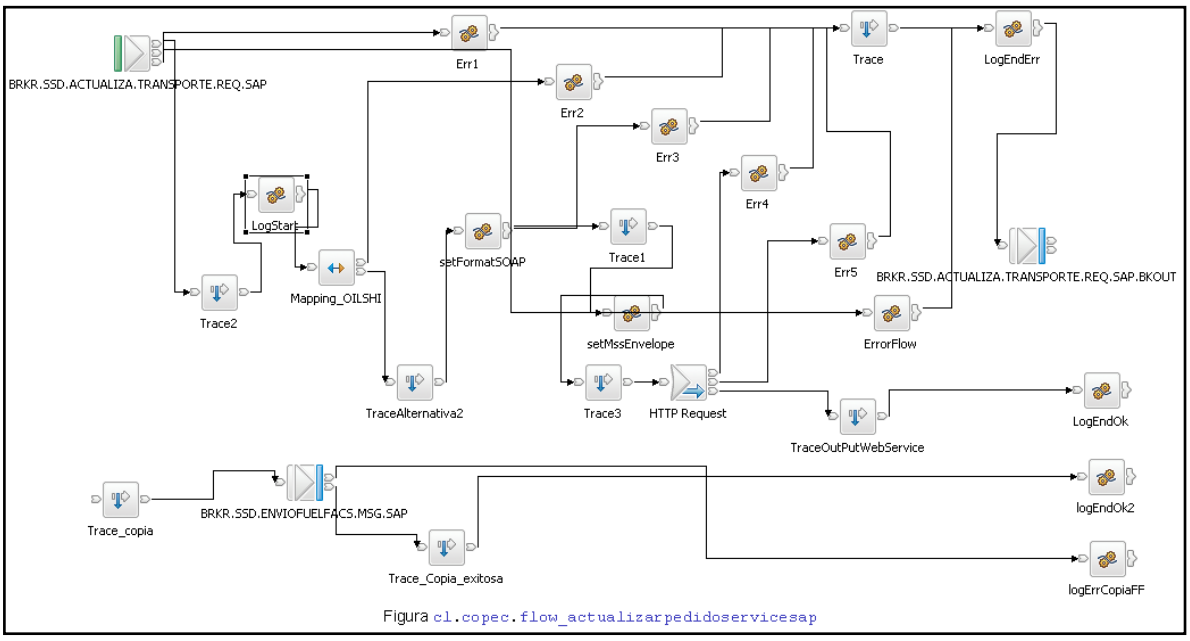


Figura 13-Flujo Programaciones

Servicio Actualiza Transportes: Este flujo es el encargado de actualizar cada un minuto los mensajes para re procesar los lotes de transportes pendientes a enviar SSD - SAP.

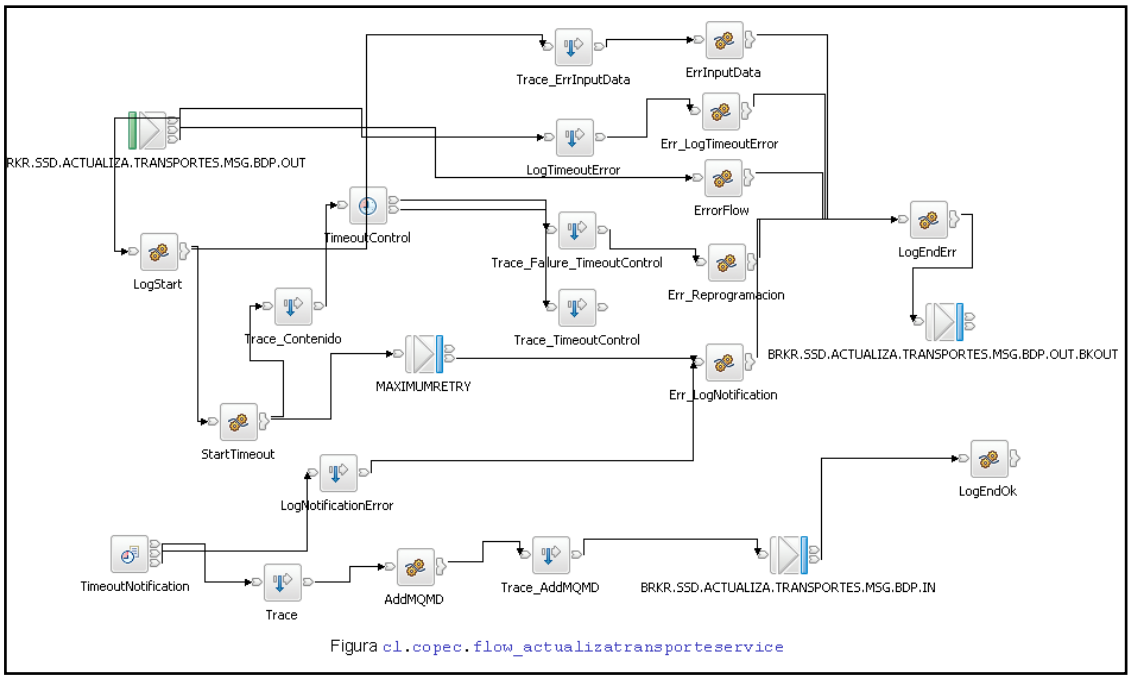


Figura 14-Flujo Actualizador Transportes

Servicio Planificar Pedidos : Este flujo es el encargado de la creación de un archivo plano, con los pedidos a planificar en BDP.

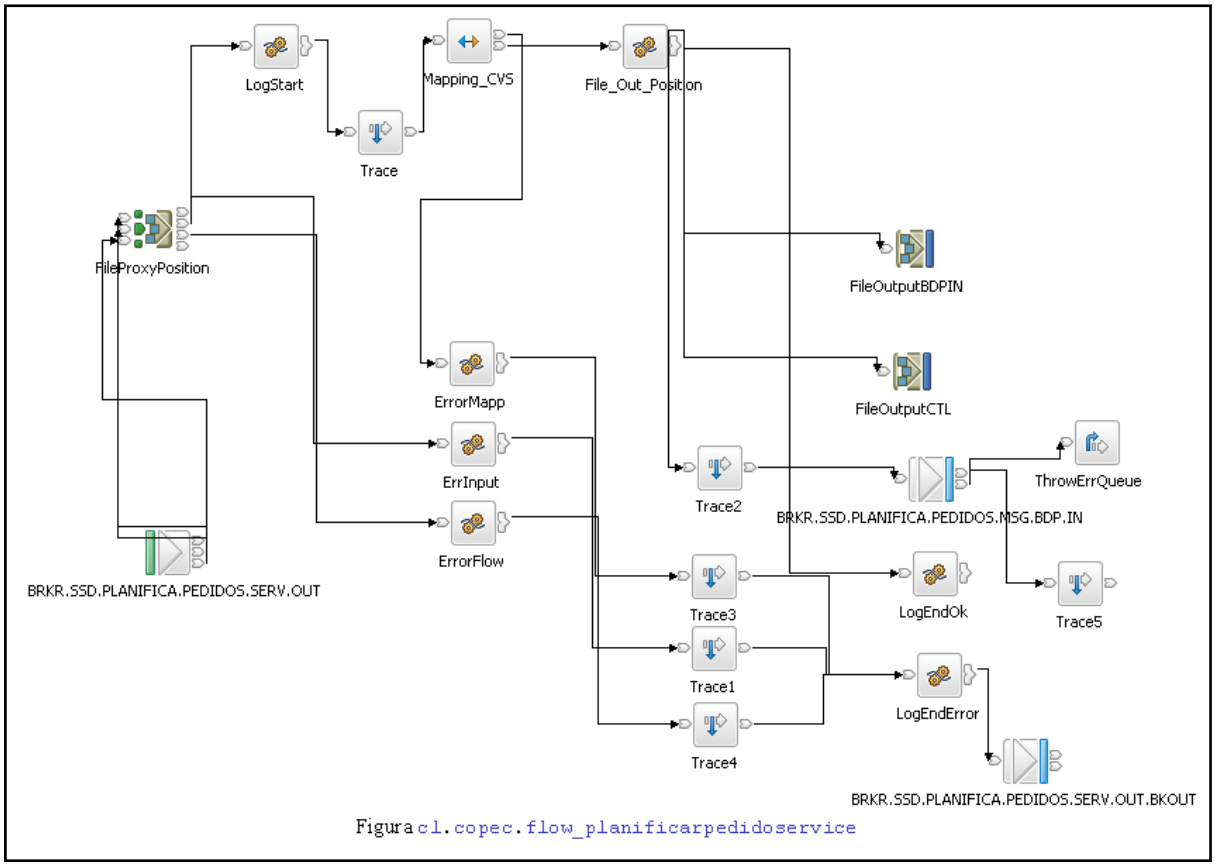


Figura 15-Flujo Planificador Pedidos

Servicio Registro Pedidos: Este flujo es el encargado de registrar los pedidos de ventas creados en SAP en la base de datos intermedia SSD (IDOC OILTPI50).

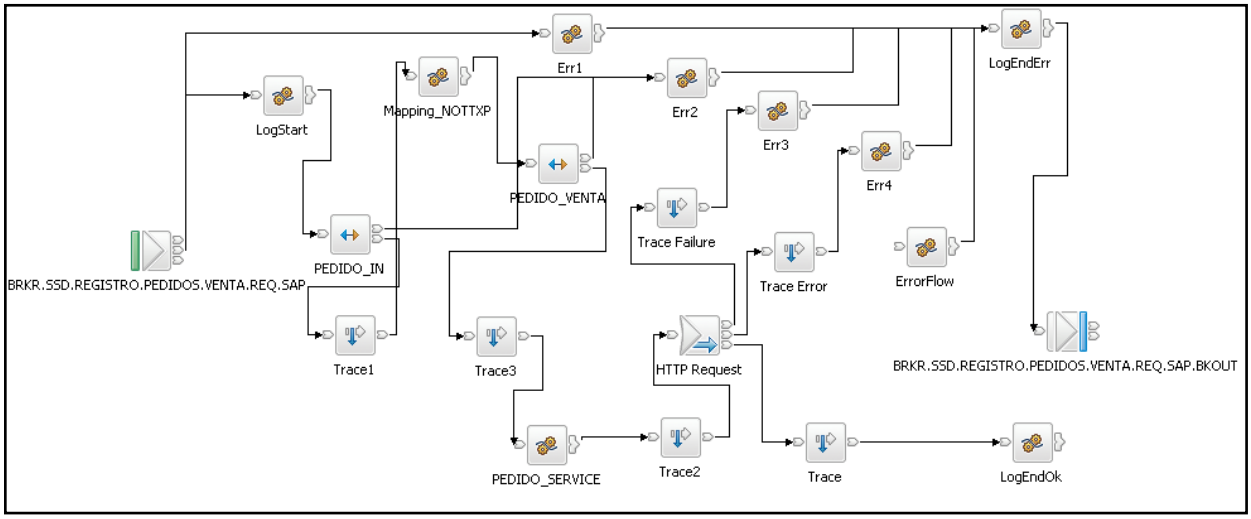


Figura 16-Flujo Registro Pedidos

Servicio Registro Pedidos Traslados: Este flujo es el encargado de actualizar/registrar los Pedidos de reposición creados en SAP en la base de datos intermedia SSD. (IDOC ORDER05).

Las distintas soluciones de arquitectura en el broker se dividen funcionalmente en dos grandes grupos: están los que reciben la información proveniente de SAP y los que son accedidos por las componentes J2EE, ya sea desde una action de la Web o de un Message-driven bean. Todos ellos a través de una mensajería JMS.

Cada Broker requiere un subyacente MQ que se utiliza para la gestión y el despliegue. WebSphere Message Broker se aprovecha de MQ como el mecanismo subyacente de la mayoría de las funciones de gestión, incluido el despliegue de los servicios basado en WebSphere Message Broker Toolkit.

Además de lo anterior, como una decisión de arquitectura se definió excluir cualquier diseño que contemplara un acceso a la base de datos desde el broker, por ser esta una política de bajo performance y por lo tanto dejar al broker de forma exclusiva sólo a la transferencia de mensajes desde SAP a los Legacy y viceversa.

5.5 Manejo de Servicios Web con IBM WebSphere Message Broker

Todos estos Flujos están relacionados con el Message Set Definition, que contiene la estructura de todos los mensajes de entrada y salida de cada uno de los flujos.

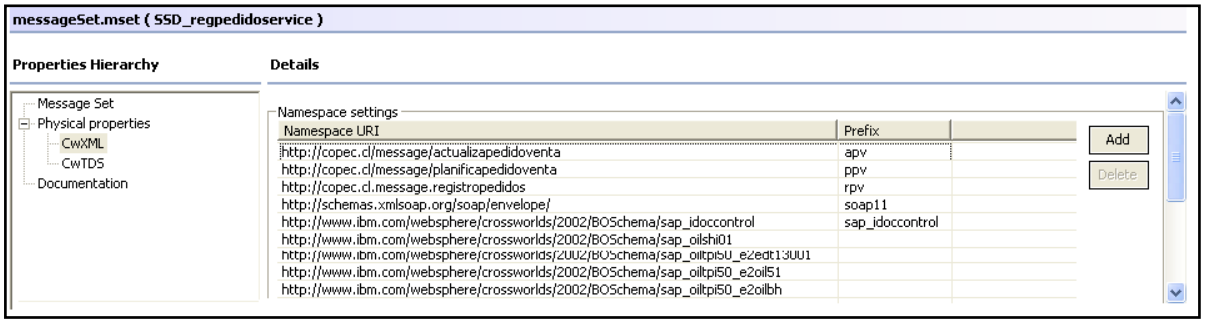


Figura 17-Message Set Definition

El archivo creado en un servicio Web es un WSDL. Este es creado a partir de un Message Set, como la "Figura 18-programados.mxsd" lo muestra, a partir de un Message Definition File "programados.mxsd".

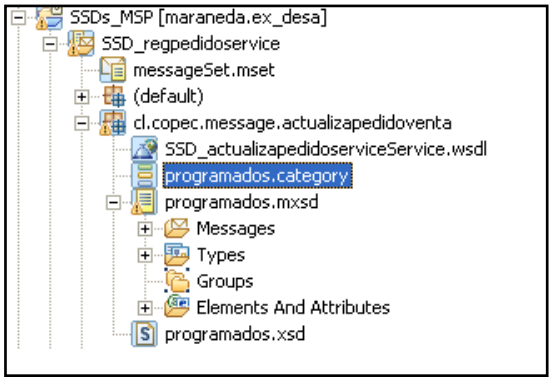


Figura 18-programados.mxsd

Este es un WSDL de los servicios Web que se implementaron (ver “Figura 19-WSDL de un WebService”):

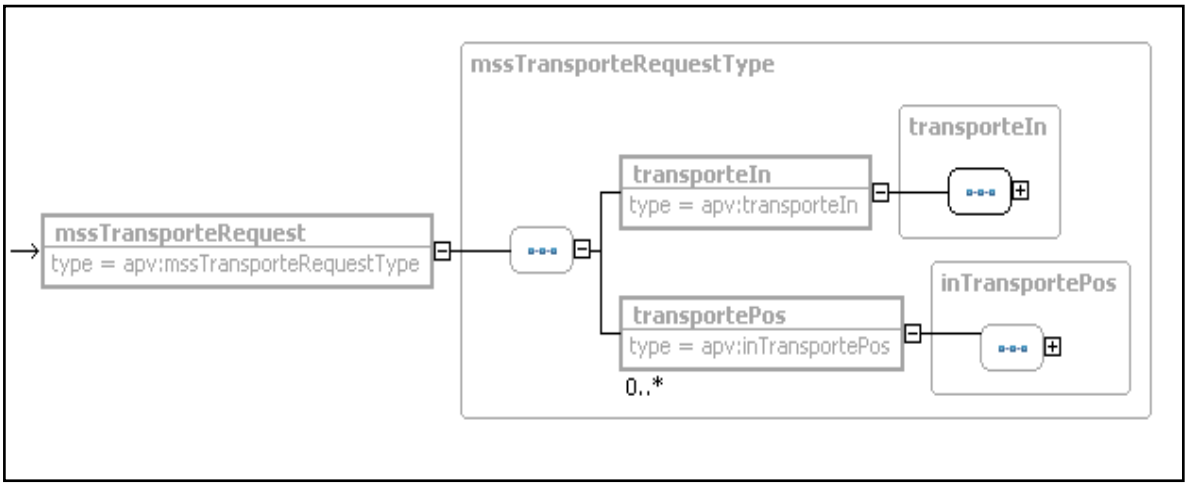


Figura 19-WSDL de un WebService

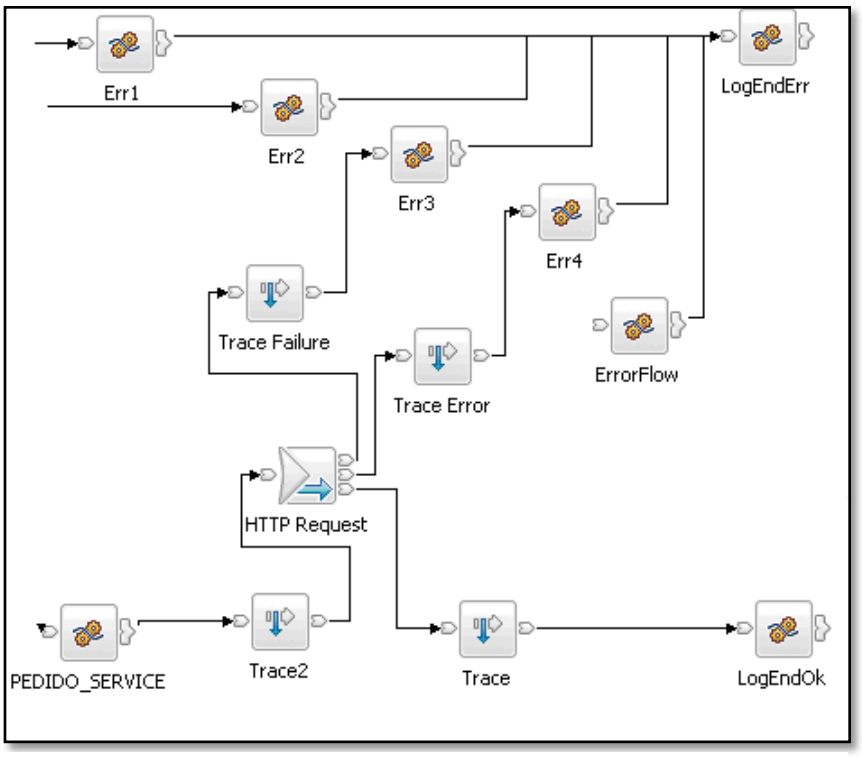
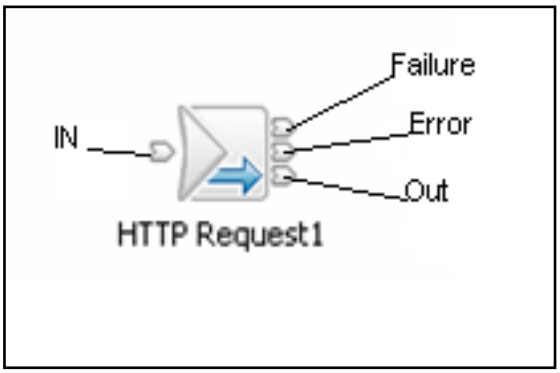


Figura 20-Mensajes en MQ

Este escenario representa una aplicación que utiliza WebSphere MQ para realizar una llamada a un servicio Web en el procesamiento de un mensaje. Un mensaje de la corriente de este escenario, y el consiguiente mensaje de los árboles se muestran en la “Figura 20-Mensajes en MQ



Cada Salida está manejada para registrar un log ya sea de éxito o fracaso (ver "Figura 21-Mensajes Message Broker"):

Figura 21-Mensajes Message Broker

- In : El terminal de entrada que acepta un mensaje para su procesamiento por el nodo.
- Failure : El terminal de salida a la que se dirige el mensaje si se detecta una falla durante el procesamiento en el nodo.
- Out : El terminal de salida a la que se dirige el mensaje si representa con éxito de la solicitud de servicio Web y si es necesaria la continuación de la tramitación dentro de este flujo de mensajes.
- Error : El terminal de salida para que los mensajes que incluyen un código de estado HTTP que no está en el rango 200 a 299, incluidos los códigos de redirección (3xx) si no se ha establecido la propiedad se debe seguir redirecciones HTTP.

Hay tres definiciones de servicios Web, dos de ellos tienen un cliente en el broker, esto significa que la llamada al servicio se implementa con el nodo HTTPRequest.

- SSD_regpedidoserviceService.wsdl
- SSD_planificapedidoserviceService.wsdl
- SSD_actualizapedidoserviceService.wsdl

El prototipo inicial se limitó a una única solicitud de servicio, el registro de pedidos anteriormente descrito. Hasta tres únicas vías de la conducción de un servicio Web de solicitud se pueden añadir sin requerir cambios significativos en la solución de la arquitectura. El número potencial se puede aumentar aún más mediante la adición de otro nodo de cómputo y "duplicar" las funciones "***PEDIDO_SERVICE"*** (nodo de cómputo).

Cada ruta a un servicio Web de solicitud se inicia en el primer nodo de cómputo y termina en un nodo de Log. La única razón para tener un nodo Log es tener un lugar para poner un punto de registro en las variables de entorno del éxito de la operación.

El mapeo entre el mensaje original y la solicitud de servicio es un simple uno-a-uno con la operación de un subconjunto de los datos de entrada que están involucrados, ya que se utiliza

con el nodo HTTPRequest. Este mapeo nos conduce a una entrada de servicios de la componente Web.

Implementación en nodo de Computo (ESQL[19]) en la construcción del mensaje SOAP [16]:

```
CALL CopyMessageHeaders();
CALL CopyEntireMessage();

-- texto del pedido
DECLARE textodelPedido CHARACTER;
SET textodelPedido = InputRoot.MRM.pedidoVentas.textoDelPedido ||
InputRoot.MRM.pedidoVentas.indGestEspecial;
SET Environment.Variables.texto = textodelPedido;
--Set envelope
SET OutputLocalEnvironment  = InputLocalEnvironment;
SET OutputRoot.Properties = InputRoot.Properties ;

SET OutputRoot.Properties.MessageSet = 'MRRC9OG002001';
SET OutputRoot.Properties.MessageType = 'Envelope' ;
SET OutputRoot.Properties.MessageFormat = 'CwXML' ;
SET OutputRoot.MRM  = NULL;

IF(LENGTH(textodelPedido)> 1)THEN
SET       OutputRoot.MRM.soap11:Body.rpv:pedidoVentasRequest.pedidoVentas.numPedidoSap       =
InputRoot.MRM.pedidoVentas.numPedidoSap;
....
```

5.6 Manejo de Mensajes con IBM WebSphere Message Broker

Se utiliza el nodo MQInput para recibir mensajes de los clientes que se conectan con el corredor del gestor de colas utilizando WebSphere MQ de Transporte.

MQInput nodo recibe el mensaje de entrada a un mensaje de un flujo de WebSphere MQ cola de mensajes definidos en el corredor del gestor de colas. Si se considera apropiado, se puede definir la entrada de la cola como una cola de WebSphere MQ agrupadas o compartida.

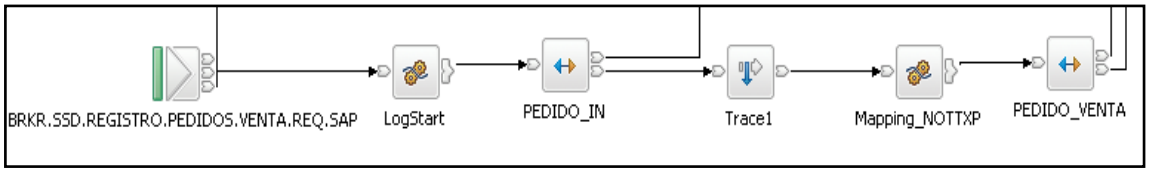


Figura 22-Nodo MQInput

[19] Extended Structured Query Language

Los mensajes que se reciben a través de conexiones WebSphere MQ siempre deben comenzar con un nodo MQInput. Puede configurar las propiedades del nodo MQInput para controlar la forma en que los mensajes se reciben. También puede solicitar que la conversión de los datos se realice en la recepción de cada mensaje de entrada.

Para esta implementación el nodo MQInput maneja mensajes en el siguiente dominio: MRM.

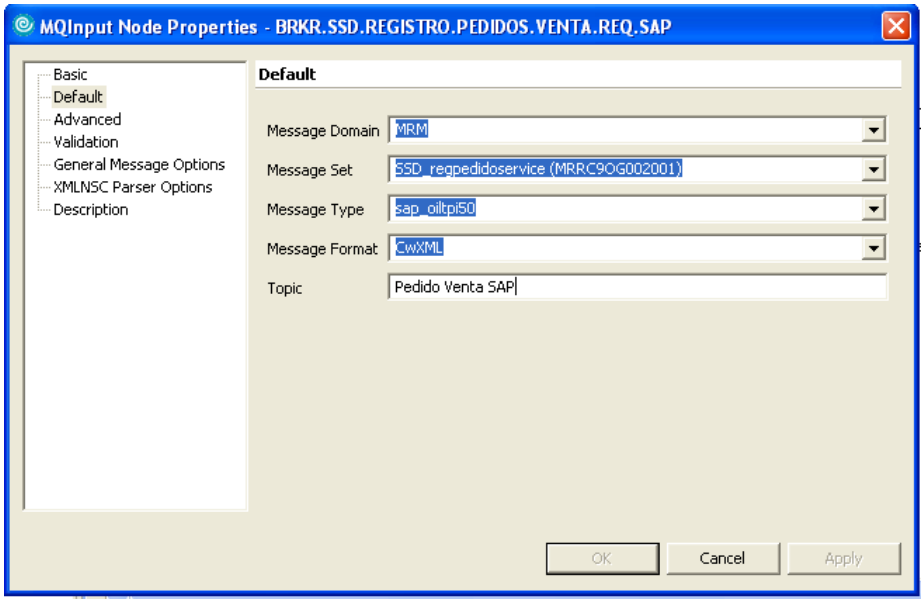


Figura 23-Formato mensaje en nodo MQInput

El formato del Mensaje nos indica según la "Figura 23-Formato mensaje en nodo MQInput" que es un IDOC OILTP50 y que éste será transformado a una estructura más simple, que será interpretada por un servicio Web. Para esto el IDOC será transformado por medio de un nodo Mapping.

El nodo Mapping tiene una o más asignaciones, que se almacenan en archivos de asignación de mensaje (. msgmap). Estos archivos se configuran usando el editor de mensajes de Mapping.

Un nodo de Mapping debe contener las siguientes entradas y salidas:

- Cero o una de las fuentes (de entrada) mensajes
- Cero o más fuentes de datos (entrada).
- Uno o más de la meta (de salida) mensajes

El origen y destino de los mensajes deben ser trazados y para esto se define en el mensaje de los archivos de definición de un Message Set. El parser de mensajes de la fuente se puede especificar en tiempo de ejecución (por ejemplo en una cabecera MQRFH2), pero el meta mensaje es construido usando el analizador de tiempo de ejecución que se especifica en el Message Set.

Si un mensaje se mapea entre los elementos de diferentes tipos, puede que tenga que incluir en su mapping las definiciones, en función del analizador de tiempo de ejecución que se especifica en el Message Set.

El nodo mapping utiliza un lenguaje para manipular los mensajes basado en XPath [20] (ver Figura 24-Editor de Mensajes de Mapping).

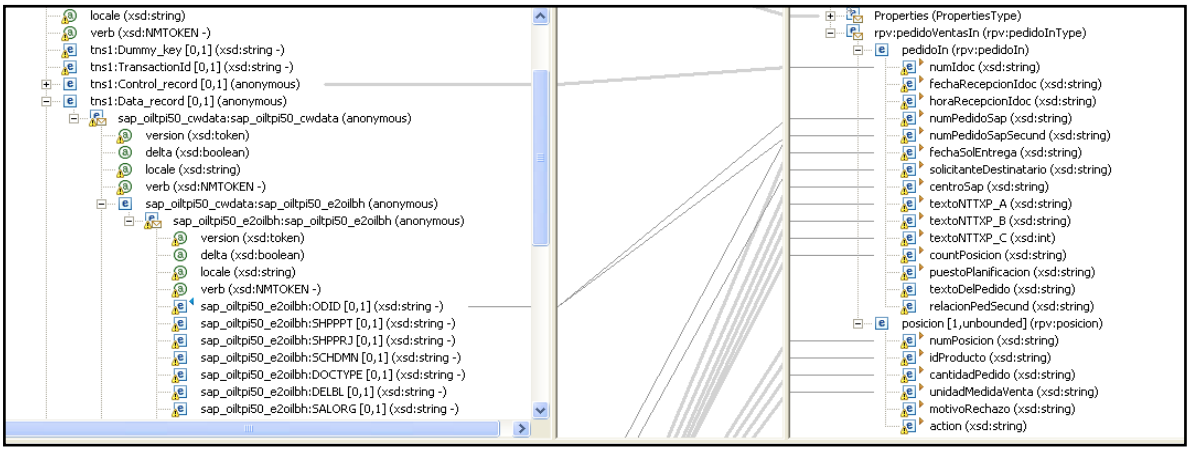


Figura 24-Editor de Mensajes de Mapping

Para desarrollar asignaciones de mensaje para un nodo de mapping, se utilizó el editor de mensajes de mapping, que prevé paneles separados para trabajar con las fuentes, los objetivos y las expresiones.

Todos los flujos que vienen de SAP tienen una lógica similar. Se recibe el Mensaje del Adapter luego se deriva al flujo externo el cual lo envía a los diferentes flujos de la aplicación según corresponda.

El Message Flor Project en SSD se denomina SSDs_MFP y su respectivo Message Set Project es SSDs_MSP.

5.7 Manejo de Archivos con IBM WebSphere Message Broker

File Extender es un conjunto de plugins para WebSphere Message Broker que permiten el manejo de archivos. File Extender consta de un conjunto de plug-in de nodos intermediarios a los actuales o nuevos flujos de mensajes Message Broker de la paleta de Herramientas. Los nodos están diseñados para encajar el modelo de administración de Message Broker a fin de que el archivo de protocolo de apoyo pueda ser fácilmente añadido a los que ya utilizan arquitecturas Message Broker. Lo que permite ampliar rápidamente los servicios en productos WebSphere de integración para apoyar el soporte a legacy basado en aplicaciones con archivos.

[20] XML Path Language

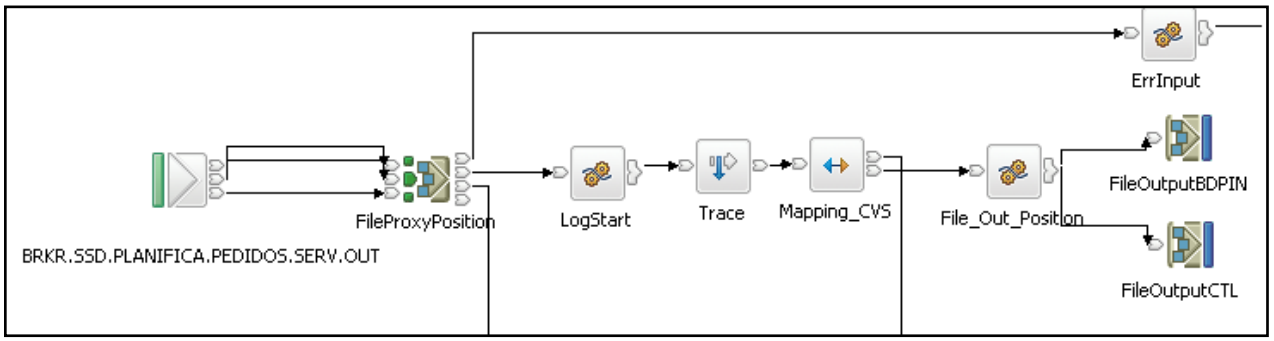


Figura 25-Flujo Broker generador de Archivos

En esta solución de arquitectura se implementó teniendo como necesidad el manipular archivos planos. Para esto se implementó una estructura de formato físico TDS [21]la que permite manipular los datos y generar un archivo plano con las características que el legacy BDP requiere para sus procesos de subida (upload) de archivos a la base de datos.

Además se tuvo en cuenta el registro de Log para el seguimiento de las operaciones de transformación de datos en archivos.

Para interactuar con este flujo en el broker se implemento un servicio Web que proporcionó a la componente Web una especie de operación en sincronía e idealmente manejar el estado de éxito o fracaso de la transacción.

En el servicio se manejan las siguientes propiedades del mensaje:

- Reply To
- Message.ID
- Correlation.ID

En Reply to se envía el nombre de la cola de respuesta a la cual el legacy debe responder y confirmar que la operación término exitosamente o no.

La propiedad Message ID da la llave del mensaje para identificar la respuesta y posterior confirmación del proceso.

La propiedad Correlation ID es un camino alternativo para identificar el mensaje original en la respuesta del proceso, ya que el legacy envía message id nuevo cada vez que confirma una operación sea de éxito o no. Por ende en esta propiedad se copia el message id original para que el proceso, cuando responda se pueda identificar la llave de la operación, puesto que esta propiedad permanece intacta cada vez que se recibe la confirmación del legacy BDP.

En resumen es así como se identifica la operación y esto no se realiza en el broker si no que forma parte de la lógica del servicio Web.

[21] Tagged/Delimited String

Capítulo 6 Implementación

6.1 Descripción general de la interfaz

Se presenta la implementación de cómo se realiza la programación de pedidos en BDP la cual consta de las etapas de:

- Identificación y registro de programaciones
- Obtención de programaciones y envío de programaciones BDP a SAP/ Web a SAP / SAP a SAP.

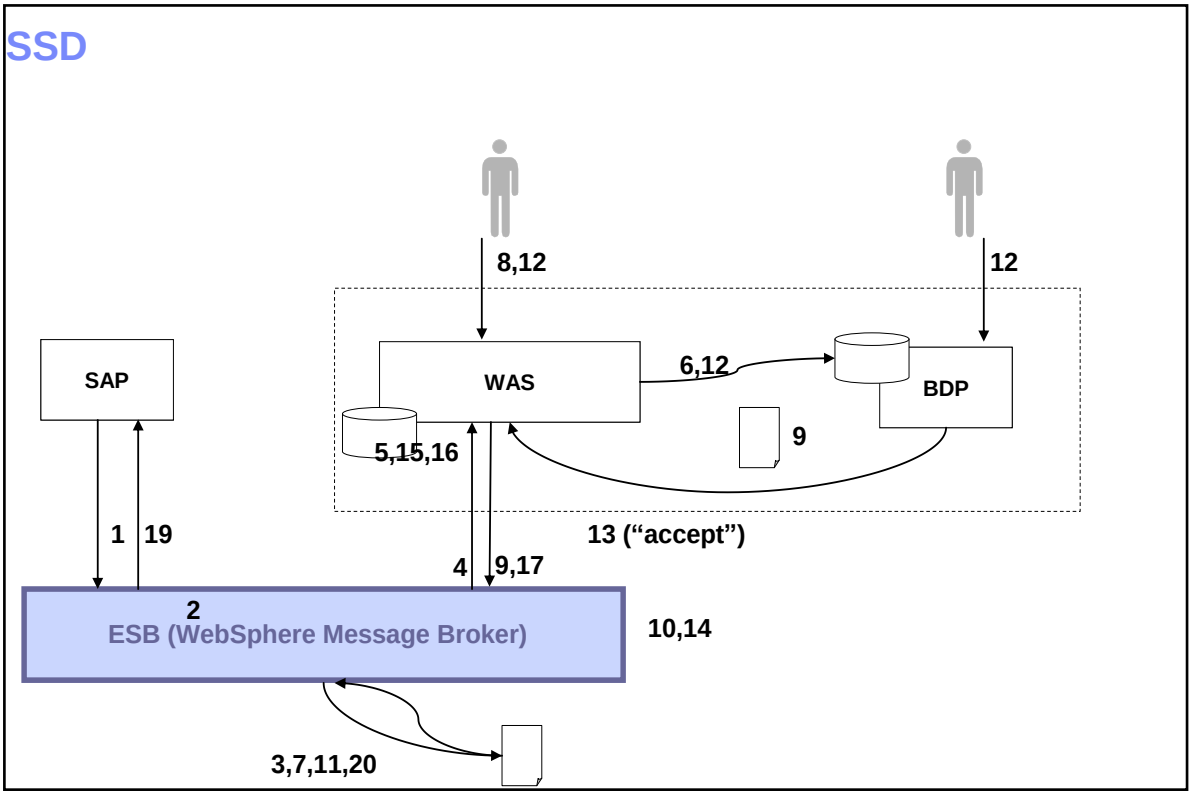


Figura 26-Programación Pedidos

1) En SAP se genera un pedido a planificar, el cuál es recibido por WebSphere Message Broker como un IDOC vía el adaptador.
2) La recepción del mensaje dispara la ejecución de un flujo en WebSphere Message Broker.
3) Se registra en el Log la recepción del requerimiento de servicio.
4) Se invoca al servicio de registro de pedidos a planificar provisto por el Seguimiento Despacho.
5) El Seguimiento Despacho inserta en la base de datos el pedido.
6) El Seguimiento Despacho accede directamente a la base de datos de BDP para obtener la información necesaria para realizar las validaciones.
7) WebSphere Message Broker registra en el Log el resultado de la ejecución del servicio de registro de pedido a planificar.

8) El usuario valida los pedidos a planificar antes de ser enviados a BDP.
9) El usuario visualiza los pedidos a planificar, selecciona un grupo y los “envía a BDP”.
10) El Seguimiento Despacho genera un archivo de envío a BDP.
11) El Seguimiento Despacho registra en el Log el envío del archivo.
12) El usuario realiza la planificación en BDP. Este número se repite en varios puntos, todos los que forman parte del servicio de planificación
13) BDP realiza el “Accept” (de las planificaciones). El Seguimiento Despacho detecta esto y :
    - Opción A: Recibe un mensaje del nuevo Accept.
    - Opción B: Accede a la base de datos de BDP para obtener la información.
14) El Seguimiento Despacho registra en el Log la recepción del archivo.
15) El Seguimiento Despacho actualiza los pedidos agrupando los pedidos programados en unidades de Transportes a enviar a SAP:
    - Evaluación de casos.
16) Identificación del proceso a ejecutar en SAP , actividades a realizar :
    - Armar una estructura IDOC.
    - Llenar una función RFC.
17) El Seguimiento Despacho invoca al servicio del ESB
18) Se registra en el Log la recepción del requerimiento de invocación al servicio
19) Se invoca (vía el adaptador) al servicio de registro de pedidos programados, provisto por SAP
20) Se registra en el Log el resultado de la invocación al servicio

Este resumen presenta a su vez, desde el punto de vista de la implementación, varias ventajas:

- Se mantiene una solución simple, sin comprometer el rehúso ni las buenas prácticas de implementación de un SOA
- Se requiere solamente la implementación de una menor cantidad de flujos de WebSphere Message Broker, lo cual implica un costo de desarrollo menor.
    - No se requiere la implementación de servicios entre el Seguimiento Despacho y BDP:
        i. servicio de acceso a información a BDP para obtener información necesaria para realizar validaciones (en su lugar, se accede directamente a la base de datos).
        ii. servicio de BDP para recibir pedidos a planificar a través de MB / NFS.
        iii. servicio de acceso a información DB de BDP para obtener la información de los pedidos programados.
- Se simplifica la interacción entre el Seguimiento Despacho y BDP al utilizar solo un flujo de servicio para generación del archivo plano con los pedidos a planificar.
- Se reutiliza el Log de registro de la ejecución de las interacciones entre sistemas, unificando su uso para los casos de invocaciones a servicios entre sistemas e invocaciones “internas”, tanto en el broker como en el was.

6.2 Registro Pedidos SSD

El registro de Pedidos realiza las siguientes actividades:

- Envío de pedidos al BDP (IDOC SAP).
- Recepción de pedidos en el sistema intermedio (SSD).
- Una vista WEB para la visualización de pedidos recibidos del R/3.
    - Validaciones para la visualización de Pedidos.

6.2.1 Escenario de Integración

SAP utiliza servicio de Legacy J2EE (Online):

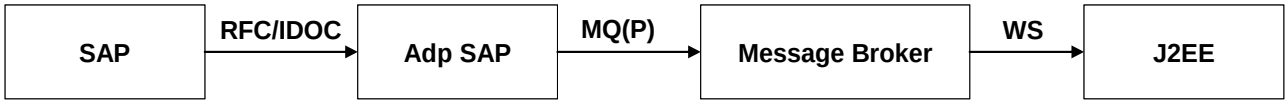


Figura 27-Escenario 1: SAP utiliza legacy J2EE

Nota : RFC/IDOC : Mensajes; MQ(P): Mensaje MQ Persistente; WS: Web Services.

En SAP (R/3) se crean los pedidos, al momento de gravarse la información esta es enviada como un IDOC. Un flujo que expone el servicio a nivel del ESB para que sea utilizado por el adapter que lo requiere ("Flujo Externo") recibe el mensaje y lo coloca en el flujo que realizará la transformación del mensaje a una estructura de datos que pueda ser enviada por medio de una llamada Web Services J2EE a una BD. Este flujo considera las particularidades de la tecnología de transporte de mensajes y utiliza MQ, para escribir, leer y consumir los mensajes de las colas de mensajes.

6.2.2 Lógica de Implementación

Para el registro de la información de los pedidos hay que tener en cuenta ciertas consideraciones, sin embargo se explica lo medular de la lógica de registro de pedidos. Ésta básicamente consiste en manejar un contador alfabético de las versiones que se registren de los pedidos en SSD. EL criterio para una versión dada define que un mismo pedido de SAP puede sufrir variaciones en el tiempo, estas variaciones están identificadas y clasificadas lo que indica que cuando una de ellas es detectada, se registre una nueva versión del pedido.

Hay dos tipos de pedidos: de ventas y reposición (IDOC OILTPI50 y ORDER05 respectivamente), para la persistencia de pedidos se definieron dos tablas ***pedido*** y ***posición***. En

la tabla pedido están los datos de cabecera y en la tabla posición están los datos de productos ingresados en cada pedido.

Cada tipo de mensaje ya sea venta o reposición es tratado como una mediación independiente, aunque ambos invocan el mismo Web services cada uno es tratado en el Message Broker como un flujo individual.

Además de registrar los datos de pedidos, se registran datos de status del pedido que corresponde a identificar si la información del pedido es consistente y si no lo es, el motivo del problema o error que reporte el proceso de validación. Esta es realizada en un proceso a parte del registro del pedido, como una acción antes del despliegue de los datos en la Web, cada vez, que se ingresa al reporte 1 y 2, consulta pedidos a validar, consulta pedidos a enviar respectivamente.

En el tipo de reporte uno solo serán desplegados los pedidos que presenten un estatus de error o problema, cada vez que se cargue la consulta de validación (reporte n° 1), se gatilla la validación de pedidos, finalizado este proceso solo se despliegan en la vista aquellos registros de datos que conservan el tipo reporte uno. El grupo de datos que pasa a reporte dos solo se puede desplegar en la consulta tipo reporte dos, que corresponde a la planificación de pedidos.

El tipo de reporte dos consiste en los pedidos que fueron validados exitosamente. Cada vez que se gatilla esta consulta, el objeto de negocio de la acción vuelve a ejecutar la validación para aquellos registros que aún no han sido validados, como son los que cumplen con el siguiente criterio:

- Tipo de reporte uno.
- Valida estatus del pedido en cero (no validado).

Hay que destacar que solo considera los registros de datos que además cumplen con el criterio de consulta del reporte, la que tiene trece campos para realizar un filtro de los pedidos que se quiere consultar. Esta consulta antecede a cada uno de los reportes de la Web. Además es un criterio totalmente dinámico, que solo se incluye cuando el valor es distinto de “null” o blanco (ver Figura 28-Reporte Consulta Pedido).

Figura 28-Reporte Consulta Pedido

En el caso de que la consulta no tenga ningún valor asociado como criterio de filtro se considerará todo el universo de datos que estén registrados en la base de datos. Este tipo de consulta consume mucho recurso, tanto en memoria de la componente Web, como en la base de datos donde se extrae la información. No solo se trata de extraer datos y desplegarlos en la vista del reporte, sino que hay un proceso de validaciones que se ejecuta junto a la consulta de datos que cumplen con el criterio de selección y esto significa un número determinado de consultas a las bases de datos iSeries y Oracle respectivamente, cuestión que no es menor ya que esta se aplica a nivel de cada pedido o dicho de otro modo de cada registro. Por ende se recomienda siempre aplicar un criterio a la consulta como por ejemplo el rango de fechas, para que esta sea performante.

Como se mencionó anteriormente, en el proceso de validación intervienen dos motores de base de datos: DB2 iSeries y Oracle 8i.

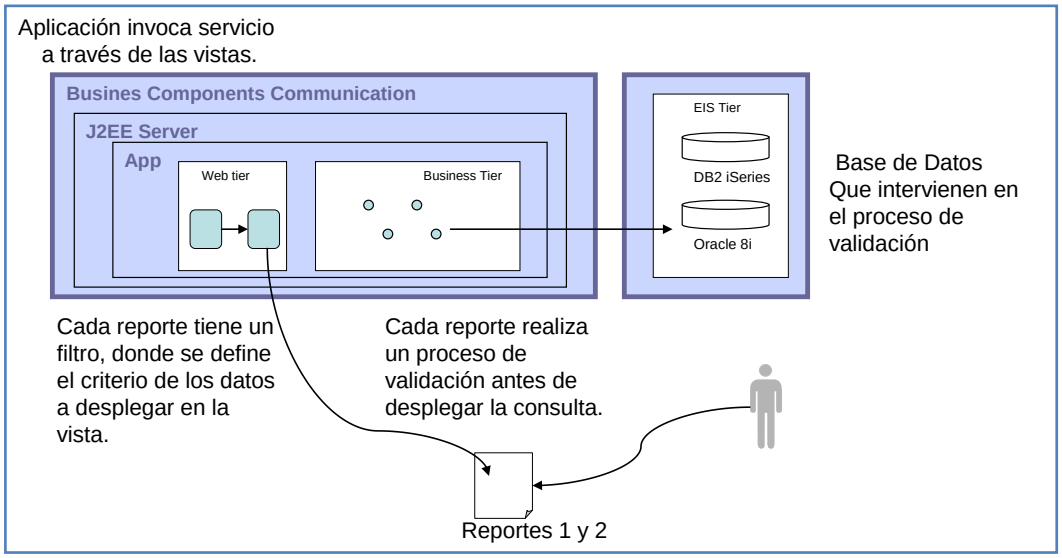


Figura 29-Componentes Aplicaciones J2EE

- Reporte 1: /validacionPedidos.jsp
- Reporte 2: /planificacionPedidos.jsp

Las aplicaciones J2EE están compuestas de diferentes componentes. Un componente J2EE es una unidad de software funcional auto-contenido que se ensambla dentro de una aplicación J2EE, son clases, ficheros y que se comunica con otros componentes de la aplicación.

Estas validaciones son previas al envío de pedidos al BDP:

- Material – Centro
  - Se realiza el JOIN de tablas del BDP:
    - LOCATION_PRODUCT_TANK
    - PRODUCT_TANK_GROUP
    - PRDCT_TNK_GRP_TERMINAL
- ID BDP del Cliente – RUT del Cliente – Punto de Consumo del Cliente
  - Tabla: LOCATION
- Centro – Cliente – Centro
  - Tabla: NETWORK
- Número Estanque – Producto
  - Tabla: LOCATION_PRODUCT_TANK

Adicionalmente, todos los reportes tienen como restricción el no mostrar todos aquellos registros (personalizados en SAP) que tengan relación con la siguiente lista.

- No mostrar los registros en donde los valores del campo Clase Doc Venta (P0025) sea igual a:
  - Reposición de TCT, TAE, CUPON (ZTAR)
  - Cumplimiento de storage (ZCST)
  - Cumplimiento de almacenamiento (ZCUA)
  - Reposición de almacenaje (ZRAL)

- Pedidos consolidados secundarios (ZVGC).

Para tal efecto se mantiene una tabla CLASE_DOC_VENTA local en DB2 iSeries de configuración y mantenimiento (en el SSD), la que identifica toda esta lista con el ID del documento de venta.

En cuanto al manejo de versiones del pedido solo se considera para efectos de la planificación la última versión del pedido. Para esto se implementó una forma de dar de baja el registro con la versión anterior a través de una marca. Esta marca se inserta en el tipo de reporte asignándole una "N".

Resumiendo técnicamente las validaciones en accesos a la BD, queda como sigue:

Sentencia que consume connection	***N° reg.***	***Motor Oracle***	***Motor DB2***
st = bean.validaIdEnSap(pedido,st);	1	1	
st = bean.validaProducto(pedido,st);	1	2	2
st = bean.validaPlantaDespachadora(pedido,st);	1	1	
st = bean.validaNetwork(pedido,st);	1	1	
dao.updateValidaPedidos(updatePedidos);	N		1
Sub Total (con N = 1)		5	3
Total			8

Nota : N corresponde al número total de registros de la consulta.

Todas y cada una de ellas, consultan diversos datos de las tablas nombradas anteriormente, para chequear que la información de los pedidos es válida previo a su envío al BDP, de otro modo si los datos son inconsistentes o erróneos, daría como resultado una caída en la planificación de pedidos BDP.

6.2.2.1 Implementación de Versión de Pedidos

El método encargado de la implementación de la lógica de versión dado es ***"nextContador",*** esta lógica define cual será el contador a designar en el pedido, vale decir cuál será la versión del registro del pedido. En el caso que el pedido no exista se creara la primera versión, en el siguiente caso se validará si se trata de una actualización del registro de pedidos o si se trata de una nueva versión la que será un incremental de la anterior versión. Como las versiones se identifican alfabéticamente de A - Z, esta inicia de la siguiente forma:

Sufijo: S + [A-Z] + (número de pedido SAP)

La casuística (caso por caso) define distintas instancias para considerar un nuevo registro del pedido o una nueva versión.

Para realizar la verificación de los datos del pedido se recogen los datos del pedido si existe en la base de datos y los datos provenientes de SAP. Estos datos se comparan y dependiendo del resultado de esta evaluación es que se considera si es una nueva versión.

Las que se detallarán como siguen:

En el caso de que el pedido no esté programado “statusprog” igual a cero:

- Si se trata de un cambio MIX, si el motivo del pedido es igual a ZC1, ZC4.
- Si se trata de un cambio de centro (centro BDP)
- Si se trata de un cambio en la ventana horaria.

En el caso de que el pedido si este programado el “statusprog” es igual a uno:

- Si se trata de un cambio de MIX, si el motivo es igual a ZC4.
- Si se trata de un cambio de MIX, si el pedido proviene de la lógica del “Accepts”.

Para todos los casos en que se genere una nueva versión del pedido, el anterior registro de datos que está en la base de datos se da de baja. Este procedimiento corresponde a una marca de tipo de reporte igual a “N”.

6.2.3 Modelo de Datos

6.2.3.1 Modelo Datos Tablas Registro Pedidos SSD

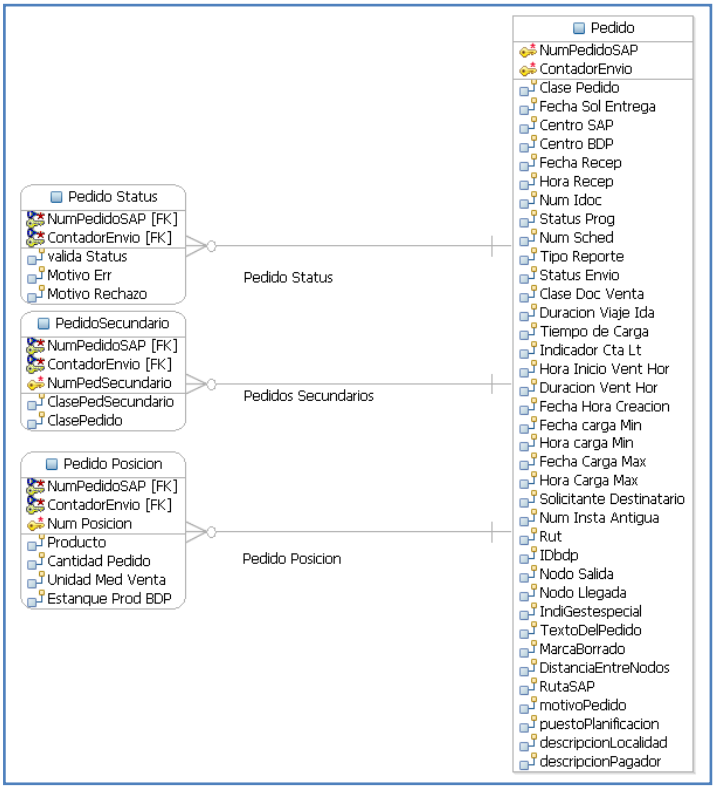


Figura 30-Modelo de datos Registro Pedidos

6.3 Planificación Pedidos SSD

La planificación de Pedidos consiste en el envío de Pedidos al BDP y en el resto del documento se entenderá como el tipo reporte 2 “Envío de Pedidos”.

6.3.1 Escenario Integración

Figura 31-Escenario 2: envío Pedidos a BDP

El envío de Pedidos esta implementado una parte J2EE como un servicio Web, que es invocado por una vista WEB que despliega los datos validos de la tabla pedidos (DB2 iSeries). Esta vista pertenece a la componente Web “***SSDConsultasWeb***. Cada envío es originado por una instancia de servicio Web denominado:

SSD_planificapedidoserviceSOAP_HTTP_Service

el cual inserta un mensaje MQ que pasa a la capa de integración WebSphere Message Broker donde los datos compuesto por una cadena con la lista de pedidos a planificar son transformados en un archivo plano.

El servicio hace un put a la cola: BRKR.SSD.PLANIFICA.PEDIDOS.SERV.OUT. Este mensaje es una cadena de texto con los datos para construir el archivo plano. Este mensaje en la cola da inicio a la ejecución del flujo en el broker, este servicio está encargado de la construcción del archivo plano con el formato que requiere la aplicación del BDP, para realizar la carga de los datos a la base de datos.

En este escenario la integración es con el paquete BDP por medio de archivos planos.

6.3.2 Lógica de Implementación

La componente Web contiene cuatro reportes el primero pertenece a la validación (ya revisado), el segundo es el encargado de traspasar los pedidos a la capa de integración, el tercero es el encargado de mostrar los pedidos que ya han sido planificados y el cuarto es el que muestra un Log de error en la operación de la programación proveniente de BDP.

Las páginas que conforman las distintas vistas son:

- Reporte 2: /planificacionPedidos.jsp
- Reporte 3: /actualizacionPedidos.jsp
- Reporte 4: /logErrorPedidos.jsp

Los action del Struts que se relacionan a cada una de estas vistas son las contenidas en el package ***segdespconsultasweb.actions***:

- Reporte 2: / ActionPlanificaPedidos
- Reporte 3: / ActionConsultaProgramados

- Reporte 4:./ ActionConsultaPedidoLog

En el caso del reporte dos corresponde a la planificación, esto es el envió de los pedidos a BDP. En el caso del reporte tres, se muestra los pedidos programados y existe la opción de actualizar el estatus de envió a BDP de enviado o no enviado. Con el objeto de poder re enviar este pedido nuevamente a BDP, las veces que se necesite. El reporte cuatro será explicado en el tema de actualización de pedidos, más adelante en este capítulo.

Para proceder al envió de pedidos se definió que los datos que inicialmente viajan por la petición del servicio Web como un texto, serán transformados en un archivo plano, en la capa de mediación del broker. Este archivo plano tiene el formato que requiere la aplicación de BDP en su estándar para el ingreso masivo de datos a la base de datos del BDP.

Para tal efecto se definió un formato físico del cual se desprenden una serie de definiciones de mensajes para llevar a cabo esta transformación.

El Formato Físico está definido en el dominio MRM.

Cada archivo de definición de mensaje describe tanto estructura lógica de sus mensajes como los formatos físicos que describen el aspecto preciso de su flujo de BIT de mensaje durante la transmisión. El dominio MRM (la información de formato físico debe ser proporcionada) le dice al analizador sintáctico exactamente como analizar el flujo de BIT del mensaje.

El formato que estos archivos planos usan es el Tagged Delimited String Format (TDS) formato físico para modelar mensajes de texto formateados (ver Figura 32-TDS) con el contenido de los datos identificado por etiquetas y separado por delimitadores específicos. Este apoyo es bastante usado para modelar estándares en la industria.

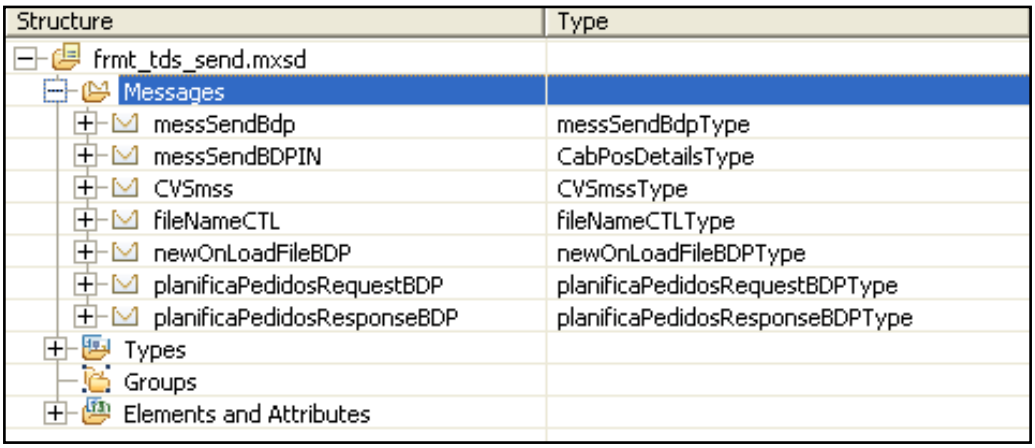


Figura 32-TDS

6.3.2.1 Envío de Datos con Archivos Planos

Para realizar una subida de archivo(s) a paquete BDP se debe realizar lo siguiente:

MB File Extender solo escribe y lee localmente en el sistema de archivos del servidor donde está instalado, por ende para poder enviar este archivo plano al paquete, en este caso BDP, se implementó un sistema de carpetas compartidas a través del protocolo NFS[22], para dejar los archivos CTL y BDPIN (donde CTL_xx es el nombre del archivo de control y BDPIN_xx es el archivo de datos) en la carpeta compartida por el servidor NFS (ver Figura 33-Arquitectura File Extender). El archivo de control tiene en su interior el nombre del archivo de datos a ser cargados por el proceso de upload en BDP (donde xx corresponde a un formato de fecha con máscara de año mes día hora minuto segundo).

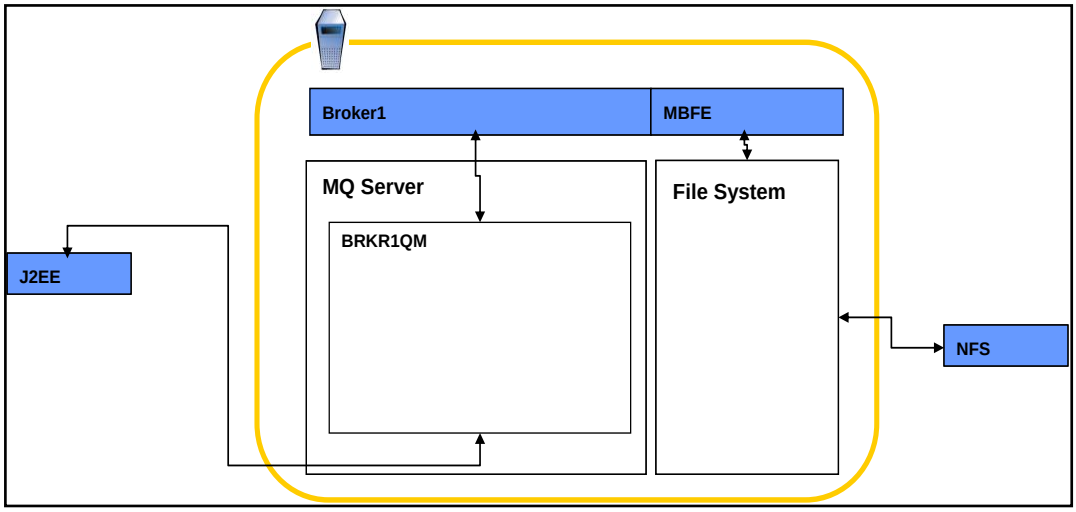


Figura 33-Arquitectura File Extender

Archivo de Datos

- Formato Nombre Archivo : BDPIN208.yyyyMMddHHmmssSSSHHmmss
- Ejemplo Nombre : BDPIN208.N.20071107164223354.204223

Archivo de Control

- Formato Nombre Archivo : CTL208.yyyyMMddHHmmssSSSHHmmss
- Ejemplo Nombre : CTL208.N.20071107164223354.204223

Ruta file system local en el broker para la carpeta compartida `..\outbox\BDP\SSD\SST`

Esta carpeta disponibiliza el recurso para luego realizar un upload del archivo el cual es tomado por un proceso propio del paquete BDP el que finalmente inserta los datos de la planificación del o los pedidos en una base de datos oracle (BDP).

El recurso se disponibiliza enviando un mensaje MQ a BDP informando que hay un nuevo archivo plano que se debe extraer.

[22] Network File System

El paso de mensajes es síncrono, el servicio Web que envía el mensaje espera a que BDP lo reciba y responda para continuar su ejecución.

- Requestor receptor = recepto .Send(txtMsgSend);
- Consumer consumer = consumr.consumer(id_in);

Los elementos principales que intervienen en el paso de mensajes son el proceso que envía, (SSD) el que recibe y responde el mensaje.

En el paso de mensajes síncrono el BDP enviará un mensaje de confirmación, el que se detallara más adelante.

Por ende en el Queue Manager (broker BRKR) se definió dos colas de requerimiento una de entrada y una de salida.

- Entrada : BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.IN
- Salida : BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT

6.3.2.2 Formato del mensaje Salida a ser enviado (SSD-BDP)

Los valores son el nombre de la Queue remota a la cual va destinado el mensaje de aviso en BDP y el nombre del archivo de control que disponibiliza los datos y define donde serán cargados en la base de datos de BDP. Para esto se define una cola local en el flujo la que está configurada con el nombre de la cola remota.

- Nombre cola salida remota : BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.IN
- Nombre cola entrada remota : LS_UP_FILES_BDP.REQ

Formato del Mensaje String:

```
'<TRX><SP>LS_PROCESA_FILE_BDP</SP><DATA><FILE>'|| nombre_archivo_CTL_NAME ||
'</FILE></DATA></TRX>';
```

Con esto se subirá a BDP todos los archivos mencionados en los tag “FILE”. El nombre del archivo es “case sensitive” y debe corresponder al nombre del archivo de control (el que comienza con CTL208).

En el encabezado del mensaje se indica el Qmanager y cola de respuesta.

El put del mensaje da fin a la instancia del flujo y a la operación misma de la escritura del archivo plano en el sistema de archivos definido para la carpeta compartida.

6.3.2.3 Formato del mensaje Entrada a ser recibido (BDP-SSD)

- Nombre cola entrada local : BRKR.SSD.PLANIFICA.PEDIDOS.MSG.BDP.OUT

Formato del Mensaje String:

```
<TRX><ERROR><CC>0000</CC><CODE>XPIN70</CODE><73es>LS_PROCESA_DISP_BDP</73es
></ERROR><DATA>PROCESO CORRECTO</DATA></TRX>
```

Cuando en el tag CC viene el valor "0000" esto indica el éxito de la operación, cualquier otro valor indica error en la operación.

El estado de la operación exitoso le indica al proceso la inmediata actualización del campo STATUS_ENVIO en la tabla PEDIDOS en el SSD de cero a uno: "enviado".

Finalmente terminada la actualización del estatus el reporte contabiliza los registros enviados y los reporta en la Web como confirmación del éxito de la operación. En el caso de que la operación no sea exitosa el despliegue del resultado de la operación dará como reporte cero registros enviados a BDP.

Finalmente este mensaje da término a la operación del servicio en la componente J2EE.

6.3.3 Modelo de Datos

6.3.3.1 Modelo de Datos Proceso Validación en BDP

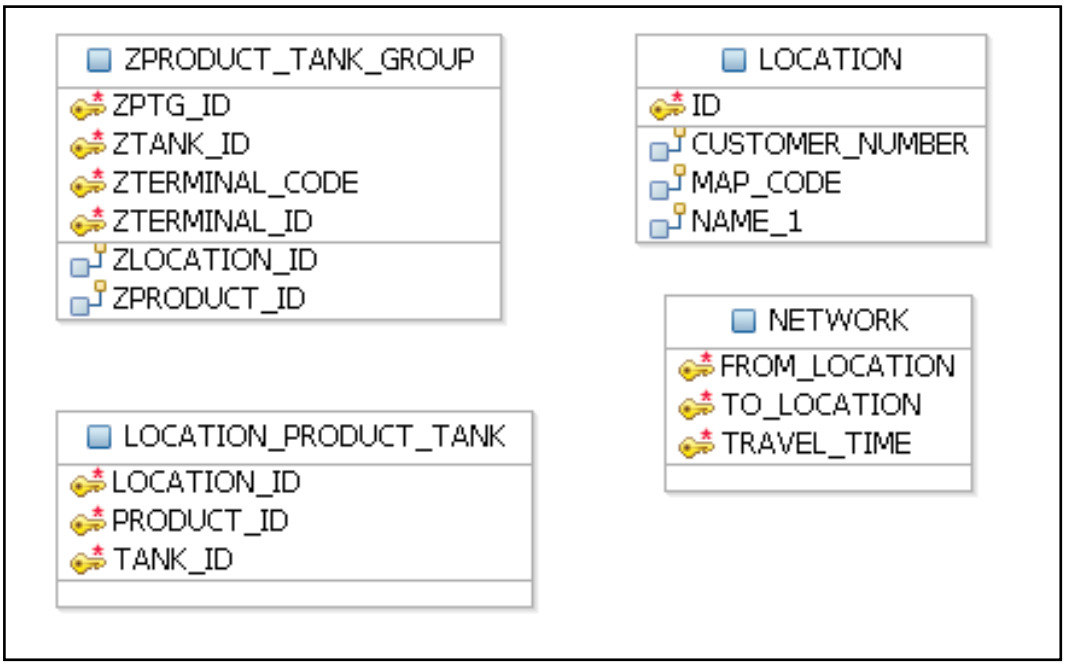


Figura 34-Modelo de Datos Validación BDP

6.3.3.2 Modelo de Datos Tablas Mantenimiento Códigos SAP

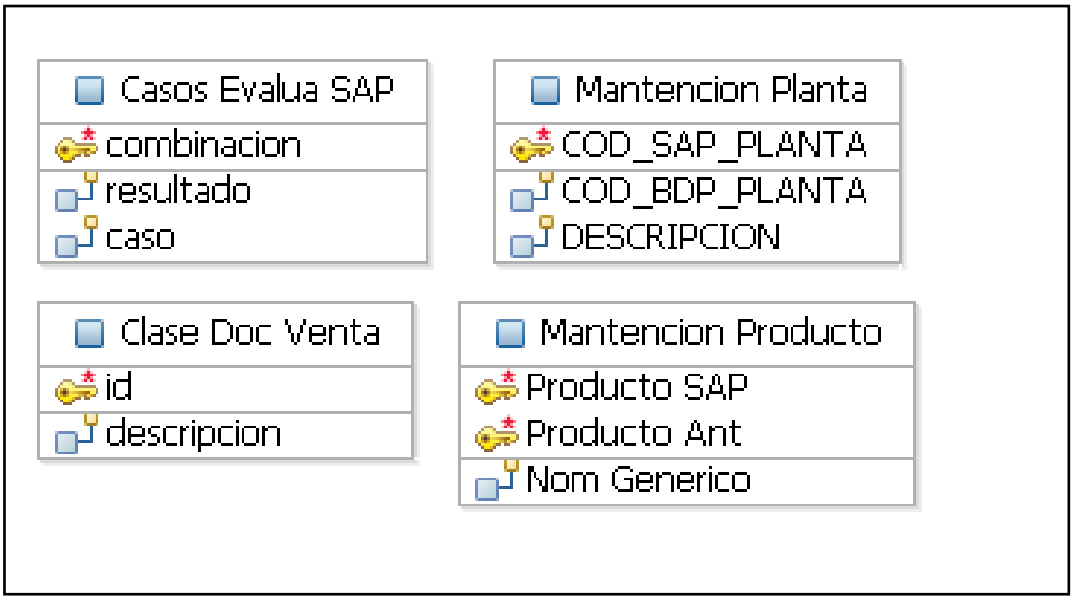


Figura 35-Modelo de Datos Mantenimiento Códigos

Nota: Estas tablas son las utilizadas para la conversión de códigos en SSD. Son especialmente utilizadas en la componente de Actualización de Pedidos.

6.4 Actualización Pedidos SSD

Es el envío de pedidos programados desde el BDP al SAP R/3.

Este proceso se divide en dos partes primordiales las que son el envío de pedidos programados al SAP y la confirmación de la programación en SSD, una vez que ésta ya ha sido contabilizada en SAP.

Cada vez que se realiza una planificación de pedidos en BDP, esta información regresa a SSD con la lógica de despacho de esos pedidos. Esta actividad se resume en una serie de registros de programación y algunos datos adicionales como la ruta, camión fechas y horas de despacho, etc.

6.4.1 Escenario de Integración

Legacy J2EE utiliza servicio de SAP (Online):

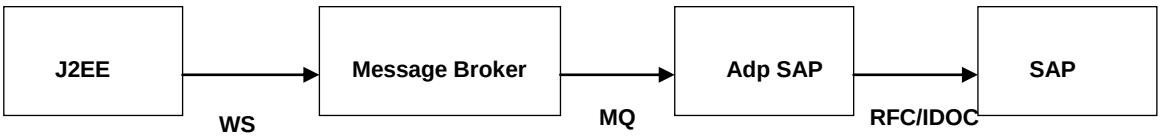


Figura 36-Escenario 3: legacy J2EE utiliza SAP

Se cuenta con la posibilidad de distintos mecanismos de transporte de mensajes, principalmente:

- Colas de mensajes (WebSphere MQ), mecanismo propio de la plataforma IBM, orientado a invocaciones asincrónicas.

Este escenario esta visto de la perspectiva de la aplicación J2EE hacia SAP. Hay que recordar que la información proviene del paquete BDP.

6.4.2 Lógica de Implementación

6.4.2.1 Definición del Proceso

Este proceso tiene por objetivo enviar los pedidos pertenecientes a la programación de combustibles en BDP al sistema intermedio “Seguimiento de Despacho” desarrollado en SAP.

Para ello es necesario detectar los distintos “Accept” realizados en BDP y a partir de esta acción actualizar los pedidos en SAP.

Accept es conocido como la aceptación de la programación en BDP y el posterior envío del mensaje a SSD informando que una nueva programación se ha aceptado (ver Figura 37-Envío de Pedidos desde BDP a SAP).

Este mensaje es el inicio del proceso en el objeto de negocio MDB.

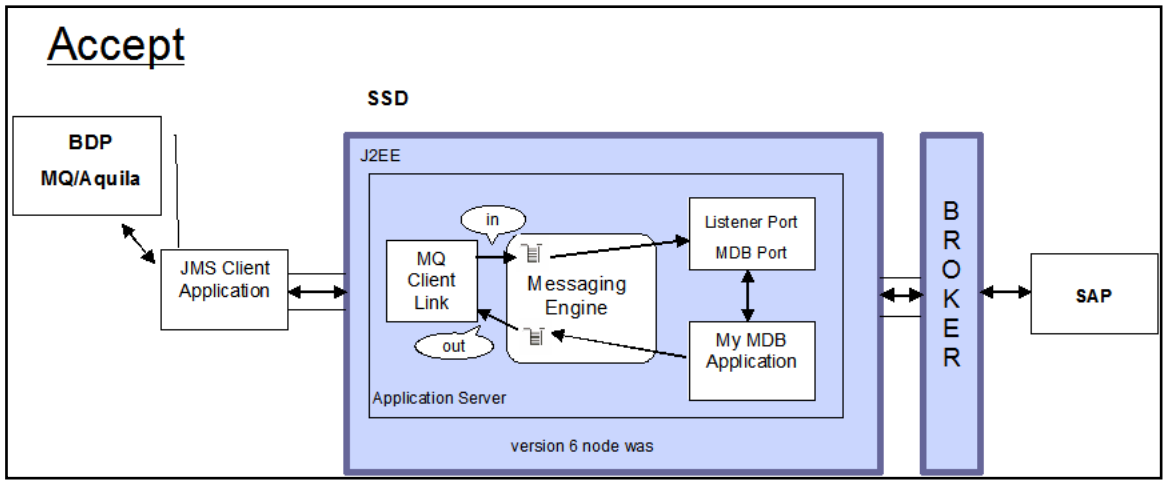


Figura 37-Envío de Pedidos desde BDP a SAP

Integración sin requerimientos de mediación entre BDP – SSD: Este caso en que una aplicación requiere utilizar un servicio sin existir la necesidad de incluir ningún tipo de mediación (cambio del mensaje, formato de mensaje, etc.).

Se decidió utilizar conexiones directas entre el proveedor del servicio y la aplicación que lo consume. Sin embargo la mediación si es requerida solo existe entre SSD – SAP.

6.4.2.2 Definición del tipo de envío del mensaje

El paso de mensajes será síncrono, el proceso que envía el mensaje espera a que SSD lo reciba para continuar su ejecución. Los elementos principales que intervienen en el paso de mensajes son el proceso que envía, el que recibe (SSD) y el mensaje. En el paso de mensajes síncrono el SSD enviará un mensaje de confirmación, el que se detallará más adelante.

Por ende en el Queue Manager, se definieron dos colas de requerimiento una de entrada y una de salida:

- Entrada : SSD.ACTUALIZA.PEDIDOS.MSG.BDP.IN
- Salida : BRKR.SSD.ACTUALIZA.PEDIDOS.MSG.BDP.OUT

6.4.2.3 Paso de Mensajes proceso que envía. (BDP)

Descripción de campos del mensaje

- processNumber CHAR(14) Formato YYYYMMDDHHMSS
- schedulerNumber CHAR(12) Formato XXXXXXXXXXXX

XML Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<TRX><SP>SSD</SP><DATA><processNumber>20071221173550</processNumber><schedulerNumber>22122101</schedulerNumber></DATA></TRX>
```

6.4.2.4 Paso de Mensajes proceso que recibe (SSD)

Descripción del campo del mensaje

- status CHAR(4) Formato DDDD

Valores para el campo Status:

- “0000" OK .
- “0001” No existe Registro.
- “0002” No es un mensaje de Texto.

XML Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<TRX><ERROR><CC>status</CC><CODE>status</CODE><DESC>des</DESC></ERROR><DATA><processNumber>processNumber</processNumber><schedulerNumber>schedulerNumber</schedulerNumber></DATA></TRX>
```

6.4.2.5 Lógica de Negocios

Cuando una programación es aceptada en BDP, se produce un registro equivalente al enviado en la planificación en BDP. Además un mensaje es enviado con los datos de fecha de proceso y escenario de la programación a SSD.

Con este mensaje se gatilla todo el proceso de Actualización Pedidos en SSD. Esto es extraer los datos de BDP, evaluar y agruparlos pedidos por ruta. Esta agrupación se lleva a cabo utilizando para ello estructuras para organizar una colección de objetos en memoria de modo que también se manejarán distintos algoritmos para agilizar el acceso a los datos.

Después de organizar esta colección de objetos, hay una serie de validaciones que se realizan para determinar si es una programación viable, para eso se definió un modelo de datos que es

capaz de llevar un registro histórico de las programaciones y de los pedidos de SAP de manera consistente en la base de datos. Pero la persistencia en la base de datos se realiza al final del proceso de validación.

A grandes rasgos y a modo de resumen una parte importante de la lógica de negocio, específicamente de las validaciones, es encontrar el tipo de formato de mensajes a enviar a SAP y no es solo una cuestión de formatear datos sino que también es una operación a realizar en SAP, la que se debe detectar como resultado de esta evaluación.

El proceso de programación de pedidos en BDP tiene por objetivo principal enviar esa programación a SAP. Por ende se divide en tres grandes grupos lógicos los tipos de acciones a ejecutar en SAP, desde un punto de vista funcional se agrupan en: pedidos de ventas, traslados y "MIX" (es una combinación pedidos de ventas y traslados).

Para el caso de mantener las programaciones en la base de datos se creó una llave en la tabla que registra las programaciones, esta llave consiste en un número de proceso, escenario y versión del pedido. Solo se considera en la evaluación de datos o más bien proceso de validación aquellos registros que mantienen la última versión del pedido y de programación.

El siguiente recuadro resume los casos que podrían presentarse en la identificación de la programación o reprogramación enviados desde el. Esta lógica de programación es esencial para identificar el formato físico del mensaje que se debe enviar a SAP. Los formatos utilizados son IDOC y las llamadas a funciones RFC.

Lógica de programación.
Interfaz BDP-SAP

		Ventas		Traslados		Mix (Pedidos + Traslados)	
N°	Casos	Programación	Reprogramación	Programación	Reprogramación	Programación	Reprogramación
		Alt 1	Alt 1	Alt 1	Alt 1	Alt 1	Alt 1
1	- Fe. Ent 1 <> Fe. Ent 2 ó - Cab 1 <> Cab ó - Eventos 1 <> Eventos 2	1) F10	1) F12	1) F6 2) F7	1) F4 + F7	1) F6 2) F7	1) F4 + F7
2	- Ce 1 <> Ce 2 ó - Vol 1 <> Vol 2 y - Pedidos = ZCES	1) Log SSD 2) Modif. ZCES 3) F10	1) F11 2) Log SSD 3) Modif. Manual ZCES 4) Reenvío a BDP 5) F10			1) Log SSD 2) Modif. ZCES o Trasl. 3) F6 + F7	1) Log SSD 2) F4 + F5 3) Mod. Manual 4) F6 + F7
3	- Ce 1 <> Ce 2 y - Pedidos <> ZCES	1) F10	1) F11 2) F10	1) Log SSD 2) Mod. Trasl (Anular y crear) 3) F6 + F7	1) F4 + F5 2) Log SSD 3) Mod.Trasl. (anular y crear) 4) F6 + F7	1) Log SSD 2) Mod. Trasl o Ped. 3) F6 + F7	1) Log SSD 2) F4 + F5 3) Mod. Manual (anular y crear) 4) F6 + F7
4	Asig. Ped 1 <> Asig. Ped 2		1) F11 2) F10		1) F4 + F5 2) F6 + F7		1) F4 + F5 2) F6 + F7
5	Sin modificación	1) F10		1) F6 2) F7		1) F6 2) F7	
6	- Vol 1 <> Vol 2 y - Pedidos <> ZCES	1) F10	1) F11 2) F10	1) F6 con Vol 2 2) F7	1) F4 + F5 2) F6 + F7	1) F6 2) F7	1) F4 + F5 2) F6 + F7
	(X, Y)	1	2	3	4	5	6

F4 Borra Transporte, F5 Borra Entrega, F6 Crea Entrega, F7 Crea Transporte.

F10 Crea Transporte, F11 Elimina Transporte, F12 Modifica Transporte.

Esta matriz es el resumen de combinaciones posibles que en cada caso pueda presentar un conjunto de datos de programación. Por cada registro distinto según la cabecera general, de la

tabla de parámetros de ingreso, se deberá lanzar un evento que ejecute un job [23] de fondo (en SAP) en el cual se pueda correr el o los procesos identificados según sea el caso asignado.

Los recuadros achurados con gris son aquellas combinaciones que nunca se darán, pues la lógica de negocio no los contempla.

En conclusión se obtiene la información de programación, se identifica el caso presentado y ejecuta la operación en SAP. Sólo aplica para los pedidos de ventas y los MIX (pedidos de ventas y traslados).

El modelo de datos que se utiliza contempla tablas para el registro de las programaciones, una tabla de configuración, una de transportes, una tabla para el control de transportes enviados, y una tabla de control de errores.

Las programaciones se registran en las tablas:

- Pedidos_cab : datos de la cabecera del transporte y del pedido.
- Pedidos_pos : datos de la programación producto y cantidad, etc.
- Pedidos_conf : datos del resultado de la validación de las programaciones.
- Pedidos_log : datos de registro de errores en el proceso de validación.
- Transporte : datos que contabilizan los transportes programados de un grupo de pedidos.
- Temp_envio_sap: datos del registro de envío efectivo a SAP de la programación.

Para La Actualización de Pedidos (ver Figura 38-Actualización de Pedidos) se utiliza IBM WebSphere MQ en el envío de mensajería e IBM WebSphere Application Server la que despliega la aplicación SSD en su plataforma Java 2 Enterprise Edition (J2EE ™).

Para esta implementación se utilizó un objeto MDB EJB 2.1, que contenga la lógica de negocio necesaria para manejar la mensajería entre el legacy BDP y la plataforma MQ y WAS.

[23] Proceso corriendo en modo batch o background

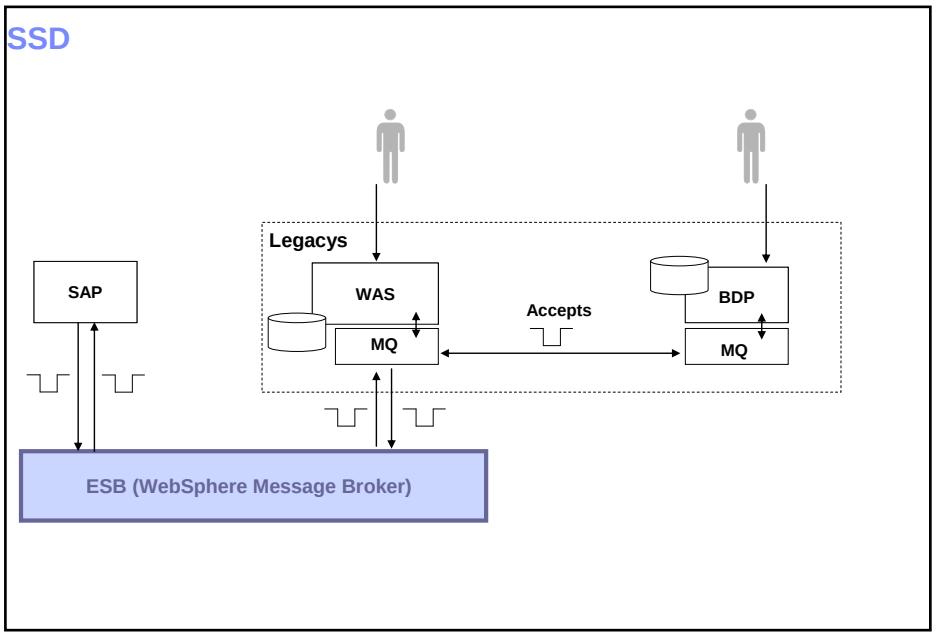


Figura 38-Actualización de Pedidos

Las interfaces de comunicación entre el SSD y BDP no son “servicios” (desde un punto de vista de arquitectura SOA), sino que es más correcto verlo como interfaces de comunicación entre dos subsistemas de un mismo sistema y por lo tanto no existiría una razón desde el punto de vista de arquitectura para que esta interacción se realice a través del ESB.

No se identifica un rehúso posible de las interfaces de comunicación entre el SSD y BDP. Existe una posible sustitución en un futuro de BDP y el SSD por un único sistema de planificación Logístico.

6.4.2.6 Definición de Algoritmos y Estructura de Datos

Un tema que aparece frecuentemente en la programación es el de organizar una colección de objetos en memoria. Este tema también es conocido con el título "estructuras de datos" y se relaciona también a distintos algoritmos para agilizar el acceso. Antes de hablar específicamente de las colecciones de Java que se utilizó, hablaremos un poco sobre el problema en general de almacenar colecciones de datos en memoria.

Podríamos decir que la primera estructura de datos utilizada es la variable. Almacena un solo objeto y el tiempo de acceso es ínfimo. La segunda estructura de datos que se utiliza es el arreglo (array). Esta colección nos permite obtener un valor dado de una posición en una tabla. Es muy veloz ya que esta estructura tiene un fuerte paralelo con cómo se organiza internamente la memoria de una computadora. Los valores en un arreglo están en orden y en ese mismo orden están dispuestos en la memoria, por lo tanto el acceso lo maneja directamente el hardware (porque la memoria funciona físicamente como un arreglo).

Asimismo cada una de las soluciones puede traer problemas secundarios que serán más o menos importantes de acuerdo a nuestras necesidades. En el proceso se necesita mantener una serie de datos que constantemente cambiaran en su estructura y volumen. Por ende se utilizan "ArrayList" que es una colección de datos dinámica en cuanto a su tamaño y en cuanto al formato de los objetos que almacena. Esta estructura de datos que facilitan el proceso de validación sin tener que utilizar la base de datos.

Por ejemplo, luego del proceso de recolección de datos en los distintos motores de base de datos, tenemos el material suficiente para construir la estructura de datos que compone a un transporte y por lo tanto la siguiente etapa es la validación.

Para el proceso de validación se sobrecargó el método con variantes que permiten validar distintos instancias de las programaciones, de este modo se separa de manera precisa las instancias de un proceso inicial de la programación de los re procesos denominados sub-grupos de lotes.

En la primera instancia se evalúan los datos, se construyen las estructuras de transportes, se validan y se registran en la base de datos.

En la segunda etapa solo se recoge la información se evalúa y valida el sub-grupo a enviar. El objeto de negocio sigue en forma recursiva re evaluando los datos de sub-grupos pendientes de evaluación hasta que no quede ninguno, solo entonces se da término a la programación del escenario aceptado en la primera parte del proceso.

```
/**
 * @throws PedidosDAOException
 *
 */
public void validaTransportes(String processNumber) throws PedidosDAOException {
    validaProcesoAEjecutar(transportes);
    separaTiposTransportes(processNumber);
    saveTransportes();
    separaLotesTransportes(processNumber);
    asignaDatosTransportesAll();
}
```

Figura 39-Código Validación de Datos

La "Figura 39-Código Validación de Datos" muestra la lógica de validación de datos, luego de la creación de la estructura del transporte.

ValidaProcesoAEjecutar: es un método que recibe una colección de transportes, los que evalúa y valida según una plantilla que corresponde a la matriz anteriormente descrita con la serie de combinaciones de casos que podrían presentarse en la lógica de negocios del accept. Al finalizar la ejecución de este método se reasigna los datos a la colección de transportes, para adicionar el caso asignado al transporte junto con algunos datos adicionales.

SeparaTiposTransportes: es un método que rescata de memoria una colección de transportes los que evalúa para asignarlos a colecciones de datos de transportes con distintos

formatos, en los que se incluyen formatos de IDOC de creación de desprogramación y formatos a funciones RFC.

SaveTransportes: es un método que rescata de memoria una colección de datos ya filtrada y consistente para ser grabados en la base de datos.

SeparaLotesTransportes: es un método que rescata de memoria una colección de datos de transportes viables para su contabilización en SAP, los reordena para que no se repitan los números de camiones ya que esto es una restricción de SAP, el que no puede contabilizar en un mismo envío las distintas rutas de un camión. Los restantes quedan pendientes para ser procesados por el objeto de negocio que procesa los sub-grupos de lotes.

AsignaDatosTransportesAll: es un método que rescata de memoria una colección de datos de transportes viables para su contabilización en SAP y le adiciona el segmento de eventos para cada pedido en el transporte.

```
public void validaTransportesc(String processNumber) throws PedidosDAOException {
public void validaTransportes2(String processNumber) throws PedidosDAOException {
public void validaTransportes3(String processNumber) throws PedidosDAOException {
public void validaTransportes4(String processNumber) throws PedidosDAOException {
```

Figura 40-Código validación datos transportes

La "Figura 40-Código validación datos transportes" muestra los distintos métodos de validación implementados para evaluar los datos de transportes en las distintas instancias en que estos se producen ya sea en el proceso inicial del accept o en los sub-grupos de lotes y también en la evaluación de programación por contingencia de la Web.

ValidaTransportesc: corresponde al proceso de validación de transportes creados/programados desde la Web o más bien llamada "Contingencia BDP", y aplica solo cuando es una combinación F11.F10. Esto quiere decir una eliminación por IDOC y una programación por IDOC.

ValidaTransportes2: corresponde al proceso de validación de transportes creados/programados desde la Web o más bien llamada "Contingencia BDP", en el proceso inicial de la contingencia, esto es cuando se da "submit" a la programación editada en la Web.

ValidaTransportes3 : corresponde al proceso de validación de transportes creados/programados en el sub-grupo de proceso por lotes. En este método adicionalmente se incorpora el método de "separaLotesTransportes", el cual tiene por objeto separar las rutas de un mismo camión de distintas agrupaciones de formatos físicos.

ValidaTransportes4: corresponde al proceso de validación de transportes creados/programados en el sub-grupo de proceso por lotes.

El objeto de negocio que implementa esta lógica se denomina ***"ProgramadosAccessBeans".***

Esta clase se definió en el DAO que se ha implementado en la componente EJB de la aplicación J2EE y es el objeto encargado de manipular las instancias DAO en la base de datos.

Este DAO obtiene los accesos a la base de datos a través de una clase DataSource, el cual está definido en un proyecto java simple como una librería de utilidades comunes "***CosmosUtil***".

Una clase de objetos de acceso de datos puede proporcionar acceso a recursos de datos sin el acoplamiento de la API de recursos a la lógica de negocio. Por ejemplo, la aplicación tiene las siguientes clases de tipos pedidos, configuración y Log de pedidos usando DAO interfaz ***PedidosDAO***.

- DB2PedidosDAO
- DB2PedidosLogDAO
- DB2PedidosConfDAO

La interfaz ***PedidosDAO*** (Figura 41-Código Interfaz PedidosDAO) define el API de DAO. Los métodos en la interfaz a continuación no se hacen referencia a un mecanismo de acceso a datos. Por ejemplo ninguno de los métodos especifica una consulta SQL y lanzan únicas excepciones de tipo ***RegPedidoServicesException***, ***PedidosDAOException***. Para evitar mecanismos específicos de la información en la interfaz DAO, incluidas las excepciones arrojadas, es esencial para ocultar los detalles de la ejecución:

```
public interface PedidosDAO {
   public boolean insertPedido(Object Pedidospedido) throws ReqPedidoServicesException, PedidosDAOException;
   public boolean deletePedido(Object Pedidospedido)throws RegPedidoServicesException, PedidosDAOException;
   public Object findPedido(Object StringidPedido)throws RegPedidoServicesException, PedidosDAOException;
   public boolean updatePedido(Object Pedidospedido)throws RegPedidoServicesException, PedidosDAOException;
   public Collection selectPedidoRS(Object Pedidospedido)throws RegPedidoServicesException, PedidosDAOException;
 }
```

Figura 41-Código Interfaz PedidosDAO

La clase DB2PedidosDAO implementa esta interfaz para la base de datos relacional DB2 iSeries, como se muestra en el siguiente fragmento de código:

```
public boolean updatePedidoReporte(Object pedido) throws RegPedidoServicesException, PedidosDAOException {
      boolean sw = false;
      Pedidos p = (Pedidos) pedido;
      Connection c = null;
      PreparedStatement ps = null;
      ResultSet rs = null;
      String contador = null;

         try {
            c = getDataSource().obtenerConnection(JNDINames.CATALOG_DATASOURCE_DB2);
            ps = c.prepareStatement(UPDATE_PEDIDO_SQL_REPORTE);

            ps.setString(1, p.getTipoReporte());
            ps.setString(2, p.getNumPedido());
            ps.setString(3, p.getContadorEnvioBdp());
            ps.execute();
            sw = true;

            } catch (SQLException se) {
                se.printStackTrace();
                throw new PedidosDAOException(se);
              } catch (ConnectionMgrException se) {
                throw new PedidosDAOException(se);
              }finally{
                 try{if(rs != null)rs.close();}catch(Exception e){System.out.println("Err :" +e.getMessage());}
                 try{if(ps != null)ps.close();}catch(Exception e){System.out.println("Err :" +e.getMessage());}
                 try{if(c  != null)c.close();}catch(Exception e){System.out.println("Err :" +e.getMessage());}
              }

   return sw;

   }
```

Figura 42-Código clase DB2PedidosDAO

La clase OraclePedidosDAO implementa esta interfaz para la base de datos relacional Oracle en BDP, solo cambia la variable que apunta al dataSource de Oracle.

Los Objetos de negocio utilizados para la utilización del DAO tienen el sufijo "AccessBeans". En este caso ***PedidosAccessBeans*** posee la lógica de negocio que es utilizada en la implementación de registro y la planificación de pedidos.

El diseño de una interfaz DAO y la ejecución es un compromiso entre la sencillez y la flexibilidad. En la aplicación se dan varias estrategias para la aplicación del modelo de acceso de datos de objetos.

Hay que destacar que los objetos de negocio están en la componente Web como en la componente EJB y el DAO se encuentra implementado en la componente EJB del package ***"registropedidosservices.dao".***

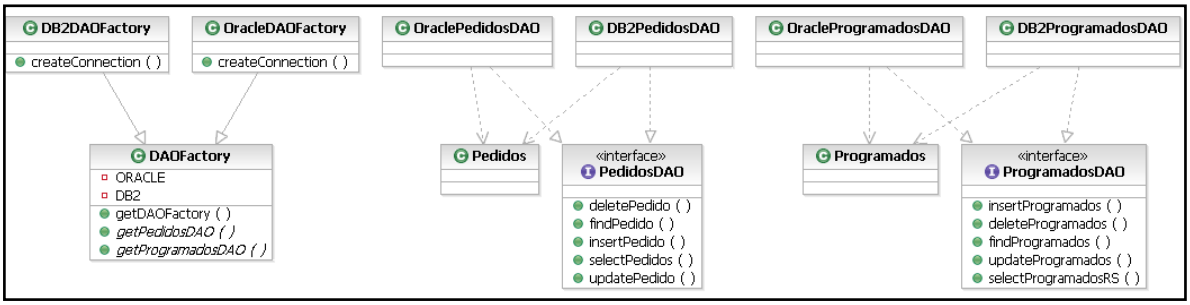


Figura 43-Clases de Objetos y sus interfaces

Lo anterior aplica para las interfaces:

- PedidosConfDAO
- ProgramadosDAO

6.4.2.7 Reportes asociados al proceso de Accepts

Los reportes que pertenecen a esta definición son el reporte cuatro y la página de contingencia.

El reporte cuatro (**Log de Errores**) es el encargado de mostrar los errores que se producen durante la ejecución de la lógica de negocio. Esta como es compleja tiene muchas aristas que necesitan ser registradas para tener claridad de los datos que no son exitosamente procesados y por ende no llegan a SAP.

Para tal efecto en la lógica del objeto "***ProgramadosAccessBeans***" se definió un arrayList que almacena todos los casos de error que corresponden a los transportes con problemas de datos o que por algún motivo se ha definido en la matriz "Lógica de Programación" de casos que no son viables para el envío a SAP.

En el método "***separaTiposTransportes***" se definen las estructuras que llenaran los distintos formatos del mensaje a ser enviado, a pesar que solo existen dos formatos IDOC y RFC hay una serie de consideraciones para el llenado de estos, como lo son por ejemplo las distintas operaciones que cada "Bussines Object" de un mensaje finalmente ejecuta en SAP. Por ende el "Bussines Object" a ser llenado debe considerar en la lógica de cada uno de sus segmentos la operación y los datos necesarios para ejecutar esa operación en SAP.

Se detalla una lista de los posibles problemas, casos de error o simplemente casos en que no se debe enviar nada a SAP.

Transportes con problemas de datos:

- Status Transporte > 3 nada en SAP
- Error DT ruta con carga iniciada
- Error CT ruta con marca Contingencia
- Error Casos 12
- ZCES sin Secundarios (Padres sin hijos)
- ZPRI sin Secundarios (Padres sin hijos)

Transportes sin modificación, no se envía transporte a SAP:

- Caso 25
- Caso 45
- Caso 65

Transporte se envía solo Desprogramación y Log de error:

- Caso 22
- Caso 23

Transportes con error en formato RFC:

- Error RFC 33

- Error RFC 43
- Error RFC 52
- Error RFC 53
- Error RFC 62
- Error RFC 63

6.4.2.8 Formato del mensaje enviado a SAP

Los formatos como se mencionaron anteriormente son IDOC (**OILSHI01**) y RFC (**sap_zvi_act_pedido_bdp2**).

En el caso del uso de las funciones RFC, la comunicación entre SAP y los sistemas no SAP se llevan a cabo por medio de lo denominado como Remote Function Call (RFC).

Remote Function Call es el estándar de interfaz de SAP para la comunicación entre sistemas SAP. El RFC pide una función para ser ejecutado en un sistema remoto.

RFC es un método de comunicación síncrona que ejecuta la función en la llamada al módulo RFC servidor.

La definición del servicio es en parte muy similar al de un IDOC y corresponde a una estructura con segmentos y definiciones que agregar al conector de SAP, etc. Sin embargo en el proceso de comunicación con el adapter es muy diferente ya que esta función denominado ***"sap_zvi_act_pedido_bdp2",*** a diferencia del IDOC, es una llamada síncrona.

La "Figura 44-Estructura Mensaje" detalla su estructura:

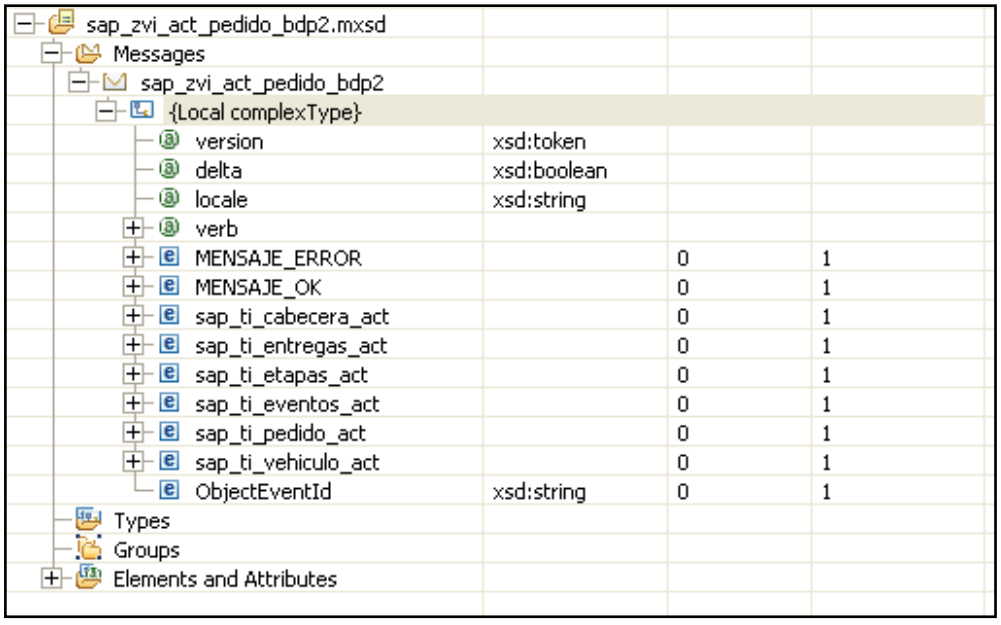


Figura 44-Estructura Mensaje

En el detalle de la definición, el mensaje de error y el mensaje ok, son los que contienen el estatus del resultado de su ejecución en SAP. Hay que destacar que en la respuesta viene el mensaje completo (toda la estructura del mensaje).

En la capa de mediación del mensaje de transporte a ser invocado desde una función RFC se incluye un sub-flujo específico encargado de realizar la llamada por medio del adapter, del cual obtiene la respuesta de SAP, la que envía inmediatamente con el estatus ya sea de error o de éxito de la operación. Esta definición no considera los datos para pedidos secundarios de estaciones de servicios, los que son manejados por el IDOC OILSHI01.

Sin embargo esta función permite realizar múltiples operaciones con el transporte en SAP sin necesidad de re enviar los datos como en el IDOC dos veces, como es el eliminar y crear nuevamente el transporte. Esto nos permite disminuir la carga en el servicio de un flujo de mensajes que se deba mediar en la capa de integración por el broker, reduciendo a un solo mensaje que sea capaz de eliminar volver a crear y modificar el transporte dado.

El sub-flujo encargado de realizar la llamada es el siguiente:

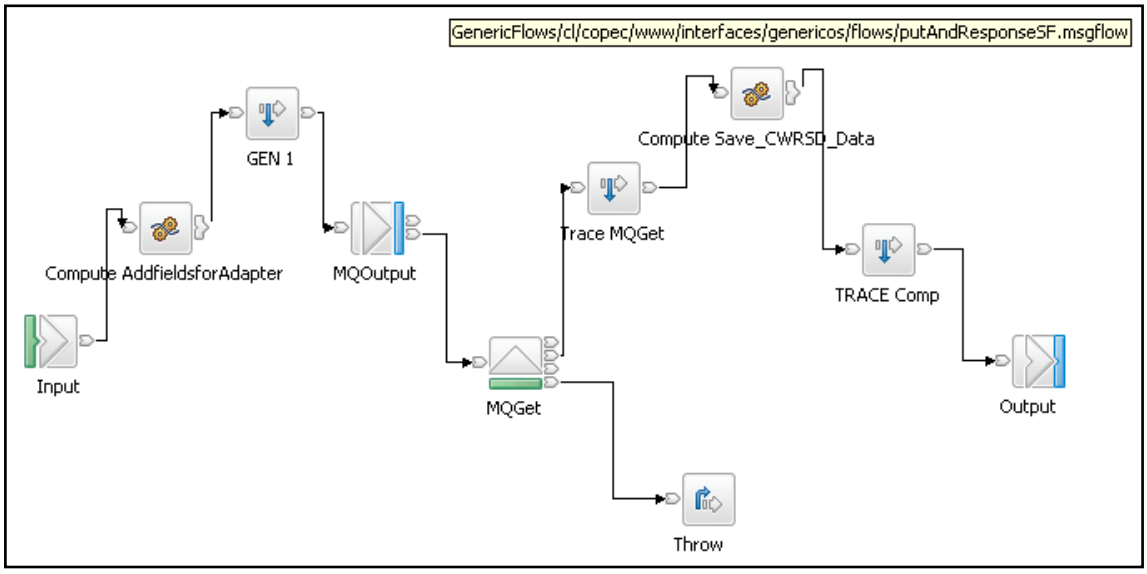


Figura 45-SubFlujo Broker para un transporte

ADP.SAP.REQUESTQUEUE la que está configurada en el nodo MQOutput. Luego en un nodo MQGet es leída la respuesta en el adapter en la cola ADP.SAP.RESPONSEQUEUE, el que finalmente devuelve la salida al flujo, desde donde fue invocado el sub-flujo.

En el caso del IDOC OILSHI01 el que se explicará en detalle más adelante, la invocación se lleva a cabo depositando el mensaje en la cola ADP.SAP.REQUESTQUEUE. Para saber si el transporte fue contabilizado en SAP, esperamos que SAP envíe el IDOC OILSHI01 como respuesta, pero esta vez con una estructura diferente al enviado inicialmente. Este mensaje que proviene desde SAP es mediado por otro servicio.

- SSD - SAP : actualizapedidosservice : envía el IDOC OILSHI01 (SSD-SAP).

- SSD – SAP : actualizarpedidoservicerfc : envía la RFC (SSD-SAP).
- SAP – SSD : actualizarpedidoservicesap : recibe el IDOC OILSHI01 (SAP-SSD).

6.4.3 Modelo de Datos

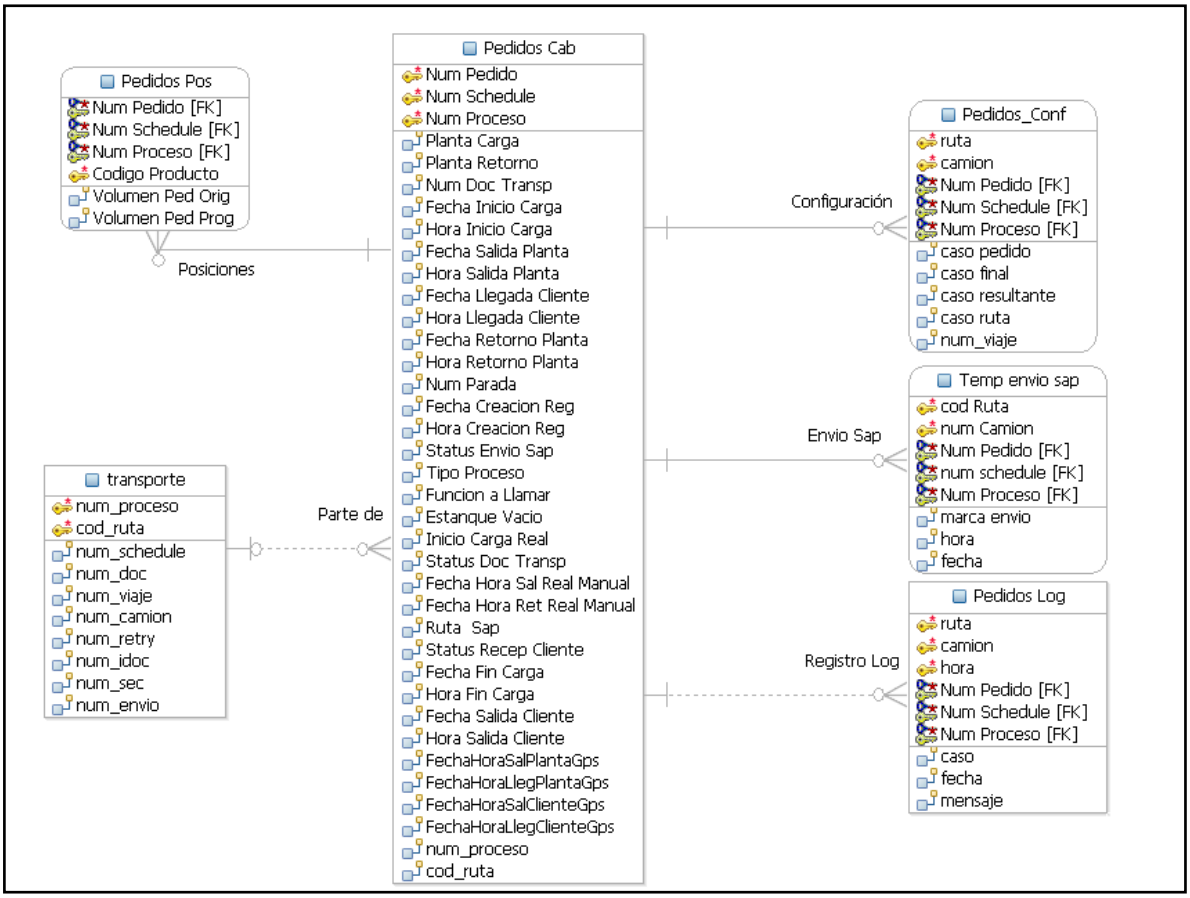


Figura 46-Modelo de datos Actualización Pedidos

6.5 Actualización de Pedidos Programados

6.5.1 Escenario de Integración

SAP utiliza servicio de Legacy J2EE (Online).

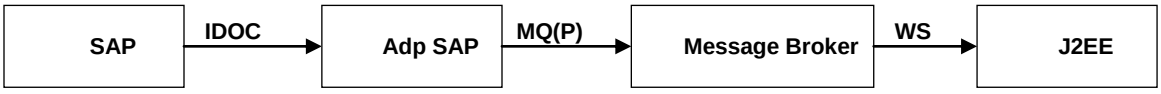


Figura 47-Escenario 4: SAP utiliza servicio J2EE

Este escenario es muy similar al de registro de pedidos, sin embargo tiene particularidades que lo diferencian de él como es que solo existe un tipo de mensaje que es tratado, es decir del mensaje que define al transporte el idoc **OILSHI01**.

6.5.2 Lógica de Implementación

En la mediación del mensaje que proviene de SAP intervienen dos flujos: uno externo y un flujo interno. Esto quiere decir que el mensaje es de un solo tipo pero a pesar de eso este mensaje que posee un único formato. También puede venir con distintas formas, vale decir, la relación de datos en la configuración de cada uno de los segmentos del idoc.

Este idoc es bastante particular porque en su estructura puede configurarse múltiples eventos en SAP:

Proceso de flujo del Mensaje

- El Sistema R/3 y/o sobre la base de los pedidos internos. Envío de datos que en R/3 se utilizan IDoc OILSHI01, el que corresponde a los transportes en SAP manejados de forma estándar.
- Los IDocs se interpretan por el sistema R/3. Ellos sirven de base para la creación, modificación, eliminación y la supresión del envío de los documentos.

El OILSHI01 realiza las siguientes funciones:

- Contiene segmentos específicos en el que el envío de datos es almacenado.
- Esos segmentos son copiados en las tablas internas.

La siguiente tabla muestra qué datos se almacenan en los segmentos del Idoc y si es necesaria para el procesamiento de entrada/salida, o si es opcional (IDoc-segmento de los datos necesarios y opcionales).

IDoc-Segment	Data	Required/Optional
E1OILSH	Shipment header	Required
E1OILSE	Shipment event	Optional
E1OILSZ	Shipment event: text lines	Optional
E1OILSA	Partner	Optional
E1OILS1	TD shipment history	Optional

E1OILSM	Material in shipment	Optional
E1OILSV	Vehicle	Optional required for E1OILSF, E1OILSC, E1OILS2, E1OILSF or E1OILS3
E1OILSF	Driver	Optional
E1OILSC	Compartment	Optional required for E1OILSQ
E1OILSQ	Compartment allocation of quantities	Optional
E1OILS2	Assignment of document items to stages	Optional
E1OILST	TD shipment stages	Optional
E1OILFQ	Assignment of document quantities to stages	Optional
E1OILS3	Quantity per material/vehicle	Optional
E1OILSI	Shipment item data	Optional required for E1OILSP
E1OILSP	Shipment item position data	Optional required for E1OILS4
E1OILS4	Storage object segment	Optional

Las entregas se crean sobre la base de los datos de envío. Cuando las funciones no se llevan a cabo con éxito el procesamiento del IDoc tendrá que ser repetido. La situación del IDoc se fija a 53 cuando todas las funciones se han ejecutado con éxito y los documentos correspondientes se han publicado.

Operaciones definidas en SAP para el idoc OILSHI

- F10 Idoc 51 Crea entrega y transporte
- F11 Idoc 53 Borra entrega y transporte
- F12 Idoc 54 Modifica transporte

En el caso de los transportes, idoc OILSHI01, solo se utiliza el formato del idoc para el envío de transportes en su proceso de creación y eliminación, para el caso de la modificación del

transporte se utilizan llamadas a funciones RFCs. Por ende la operación del idoc 54 no es utilizada en esta interfaz.

El mensaje luego de salir del adapter SAP pasa por un flujo ruteador que lo direcciona a un segundo flujo que propaga el mensaje a dos sub-flujos cada uno pertenece a distintas interfaces, estas son Fuel-Facs y SSD.

Ambos flujos finalmente luego de la lógica de transformación del mensaje finalizan con una llamada a servicios Web. En esta llamada el mensaje es tomado por una aplicación J2EE, la que finalmente termina registrando los datos significativos para el negocio en una base de datos.

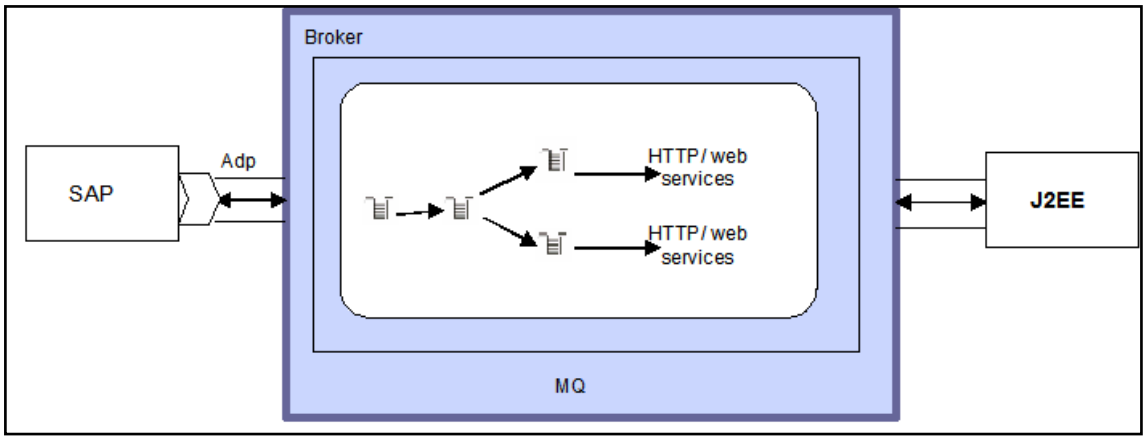


Figura 48-Ruteo de Flujos

El idoc de transportes es especialmente complejo por la serie de configuraciones que debe contener en su estructura y poder rescatar los datos necesarios para determinar en el transporte cuales son los pedidos y sus respectivas posiciones, esto es, datos del cliente y datos de cada pedido a despachar.

Toda esta información es enviada al SSD para mantener los datos consistentes con respecto a SAP. Además se necesita tener claro cuáles son los transportes que se pueden re programar y cuáles no. Los que ya no se pueden modificar o re programar son aquellos que ya han sido despachados, por ende tienen en el registro de datos la marca de estatus de documento de transporte mayor a dos, además de un valor para la fecha inicio carga.

Actualización de Transportes en SSD

Los siguientes casos tienen la relación entrega v/s pedido. Los transportes creados en Sap deben validar que estos datos existan antes de grabar el transporte:

- Pedidos Consolidados.
- Pedidos ZCES.
- Mix Pedidos Normales v/s ZCES, etc.

Se acordó incluir una relación en el campo del IDOC OILSHI Segmento e2oilse/SI_DOC_NR que contendría el numero de entrega para identificar el pedido y su programación.

Por ende también hay que formalizar este cambio en la especificación y en la operación de SAP, además de validaciones como que el dato exista para que la relación aplique.

Comentarios para el caso ZCES:

- Para identificar las entregas que están asociadas al pedido principal se deberá leer el segmento E1OILSZ, el cual está subordinado al segmento E1OILSE.
- En el segmento E1OILSZ y campo TDLINE se encuentra el código de la entrega (la misma del segmento E1OILSQ y campo DOC_NUMBER). Se deberán considerar todos los segmentos E1OILSZ subordinados al segmento E1OILSE ya que es posible que un pedido principal tenga más de una un pedido secundario y por lo tanto más de una entrega.
- En el campo "Nº de documento" del segmento E1OILSE se guarda el pedido principal el cual deberá tener la suma de las cantidades de las entregas encontradas en los segmentos E1OILSZ y E1OILSQ).

6.5.3 Modelo de Datos

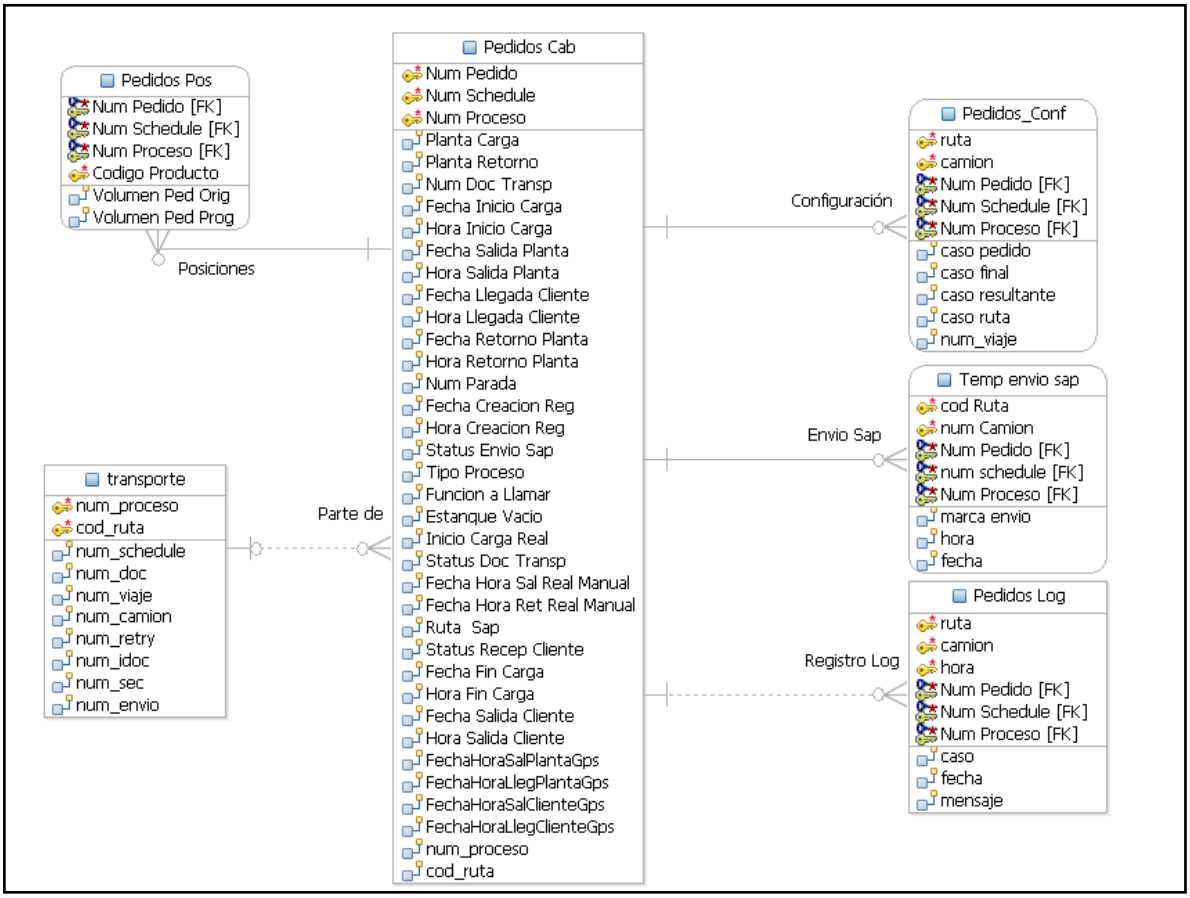


Figura 49-Modelo de Datos Actualización Pedidos Programados

Capítulo 7 Discusión y Conclusiones

Como resultado del trabajo, se cuenta con un sistema renovado e integrado que permite el despacho de combustible a los clientes de una compañía distribuidora después de la implementación de SAP.

Durante el trabajo se privilegió el disminuir las modificaciones que se le tuviesen que realizar a los legacy que permanecerían funcionando, esto con el fin de que el proyecto global al cual se vio enfrentado la compañía no aumentase de costos pero sobre todo no produjese mayores puntos de riesgos por lo que significaba toda la renovación tecnológica.

Varios de los productos de IBM que se implementaron (Message Broker, MQ, etc.) ayudaron a minimizar las modificaciones al código o funcionamiento de los legacy y crear piezas de software nuevas que fuesen reutilizables hoy y más adelante. Esto significó tener que pensar en soluciones centralizadas en el ESB del tipo: transformación de la data, almacenamiento en sistemas de datos intermedios o de paso, sincronización en el paso de la información, etc. Quedó de manifiesto lo complejo que fue integrar aplicaciones distintas.

El presente trabajo se concentró en la integración y cómo resolverlo sin entrar en detalles del origen del flujo de datos y cómo se componía ahí la información, pero se detalló el cómo el ERP de SAP enviaba la información: Idoc. De la amplia gama de mensajes idoc que SAP trae, se logró encontrar aquellos que más se acercaban a las necesidades de los legacy. Pero dado que los legacy y aplicaciones J2EE no manejan de manera nativa este tipo de mensajes, se utilizó de manera exitosa los adaptadores que trae el MessageBroker para lograr comunicar ambos extremos del flujo.

Existieron aplicaciones como el BDP y el FuelFacs que se manejaron de manera particular y especial dada sus características de ser cerrados sin posibilidad de sufrir modificaciones. En el caso del FuelFacs, encargado de inyectar el combustible en los camiones tanques, no se revisó en este trabajo dado que no sufrió ningún tipo de cambio o adaptación, funcionando éste con lectura de los datos a través de archivos planos. Por el lado del BDP si bien tampoco sufrió modificaciones, se debió entender en profundidad el tipo de información que esperaba recibir y sobre todo la información devuelta que respondía. Esto fue un trabajo de investigación particular que si bien no tomó mucho tiempo, permitió adaptar la información que salía desde el Broker hacia él de mejor manera.

Durante la implementación y luego de tener el diseño de solución terminado aparecieron complicaciones tales como el manejo de archivos que hace el Broker: éste sólo podía leer/escribir archivos que estuviesen en su propio filesystem y no ir a servidores remotos. Esto traía complicaciones en los escenarios en que la información a transmitir entre sistemas fue muy grande como para buscar alternativas tales como tomar la data y en vez de crear archivos colocarlos en mensajes en colas del MQ. La solución en este caso particular pasó por instalar el protocolo NTFS en los servidores comprometidos y así montar las carpetas de los servidores de origen/destino en el servidor del Broker.

La tecnología ofrecida hoy en día por grandes empresas como SAP e IBM para temas de integración no está del todo madura y no alcanza a dar soluciones completas de punta a punta. Esto significa que los proyectos de integración deben verse enfrentados a una gran cantidad de escollos que deben ser superados con desarrollos propios y con el uso de tecnologías afines. Si bien la tendencia de las empresas por estandarizar los protocolos y aplicaciones es cierta hasta

cierto nivel, aún se hace necesario un esfuerzo y conocimiento adicional no menor para lograr de manera armoniosa hacer funcionar un proceso de negocio como el presentado en este trabajo.

7.1 Evaluación de la integración

En la actualidad el proceso de despacho de combustible implementado de acuerdo a lo expuesto en este trabajo funciona de manera productiva y constante, pero con varias complicaciones de performance e incluso de pérdida de información.

Por el lado de la performance existieron aspectos técnicos no levantados durante el proyecto que al momento de estar en producción aparecieron como temas importantes a corregir, tales como: problemas de performance en el ERP que hacen que los idocs se muevan de manera mucho más lenta como así también los mensajes devuelta desde el SSD, el uso de las base de datos intermedias y del BDP que no fueron configuradas de acuerdo a las nuevas implementaciones (creación de nuevos índices en las tablas, etc.), el servicio que entrega el Broker no ha sido configurado por el administrador a cargo de manera correcta haciendo de éste una pieza central productora de cuellos de botella en los tiempos de espera, modificaciones en las aplicaciones J2EE del SSD que no tuvieron una mirada adecuada en la performance sino más bien ajustada a cumplir con los plazos del proyecto, etc.

En lo que a pérdida de información dice relación, una vez en producción comenzaron a aparecer situaciones anómalas como caídas del ERP que desarmaban todo el flujo de información produciendo que los pedidos que en ese momento estuviesen siendo tratados quedasen repartidos entre el Broker y el SSD, pérdida de sincronización entre el Broker y el BDP producto del uso excesivo del primero (y su falta de configuración adecuada) que no fue medida con anticipación de manera correcta, etc.

El funcionamiento actual del proceso, si bien entrega servicio, adolece de varios problemas que van repercutiendo en un extremo en la salida de los camiones desde las plantas y en el otro extremo en la facturación y contabilización de la compañía. El diseño de la solución de integración no es el culpable completo de estos problemas sino que son otros los actores primordiales: los administradores de las base de datos, del Broker y del ERP que enfocan actualmente sus esfuerzos en estabilizar la performance de los sistemas y realizan ajustes y mejoras para un mejor rendimiento; el Broker en sí que no hace un buen manejo de situaciones no programadas en los flujos o en las aplicaciones (por ejemplo campos de datos que vienen con otro formato o delimitadores) produciendo caídas del servicio en un punto neurálgico de la solución.

El intercambio de datos a través de archivos, si bien más estable que los mensajes a través del Broker, está sujeto a la estabilidad de la plataforma: Adaptadores del Broker funcionando, carpetas montadas sin caídas, etc. Sin embargo parece de menor complejidad el estabilizar la infraestructura que mejorar la performance de los mensajes a través del Broker.

7.2 Pendientes

Es necesario evaluar este sistema mediante su uso, para luego implementar una versión definitiva.

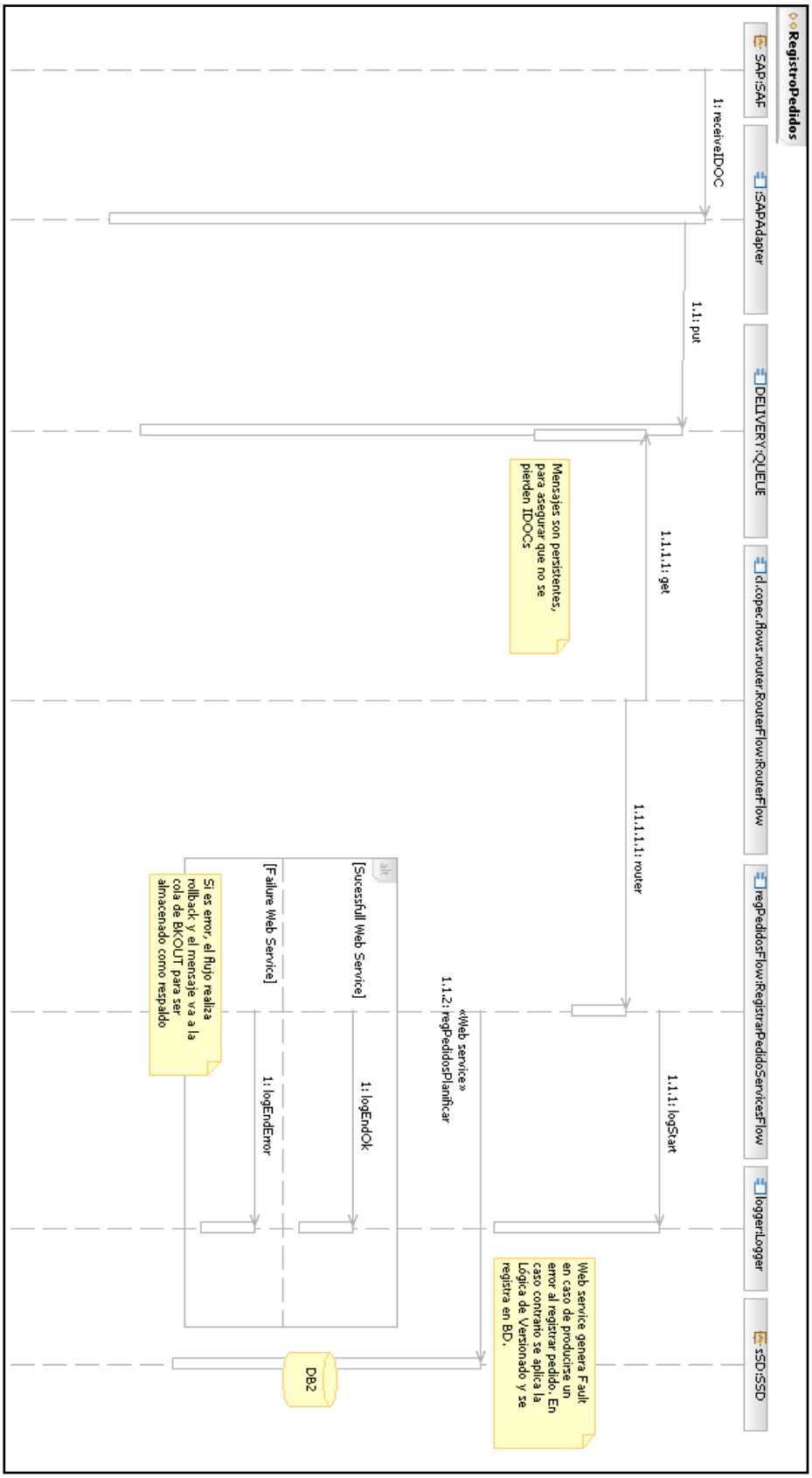
Las mejoras que se observan dependerán de las prioridades que se deseen observar. Si estas son meramente prácticas se buscará el reemplazo de algunos mensajes en el Broker por intercambio de archivos a través de carpetas montadas entre los servidores de las aplicaciones.

Sin embargo si las mejoras apuntan a hacer más eficiente la solución ya implementada se observan algunas como: creación de idocs personalizadas que envíen la información tal y como los legacy las necesitan, evitando que el Broker tenga que realizar además de un trabajo de ruteado el de transformación de la data; las RFC implementadas deben ser revisadas en su código en el ERP para realizar mejoras que agilicen su performance y así produzcan una disminución en los tiempos de ejecución; quedó fuera del alcance de este trabajo las mejoras al sistema GPS de los camiones por ser ese un proyecto en sí, pero de implementarse deberían ser revisados los flujos y programas acá implementados.

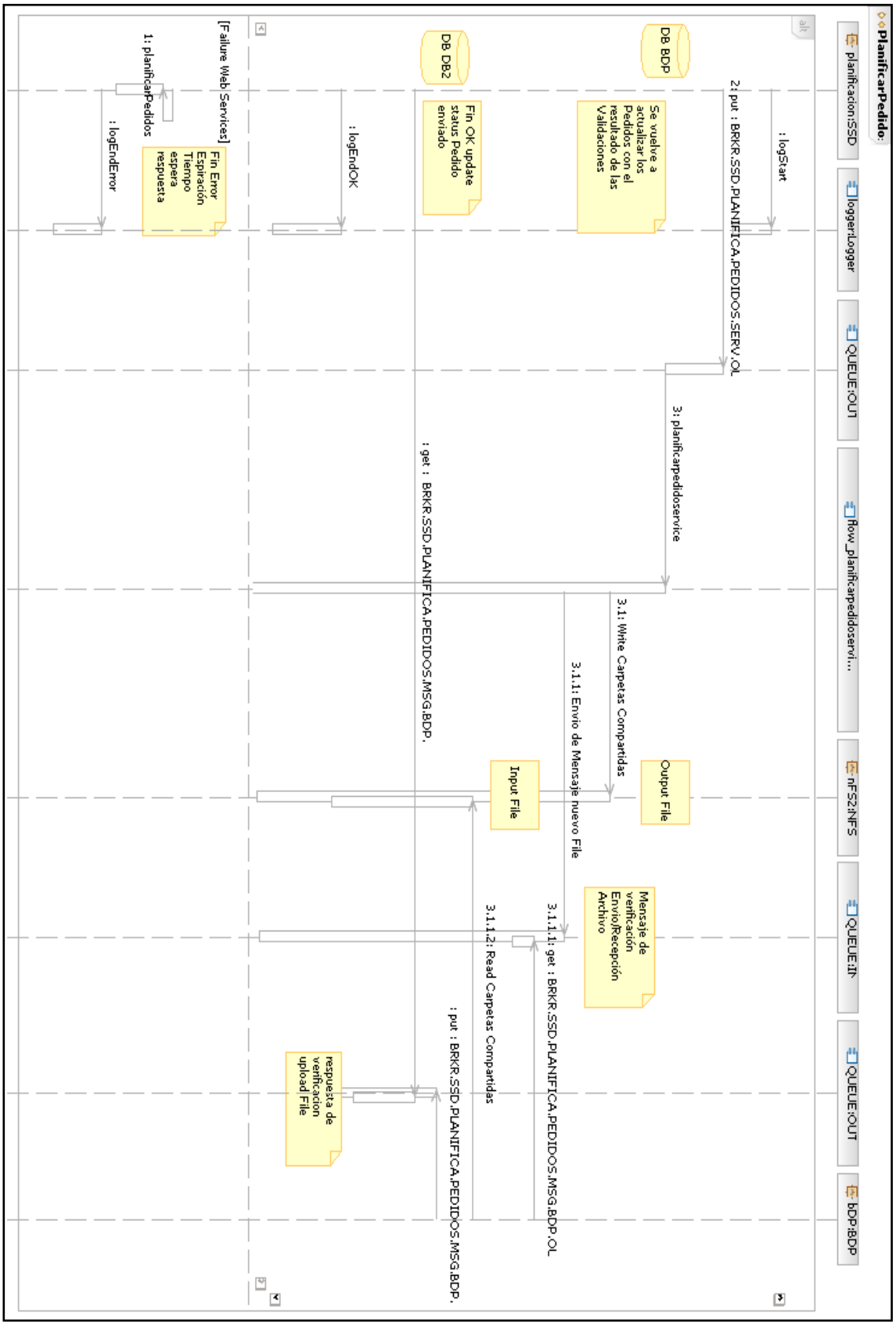
Apéndices

Diagramas de Interacción

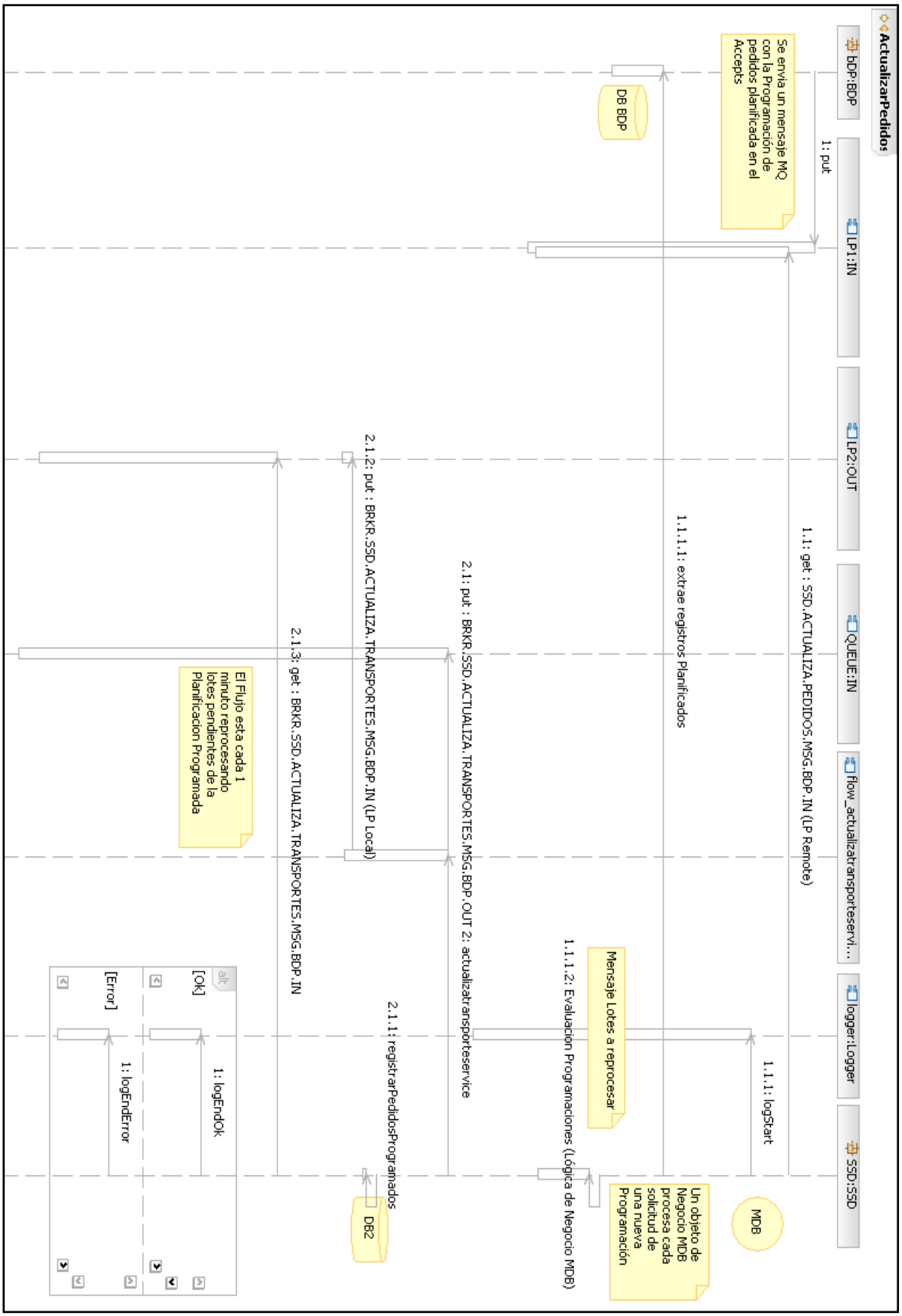
Registro Pedido (SAP-SSD)



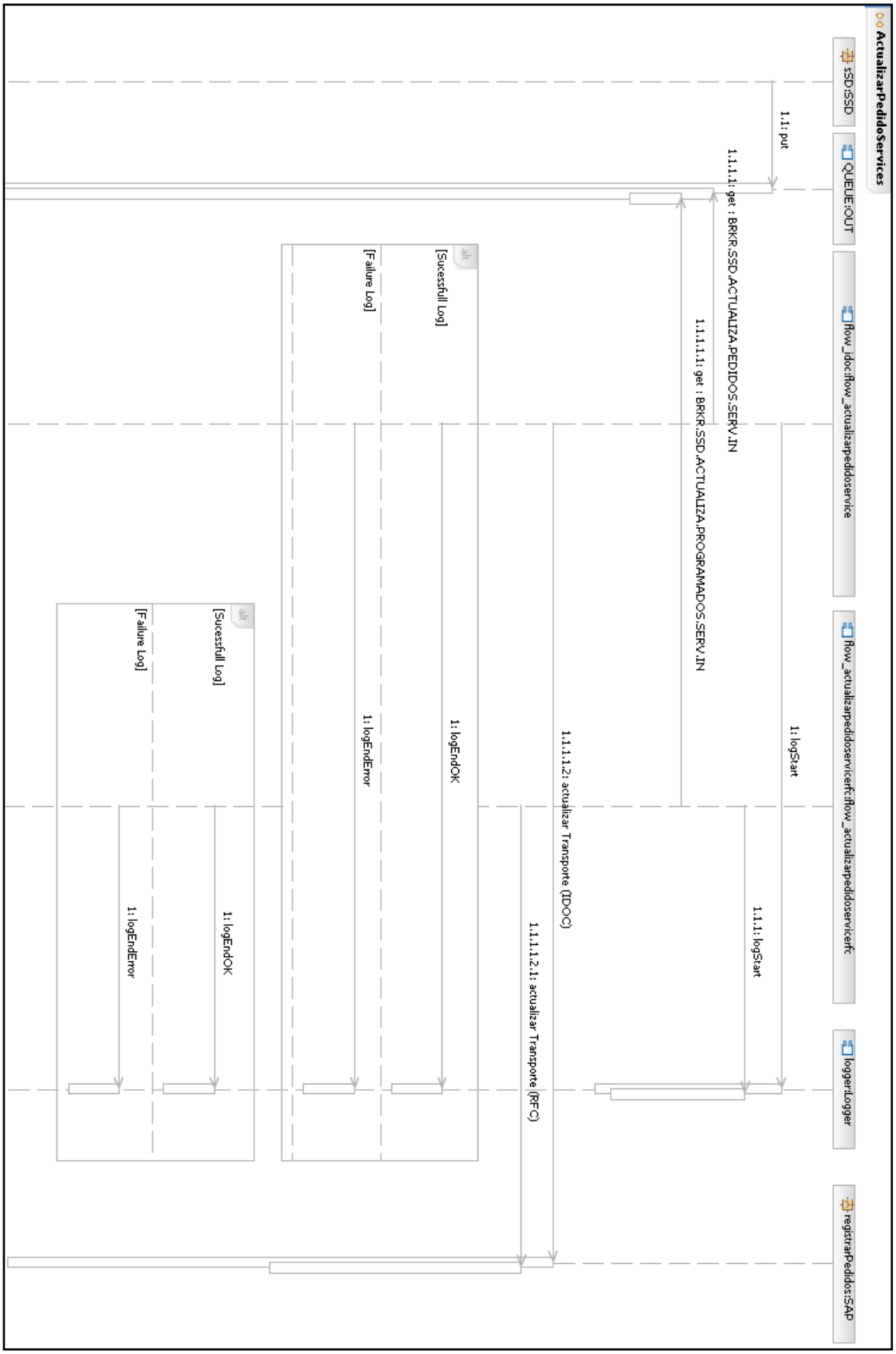
Planificación Pedidos (SSD-BDP)



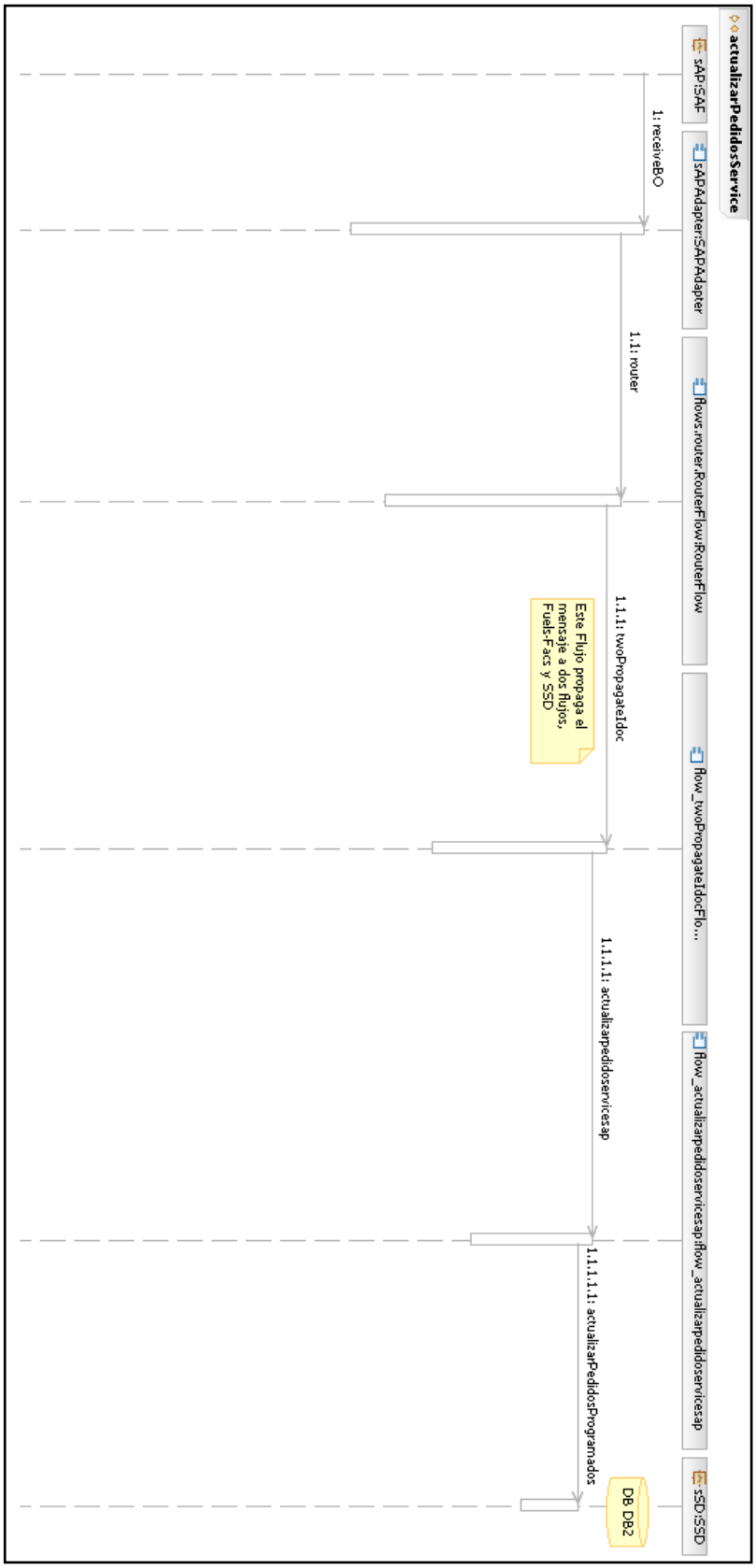
Actualización Pedidos (BDP-SSD)



Actualización Pedidos (SSD–SAP)



Actualización Pedidos Programados (SAP–SSD)



Bibliografía

1. SAP AG. *Wikipedia, the free encyclopedia.* [En línea] 12 de Julio de 2008. [Citado el: 10 de Julio de 2008.] http://en.wikipedia.org/wiki/SAP_AG.

2. **SAP.** SAP Service MarketPlace. [En línea] [Citado el: 31 de Agosto de 2007.] http://service.sap.com.

3. **Microsystems, Sun.** sun.com. *Java EE at a Glance.* [En línea] [Citado el: 30 de Agosto de 2007.] http://java.sun.com/javaee/index.jsp.

4. WikiPedia. *IBM RPG.* [En línea] 2007. [Citado el: 31 de Agosto de 2007.] http://en.wikipedia.org/wiki/RPG_programming_language.

5. **JDA.** JDA Transportation Planning. *Manugistic.* [En línea] [Citado el: 15 de Noviembre de 2007.] http://www.jda.com/solutions/transport.html.

6. **Krafzig, Dirk, Banke, Karl y Slama, Dirk.** *Enterprise SOA, Service-Oriented Architecture Best Practices.* Indianapolis : Prentice Hall, 2005. pág. 320.

7. WikiPedia . *SOAP .* [En línea] 2007. [Citado el: 07 de Septiembre de 2007.] http://en.wikipedia.org/wiki/SOAP.

8. **Vogel, Andreas y Kimbell, Ian.** *mySAP ERP for Dummies.* Indianapolis : Wiley Publishing, Inc., 2007. pág. 280.

9. NetWeaver. *Wikipedia, the free encyclopedia.* [En línea] 2007. [Citado el: 07 de Septiembre de 2007.] http://en.wikipedia.org/wiki/Netweaver.

10. **IBM.** IBM. *IBM WebSphere Software.* [En línea] [Citado el: 31 de Agosto de 2007.] http://www-306.ibm.com/software/websphere.

11. **Pune, Narendra Naidu.** Tech Talk: What are XA transactions? What is a XA datasource? *Design and Development Tips in Java and .NET areas.* [En línea] 26 de Enero de 2006. [Citado el: 21 de Abril de 2008.] http://narencoolgeek.blogspot.com/2006/01/what-are-xa-transactions-what-is-xa.html.

12. **Barcia, Roland, y otros.** *IBM WebSphere, Deployment and Advanced Configuration.* New York : Prentice Hall, 2005. pág. 540.

13. **Robinson, Scott.** Understanding the components of SAP IDocs. *TechRepublic.* [En línea] 05 de Agosto de 2002. [Citado el: 10 de Abril de 2008.]

14. SAP ALE Tutorial - Introduction. *theSpot4SAP.com.* [En línea] 01 de Agosto de 2006. [Citado el: 15 de Mayo de 2008.] http://www.thespot4sap.com/Articles/SAP_ALE_Introduction.asp.

15. **SAP.** RFC (SAP Library - Components of SAP Communication Technology). *Service MarketPlace SAP.* [En línea] 01 de Septiembre de 2005. [Citado el: 23 de Marzo de 2008.] http://help.sap.com/saphelp_nw04/helpdata/en/6f/1bd5b6a85b11d6b28500508b5d5211/content.htm.

16. **Shen, Rachel y Upadhyaya, Ankur.** ESQL code conventions in WebSphere Message Broker. *ibm.com.* [En línea] 12 de Marzo de 2008. [Citado el: 18 de Mayo de 2008.]

http://www.ibm.com/developerworks/websphere/library/techarticles/0803_shen/0803_shen.html.