

Universidad de Lima
Facultad de Ingeniería y Arquitectura
Carrera de Ingeniería de Sistemas



INTEGRACIÓN DE SISTEMAS ORIENTADA A SERVICIOS

Trabajo de investigación para optar el Título Profesional de Ingeniero de Sistemas

Jorge Luis Cabrera Chiappe

Código 19960200

Asesores

Fernando Fortunato Sotomayor Aramburu

Zalatiel Carranza-Avalos

Lima – Perú
Setiembre de 2010



**SERVICE ORIENTED SYSTEM
INTEGRATION**

TABLA DE CONTENIDO

RESUMEN	viii
ABSTRACT.....	ix
CAPÍTULO I: INTRODUCCIÓN	1
1.1. Formulación del problema.....	1
1.2. Justificación y alcances.....	3
1.3. Objetivos.....	3
CAPÍTULO II: Revisión crítica de literatura	5
CAPÍTULO III: MARCO TEÓRICO.....	9
3.1. Conceptos Principales.....	9
3.2. SOA - Arquitectura Orientada a Servicios:	10
3.3. Patrones:.....	13
3.1. Anti-patrones:	15
3.2. Arquitectura de Integración Empresarial (EIA-Enterprise Interation Arquitecture):.....	16
3.3. EIA y SOI	21
CAPÍTULO IV: Desarrollo del trabajo de investigación.....	22
4.1 Metodología.....	22
4.2 Desarrollo.....	27
CAPÍTULO v: RESULTADOS	48
5.1 Verificación de Requerimientos	48
5.2 Comprobación de Uso de Patrones	50
5.3 Evaluación	51
5.4 Contrastación del Método.....	55
CAPÍTULO vI: RESULTADOS DE LA INVESTIGACIÓN	57
CONCLUSIONES	58
RECOMENDACIONES	59
GLOSARIO DE TÉRMINOS.....	60
REFERENCIAS.....	61
BIBLIOGRAFIA	63
ANEXOS.....	64

ÍNDICE DE TABLAS

Tabla 3.1 Niveles de Adopción SOA.....	11
Tabla 3.2 Formas de Adopción SOA.....	12
Tabla 3.3 Niveles de Adopción para integración.....	13
Tabla 3.4 Anti-patrones	15
Tabla 4.1 Sub-Controlador Agnóstico	29
Tabla 4.2 - Recursos Canónicos (Canonical Resources)	30
Tabla 4.3 Reutilización de componentes entre dominios	31
Tabla 4.4 Transacciones entre múltiples servicios	32
Tabla 4.5 Mensajes enviados por eventos	33
Tabla 4.6 Ruteo dinámico.....	33
Tabla 4.7 Proveedor Virtual.....	34
Tabla 4.8 Bus de integración	35
Tabla 4.9 Bus de integración como “Gateway”	36
Tabla 4.10 Procesador de documentos	37
Tabla 4.11 Mensaje inalterable	38
Tabla 4.12 Integración desde capa de negocio	38
Tabla 4.13 Tren Tecnológico.....	39
Tabla 4.14 ¿Qué hay de Nuevo?	40
Tabla 4.15 El Big Bang.....	40
Tabla 4.16 Servicio Web = SOA	41
Tabla 4.17 El enfoque Silo (The Silo Approach)	41
Tabla 4.18 Registros de mal comportamiento	42

Tabla 4.19 Servicios Parlanchines (Chatty Services)	42
Tabla 4.20 Servicios punto a punto	43
Tabla 4.21 Servicios sin componentes.....	43
Tabla 5. 1 - Resultados	48



ÍNDICE DE FIGURAS

Figura 2.1 Magic Quadrant for Integration Backbone Software, 1H05	8
Figura 3.1 Integración mediante base de datos	16
Figura 3.2 Integración mediante base de datos	17
Figura 3.3 Integración mediante llamadas remotas	17
Figura 3.4 Integración mediante mensajería	18
Figura 4.1 Metodología.....	27
Figura 4.2 Diagrama de Clases Sub Controlador Agnóstico	29
Figura 4.3 Diagrama de Clases Recursos Canónicos	31
Figura 4.0.4 Reutilización entre dominios	32
Figura 4.5 Cross-Service Transaction.....	32
Figura 4.6 Cross-Service Transaction.....	34
Figura 4.7 Proveedor Virtual	35
Figura 4.8 Bus de integración	36
Figura 4.9 Bus de integración como “Gateway”	37
Figura 5.1 Constratación del método	56

ÍNDICE DE ANEXOS

Anexo 1: A.1.1 - Documento de Requerimientos.....	65
Anexo 2: A.1.3 - Documento de Análisis.....	69
Anexo 3: A.1.2 - Documento de Arquitectura (Inicial).....	71
Anexo 4: A.2.1 - Documento de Diseño.....	76
Anexo 5: A.2.2 - Documento de Arquitectura (Final).....	117
Anexo 6: A.3.1 – Código Fuente	122
Anexo 7: A.4.2 – Diseño de Pruebas	143
Anexo 8: A.4.3 – Plan de Pruebas	146
Anexo 9: A.4.4 – Casos de Prueba	149
Anexo 10: A.4.1 – Informe de Pruebas	152

RESUMEN

La integración de sistemas y el despliegue de arquitecturas de Integración Empresarial EIA (Enterprise Integration Architecture - Arquitectura de Integración Empresarial) constituyen, actualmente, algunas de las tareas más demandantes en la industria de tecnologías de información. En la medida de que las empresas poseen aplicaciones desarrolladas en diversas tecnologías, entre ellas, Sistemas de Planeación de Recursos Empresariales ERP (Enterprise Resource Planning), correo electrónico (email), sistemas de Planeamiento de la Producción, sistemas de Gestión de Ventas e Inventarios, Sistemas de Recursos Humanos, Gestión de clientes (Customer Relationship Management), las cuales requieren ser integradas. Dentro de este contexto, SOI (Service Oriented Integration o Integración de Sistemas Orientada a Servicios) propone dicha integración entre soluciones computacionales utilizando interfaces basadas en servicios con interfaces y servicios estandarizados.

La presente investigación demuestra que la integración SOI permite:

1. Aumentar la eficiencia en el desarrollo y mantenimiento de interfaces.
2. Reducir del esfuerzo en el proceso de desarrollo y mantenimiento de interfaces entre sistemas.
3. Unificar y estandarizar las interfaces hacia el uso de Servicios Web (Web Services) y XML (Extensible Markup Language), en los casos donde sea aplicable.
4. Mejorar la interoperabilidad de las distintas plataformas en la empresa.
5. Aumentar la reutilización de interfaces y componentes desarrollados.
6. Facilitar la integración con distintas entidades de negocio (por ejemplo: proveedores y clientes), mediante un estándar de facto (Web Services).

Palabras clave:

EIA, SOA, Integración, Sistemas, XML, Web Services.

ABSTRACT

System integration and the EIA (Enterprise Integration Architecture) discipline conform nowadays one of the most demanding in the IT industry. This respond to the fact that organizations own already implemented applications, develop in many different technologies like ERP, emails, Production Planning, Inventory, Sales, Human Resources, CRM, etc.; Those require to be integrated. Is into this context that SOI (Service Oriented Integration) proposes the integration between those computational entities use interfaces based in standardized services.

The present work investigates and probes that the following benefits are achievable following SOI:

1. Improve the efficiencies in the interface development.
2. Reduction of the effort for the develop and maintenance of the interfaces.
3. Unification and standardization in favor of Web Services (Web Services) y XML (Extensible Markup Language), where applicable.
4. Improve the interoperability of the organization applications.
5. Increase the re-use of interfaces and related.
6. Facilitate the integration with other entities like providers , customers and financial services throw this standard.

Keywords:

EIA, SOA, Integration, System, XML , Web Services.

CAPÍTULO I: INTRODUCCIÓN

1.1. Formulación del problema

La ausencia de un correcto enfoque de arquitectura de sistemas, el uso inadecuado de patrones y anti-patrones o la no utilización de éstos, ha generado que diversas empresas incurran en formas de integración de sistemas informáticos con un porcentaje bajo de re-utilización. Estas se caracterizan por un manejo de errores, transacciones y excepciones ineficientes además, por ser demandantes de recursos en las etapas de desarrollo construcción y de mantenimiento, principalmente en lo referido a tiempos, recursos humanos y capacidad computacional.

La situación antes descrita encuentra como principales factores los que a continuación se detallan:

1. Carencia de un planeamiento estratégico de las tecnologías de la información. En muchas organizaciones no existe un gobierno de TI (IT Governance).
2. Múltiples sistemas de información y/o aplicativos de distintas arquitecturas, proveedores y plataformas tecnológicas en las empresas, siendo las más usuales las siguientes:
 - Correo electrónico.
 - Sistema de gestión de contenidos (edición y publicación de documentos, imágenes y contenidos de distintos tipos).
 - Sistemas ERP (Enterprise Resource Planning).
 - Sistemas contables.
 - Sistemas de ventas en línea.
 - Sistemas CRM (Customer Relationship Management).
 - Sistemas de recursos humanos y/o planillas.

3. Necesidad de una empresa integrada de extremo a extremo (“End to End”), en donde la información de los aplicativos fluya a través de los sistemas de información y no se encuentre en silos de información.
4. Múltiples plataformas de aplicaciones, las cuales, en muchos casos, implican numerosos sistemas operativos, distintas bases de datos, diversos servidores de aplicaciones y componentes de hardware distintos.
5. Múltiples lenguajes de programación y plataformas de desarrollo.
6. Información distribuida en distintos sistemas y bases de datos, la cual es difícil de integrar y aprovechar.
7. Ausencia de un rol de arquitecto de sistemas dentro de la empresa ó un arquitecto de integración.
8. Desarrollo de interfaces heterogéneas punto a punto entre sistemas.
9. Diversos tipos y medios de integración en una misma empresa, entre ellos:
 - Integración mediante bases de datos (tablas de interfaz o intermedias).
 - Archivos planos de intercambio de datos.
 - Conmutadores transaccionales (programas que manejan operaciones que requieren atomicidad y modifican o crean datos en múltiples sistemas en línea).

Es el caso que, la confluencia de los factores antes descritos en las empresas trae consigo los inconvenientes que se describen a continuación:

1. Alta dedicación de los recursos de las áreas de sistemas a labores de desarrollo y mantenimiento de las interfaces entre sistemas. Ello se traduce económicamente en un alto costo respecto de dichas labores.
2. Ausencia de un estándar o política de desarrollo de interfases para la integración de sistemas.
3. Interfaces poco re-utilizables y, en muchos casos, ineficientes.

4. Manejo de errores y excepciones mediante procedimientos manuales, así como carencia de transaccionalidad y atomicidad en las operaciones, donde muchas veces quedan datos inconsistentes en distintos sistemas.

1.2. Justificación y alcances

El empleo de patrones plantea, por definición, una forma eficiente y probada para la resolución de problemas. De ello que con la creación de un catálogo de patrones y anti-patrones SOI (Service Oriented Integration o Integración de Sistemas Orientada a Servicios) y su utilización en conjunto con el método propuesto, se pretende proporcionar a los arquitectos de sistemas y profesionales afines una herramienta que permita, entre otros, los beneficios mencionados en el resumen.

La tesis propuesta incluye la implementación de una prueba de integración, la cual permitió evaluar el grado en que se alcanzan los citados beneficios. En ese sentido, el alcance de la tesis propuesta abarca lo siguiente:

1. Catálogo de patrones y anti-patrones.
2. Diseño de una metodología de integración de sistemas.
3. Prueba de concepto de integración para comprobar los beneficios relacionados a SOI mediante la metodología propuesta.

1.3. Objetivos

1.3.1 Objetivo Principal

Probar que la integración Orientada a Servicios (SOI) permite alcanzar, en proyectos de integración, los siguientes beneficios.

1. Aumentar la eficiencia en el desarrollo y mantenimiento de interfaces.
2. Reducción del esfuerzo en el proceso de desarrollo y mantenimiento de interfaces entre sistemas.

3. Unificación y estandarización las interfaces hacia el uso de Servicios Web (Web Services) y XML (Extensible Markup Language), en los casos donde sea aplicable.
4. Mejorar la interoperabilidad de las distintas plataformas en la empresa.
5. Aumentar la re-utilización de interfaces y componentes desarrollados.
6. Facilitar la integración con distintas entidades de negocio (por ejemplo: proveedores y clientes), mediante un estándar de facto (Web Services).

1.3.2 Objetivo Secundario

Crear un catálogo de patrones y anti-patrones SOI que sirva de herramienta a los arquitectos de sistemas y profesionales afines para alcanzar los beneficios señalados en el numeral precedente.

CAPÍTULO II: REVISIÓN CRÍTICA DE LITERATURA

En la actualidad no existe gran variedad de fuentes sobre integraciones de sistemas orientadas directamente a servicios web. No obstante ello, y aún cuando la mayor parte de la literatura encontrada sobre integración de sistemas está clasificada como patrones de EIA (Enterprise Integration Architecture), no es menos cierto que gran parte de dichos patrones son utilizables o extensibles al uso de XML y servicios Web, y además son la base para los patrones SOA y SOI. Es así que, la fuente de información más significativa, a la fecha, es “*Enterprise Integration Patterns*” , 26 de Abril 2010, (<http://www.eaipatterns.com/>). Cabe mencionar el libro del correspondiente sitio web: “*Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*”, por The Addison-Wesley Signature Series, 2010.

Por otro lado, debe tenerse presente que las principales empresas de software manejan información sobre SOA y SOI, así como sobre sus beneficios, no obstante lo cual dichas fuentes no son públicas en su mayoría, y en ese sentido, el acceso a las mismas es restringido. Sin perjuicio de lo expuesto, se proponen las siguientes direcciones electrónicas como fuentes de información relacionada:

1. IBM Developer Works: El sitio web de IBM “Developer Works” por IBM, 26 de Abril 2010 (<http://www.ibm.com/developerworks/>) ; contiene una gran cantidad de información sobre patrones y mejores prácticas para la implementación de servicios Web, BPM (Business Process Management), SOA (Service Oriented Architecture) y la metodología de IBM SOMA (Service Oriented Modeling Architecture). En dicho sitio web se pueden encontrar patrones, anti-patrones y además artículos que en su mayoría están referidos a las tecnologías J2EE y productos de la corporación IBM, tales como Websphere Message Broker, Websphere MQ Series, entre otros. En la medida de que IBM es considerado uno de los líderes SOA (Service Oriented Architecture), el sitio web en mención puede ser considerado un gran recurso teórico y práctico.

2. SOA Design Patterns: El sitio de internet “SOA Patterns”, de Arcitura Education, 26 de Abril 2010 (<http://www.soapatterns.org>), y el libro sobre el mismo “SOA Design Patterns” , 2010, de Thomas Erl constituyen una valiosa referencia sobre patrones orientados a servicios, ya sean de integración o arquitectura.
3. Integration Patterns - Microsoft: La página web “Integration Patterns” por Microsoft, 26 de Abril 2010 (<http://msdn.microsoft.com/en-us/library/ms978729.aspx>) es la referencia completa de patrones de integración propuestas por la Microsoft Corporation. Este libro constituye una referencia coherente y orientada a casos reales de implementación, muy comprensible y dinámica. Asimismo, se ofrece el sitio web “Patterns & Practices” ” de Microsoft, 26 de Abril 2010 (<http://msdn.microsoft.com/en-us/practices/default.aspx>), el cual contiene patrones y mejores prácticas para la plataforma .Net y el bus de integración sobre el producto de software Microsoft BizTalk Server.
4. Core J2EE Patterns: Sun Microsystems ofrece en su sitio web los principales patrones de arquitectura sugeridos para la plataforma J2EE, entre los cuales podemos diferenciar los patrones de integración propuestos para esta plataforma. Su sitio web “Core J2EE Patterns”, por Sun Microsystems, 26 de Abril 2010 (<http://java.sun.com/blueprints/corej2eepatterns/index.html>), contiene una cantidad reducida de patrones de integración que sin embargo son útiles y comprensibles. Pero debemos de considerar que estos no adentran en patrones, antipatrones ya sea para buses de integración ni portales, solo encontraremos los relativos a aplicaciones J2EE.

En adición a lo expuesto, cabe señalar que haciendo uso de la Internet se puede acceder a gran cantidad de artículos relacionados al tema propuesto que respaldan las ideas principales de la presente tesis. Así, por ejemplo, se tiene lo señalado por Gartner:

“Gartner afirma que el 72 % de los proyectos relacionados con integración de sistemas están relacionados con servicios Web. Los analistas afirman que la

experiencia de vendedores y proveedores de servicios externos (ESPs) será determinante para la adopción de servicios Web.

“El 95% de las empresas contempladas en el estudio, planean ocupar a su propio personal para implementar soluciones de servicios Web y el 47% de las mismas, piensan recurrir de manera adicional, a un proveedor externo de servicios”, comentó Benoit Lheureux, Director de Investigación de Gartner. “Por ello, la experiencia que tengan los vendedores de servicios profesionales de TI y los ESPs será un factor crítico en la adopción de servicios Web, ya que los usuarios finales necesitan de su capacidad para orientarlos en el camino hacia el uso de esta tecnología”.

Gartner afirma que a pesar de que los servicios Web son una tecnología que permite la reutilización de los recursos y agilidad en los sistemas, al tiempo que reduce el costo total de propiedad (TCO), su adopción más que considerarse una ventaja competitiva, será motivada precisamente por el deseo de reutilizar, consolidar y en última instancia, cambiar el paradigma del desarrollo, integración y administración de aplicaciones en las empresas.

Los estudios de Gartner permiten afirmar que a pesar de que los servicios Web son una tecnología desarrollada para promover el uso de ambientes compartidos, hasta el año 2007 la mayor parte de estos continuarán llevándose a cabo dentro de las cuatro paredes de las organizaciones, o en sistemas que involucran un solo ambiente empresarial.

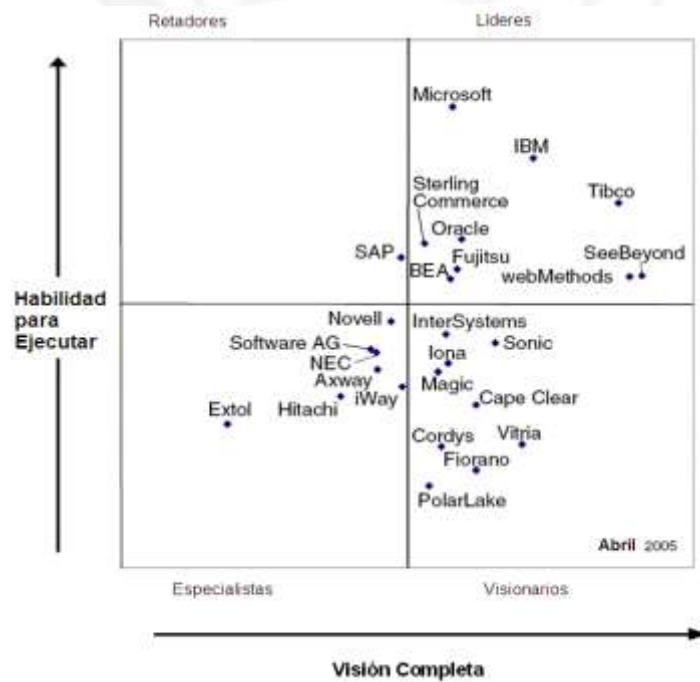
Gartner considera que las áreas de consultoría e integración de sistemas (SI) serán líderes en la adopción de servicios Web, y su consolidación ocurrirá dentro de los mercados de software e infraestructura. Como resultado, los ESPs deben desarrollar sus capacidades en servicios Web para dirigir las necesidades de las corporaciones hacia los nuevos paradigmas de desarrollo, integración y administración de aplicaciones.”

(Gartner, 2010, párr. 1-4, <http://www.ebizlatam.com/noticias/wmview.php?ArtID=1619>)

Finalmente, se considera pertinente presentar el siguiente cuadro, que si bien es de hace algunos años, nos muestra a los principales competidores, en el campo de Brokers de Integración.

Figura 2.1

Magic Quadrant for Integration Backbone Software, 1H05



Nota. De "Magic Quadrant for Integration Backbone Software, 1H05", por Gartner, 2005

CAPÍTULO III: MARCO TEÓRICO

3.1. Conceptos Principales

3.1.1. Integración Empresarial (Enterprise Integration):

Se refiere a la tarea de lograr que aplicaciones separadas funcionen juntas para producir un conjunto unificado de funcionalidades, ya sea que éstas sean desarrolladas a medida o desarrolladas por terceros como productos de software. Las aplicaciones en mención pueden ejecutarse en múltiples plataformas de software, sistema operativo y hardware. Asimismo, las aplicaciones pueden estar geográficamente dispersas, como es el caso, por ejemplo, de las aplicaciones de aliados de negocios, proveedores y clientes. Deben considerarse también, el caso de aplicaciones que no fueron diseñadas para ser integradas, que no pueden ser modificadas pero que aun así son necesarias de integrar.

3.1.2. Servicio Web (Web Service):

La definición de servicio web que nos da la organización W3C es:

“Un servicio web es una aplicación de software identificada por una URI (Uniform Resource Identifier o Identificador Único de Recurso), cuyas interfaces y relaciones son posibles de definir y descubrir mediante artefactos XML, y que soporta interacciones directas con otras aplicaciones utilizando mensajes basados en XML mediante protocolos de internet.” Extraído de : World Wide Web Consortium, 26 de Abril 2010, (www.w3c.org)

De lo expuesto, se extrae que un servicio web no está limitado a utilizar mensajes tipo SOAP utilizando el protocolo de transmisión HTTP. De ello que pueda afirmarse, lo siguiente:

- Un servicio web no requiere necesariamente SOAP.
- Un servicio web no requiere necesariamente utilizar HTTP, como protocolo de transmisión de datos.

- No considera necesariamente a UDDI como mecanismo para descubrir dichos servicios web.

Por tanto, los servicios web son componentes desarrollados usando tecnologías de principalmente 3 categorías:

- Una forma de descripción basada en XML (WSDL) (Ej.: SOAP).
- Un protocolo de mensajes XML (Ej.: SOAP).
- Un protocolo de transporte (Ej.: http, SMTP).

3.1.3. UDDI (Universal Description, Discovery and Integration):

Registro de servicios web, independiente de la plataforma, basado en XML para ser usado entre compañías como directorio de servicios mediante el internet. Esta solución libre está respaldada por OASIS como iniciativa para que las compañías publiquen y listen sus servicios, poder descubrirse unos a otros y definir como interactuar.

3.2. SOA - Arquitectura Orientada a Servicios:

(Service Oriented Architecture): Es una filosofía arquitectural de construcción de sistemas computacionales para crear procesos de negocios empaquetados como servicios. Asimismo, define el intercambio de datos mediante aplicaciones a través de servicios web y así participar de los procesos de negocio. Estas funciones son independientes de los sistemas operativos y lenguajes en que están desarrollados. SOA separa las funcionalidades en distintas unidades (Servicios) que pueden ser distribuidos en una red, pudiendo ser combinados y reutilizados para crear nuevas aplicaciones de negocio.

De lo expuesto, se colige que SOA es:

- Un estilo arquitectural IT y principios de diseño para el desarrollo de aplicaciones e integración de sistemas que considera componentes débilmente acoplados y altamente íter-operables.
- Una forma de desarrollar sistemas de software de tal manera que estos prevean servicios a usuarios finales, otros sistemas y otros servicios.

- Una evolución natural del modelo de programación orientada a objetos y enfoques orientados a datos. Al crear un sistema SOA los servicios serán implementados una o más tecnologías.
- La integración de aplicaciones y fuentes de datos mediante el intercambio de información basado en una semántica y vocabulario común para definir la estructura de este intercambio (interfaces).
- Un mapa o plano (“Blueprint”) de los sistemas orientados a servicios.

Sus objetivos son:

- Proveer servicios a usuarios, aplicaciones y otros servicios utilizando mensajes con semántica común y estándar.
- Describir los componentes de un sistema y la interacción entre los mismos.
- Promover la orquestación de negocios a un nivel de servicios empresariales utilizando un modelo distribuido.
- Lograr baja cohesión entre los sistemas y las entidades que interactúan.

Niveles de Adopción de SOA:

Según muchos expertos, podemos encontrar que las empresas pasan por distintos niveles o formas de adopción a las arquitecturas SOA. Dichas etapas y/o formas de adopción son:

Tabla 3.1

Niveles de Adopción SOA

Nombre	Descripción
Implementación de servicio Web Individuales	Crear servicios de tareas contenidas en aplicaciones nuevas o existentes.
Integración orientada a servicios	Integrar múltiples sistemas al interior y exterior de la empresa utilizando servicios con un propósito de negocio.
Cambio completo del Área de IT	Implementación de una arquitectura que permita la integración entre funciones de negocio dentro y fuera de la empresa.

Nombre	Descripción
Transformación en demanda	Transformación progresiva de los procesos actuales de negocio a nuevos procesos utilizando la arquitectura SOA.

Nota. Adaptado de “IBM Developer Works” por IBM, 2010 (<http://www.ibm.com/developerworks/>)

Formas de adopción SOA

Muchas empresas adoptan SOA utilizando, para ello, alguna de las siguientes tácticas de aproximaciones.

Tabla 3.2

Formas de Adopción SOA

Aproximación	Descripción	Tipo
Por procesos del negocio	Los procesos de negocio de la empresa necesitan acceder a muchas funcionalidades en distintos sistemas, además dichos procesos deben de poder cambiar fácilmente.	Arriba Abajo
Basado en la Herramienta	Modelar los procesos de negocio y luego dejar que las herramientas generen el detalle.	Arriba Abajo
Reutilizar la funcionalidad existente	Reutilizar o exponer funcionalidades de sistemas ya desarrollados de manera rápida y reutilizable.	Abajo a Arriba
Convertir a Componentes aplicaciones existentes	Descomponer los sistemas ya desarrollados en módulos utilizando herramientas basadas en compiladores.	Abajo a Arriba
Necesidad de información	Proveer acceso a la información utilizando servicios.	Orientado a Datos
Orientado a Mensajes	Necesidad de comunicar sistemas mediante un estándar y protocolos no propietarios.	SOI

Nota. Adaptado de “IBM Developer Works” por IBM, 2010 (<http://www.ibm.com/developerworks/>)

Adopciones de integración de sistemas SOA

Muchos de los autores coinciden en que las principales formas y etapas de adopción para integración utilizando SOA son:

Tabla 3.3*Niveles de Adopción para integración*

Contexto	Patrón	Aplicabilidad
Silo, funcionalidad concentrada	Hard-Code (no es un patrón si no un estado)	Bajo Riesgo, poco cambio, alta performance.
Distribuida, muchos puntos de acceso	Exposición punto a punto	Exponer la funcionalidad existente rápidamente, da resultados de manera rápida.
Exponer funcionalidades de sistemas "legacy" mediante servicios web	Service Adapter (adaptador de servicio)	El consumidor (de servicios) necesita acceder a alguna funcionalidad que no se encuentra expuesta como servicios.
Acceder a servicios utilizando un "proxy" si no es posible acceder directamente a la funcionalidad deseada.	Service Proxy	Provee una interfaz de servicios web.
Proveer flexibilidad en la elección del servicio	Remote Service Strategy	Proveer flexibilidad en cambiar el proveedor de servicios basado en la calidad del servicio o consideraciones funcionales.
Eliminar funcionalidad redundante, reagrupar, consolidar o reemplazar sistemas existentes	Single Point of Access	Provee un solo punto de acceso a un servicio con múltiples variantes.
Se requieren funcionalidades aun no expuestas como servicios	Virtual provider	
Un solo punto de acceso	Service Integrator	Direccionamiento y transformación.
Integración a nivel de empresa	Enterprise Service Bus	Mediación, direccionamiento, reglas de transformación, eventos.
El nirvana de SOA, reconfiguraciones dinámicas a través de servicios inteligentes	n/a	n/a

Nota. Adaptado de "IBM Developer Works" por IBM, 2010 (<http://www.ibm.com/developerworks/>)

3.3. Patrones:

En el libro: "A Timeless Way of Building", obra de referencia donde se plantea por primera vez la teoría de los patrones aplicada a la Arquitectura Civil y Urbanismo. El Arquitecto Christopher Alexander define al patrón en la siguiente manera:

"Cada patrón es una regla de 3 partes, que expresa una relación entre un contexto, un problema y una solución. Como un elemento en el mundo, cada patrón es una

relación entre un contexto, un sistema de fuerzas que ocurren repetidamente en ese contexto y una configuración espacial que permite que esas fuerzas se resuelvan entre sí.

(...)

Como elemento de un lenguaje, un patrón es una instrucción que muestra como puede ser usada esta configuración espacial una y otra vez para resolver el sistema de fuerzas, siempre que el contexto lo haga relevante.

(Alexander, 1979, *A Timeless Way of Building*, pp.123-124, Oxford University Press)

Continuando con la definición anterior, podemos agregar otro párrafo de Alexander, citado en la obra de referencia:

“Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma.”

(Alexander, 1979, *A Timeless Way of Building*, pp.100-100, Oxford University Press)

Para Alexander, estos patrones son ubicuos y permiten alcanzar la “calidad sin nombre” (Quality Without a Name ó QWAN). Para aclarar estas definiciones, podemos utilizar un ejemplo concreto aplicado a la construcción, tomado también de la obra de Alexander:

Si nos fijamos en las construcciones de una determinada zona rural, observaremos que todas ellas poseen apariencias parejas (tejados de pizarra con gran pendiente, etc.), pese a que los requisitos personales por fuerza han debido ser distintos. De alguna manera la esencia del diseño se ha copiado de una construcción a otra, y a esta esencia se pliegan de forma natural los diversos requisitos. Diríase aquí que

existe un patrón que soluciona de forma simple y efectiva los problemas de construcción en tal zona.”

(Alexander, 1979, A Timeless Way of Building, pp.100-100, Oxford University Press)

3.1. Anti-patrones:

Los anti-patrones consisten en soluciones utilizadas frecuentemente, pero altamente inefectivas y, en muchas ocasiones, con un efecto contraproducente. Estas soluciones se recopilan mostrando sus síntomas, causas principales y posibles soluciones.

Como muestra de los principales anti-patrones aceptados por la comunidad de desarrollo de software que pueden afectar el desarrollo de interfaces orientadas a servicios, se tienen, entre otros, los siguientes:

Tabla 3.4

Anti-patrones

Categoría	Nombre	Descripción
Diseño	Blob	Una clase con demasiados atributos la cual es “el corazón” del sistema.
Diseño	Poltergeists	Demasiadas clases y demasiada abstracción, clase que aparecen y desaparecen si ejecutar cambio alguno.
Estructural	Spaghetti Code	Código de programa sin estructura definida.
Estructural	Stovepipe Systems	Las aplicaciones están dispersas y aisladas.
Tecnología	Wolf Ticket	Una tecnología que dice ser abierta pero no tiene pruebas de serlo.
Tecnología	Continuos obsolecense	Intentar utilizar siempre la última versión.
Rehúso	Cut-and-paste	Errores de software duplicados.
Rehúso	Golden Hammer	Tratar de forzar los sistemas a una herramienta única.

Nota. Adaptado de “IBM Developer Works” por IBM, 2010 (<http://www.ibm.com/developerworks/>)

3.2. Arquitectura de Integración Empresarial (EIA-Enterprise Interation Architecture):

EIA es un término IT para los planes, métodos y herramientas destinados a modernizar y consolidar las aplicaciones computacionales en una empresa.

Típicamente una empresa tiene aplicaciones “heredadas”, bases de datos y desea continuar utilizándolas mientras se añaden nuevas funcionalidades como por ejemplo, para explotar los patrones de Internet, comercio electrónico, extranet entre otras nuevas tecnologías. EIA implica desarrollar una nueva visión del negocio de la empresa y por tanto, en la medida de que las aplicaciones actuales encajan en el, requiere diseñar formas de usar de manera eficiente lo que ya existe mientras que se añade nuevas aplicaciones e información.

EIA utiliza metodologías tales como la OO (Programación Orientada a Objetos), objetos distribuidos, comunicaciones entre plataformas utilizando diversos métodos de comunicación como gestores de colas y componentes COM+ y otros enfoques como SOA y SOL.

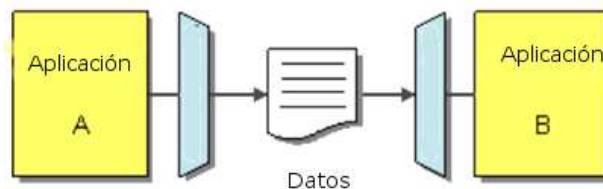
Tipos de integración

Según las teorías de EIA, los tipos de integración pueden clasificarse en los siguientes estilos:

- File Transfers: (Transferencia de Archivos): Intercambio de información mediante archivos, donde el nombre y la ubicación del mismo es predeterminado.

Figura 3.1

Integración mediante base de datos

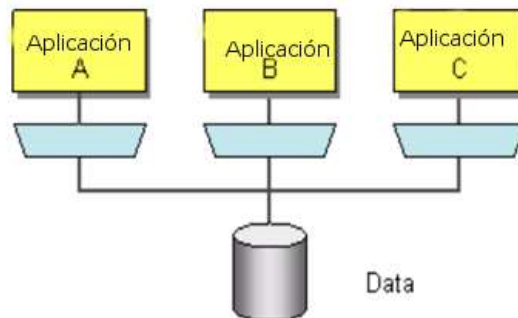


Nota. Extraído de EIA Patterns, por EIA Patterns, 2010 (<http://www.eaipatterns.com>)

- Shared Database: (Base de Datos Compartida), varias aplicaciones utilizan el mismo esquema (“schema”) de bases de datos. No hay necesidad de transferencia de datos entre dos repositorios distintos.

Figura 3.2

Integración mediante base de datos

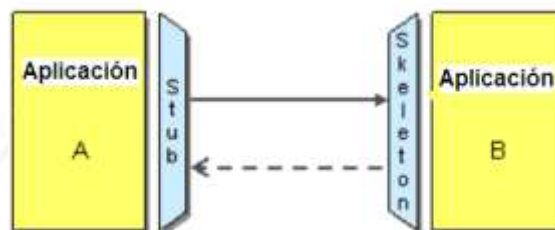


Nota. Extraído de EIA Patterns, por EIA Patterns , 2010 (<http://www.eaipatterns.com>)

- Remote Procedure Invocation: (Llamadas Remotas): Una aplicación expone una o varias de sus funcionalidades para que puede ser accedida remotamente desde otra aplicación. La comunicación ocurre en tiempo real.

Figura 3.3

Integración mediante llamadas remotas

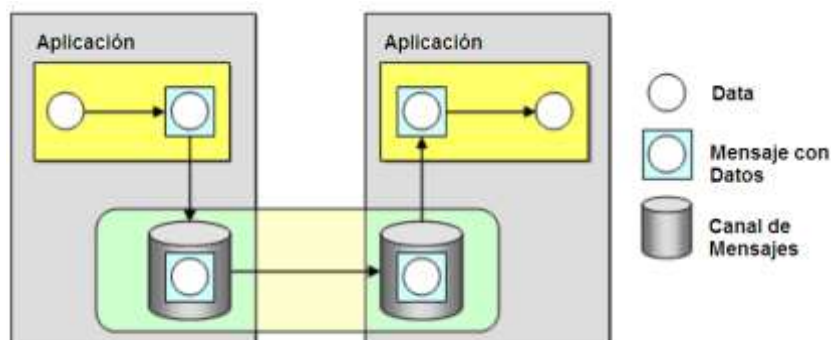


Nota. Extraído de EIA Patterns, por EIA Patterns, 2010 (<http://www.eaipatterns.com>)

- Mensajería (Messanging): Una aplicación pública expone un mensaje en un canal común para que otras aplicaciones puedan leerlo luego. Las aplicaciones deben de coincidir en el canal y el formato del mensaje.

Figura 3.4

Integración mediante mensajería



Nota. Extraído de EIA Patterns, por EIA Patterns, 2010 (<http://www.eaipatterns.com>)

Conceptos principales de EIA

Según las teorías de EIA, la más ventajosa forma de integrar sistemas es a través de los patrones relacionados a “messanging” (sistemas de mensajes).

Los conceptos más importantes de este tipo de integración son:

- **Messaging:** Es una tecnología que permite el intercambio rápido y asíncrono de grandes cantidades de datos en paquetes llamados mensajes entre aplicaciones de una manera segura y confiable.
- **Channels/Queues (Colas):** son rutas lógicas que conectan las aplicaciones y el estándar de mensajes. Se comporta como una colección o arreglo de mensajes usado y compartido en varios equipos (computadores) concurrentemente.
- **Sender/Producer (Emisor):** Es el programa que envía el mensaje escribiéndolo en el canal.
- **Reciber/Consumer (Receptor):** Es el que recibe el mensaje leyéndolo y eliminándolo del canal o cola.

Ventajas principales

Las ventajas principales de usar el esquema de integración a través de los patrones relacionados a “messanging” son:

- **Proveen Comunicación:** Permiten el transporte de datos entre diferentes aplicaciones separadas (en computadores distintos).
- **Platform/Lenguaje Independent (Independientes del lenguaje y plataforma):** Al conectar distintos sistemas es posible estos utilicen distintas tecnologías, lenguajes de programación, sistemas operativos, etc. En este escenario el SW de integración puede ser el “traductor” universal entre aplicaciones.
- **Asynchronous Communication (Comunicación Asíncrona):** Permite un tipo de comunicación en el que el emisor puede dejar el mensaje y no esperar una respuesta.
- **Variable Timming (Tiempo Variable) :** En la comunicación sincrónica el emisor tiene que esperar a que el receptor termine de procesar el mensaje, en la comunicación asíncrona ambos sistemas podrán trabajar a su conveniencia y propia capacidad no haciendo “perder” el tiempo a la otra mientras una espera a la otra.
- **Throttling (regulación):** Un problema en la comunicación sincrónica es la concurrencia de mensajes lo cual puede degradar la performance del receptor e inclusive causar la caída del mismo.
- **Reliable Communication (Comunicación Confiable):** Este método provee la confiabilidad que una integración RPC (Remote Procedure Call) no puede dar, ya que los mensajes son transmitidos como unidades atómicas.
- **Disconnected Operation (Operaciones fuera de línea):** En aplicaciones diseñadas para trabajar fuera de línea o conectadas solo por lapsos de tiempo (PDA’s Laptops), este mecanismo sirve para sincronizar los mensajes enviados fuera de línea en el lapso que la comunicación esté disponible.
- **Mediación:** Este tipo de sistema de integración puede actuar como mediador de varios sistemas, de manera tal que si un sistema se desconecta o deja de funcionar al re-iniciarse solo tendrá que conectarse al sistema de integración y no a todas las aplicaciones.

- Thread Management (Manejo de Procesos): En una comunicación sincrónica pueden generarse muchos procesos inactivos esperando una respuesta lo cual puede afectar negativamente la performance. En una comunicación mediante mensajes y colas no existe la necesidad de esperar una respuesta por lo cual esto se minimiza.

Retos de Implementación relacionados a EIA

Como principales retos de este tipo de Integración de sistemas tenemos:

- Modelo de Programación Complejo: La comunicación asíncrona de mensajes en donde la lógica está dividida en varios componentes y la programación es orientada a eventos tiene un nivel de complejidad más alto en donde deben de manejarse como por ejemplo colas, correlativos de mensaje entre otras.
- Secuencia: Este tipo de “MiddleWare” o sistema de integración garantiza que los mensajes llegaran a su destino pero no cuando ni el orden de los mismos, en situaciones donde un mensaje dependa del otro hay que considerar la manera de re establecer la secuencia de los mismos.
- Escenario Síncrono: En algunos casos es necesaria una respuesta aunque esta sea parcial o solo para confirmar que el mensaje fue recibido y está siendo procesado. Como puede ser el caso de una aplicación de reserva de tickets, en donde se debe de saber que la reserva fue recibida.
- Performance: Este tipo de integraciones suelen añadir información adicional sobre los datos con la finalidad de manejar los canales colas y correlación de mensajes así como el procesamiento necesario para la sincronización de datos los cuales llevan a un aumento en el procesamiento a comparación de una transmisión de datos sincrónica.
- Limited Platform Support (Soporte Limitado de Plataformas): Al ser propietarios muchas plataformas no están soportadas siendo muchas veces más fácil usar transferencia de archivos mediante FTP.

- Vendor lock-in (Plataformas Incompatibles): Muchos productos de Integración utilizan protocolos de mensajes propietarios incompatibles entre sí, lo cual muchas veces obliga a tener que integrar soluciones de integración entre sí.

3.3. EIA y SOI

Muchos de los lineamientos, patrones y anti-patrones de SOI están basados en EIA y por tanto podemos decir que EIA es el fundamento para SOI.

En su mayoría los patrones de EIA son extensibles a usar XML y servicios web, es decir pueden ser aplicados en integraciones orientadas a servicios, como por ejemplo utilizando XML para los mensajes y servicios web para exponer las interfaces a ser llamadas, así como para la transformación y enrutamiento de los mismos.

Si consideramos que los mensajes pueden envarse utilizando XML mediante un protocolo y una descripción del mensaje (Ej: SOAP). Y además utilizamos el bus de integración como protocolo de comunicación, los conceptos y patrones de EIA son aplicables SOI.

Sin embargo debemos de evaluar los principios de arquitectura de SOI y SOA para utilizar dichos patrones de manera coherente.

CAPÍTULO IV: DESARROLLO DEL TRABAJO DE INVESTIGACIÓN

4.1 Metodología

La presente investigación se divide en dos secciones principales:

a) Catálogo de Patrones y Anti-patrones

Para la creación de un catálogo de patrones y antipatrones SOI, se utilizó la siguiente metodología.

- Investigación y recopilación de información.
- Elaboración de un catálogo de patrones y antipatrones.

b) Evaluación de Beneficios

En esta etapa se llevó a cabo la integración de dos sistemas bajo el escenario propuesto utilizando la siguiente metodología, la cual está basada en el modelo clásico de desarrollo de software.

Al ejecutar una tras otra, se conforma las siguientes etapas, las cuales conforman la metodología empleada:

4.2.1 Requerimientos:

Definir los Requerimientos de Integración: En esta etapa se identificarán los requerimientos de integración de una manera macro. El objetivo es identificar:

- Sistemas a Integrar: Identificar los sistemas a Integrar de acuerdo a los requerimientos.
- Requerimientos Funcionales: Identificar las funcionalidades candidatas a requerir integración con otros sistemas, dichas funcionalidades se puede indentificar mediante Casos de Uso. Es así que es posible elaborar para este fin el diagrama de casos de uso de dichas funcionalidades candidatas, ó simplemente identificarlas y/o ubicarlas de los diagramas ya existentes.

- **Requerimientos No Funcionales:** Se deben de identificar los requerimientos tales como performance, escalabilidad, disponibilidad, mantenibilidad y seguridad.

Entregables de Requerimientos:

- Documento de Requerimientos.

4.2.2 Análisis:

Delinear la Arquitectura de Integración: En esta etapa se debe delinear la arquitectura de Integración en su primera versión, tomando en cuenta los requerimientos y la plataforma actual de Integración, es decir especificar la arquitectura.

En esta etapa se debe elaborar un catálogo de interfaces a partir de los casos candidatos a requerir interfaces. Dicho catalogo debe contener al menos el nombre y/o código de la interfase y los sistemas involucrados. Para ello es posible generar los siguientes documentos y diagramas:

- **Diagrama de Arquitectura:** El cual muestra los nodos, componentes conectores e interfaces.
- **Diagrama de Interfaces:** Mostrando las interfaces identificadas en esta etapa.
- **Catálogo de Interfaces:** Breve descripción de cada interfaz.

Definir los requerimientos de Transformación de Datos: En esta etapa se deben de identificar los requerimientos de transformación de datos que se requiere para las interfaces, aquí debemos identificar lo siguiente:

- **Identificar los Requerimientos de Replicación y Sincronización de Datos:** Consiste en ver la necesidad de replicar tablas y/o bases de datos entre distintos sistemas y/o analizar la posibilidad de eliminar esta redundancia mediante el uso de herramientas de software o modificando los sistemas, de ser necesario o conveniente la adquisición de nuevo software se recomienda hacer un estudio de costos y beneficios.
- **Identificar el Impacto de la Consolidación y Limpieza de Datos:** En muchos casos existen Bases de datos redundantes las cuales pueden llegar a ser consolidadas, en

este caso es necesario elaborar un plan de Aseguramiento de Calidad de la Información, la limpieza de datos debe de ser manejada como un proyecto separado y debe de estar alineado a el proyecto de integración.

- Identificar los posibles requerimientos de Llaves primarias de referencia cruzada: Aquí debemos definir los posibles campos candidatos que servirán de llaves primarias entre los distintos sistemas para identificar datos específicos o comunes.

Entregables del Análisis

- Documento de Análisis.
- Documento de Arquitectura (Inicial).

4.2.3 Diseño

Diseñar las Integraciones y Procesos: Aquí debemos diseñar las interfaces concentrándonos en el flujo de información considerando llamadas en línea o fuera de línea, se debe analizar cada flujo por cada escenario de negocio. Aquí podemos utilizar las siguientes herramientas, según sea adecuado:

- Diagramas de Secuencia o Actividad UML: Es posible utilizar alguno u ambos diagramas para definir cuál será el flujo de los mensajes y los componentes involucrados.
- Identificar los Patrones de Integración Adecuados.

Diseñar las Transformaciones de Datos: Una vez diseñados los flujos e iteraciones es posible diseñar las transformaciones necesarias ente los sistemas, estos pueden ir desde su representación binaria (EBCDIC, ASCII, ETC), formato (XML, TEXTO, ETC) hasta cambios de moneda, idioma, código, entre otros. También se debe de definir que transformaciones se realizaran en el bus de integración (si es el caso) o en las mismas interfaces.

Definir la Arquitectura de Integración: En esta etapa se debe definir cuál será la arquitectura de integración más adecuada de acuerdo a los requerimientos reunidos. Se recomienda realizar lo siguiente:

- Realizar el Mapeo de Productos de Software: Se debe de elegir los productos de software adecuados de acuerdo a la plataforma actual y requerimientos del cliente entre otros factores (económicos, conocimientos, planes a largo plazo).
- Plan de Instalación y Configuración: Se debe elaborar un plan de instalación y configuración de la plataforma de software elegida, considerando las nuevas adquisiciones y las capacidades requeridas.
- Definir el Modelo de Componentes del Sistema: Es necesario realizar un diagrama donde se muestren los componentes de software y como estos interactuaran entre si. (Diagrama de Componentes y Emplazamiento UML).

Entregables del Diseño

- Documento de Arquitectura (Final).
- Documento de Diseño.

4.2.4 Desarrollo

Construcción: Se construirán los componentes de código.

Codificación y Pruebas Unitarias: Se creará el código fuente para las interfaces identificadas en los sistemas involucrados y se realizarán las pruebas unitarias.

Configuración de los componentes de Integración: Es necesario configurar los componentes de la arquitectura de integración definida tales como buses de integración, redes y adaptadores contemplados.

Entregables del Desarrollo:

- Código Fuente.

4.2.5 Pruebas

Pruebas y Mejoras: Esto se refiere a las pruebas propiamente dichas, aquí se debe asegurar que la interfaz hace lo identificado y realizar las mejoras correspondientes de ser identificadas. Para la ejecución de dichas pruebas es necesario el diseño, plan de ejecución y casos de prueba correspondientes.

Entregables de las Pruebas:

- Plan de Pruebas
- Diseño de Pruebas
- Casos de Prueba
- Informe de Pruebas

4.2.6 Evaluación

Luego de realizada la prueba de concepto se evaluaron los siguientes puntos para verificar la obtención de los beneficios esperados. A continuación se muestra un gráfico mostrando las fases, actividades y las interrelaciones entre las actividades.

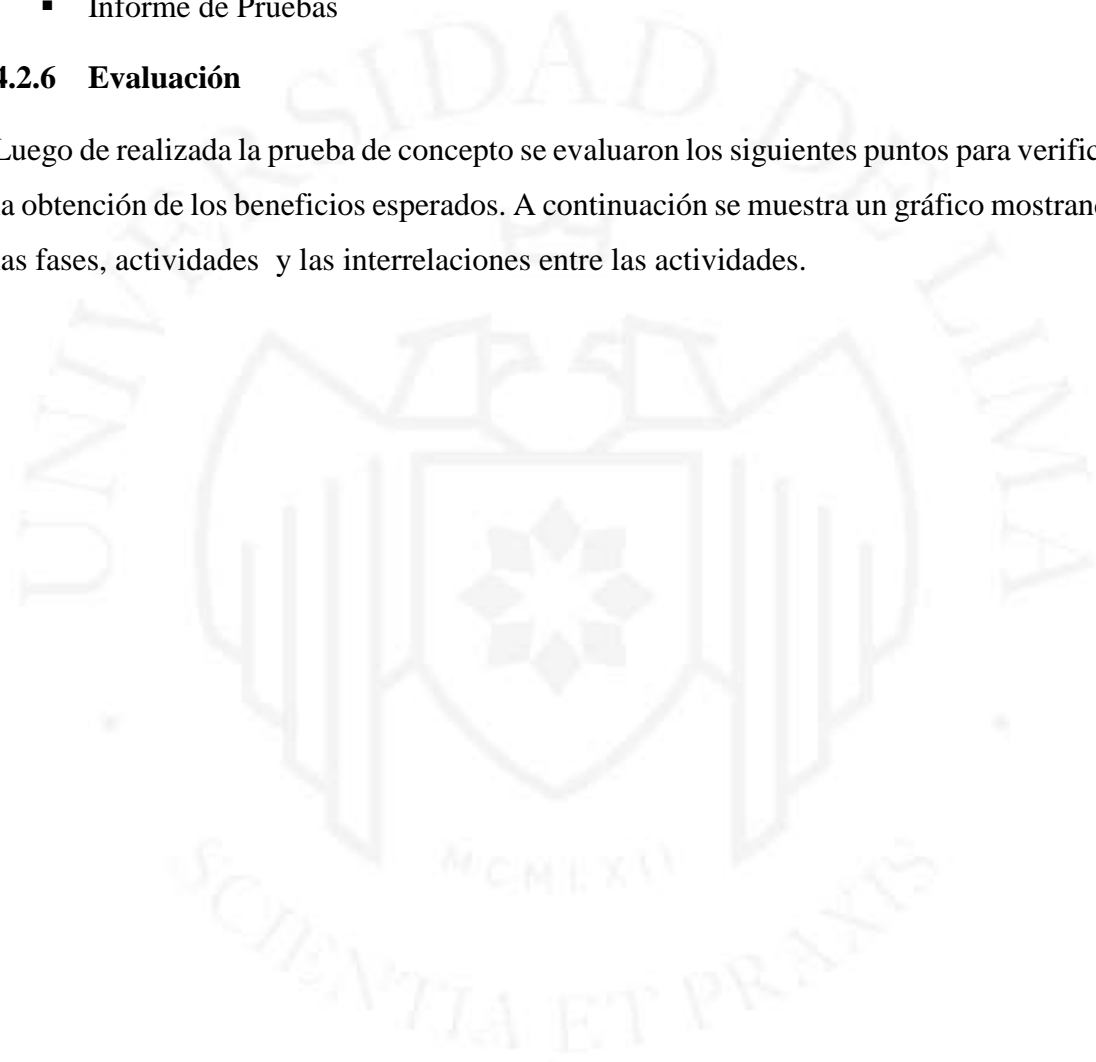
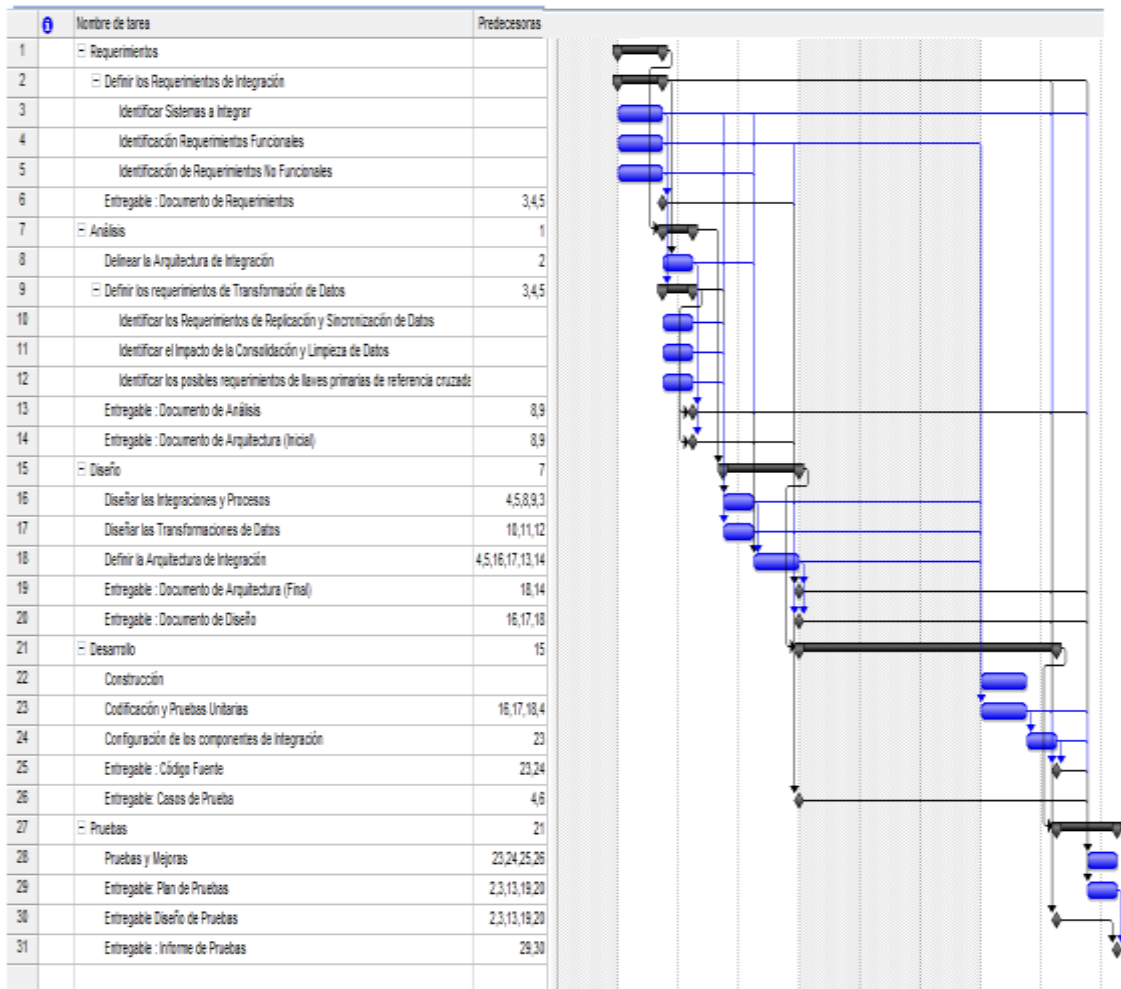


Figura 4.1

Metodología



4.2 Desarrollo

4.2.1 Catálogo de Patrones y Anti-Patrones SOI

Los patrones SOA así como todos los patrones, están basados en principios de diseño centrales que a continuación listamos los lineamientos principales que las principales fuentes de información (investigadores y grandes casas de software) delinear.

Principios de Diseño:

- a) Contrato Estandarizado de Servicios (Standardized Service Contract): Los servicios deben tener un mismo inventario (directorío) para asegurar que su “contrato” sea estándar, este principio también es conocido como “Boundaries are Explicit”.
- b) Baja cohesión entre servicios (Service Loose Coupling): Los “contratos” entre servicios debe impulsar una baja cohesión entre ellos y el entorno.
- c) Abstracción de servicios (Service Abstraction): Los “contratos” entre servicios solo contienen información esencial y limitada a que es lo publicado en el servicio.
- d) Re utilización de servicios (Service Reusability): Los servicios deben tener lógica aislada e independiente para ser reutilizables como recursos de la empresa.
- e) Autonomía de Servicios (Service Autonomy): Los servicios deben ser autónomos para ejecutarse sobre su plataforma.
- f) Servicios sin estado (Service Statelessness): Los servicios no deben manejar estados salvo cuando sea necesario.
- g) Deben de poder ser descubiertos (Service Discoverability): Los servicios deben poder ser descubiertos e interpretados dinámicamente mediante la meta data de un registro o directorío.
- h) Composición de Servicios (Service Composability): Los servicios pueden ser composiciones de varios participantes no importa la complejidad de estos.

4.2.2 Patrones SOI

Los principales patrones de implementación de servicios, encontrados y catalogados a lo largo de esta investigación son los siguientes:

- a) Sub Controlador Agnóstico (Agnostic Sub Controller)
- b) Recursos Canónicos (Canonical Resources)
- c) Reutilización de componentes entre dominios
- d) Transacciones entre múltiples servicios

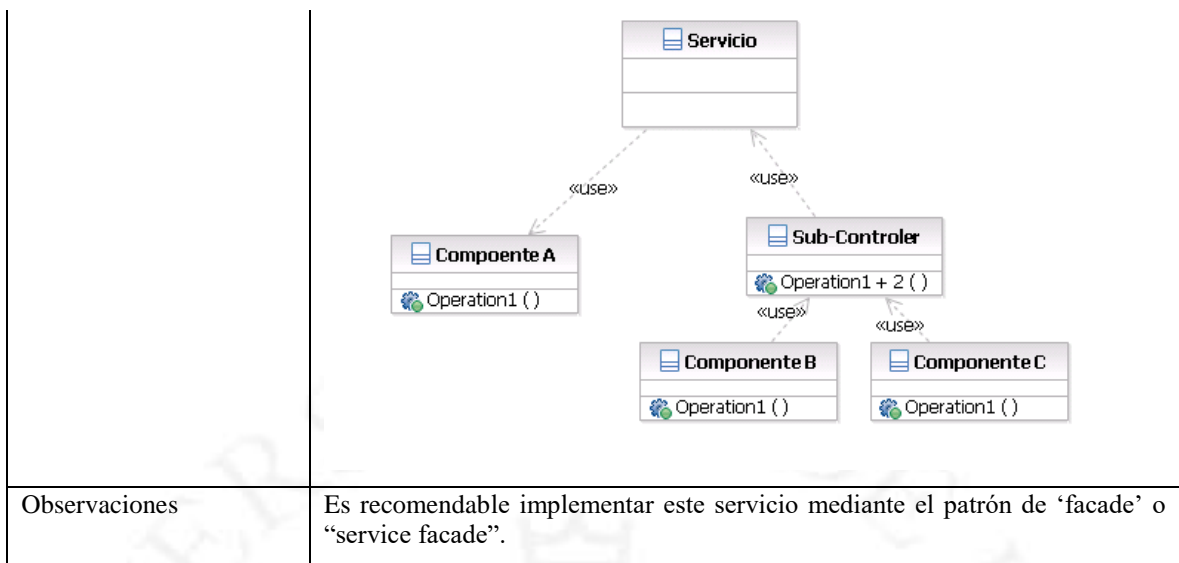
- e) Mensajes enviados por eventos
- f) Ruteo dinámico
- g) Proveedor Virtual
- h) Bus de Integración
- i) Bus de Integración como “Gateway”
- j) Procesador de documentos
- k) Mensaje inalterable
- l) Business Layer Integration

A continuación se presentan tablas resumen de cada uno de ellos:

Tabla 4.1

Sub-Controlador Agnóstico

Patrón	Sub Controlador Agnóstico (Agnostic Sub Controller)
Problema	Al crear servicios estos usualmente están compuestos de llamadas a varios componentes específicos y es muy posible esto evite la re utilización de sub llamadas.
Solución	Implementar sub-controladores “agnósticos” para re-utilizarlos en varios servicios.
Gráfico	<p>Figura 4.2 <i>Diagrama de Clases Sub Controlador Agnóstico</i></p> <pre> classDiagram class Servicio class ComponenteA { Operation1() } class ComponenteB { Operation1() } class ComponenteC { Operation1() } Servicio ..> ComponenteA : «use» Servicio ..> ComponenteB : «use» Servicio ..> ComponenteC : «use» </pre>



Nota. Adaptado de "SOA Patterns" por SOA Patterns, 2010 (<http://www.soapatterns.org>)

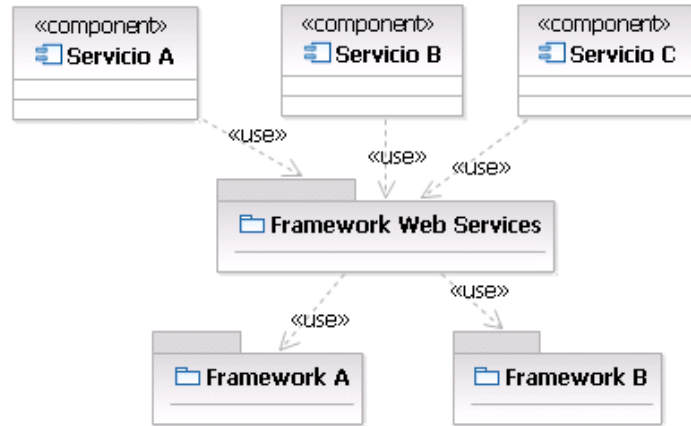
Tabla 4.2

Recursos Canónicos (Canonical Resources)

Patrón	Recursos Canónicos (Canonical Resources):
Problema	La implementación de servicios puede ser soportada por múltiples recursos y artefactos de implementación, lo cual lleva al riesgo de múltiples implementaciones distintas e innecesarias.
Solución	La arquitectura y la infraestructura que soporta la implementación de servicios web debe de tener recursos comunes y artefactos reutilizables en la implementación de servicios.
Gráfico	

Figura 4.3

Diagrama de Clases Recursos Canónicos



Observaciones	Es así que se requiere la formalización de los componentes arquitecturales que conforman la plataforma y asegurar que las implementaciones de los servicios sigan los patrones establecidos.
---------------	--

Nota. Adaptado de "SOA Patterns" por SOA Patterns, 2010 (<http://www.soapatterns.org>)

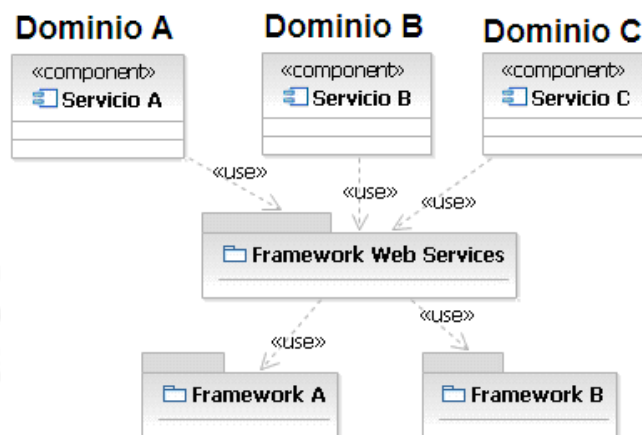
Tabla 4.3

Reutilización de componentes entre dominios

Patrón	Reutilización de componentes entre dominios
Problema	<p>Muchas empresas implementan distintas infraestructuras (dominios), para poder administrar y controlar cada uno independientemente del otro. Estos dominios usualmente están relacionados a áreas del negocio.</p> <p>Muchas veces debido a esta misma independencia las implementaciones suelen ser distintas, muchas veces más de un dominio implementa artefactos (objetos) que otra también implementa (capa de utilidades o “service utility layer”). Creando re trabajo y dificultad ante los cambios en dichas entidades.</p>
Solución	La implementación de artefactos (objetos) comunes deben de ser creados para que todos los dominios los utilicen. Esto facilita el cambio y la implementación de los servicios en los distintos dominios.
Gráfico	

Figura 4.0.4

Reutilización entre dominios



Observaciones

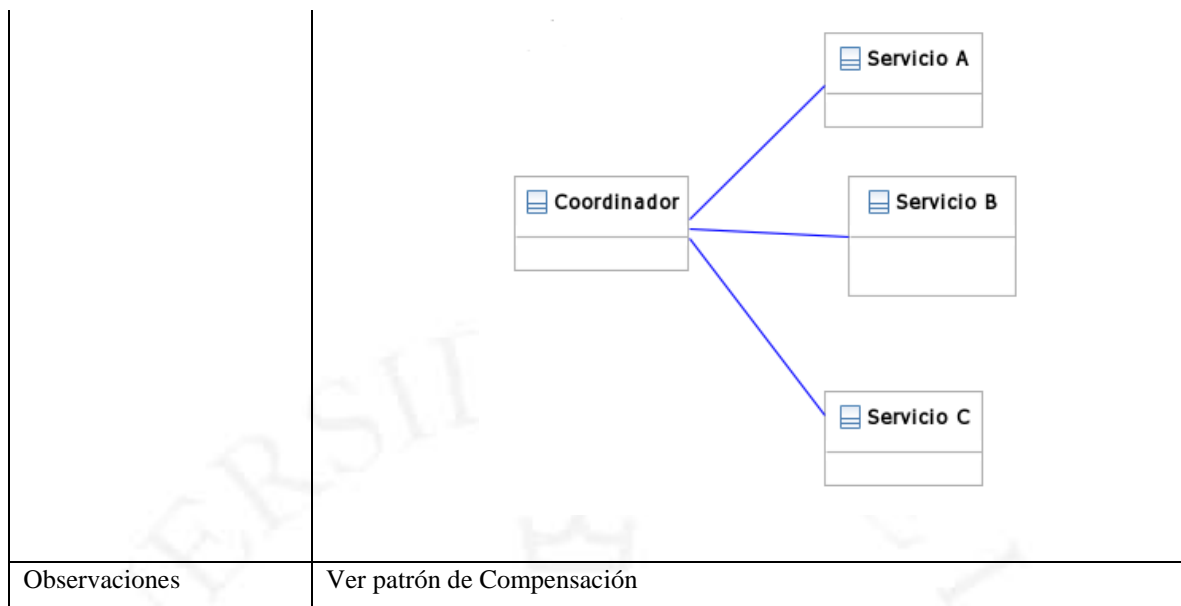
Es así que se requiere la formalización de los componentes arquitecturales que conforman la plataforma y asegurar que las implementaciones de los servicios sigan los patrones establecidos.

Nota. Adaptado de "SOA Patterns" por SOA Patterns, 2010 (<http://www.soapatterns.org>)

Tabla 4.4

Transacciones entre múltiples servicios

Patrón	Transacciones entre múltiples servicios (Cross-Service Transacción)
Problema	Muchas veces una operación necesita manejarse como atómica en múltiples plataformas. Pudiendo manejar que si ocurriera algún error o excepción en alguno de los servicios, realizar una operación de “rollback” en los otros servicios.
Solución	Registrar todos los servicios como parte de una sola transacción (muchos buses de integración implementan sistemas transaccionales para este fin), dichos servicios deberán de poder comunicar el estado de una transacción al sistema. Así mismo deberá de brindar la capacidad de anular una transacción específica (Rollback) además de realizar “commit” (aceptar cambios) a una transacción determinada.
Gráfico	<p>Figura 4.5</p> <p><i>Cross-Service Transaction</i></p>



Nota. Adaptado de: "SOA Patterns" por SOA Patterns, 2010 (<http://www.soapatterns.org>)

Tabla 4.5

Mensajes enviados por eventos

Patrón	Mensajes enviados por eventos (Event Driven Mensaje)
Problema	Usualmente los consumidores de un servicio necesitan saber cuando ocurre un evento relacionado con el proveedor. Muchas veces se opta por que el consumidor del servicio pregunte sobre dicho evento al proveedor. Este método puede ser ineficiente ya que requiere de muchas llamadas e intercambio de información muchas veces innecesaria. Además del retraso ocasionado por el tiempo entre consultas que realiza el consumidor.
Solución	Utilizar un programa de manejo de eventos (event manager) el cual permita al consumidor o consumidores estar suscrito al evento del servicio proveedor (publicador), el servicio proveedor puede definir muchos tipos de eventos y el suscriptor decide a qué tipo de evento suscribirse.
Gráfico	N/A
Observaciones	La mayoría de buses de integración implementan este mecanismo.

Nota. Adaptado de "SOA Patterns" por SOA Patterns, 2010 (<http://www.soapatterns.org>)

Tabla 4.6

Ruteo dinámico

Patrón	Ruteo dinámico
Problema	Muchas veces el destino del mensaje es decidido por el mismo servicio, esto permite la ejecución de rutas predeterminadas, sin embargo puede haber

	<p>factores no previstos que causen fallas en el correcto funcionamiento de las interfaces, como, por ejemplo:</p> <ul style="list-style-type: none"> ▪ El destino final de un mensaje esta fuera de línea temporal o permanentemente. ▪ La ruta embebida en el servicio contiene un manejo de excepciones incorrecto el cual causa que el mensaje sea enviado pero su destino sea incorrecto. <p>La entrega del mensaje no puede ser llevada a cabo resultando en un rechazo del mensaje.</p>
Solución	<p>Implementar la lógica de ruteo de manera genérica y separada mediante agentes y eventos en el bus de integración.</p> <p>En muchos casos es posible implementar el ruteo mediante el contenido del mensaje utilizando tecnologías como XQuery o XPath.</p>
Gráfico	<p>Figura 4.6</p> <p><i>Cross-Service Transaction</i></p> <p>El diagrama muestra tres componentes rectangulares con un icono de lista a la izquierda: 'Mensaje' en la parte superior izquierda, 'Reglas Ruteo' en la parte inferior izquierda, y 'Bus' en la parte central derecha. Una línea azul conecta 'Mensaje' con 'Bus', y otra línea azul conecta 'Reglas Ruteo' con 'Bus', indicando que tanto el mensaje como las reglas de ruteo interactúan con el bus.</p>
Observaciones	Ninguna.

Nota. Adaptado de "SOA Patterns" por SOA Patterns, 2010 (<http://www.soapatterns.org>)

Tabla 4.7

Proveedor Virtual

Patrón	Proveedor Virtual (Virtual Provider Pattern)
Problema	Se desea obtener un nuevo servicio a partir de componentes o servicios ya desarrollados, además es posible que los componentes actuales no provean el servicio como es requerido (tiempo de respuesta, funcionalidad diferente, calidad de servicio).
Solución	Crear un proveedor de servicios virtual especificando el servicio requerido y asumiendo que este será proveído por los componentes actuales, además seguir las siguientes recomendaciones:

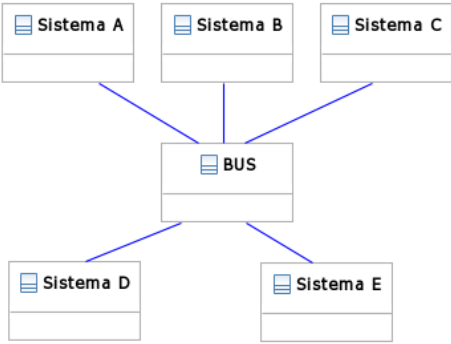
	<ul style="list-style-type: none"> ✓ Utilizar un “proxy” para comunicarse con el sistema “legacy”. ✓ Utilizar un adaptador para transformar el protocolo del sistema actual al que se necesita en la descripción del servicio. ✓ Encapsular el “proxy” y el adaptador en un patrón de “facade”. <p>Utilizar un patrón de “facade” para encapsular los adaptadores que permiten la comunicación con los componentes del sistema actual (Legacy).</p>
Gráfico	<p>Figura 4.7 <i>Proveedor Virtual</i></p> <pre> graph LR Consumidor --> Proxy Proxy --> Facade Facade --> Adaptador Adaptador --> Legacy </pre>
Observaciones	Ninguna.

Nota. Adaptado de "IBM Developer Works", por IBM, 2010 (<http://www.ibm.com/developerworks/webservices/library/ws-soa-soi/>)

Tabla 4.8

Bus de integración

Patrón	Bus de Integración
Problema	<ul style="list-style-type: none"> ▪ Necesidad de integrar un gran número de sistemas e interacciones. ▪ Necesidad de proveer funcionalidades avanzadas de integración como: consistencia transaccional, transformación, seguridad, calidad de servicio. ▪ Necesidad de distintos tipos de mensajes como sincrónicos, publicadores, subscriptores y eventos. ▪ Necesidad de una infraestructura robusta de integración integrada a servicios. ▪ Necesidad de integrar múltiples protocolos y procesamiento (transformación) de mensajes.
Solución	<ul style="list-style-type: none"> ▪ Implementar un Bus de Integración con las capacidades necesarias y con soporte a las tecnologías (protocolos) necesarios en la empresa como pueden ser : <ul style="list-style-type: none"> ▪ EIA / Tecnología de mensajes. ▪ Productos ESB.

Gráfico	<p>Figura 4.8 <i>Bus de integración</i></p> 
Observaciones	La mayoría de fabricantes implementa este patrón mediante productos de software.

Nota. Adaptado de "Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6", por IBM , 2010 (<http://www.redbooks.ibm.com/abstracts/sg246494.html>)

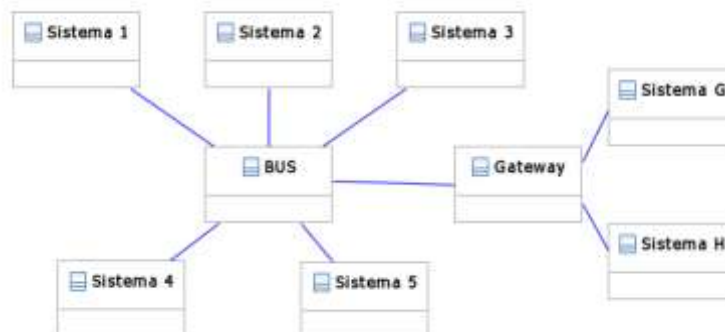
Tabla 4.9

Bus de integración como "Gateway"

Patrón	Bus de Integración como "Gateway"
Problema	<p>Este patrón es muy parecido al de Bus de Integración y tiende a utilizarse para definir límites entre empresas o unidades de una misma empresa o reforzar la seguridad y capacidad de aprovisionamiento.</p> <ul style="list-style-type: none"> ▪ Manejar múltiples interacciones entre servicios. ▪ Remover las integraciones tipo "spaguetti" punto a punto. ▪ Proveer soporte avanzado para integraciones de sistemas. ▪ Proveer límites entre áreas y administraciones distintas.
Solución	<p>Implementar un Bus de Integración con las capacidades necesarias y con soporte a las tecnologías (protocolos) necesarios en la empresa como pueden ser:</p> <ul style="list-style-type: none"> ▪ EIA / Tecnología de mensajes. ▪ Productos ESB. ▪ Tecnologías de Gateway.
Gráfico	

Figura 4.9

Bus de integración como "Gateway"



Observaciones

Ninguna.

Nota. Adaptado de "DeveloperWorks WebSphere Technical library IBM SOA Foundation product integration", por IBM, 2010 (http://www.ibm.com/developerworks/websphere/techjournal/0812_tellado/0812_tellado.html) y "SOA and the Art of Riding the Enterprise Service Bus, Part II", por IBM, 2010 (<http://soa.sys-con.com/node/46559>).

Tabla 4.10

Procesador de documentos

Patrón	Procesador de documentos
Problema	¿Cómo se crea un contrato bien definido y fácil de utilizar que sea compatible con los principios de diseño orientado a servicios?.
Solución	<p>Defina o vuelva a utilizar un esquema XML para representar mensajes de solicitud y respuesta para el servicio. Compruebe que todas las interacciones públicas con el servicio emplean estos esquemas.</p> <p>Genere objetos directamente desde los esquemas para agilizar el desarrollo. A este proceso se suele hacer referencia como un enfoque de "contrato como primer paso". El contrato se define antes del desarrollo del servicio actual. El contrato se puede utilizar posteriormente para generar el código del servicio. El enfoque de "contrato como primer paso" resulta eficaz a la hora de eliminar barreras en la interoperabilidad, ya que se basa en décadas de experiencia (CORBA, COM y DCE utilizaban lenguajes de interfaz). Los servicios Web suelen adoptar un enfoque de "contrato como último paso", ya que a menudo SOAP se utiliza sin WSDL para soluciones simples. No obstante, muchos entornos de desarrollo admiten el "contrato como primer paso", mientras que herramientas como WSCF y el generador de código de objetos XSD se encuentran disponibles para ayudar a automatizar este proceso.</p> <p>Como se mencionó con anterioridad, los proveedores de servicios no pueden esperar que los usuarios llamen o utilicen su servicio de forma específica. Esto significa que el uso de un proceso de compensación para una transacción determinada resulta aceptable, pero el proveedor no debe confiar en que el</p>

	usuario del servicio desencadene la compensación. Los patrones de reserva ofrecen la protección más autónoma para las transacciones, y elimina la dependencia del usuario del servicio.
Gráfico	N/A.
Observaciones	Los patrones de integración sugeridos por Microsoft tienen una fuerte inclinación hacia su plataforma y capacidades de la misma. Sin embargo esto no impide la implementación del patrón en otra plataforma.

Nota. Adaptado de "Principios de diseño de servicios: patrones y antipatrones de servicios", por Microsoft, 2010 (<http://msdn.microsoft.com/es-es/library/ms954638.aspx>)

Tabla 4.11

Mensaje inalterable

Patrón	Mensaje inalterable
Problema	¿Cómo se puede garantizar que los mensajes sean inalterables?.
Solución	<p>El contrato de servicio (esquema) debe indicar que los mensajes del usuario se etiqueten con un identificador de unidad de trabajo (al que se hará referencia como Id. de UDT).</p> <p>El contrato no puede requerir que el Id. de UDT sea siempre exclusivo.</p> <p>El identificador representará una unidad de trabajo que sólo se realizará una vez, independientemente del número de mensajes que se reciban con el mismo valor de Id. de UDT.</p> <p>El servicio utilizará el identificador para determinar si una unidad ya se ha utilizado antes de iniciarla. Existen tres opciones posibles para administrar los identificadores asociados con trabajo ya realizado.</p> <p>Devolver una copia almacenada en caché de la respuesta.</p> <p>Procesar el mensaje de nuevo como si la primera solicitud nunca se hubiera recibido.</p> <p>Producir una excepción y devolver un error.</p>
Gráfico	N/A
Observaciones	Debe de observarse que los patrones de Mensajería de EIA son aplicables.

Nota. Adaptado de "Messaging Patterns in Service-Oriented Architecture, Part 1" por Microsoft , 2010 (<http://msdn.microsoft.com/en-us/library/aa480027.aspx>)

Tabla 4.12

Integración desde capa de negocio

Patrón	Integración desde capa de negocio (Business Layer Integration)
Problema	¿Cómo integrar aplicaciones desde la capa de negocio?

Solución	<p>Integrar aplicaciones desde la capa de negocio permite a los sistemas consumir y exponer sus interfaces como servicios web y utilizar un esquema WSDL (Web Service Definition Language) como contrato y describir las interfaces entre ambos. Para asegurar la interoperabilidad entre ambas es necesario hacer la implementación de dichos servicios de acuerdo a las especificaciones de de WS-I (<www.ws-i.org> 26 de Abril 2010). La opción más estándar y aceptada para esto es usar SOAP y el estilo del mensaje como document-literal.</p> <p>Una manera de exponer la funcionalidad existente como un servicio web es usar el patrón “Service Interfaz”. Para encapsular la lógica necesaria para consumir servicios web es recomendable utilizar el patrón “Service Gateway”.</p>
Gráfico	N/A.
Observaciones	El nombre del patrón fue creado para su catalogación.

Nota. Adaptado de “Extensibility Points” por Microsoft , 2010 (<http://msdn.microsoft.com/en-us/library/aa480027.aspx>)

Anti-patronos SOI

A su vez durante la investigación catalogamos los siguientes anti-patronos:

- a) Tren Tecnológico
- b) ¿Qué hay de Nuevo?
- c) El Big Bang
- d) Servicio Web = SOA
- e) El enfoque Silo (The Silo Approach)
- f) Registros de mal comportamiento
- g) Servicios Parlanchines (Chatty Services)
- h) Servicios punto a punto
- i) Servicios sin componentes

A continuación se presentan tablas resumen de cada uno de ellos.

Tabla 4.13

Tren Tecnológico

Anti-Patrón	Tren Tecnológico (Technology Bandwagon)
-------------	---

Problema	Saltar a una nueva tecnología sin haber evaluado si realmente tiene impacto sobre el negocio, muchas veces puede ser exitosa pero no se puede medir el impacto en el negocio ya que no fue considerado desde el comienzo.
Síntomas	Imposibilidad de articular y presentar el valor del proyecto de TI dentro de la empresa. También la falta de alineamiento de los proyectos de IT al negocio puede ser un síntoma.
Consecuencias	Los presupuestos de IT suben y no se detectan los retornos de inversión en el negocio.
Causas	En muchos casos se debe a igualar o superar los anuncios de empresas competidoras.
Solución	Establecer proyectos independientes con valores reales y medibles para la organización que sigan la filosofía de SOA, describiendo como esta tecnología se alinea a los requerimientos del negocio.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010

(<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns/>)

Tabla 4.14

¿Qué hay de Nuevo?

Anti-Patrón	¿Qué hay de Nuevo? (So, What's New?)
Problema	Escepticismo con respecto a la diferencia entre SOA y otras tecnologías llevan a pensar de que SOA no es más que un conjunto de técnicas ya conocidas las cuales no darán ningún valor al negocio.
Síntomas	Oposición de gerentes a considerar SOA como una forma seria de arquitectura empresarial.
Consecuencias	Falta de soporte a proyectos SOA.
Causas	Si bien SOA está basado en los principios introducidos anteriormente muchos grupos de desarrolladores experimentados no pueden entender el cambio de paradigma.
Solución	Enfatizar que SOA es distinto a todo lo anterior, educación, evangelización.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010

(<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns/>)

Tabla 4.15

El Big Bang

Anti-Patrón	El Big Bang
-------------	-------------

Problema	Ocurre cuando SOA es considerada una “panacea” (cura para todos los males) tratando de cambiar todos los sistemas a esta arquitectura a la vez, lo cual usualmente lleva a un fracaso culpando injustamente a SOA.
Síntomas	Preocupación de las áreas usuarias por los cambios que está ejecutando TI en los sistemas.
Consecuencias	Falla al ejecutar y cumplir con los beneficios y cambios comprometidos, otra consecuencia es buscar motivos y fallas técnicas para justificar los retrasos.
Causas	Líderes de IT, focalizados en la tecnología con poder de decisión.
Solución	Realizar un plan de adopción de SOA a largo aliento e incremental.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>

Tabla 4.16

Servicio Web = SOA

Anti-Patrón	Servicio Web = SOA
Problema	Los arquitectos pueden reemplazar erróneamente API (Aplicación Programing Interfaz) por muchos servicios web que no están alineados al negocio. También conocido como “síndrome de proliferación de servicios”.
Síntomas	Muchos de los servicios implementados son simples mensajes o invocaciones a procedimientos remotos, pero no están alineados al negocio ni son SOA.
Consecuencias	Proliferación de servicios web no alineados al negocio.
Causas	Principalmente son dos, la primera es tratar de tomar atajos y realizar servicios sin haber realizado un análisis y la segunda es la falta de conocimiento entre servicios web y SOA.
Solución	Educar sobre las diferencias entre SOA y servicios Web.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>

Tabla 4.17

El enfoque Silo (The Silo Approach)

Anti-Patrón	El enfoque Silo (The Silo Approach)
Problema	En este anti-patrón los servicios son identificados por aplicación teniendo distintos nombres como resultado no existen servicios comunes ni reutilización.
Síntomas	Encontrar el mismo servicio identificado por varios grupos sin reutilización.

Consecuencias	Servicios identificados son implementados múltiples veces duplicando el esfuerzo de implementación.
Causas	Los límites entre las áreas de negocio y restricciones entre las mismas o un análisis incorrecto no enfocado a toda la empresa.
Solución	La solución es implementar un “framework” centralizado y una identificación centralizado de servicios a lo largo de la empresa con una metodología como SOMA.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>

Tabla 4.18

Registros de mal comportamiento

Anti-Patrón	Registros de mal comportamiento (Misbehaving Registries)
Problema	La existencia de múltiples registros sin control, sobrepuestos y sin un dueño definido, lleva a un caos administrativo una mala performance y sobrecostos.
Síntomas	Servicios registrados en muchos directorios o en ninguno.
Consecuencias	La duplicación de servicios en varios registros puede crear una administración dificultosa, dificultad para identificar en caso de fallas cual es el servicio que está causando problemas, problemas en la seguridad y costos debido a la duplicación.
Causas	La falta de un diseño coherente de una solución para manejar y gobernar una arquitectura SOA, en particular en la definición de registros, responsabilidades y dueños.
Solución	Establecer un modelo de manejo y gobierno para la arquitectura, el cual defina las mejores prácticas para los registros de servicios, sus dueños y la comunicación entre equipos.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>

Tabla 4.19

Servicios Parlanchines (Chatty Services)

Anti-Patrón	Servicios Parlanchines (Chatty Services)
Problema	Los desarrolladores implementan un solo servicio web como varios, con porciones de datos pequeñas lo cual lleva a la implementación de muchos servicios los cuales son difíciles de manejar y degradan la performance.
Síntomas	La implementación de muchos servicios muy granulados.

Consecuencias	Degradación en performance y sobre-costos en desarrollo además de que un esfuerzo extra al consumir dichos servicios.
Causas	El cambio de paradigma lleva a los desarrolladores a estas implementaciones al igual que sucedió con los desarrolladores que utilizaron lenguajes orientados a objetos para desarrollar programas lineales.
Solución	Reagrupar y rediseñar los servicios además de capacitar a los desarrolladores sobre las diferencias entre un servicio y un API (application programming interfaz).
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>

Tabla 4.20

Servicios punto a punto

Anti-Patrón	Servicios punto a punto (Point-to-point Services)
Problema	La integración de aplicaciones utilizando servicios punto a punto sin usar un bus de integración, llevan a que estas se vuelvan el estándar de integración.
Síntomas	Sobre utilización de servicios SOAP, HTTP entre aplicaciones.
Consecuencias	Se toma este patrón como estándar de facto para la integración de sistemas impidiendo una solución SOA o de bus de integración y sus beneficios.
Causas	Un enfoque a corto plazo y falta de visión sobre el mantenimiento y el crecimiento total de la arquitectura.
Solución	Evaluar e implantar solución como un bus de integración para la integración de aplicaciones.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>

Tabla 4.21

Servicios sin componentes

Anti-Patrón	Servicios sin componentes (Component-less Services)
Problema	Al implementar servicios directamente sin una arquitectura adecuada es posible que la implementación de los mismos no utilice capas lo cual lleva a una inflexibilidad para el cambio de la implementación de dichos servicios.
Síntomas	Al analizar las implementaciones de los servicios se encontrarán accesos directos a funcionalidad del sistema sin un concepto de capas.
Consecuencias	La falta de flexibilidad en la implementación de servicios puede llevar a mantener las limitaciones de los sistemas heredados (Legacy) dificultando una futura reingeniería de los mismos.
Causas	Falta de un buen diseño.

Solución	La utilización de patrones para el desarrollo de los servicios web y aplicaciones en general.
Observaciones	Ninguna.

Nota. Adaptado de "SOA antipatterns" por IBM, 2010

(<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns>)

Evaluación de beneficios

Para el desarrollo de la evaluación propuesta analizamos el caso de una empresa ficticia que presenta una necesidad de integración de sus sistemas informáticos y cuya problemática podemos considerar como común.

Este escenario obedece al caso de muchas empresas medianas y grandes que tienen Sistemas Contables a Medida (Desarrollados en Casa) e Implementan Sistemas de Ventas Web en sus ambientes de IT o en un Servicio de Hosting Exterior.

Caso de “Mi Mascota”

La empresa ficticia “Mi Mascota”, se dedica a la comercialización de animales domésticos o de compañía (mascotas) dentro del territorio nacional. Con 5 años en el mercado la empresa ha decidido abrir una tienda virtual para la venta de mascotas.

“Mi Mascota” actualmente cuenta con alrededor de ochenta (80) trabajadores y un área de Sistemas de ocho (8) personas destinando cuatro (4) personas para el desarrollo de sistemas de información.

La empresa “Mi Macota” es parte de la corporación “Pet United” por lo cual posee acuerdos trasnacionales para el uso de licencias de diversos productos Microsoft e IBM.

A nivel corporativo se decidió iniciar la venta de mascotas a través de Internet, para este fin se utilizó el código fuente de la aplicación “Pet Shop”, cuyo código esta licenciado bajo el acuerdo Microsoft Public License (Ms-PL), mas información en “Ms-PT”, por Microsoft, 26 de Abril del 2010, (<http://petshopvnext.codeplex.com/license>) .

Dicha aplicación considera las funcionalidades necesarias para el registro de usuarios (clientes), el catálogo de mascotas y un carrito virtual de compras el cual genera los pedidos necesarios.

Dado que las características contables de cada país varían de manera sustancial se tomó la decisión de que cada país implemente o adquiera un paquete contable, es así que “Mi Mascota” está implementando un sistema contable, el cual está siendo desarrollado en la plataforma J2EE utilizando Base de Datos MySQL y WebSphere Application Server. Dicho sistema actualmente contempla:

- Maestro y Mantenimiento de Productos.
- Ingreso y Reportes de Asientos Contables.
- Ingreso y Reportes de Kardex Valorizado.

Dichos reportes son exportados a una hoja de cálculo (MS Excel) para elaborar la contabilidad de la empresa mientras el sistema es concluido.

Es así que se ha identificado necesario llevar a cabo un proyecto de integración para el envío de las órdenes realizadas en el Sistema de Ventas Web al sistema contable.

Implementación

a) Aplicaciones

Para emular el ambiente descrito se utilizó y modificó la aplicación MS-Petstore y se desarrolló una aplicación que emula el sistema contable escrito en el lenguaje de programación Java, además se utilizó la base de datos MySQL.

b) Ambiente de Desarrollo y Pruebas

El desarrollo fue llevado a cabo por una persona a tiempo parcial, utilizando las siguientes herramientas.

- VM Workstation 6.5.3.
- IBM Websphere Integration Developer 6.0.2.
- Microsoft Visual Studio 2005.
- Windows XP Service Pack 3.
- Windows 7.

- Sistema Operativo Linux – TurnKey MySQL Appliance - : 2009.10.
- MySQL v.5.0.5.
- IBM Rational Software Architect v.7.5.

El ambiente de desarrollo y pruebas estuvo conformado por 2 laptops, implementándose dos sistemas virtuales en cada una de ellas, donde se emplazaron las siguientes imagines virtuales de sistemas operativos:

Sistema Uno:

Imagen: Base de Datos

- Sistema Operativo Linux (Turnkey MySQL Appliance).
- MySQL database.

Sistema Dos:

Imagen: WAS

- Sistema Operativo Windows XP.
- WebSphere App Server v.7.5.
- Sistema Contable J2EE.

Sistema Tres:

Imagen: Petstore

- Sistema Operativo Windows XP.
- Microsoft SQL Server 2005.
- Internet Information Server 5.1.
- Microsoft Developer Studio 2005.
- Aplicación Ventas Web.

Sistema Cuatro:

Imagen: Process Server

- Sistema Operativo Windows XP.
- Websphere Integration Developer 6.0.2.
- Websphere Process Server.
- Microsoft Office 2007 Profesional.

c) Entregables

Siguiendo la metodología descrita en el numeral 4.1 del Capítulo IV de la presente investigación, se elaboraron los entregables correspondientes a las fases de Análisis, Diseño, Desarrollo y Pruebas, los cuales se listan a continuación y se acompañan como Anexos al presente documento.

- A.1.1 - Documento de Requerimientos
- A.1.3 - Documento de Análisis.
- A.1.2 - Documento de Arquitectura (Inicial).
- A.2.1 - Documento de Diseño.
- A.2.2 - Documento de Arquitectura (Final).
- A.3.1 - Código Fuente.
- A.4.4 - Casos de Prueba.
- A.4.2 - Diseño de Pruebas.
- A.4.1 - Informe de Pruebas.
- A.4.3 - Plan de Pruebas.

CAPÍTULO V: RESULTADOS

La integración de ambos sistemas se realizó de manera exitosa, a continuación se muestra el tiempo aproximado por etapa:

Tabla 5. 1

Resultados

Etapa	Tiempo Aproximado
Análisis	10 días
Diseño	10 días
Desarrollo	15 días
Pruebas	5 días

5.1 Verificación de Requerimientos

A continuación se muestran los requerimientos y como fueron concretados:

a) Requerimientos Funcionales

Realizar Pedido: Este caso de uso ya se encontraba implementado y no se realizaron modificaciones, sin embargo se identificó como probable caso de uso de integración.

Envía Pedidos: Este caso de uso fue implementado en WebSphere Process Server como parte los componentes de integración.

b) Requerimientos No Funcionales

La verificación de los requerimientos no funcionales requiere para su comprobación la realización de pruebas y su posterior comparación con un estándar y/o niveles de servicio definidos específicamente para el caso. Sin embargo dichas labores se encuentran fuera

del alcance propuesto por requerir recursos no disponibles en la presente investigación y además por no invalidar o afectar negativamente dicho la investigación realizada.

Sin perjuicio de lo expuesto, en un plano teórico pueden establecerse los siguientes factores:

Performance: Al implementarse el proceso de transferencia e integración tipo “Batch” y no un esquema en línea de llamada directa a servicios no se modificaron las funcionalidades ya implementadas y por tanto, no se afectaría su performance o disponibilidad directamente minimizando este impacto.

Escalabilidad: Dado que las aplicaciones en este caso son basadas en web y al haber utilizado servicios web para la implementación de las interfaces, la escalabilidad de las interfaces dependería de los mismos componentes y servicios (Hardware, Software, Comunicaciones , entre otros) que la aplicación, por lo tanto la escalabilidad no sería afectada al no requerir componentes o servicios adicionales.

Disponibilidad: Se mantendría dadas las razones expuestas en los numerales correspondientes **a performance y escalabilidad.**

Mantenibilidad: La mantenibilidad no sería afectada al haber utilizado los mismos lenguajes de programación, las clases ya desarrolladas y la baja cohesión lograda entre ambos sistemas gracias a la utilización de SOI.

Seguridad: Dado que tanto aplicación como sus interfaces son web y basadas en el protocolo y servicio HTTP (HyperText Transfer Protocol), su seguridad dependería de los mismos recursos y configuraciones para asegurar dicho servicio, es así que no se afectaría la seguridad de la misma. Cabe resaltar que adicionalmente sería posible utilizar los siguientes recursos en caso sea necesario:

- Autenticación Simple.
- SSL (Secure Socket Layer).
- HTTPS (Hypertext Transfer Protocol Secure).
- Firewall y reglas de acceso.

Como resultado entonces podemos afirmar que la Integración Orientada a Servicios puede ser aplicada en los proyectos de desarrollo de sistemas similares al modelo presentado y mediante las etapas planteadas en la metodología que la presente investigación utilizó.

5.2 Comprobación de Uso de Patrones

A continuación se muestran los patrones y su utilización en el código:

4.2.7 Sub Controlador Agnóstico:

Dentro del código de los objetos “OrderWS”, “KardexWS”, “AsientoWS”, se comprueba el patron al utilizar el Objeto “Order”, “Kardex” y “Asiento”, respectivamente. Dichos métodos manejan múltiples objetos dentro de sus métodos de tal manera que el Objeto que hace uso de ellos no realiza múltiples llamadas.

4.2.8 Recursos Canónicos:

Se reconoce mediante las librerías utilizadas en los servicios web, dichas librerías implementan los componentes comunes para los servicios web, para el caso de la plataforma Microsoft .NET:

- System.Web.Services
- System.Web.Services.Protocols

Y para la plataforma J2EE:

- @javax.jws.WebService

4.2.9 Transacciones entre múltiples servicios:

Este patron se reconoce dentro del código implementado en el bus de integración, esa así que podemos apreciar lo siguiente:

- Inclusión de las importaciones de los servicios web a utilizar en los objetos: OrderWSSoapImport1 y ProxyWSDelegateImport1 que corresponden a los servicios de Ordenes, Asientos y Kardex.

- Los objetos tipo “invoke”: ObtenerOrdenes, CrearAsiento y CrearKardex, donde se realizan las llamadas a los servicios web correspondientes.

4.2.10 Bus de Integración:

Se demuestra su utilización mediante la inclusión del código fuente de “Websphere Process Server”.

5.3 Evaluación

A continuación se presenta la evaluación sobre los beneficios obtenidos:

5.3.1 Aumento de la eficiencia en el desarrollo y mantenimiento de las interfaces.

Durante la implementación de los servicios necesarios en ambas plataformas se pudo observar que ambas permiten la creación casi inmediata de servicios Web desde métodos y clases comunes.

Es así que se implementó un paquete más en cada aplicativo para los métodos necesarios de modo de no afectar las clases de negocio. Luego se procedió a implementar de manera automática los servicios que la aplicación requería con las herramientas que brinda cada herramienta de desarrollo (WebSphere Software Architect y .Net Studio 2005).

Sin embargo, cabe resaltar que esto fue posible de manera transparente debido a que la aplicación fue desarrollada usando el paradigma de orientación a objetos e implementa tipos de datos estándar para los servicios web.

En acuerdo con lo anteriormente expuesto, la aplicación debe de implementar objetos de tipo “entidad” como por ejemplo los denominados POJO (“Plain Old Java Objects”) o EJB Enterprise Java Beans, que son utilizados como “moneda” a través de la aplicación.

Además la lógica de negocio debe de estar separada y utilizar estos objetos como respuesta y parámetros de entrada.

Con respecto a la implementación en el bus de integración, no se encontró mayor dificultad de compatibilidad entre ambas plataformas utilizando el protocolo SOAP.

En el caso presentado fue necesario implementar:

- Capa Proxy.
- Generación de servicios (automática).
- Orquestación (Programa de Integración en el Bus de Integración).

En el supuesto de haber realizado la implementación bajo un gestor de colas tal como MSMQ (Microsoft Message Queuing) ó IBM MQ (Message Queue) Series, siguiendo los lineamientos que la documentación de dichos productos nos brinda hubiese sido necesario implementar lo siguiente¹:

- Capa Proxy.
- Método de acceso al Gestor de Colas (Queue Connection Factory), una por cada plataforma.
- Método para recibir el mensaje, en este caso una por plataforma.
- Método para interpretar los datos del mensaje
- Elaboración de código (Clases) para la lectura e interpretación de mensajes, al menos uno por cada tipo de mensaje y plataforma.
- Configuración de Colas y Canales para mensajes

Tomando en cuenta que el número de programas y su complejidad dependerá de la implementación hecha y existiendo un número muy grande de implementaciones posibles no podemos tener certeza sobre el número exacto de programas y su complejidad, pudiendo variar el número de programas entre 2 métodos (un método en

¹ El sustento sobre dichas clases podemos encontrarlas en el libro “MQ Series - Using Java” para la plataforma J2EE y para la plataforma .Net, el libro “WebSphere MQ Solutions in a Microsoft .NET Environment”, en las direcciones electrónica: (i) <<http://publib.boulder.ibm.com/series/v5r2/ic2924/books/csqzaw07.pdf>>(10 de Enero 2010) y (ii) <<http://www.redbooks.ibm.com/abstracts/sg247012.html>>(10 de Enero 2010).

cada plataforma), hasta al menos 10 (5 en cada una) siguiendo las buenas prácticas de ambos fabricantes.

Sin embargo considerando las tareas, arquitectura y manuales de los fabricantes, podemos afirmar que en dicho escenario (utilizando un gestor de colas) la probabilidad de tener número mayor de programas y de mayor complejidad es mayor a comparación del escenario desarrollado.

Por los factores expuestos, y tomando en cuenta como métrica referencial el número de programas y su complejidad, podemos afirmar que el beneficio planteado será obtenible de acuerdo al caso específico y dependerá de factores como:

- Implementación los sistemas a Integrar (Alineamiento al patrón MVC)
- Código escrito para las interfaces y su porcentaje de reutilización.
- Capacidad de generación y soporte para la implementación de servicios Web.

Reducción de esfuerzo y por tanto en costo en el proceso de desarrollo y mantenimiento de interfaces entre sistemas.

Tomando como métrica referencial el número de programas y su complejidad sustentados en el numeral anterior. Podemos decir entonces que al existir una alta probabilidad de tener un número menor de programas y de menor complejidad, un ahorro de costos en tiempo y recursos humanos comparado a un desarrollo de interfaces clásico hubiese sido posible de acuerdo al caso específico.

Sin embargo, cabe resaltar que no se han analizado el costo de licenciamiento de productos de software ni el hardware necesario, dado que no es parte del alcance propuesto, por lo cual no se puede afirmar que haya un ahorro de costos en los proyectos y organizaciones de TI al utilizar esta filosofía.

Unificación y estandarización de las interfaces hacia el uso de Web Services y XML en los casos donde sea aplicable.

Este beneficio está implícito en la filosofía SOA y fue obtenido en este caso. Cabe resaltar que este beneficio se sostendrá siempre y cuando se sigan los principios de diseño

que SOI plantea. Sin embargo para llegar a una estandarización se deberá de modificar aquellas no estandarizadas que ya estén implementadas.

5.3.2 Mejorar la interoperabilidad de las distintas plataformas en la empresa.

Para el caso evaluado es posible considerar una mejora sustancial en la interoperabilidad tomando en cuenta los siguientes factores:

- Los servicios expuestos son fácilmente modificables, así como la implementación en el bus de integración.
- La manera como es implementado el servicio es transparente para quien lo consume, por lo cual cualquier modificación sobre su implementación tiene bajas probabilidades de afectar a quien lo consume.
- Es posible integrar cualquier sistema con soporte a servicios Web a través de esta tecnología o también, a través de adaptadores o conectores provistos por el software de integración.
- Las interfaces, orquestaciones e implementaciones realizadas en el bus de Integración tienen más probabilidades de ser gobernadas.
- Los servicios desarrollados son reutilizables sin ser modificados.
- No existe necesidad de utilizar librerías o clases propietarias para acceder al bus de integración.
- Dada la flexibilidad y bajo acoplamiento de las interfaces, las modificaciones sobre estas serán más sencillas.

5.3.3 Aumento de la reutilización de interfaces y componentes desarrollados.

Si bien para este caso no se reutilizaron interfaces existentes, debemos de considerar que las interfaces creadas están listas para ser reutilizadas y además que siendo estas estándares su probabilidad de reutilización es mayor.

Considerando lo anteriormente expuesto, podemos decir que este beneficio es alcanzable de manera natural con la aplicación de SOI.

5.3.4 Facilitar la integración con distintas entidades de negocio (por ejemplo: proveedores y clientes), mediante un estándar de facto (Web Services).

Si bien el caso planteado no considera este escenario (integración con distintas entidades de negocio), podemos inferir que esto será cierto siempre y cuando se cumplan en alguna medida las siguientes condiciones:

- Las plataformas a integrar sean compatibles con Web Services.
- Utilicemos un mediador o bus de integración.
- Nuestras aplicaciones estén desarrolladas de acuerdo al patrón MVC (Model-View-Controller).

5.4 Contrastación del Método

El éxito de la aplicación de la metodología desarrollada para el escenario propuesto nos indica que el mismo contiene los pasos necesarios para enfrentar un escenario similar.

A continuación mostramos el detalle sobre la metodología utilizada y lo realizado.

Figura 5.1

Constratación del método

Tarea/Etapa	Realizado	Utilizado
Requerimientos		
Definir los Requerimientos de Integración		
Identificar Sistemas a Integrar	Sí	Sí
Identificación Requerimientos Funcionales	Sí	Sí
Identificación de Requerimientos No Funcionales	Sí	Sí
<i>Entregable : Documento de Requerimientos</i>	Sí	Sí
Análisis		
Delinear la Arquitectura de Integración		
Definir los requerimientos de Transformación de Datos		
Identificar los Requerimientos de Replicación y Sincronización de Datos	Sí	Sí
Identificar el Impacto de la Consolidación y Limpieza de Datos	No	No
Identificar los posibles requerimientos de llaves primarias de referencia cruzado	Si	Sí
<i>Entregable : Documento de Análisis</i>	Sí	Sí
<i>Entregable : Documento de Arquitectura (Inicial)</i>	Sí	Sí
Diseño		
Diseñar las Integraciones y Procesos	Sí	Sí
Diseñar las Transformaciones de Datos	Sí	Sí
Definir la Arquitectura de Integración	Sí	Sí
<i>Entregable : Documento de Arquitectura (Final)</i>	Sí	Sí
<i>Entregable : Documento de Diseño</i>	Sí	Sí
Desarrollo		
Construcción	Sí	Sí
Codificación y Pruebas Unitarias	Sí	Sí
Configuración de los componentes de Integración	Sí	Sí
<i>Entregable : Código Fuente</i>	Sí	Sí
Pruebas		
Pruebas y Mejoras	Sí	Sí
<i>Entregable: Plan de Pruebas</i>	Sí	Sí
<i>Entregable Diseño de Pruebas</i>	Sí	Sí
<i>Entregable: Casos de Prueba</i>	Sí	Sí

CAPÍTULO VI: RESULTADOS DE LA INVESTIGACIÓN

Luego de haber concluido la investigación propuesta, podemos enumerar los siguientes resultados:

1. Se probó que la integración orientada a servicios (SOI) es aplicable a proyectos de desarrollo de software e integración bajo un modelo clásico de desarrollo de software bajo la metodología propuesta durante la investigación de la presente tesis.
2. Se evaluó el grado de obtención de los beneficios planteados en la presente.
3. Se construyó un catálogo con los principios, patrones y antipatrones de SOI.
4. Se comprobó que plataformas J2EE y .NET tienen un soporte necesario a las tecnologías de Web Services robusto, permitiendo que la integración utilizando SOI, el protocolo SOAP y un bus de integración.
5. Se probó que la aplicación de la metodología propuesta para el escenario planteado fue exitosa.

CONCLUSIONES

1. La utilización de SOI y la metodología propuesta en los proyectos que envuelvan integración de sistemas, puede ayudar en mayor o menor grado a obtención de los beneficios mencionados en el numeral 5.3 del Capítulo V. Asimismo, el conocimiento de patrones y anti-patrones SOI ayudará a la obtención de los referidos beneficios.
2. La reutilización del código ya desarrollado se dará de manera natural siempre y cuando se haya seguido el patron MVC (Modela Viel Controlar) o de “Tres” capas.
3. Las plataformas SUN Java J2EE - JRE 1.8 y Microsoft .Net 2005, así como WebSphere Process Server v 7.5, son adecuadas para proyectos de Integración SOI. Siendo muy probable que las versiones superiores también lo sean.
4. En el caso aplicaciones web (web enable) la implementación de servicios web para las interfaces ayudará a mantener la disponibilidad, escalabilidad, y seguridad ya implementadas en dichas aplicaciones para dichas interfaces.

RECOMENDACIONES

1. Dado que la obtención de los beneficios planteados en la presente investigación es relativa a la organización ó proyecto analizado, es recomendable evaluar el caso específico y el grado de obtención de beneficios de la aplicación de SOI.
2. La aplicación de SOI debe de estar acompañado de un proceso y disciplina de Arquitectura IT.
3. El estudio de patrones y antipatrones así como de SOI y SOA, se hacen necesarias en la formación de arquitectos de sistemas y profesionales que se desea desempeñen en tareas de de integración.
4. Para la aplicación de SOI y la obtención de sus beneficios, es recomendable considerar aspectos que no son parte del alcance de esta investigación tales como:
 - a. Gobernabilidad.
 - b. Sistemas e interfaces ya construidas.
 - c. Conocimiento y Capacidad del equipo de T.I.
 - d. Ambiente y Ecosistema T.I. actuales.
5. Es recomendable que los objetos utilizados para la generación o implementación de servicios web utilicen tipos de datos primitivos y arreglos simples de objetos, evitando así el uso de colecciones y arreglos de objetos donde se requieran formas de casting tales como los objetos Arraylists, Vector, Hashtable y Generics

GLOSARIO DE TÉRMINOS

- Eclipse IDE: “Eclipse es una comunidad de código abierto para aquellos proyectos enfocadas en la construcción y desarrollo de librerías, herramientas y plataformas de ejecución para la construcción y el manejo e implementación de proyectos de desarrollo”(Traducido de : “*Eclipse.org*<<http://www.eclipse.org>> (26 de Abril de 2010).
- IDE: siglas en inglés de Integrated Developer Environment que significa Entorno Integrado de Desarrollo.
- Microsoft Visual Studio 2005: Es la IDE de desarrollo para la plataforma Microsoft.
- SOA (Service Oriented Architecture): Un estilo arquitectural y principios de diseño e integración para el desarrollo de aplicaciones o su integración.
- Servicio Web (Web Service): W3C define un Servicio Web como un sistema de software diseñado para dar soporte e interoperabilidad “Maquina a Maquina” sobre una red. Frecuentemente son interfaces de programas que pueden ser accedidos sobre una red como Internet y ejecutar en la maquina remota el servicio.
- XML: sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

REFERENCIAS

- Alexander (1979). *A Timeless Way of Building*.: Oxford University Press.
- Eclipse.org. (2010). *Eclipse org*. <http://www.eclipse.org>.
- EbizLatam (2009). *Gartner afirma que el 72 % de los proyectos relacionados con integración de sistemas están relacionados con servicios Web*.
<http://www.ebizlatam.com/noticias/wmview.php?ArtID=1619>
- EIA Patterns. (2010). *EIA Patterns*. <http://www.eaipatterns.com>
- Garthner. (2015). *Magic Quadrant for Integration Backbone Software, 1H05*
- IBM. (2010). *IBM Developer Works*. <http://www.ibm.com/developerworks/>
- IBM. (2010). *Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6*. <http://www.redbooks.ibm.com/abstracts/sg246494.html>
- IBM. (2010). *DeveloperWorks WebSphere Technical library IBM SOA Foundation product integration*.
http://www.ibm.com/developerworks/websphere/techjournal/0812_tellado/0812_tellado.html
- IBM. (2010). *SOA and the Art of Riding the Enterprise Service Bus, Part II*.
<http://soa.sys-con.com/node/46559>
- IBM. (2010). *SOA antipatterns*.
<http://www.ibm.com/developerworks/webservices/library/ws-antipatterns/>
- Microsoft. (2010). *Principios de diseño de servicios: patrones y antipatrones de servicios*. <http://msdn.microsoft.com/es-es/library/ms954638.aspx>
- Microsoft. (2010). *Messaging Patterns in Service-Oriented Architecture, Part I*.
<http://msdn.microsoft.com/en-us/library/aa480027.aspx>
- Microsoft. (2010). *Extensibility Points*. <http://msdn.microsoft.com/en-us/library/aa480027.aspx>
- Microsoft. (2010). *Ms-PT*. <http://petshopvnext.codeplex.com/license> .
- SOA GLOSSARY. (2010). *SOA GLOSSARY*. <http://www.soaglossary.com>
- SOA PRINCIPLES. (2010). *SOA PRINCIPLES*. <http://www.soaprinciples.com/>



BIBLIOGRAFIA

- Allen P. (2006). *Service Orientation: Winning Strategies and Best Practices.*: Cambridge University Press.
- Endrei, M. (2004). *Patterns: Service-Oriented Architecture and Web Services.* : IBM Redbooks
- Foggon, D. Maharry, D. Ullman, C. y Watson, K. (2004). *Programming Microsoft .NET XML Web Services.*: Microsoft Press.
- Hurwitz, J. Bloor, R. Baroudi, C y Kaufman, M. (2007). *Service Oriented Architecture For Dummies.*: John Wiley & Sons.
- Holzner, S. (2006). *Design Patterns For Dummies*, John Wiley & Sons
- IBM, 2010. *Elements of Service-Oriented Analysis and Design.*
<http://www.ibm.com/developerworks/webservices/library/ws-soad1/index.html>
- IBM, (2010). *Service-oriented modeling and architecture (SOMA)*,
<http://www.128.ibm.com/developerworks/webservices/library/ws-soa-design1>.
- Paul, B (2003). *Web Service Patterns: Java Edition.*: Monday Apress.

ANEXOS



Anexo 1: A.1.1 - Documento de Requerimientos



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 21- Noviembre-2009 - jcabrera

Modificaciones:

Objetivo:

El objetivo del presente documento es documentar el análisis de los requerimientos para las interfaces de integración de los sistemas:

- Sistema de Ventas Web (Web Sales).
- Sistema de Control Contable (Sistema Contable).

Requerimientos:

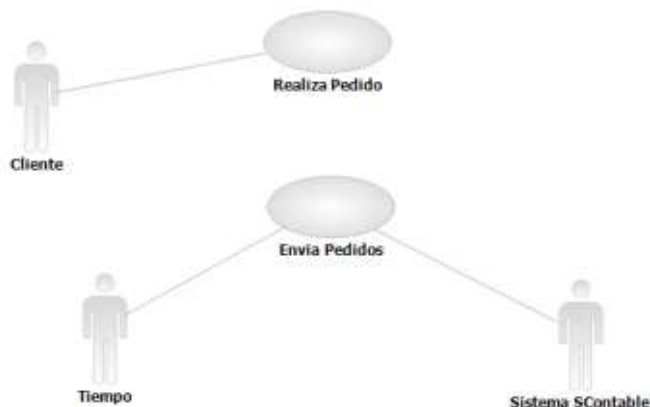
A) Requerimientos Funcionales : A continuación se muestran las funcionalidades de el Sistema de Control Contable (Sistema Contable) que en reuniones con los usuarios líderes se identificaron como candidatas a utilizar interfaces con el sistema contable:

- Realizar Pedido: Es el caso de uso o funcionalidad que permite realizar un pedido luego de haber escogido los productos, cantidades. Este caso de uso ya se encuentra implementado.
- Envía Pedidos: Este es el caso de uso que permitirá el traslado de información entre ambos sistemas, dicho caso de uso será automáticamente ejecutado una vez al día. Es aquí que deberán de crearse los movimientos de Kárdex y los asientos contables correspondientes.

B) Diagrama de Casos de Uso: A continuación se muestran los casos de uso que se han identificado como probables casos de uso de integración de ambos sistemas.

Figura A.1.1.1

Diagrama de Casos de Uso Análisis



En donde el caso de uso “Realizar Pedido” pertenece al Sistema de Ventas Web, y a su vez el caso de uso “Enviar Pedidos” pertenece al Sistema de Integración.

C) Requerimientos No Funcionales :

Los requerimientos no funcionales identificados son:

- **Performance:** La performance de ambos sistemas debe de ser impactada en lo mínimo posible, en especial la del sistema de ventas por Internet.
- **Escalabilidad:** La solución propuesta no debe de afectar la escalabilidad de la actual arquitectura de software.
- **Disponibilidad:** Se deben de mantener los niveles de disponibilidad actuales de la plataforma.
- **Mantenibilidad:** Se deben de mantener los estándares de programación (lenguajes y plataforma) actuales. Dado que el Sistema de Gestión Contable se encuentra en desarrollo a la fecha, es necesario que los cambios en este y sus interfaces afecten de la menor manera posible a las interfaces desarrolladas en el Sistema de Ventas Web.

- Seguridad: La solución no debe de afectar la seguridad de las aplicaciones y se debe de considerar que :
 - Los servidores se encuentran dentro de la DMZ de la empresa.
 - Solamente el servicio de http, es accesible desde fuera de la DMZ.
 - Dentro de la DMZ el tráfico es libre y no necesita autenticación de usuario y password.
 - Fuera de la DMZ, las aplicaciones deben de considerar un esquema de autenticación de usuario y password.



Anexo 2: A.1.3 - Documento de Análisis



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 21- Noviembre-2009 - jcabrera

Modificaciones:

Catálogo de Interfaces:

Tabla A.1.3.1

Catálogo de Interfaces

Interfaz	Sistema Expone	Sistemas que la utilizan	Descripción
WSAL-001-CPEDIDO	BUS	Ventas Web	Interfaz que envía la información sobre los pedidos creados.
SCON-001-KARDEX	Sistema de Control Contable	BUS	Interfaz que crea los movimientos de kárdex.
SCON-001-ASIENTO	Sistema de Control Contable	BUS	Interfaz que genera los asientos contables generados por un pedido.

Requerimientos de Transformación de Datos, Sincronización, Limpieza de Datos:

Se identificaron las siguientes transformaciones de datos:

- La fecha de la orden debe generar los campos año y mes del Kardex y Asiento Contable.
- Transformación de los tipos de datos SQL Server y MS-SQL.

No se considera necesaria la limpieza de datos, dado que la data histórica no será utilizada, pero si se consideran necesarias validaciones para mantener la integridad referencial entre ambos sistemas, es así que los campos identificados como llaves de referencia cruzada son:

- Identificador de producto.
- Identificador de pedido.

Anexo 3: A.1.2 - Documento de Arquitectura (Inicial)



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 21- Noviembre-2009 - jcabrera

Modificaciones:

Principios de Arquitectura:

Los principios de arquitectura contemplados relativos a este proyecto son:

1. Escalabilidad: Las aplicaciones deben de conservar o mejorar su nivel de escalabilidad en lo referente a su capacidad transaccional.

Justificación: Dada la naturaleza y expectativas de crecimiento de la empresa, es necesario sea posible aumentar las capacidades transaccionales de la plataforma mediante el aumento de nodos computacionales, sin que esto no requiera modificaciones sobre las aplicaciones en la medida de lo posible.

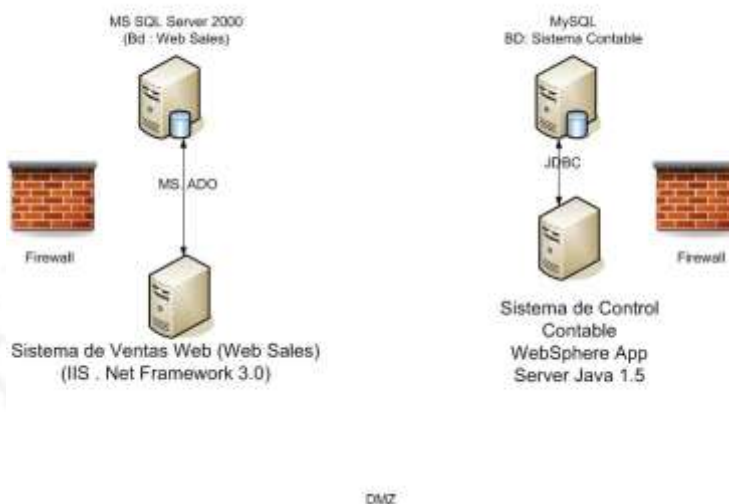
2. Interoperabilidad: Las aplicaciones deben de mantener un nivel adecuado de interoperabilidad entre si mediante tecnologías estándares.

Justificación: Tomando en cuenta la diversidad de plataformas actuales y reconociendo la necesidad del intercambio de datos entre dichas aplicaciones, se requiere la utilización de interfaces con un nivel alto de interoperabilidad y que sigan, en la medida de lo posible, estándares abiertos soportados por la mayoría o todas las aplicaciones y/o plataformas de la empresa.

Arquitectura Actual:

Figura A.1.2.1

Diagrama de arquitectura inicial de ambas aplicaciones



Descripción:

Como se puede observar los servidores se encuentran dentro de una DMZ (zona desmilitarizada) dentro de la cual existe comunicación libre y de confianza entre los servidores. Hacia fuera de la DMZ solo se permite el tráfico http, los nodos relacionados al proyecto son:

1. Servidor IIS: Donde se encuentra la aplicación de Ventas Web (Web Sales), la cual esta desarrollado con Visual Studio 2005 y .Net framework 3.0.
2. Servidor MS SQL Server: Nodo donde se encuentra la base de datos del Sistema de Ventas Web.
3. Web Sphere App Server: Nodo donde se encuentra el Sistema de Gestión Contable que esta siendo desarrollado.
4. MySQL: Nodo que contiene la base de Datos MySQL que aloja las tablas de el Sistema Contable.

Decisiones de Arquitectura

A continuación se presenta y justifican las decisiones de arquitectura:

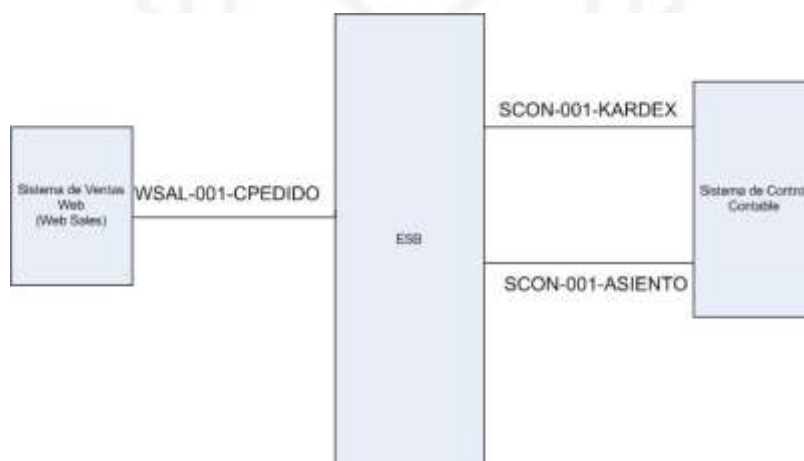
1. Utilización de servicios Web y SOI: La finalidad de esta decisión de arquitectura es lograr las ventajas que SOI plantea tales como rehúso de interfaces y reducción del tiempo de desarrollo.
2. Utilización del estándar SOAP: Esta decisión se sustenta en la capacidad de ambas plataformas para soportar dicho protocolo para el intercambio de datos.
3. Utilización de un BUS de integración: Esta decisión busca poder reutilizar las interfaces desarrolladas y se toma previendo que los sistemas involucrados deberán de intercambiar información con otros sistemas y entre si en un futuro próximo.
4. Proceso en “Batch” para transferencia de Órdenes: Con la finalidad de afectar lo menos posible la performance sobre la actual aplicación de ventas por Internet, la transferencia de órdenes se hará una vez al día durante la hora de menos uso del sistema.

Diagrama de Interfaces:

A continuación se muestran las interfaces que han sido identificadas:

Figura A.1.2.2

Diagrama de Interfaces



Para las interfaces se tomó la siguiente convención:

- ABCDE-9999-NOMBRE: donde ABC son las siglas de el sistema, “9999” el número de interfaz relativa al sistema y “NOMBRE” el nombre de dicha interfaz.



Anexo 4: A.2.1 - Documento de Diseño



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 21- Noviembre-2009 - jcabrera

Modificaciones:

Objetivo:

El objetivo del presente documento es el diseño de las interfaces de integración de los sistemas:

- a) Sistema de Ventas Web (Web Sales)
- b) Sistema de Control Contable

Flujos Identificados:

Los flujos identificados como Casos de Uso de Integración se muestran a continuación:

Figura A.2.1.1

Diagrama de actividad caso de uso comprar en línea

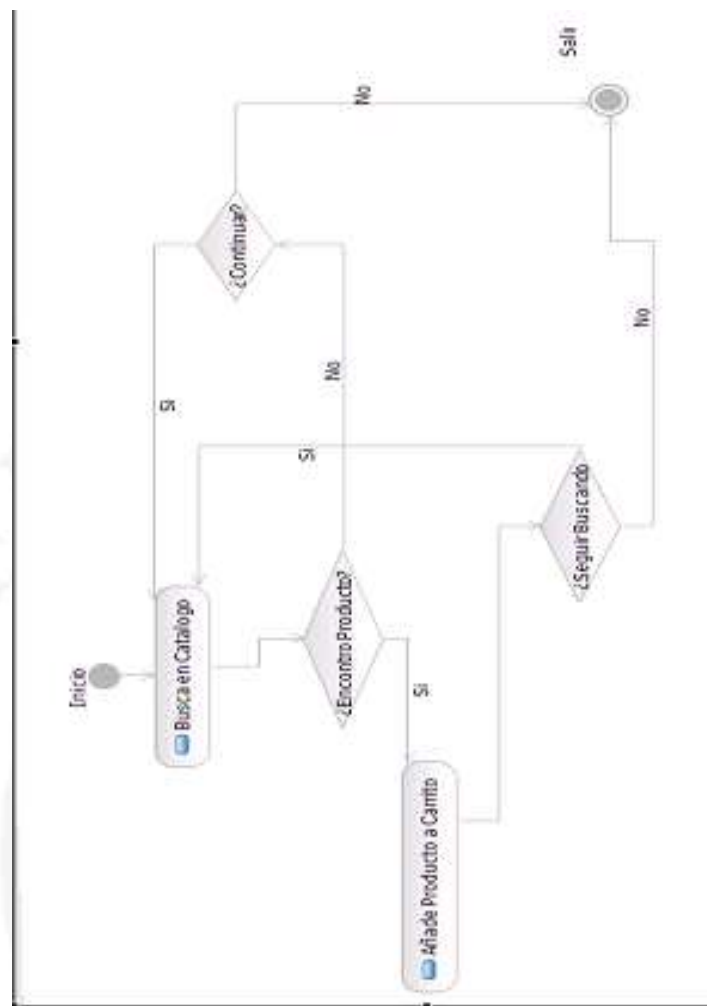
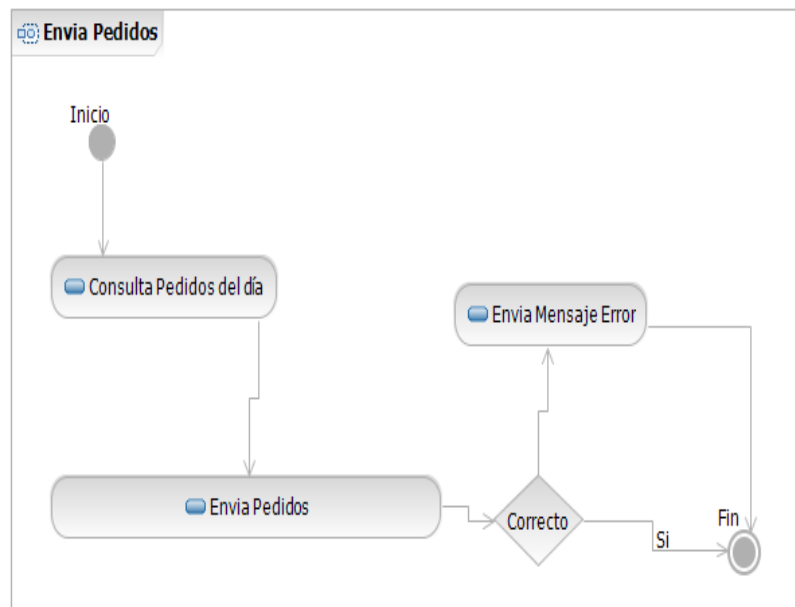


Figura A.2.1.2

Diagrama de actividad caso de uso envía pedidos



Patrones Identificados:

Se identificó el uso tentativo de los siguientes patrones en los siguientes casos:

1. Sub Controlador Agnóstico: Para la implementación de servicios en ambas aplicaciones.
2. Recursos Canónicos: Se reconoce el uso recursos (frameworks) para todos los servicios a ser desarrollados durante este proyecto y los siguientes en ambas plataformas.
3. Transacciones entre múltiples servicios: Se reconoce la participación de al menos 3 servicios.
4. Bus de Integración: Se utilizará este patrón reconociendo la necesidad de orquestación de múltiples servicios y necesidades futuras de integrar ambas aplicaciones con diversos sistemas.

De la misma manera, se reconoce que los siguientes anti-patrones deben de ser evitados durante la implementación:

1. Chatty Services.
2. Servicios punto a punto.
3. Servicios sin componentes.

Transformaciones de Datos:

Al momento no se ha detectado alguna necesidad de transformaciones de datos dado que los campos utilizan el protocolo SOAP, de ser necesarias estas podrían ser resueltas mediante la capa de integración dado que no son masivas.

Diseño de Interfaces:

Interfaces Identificadas²:

Siguiendo los lineamientos de diseño, se diseñaron y se generaron las definiciones WSDL para los siguientes servicios que fueron identificados durante presente proceso de diseño, adicionalmente se inicia cada descripción con los gráficos generados con Rational Software Architect V7.5 de las mismas: WSDL- WSAL-001-CPEDIDO

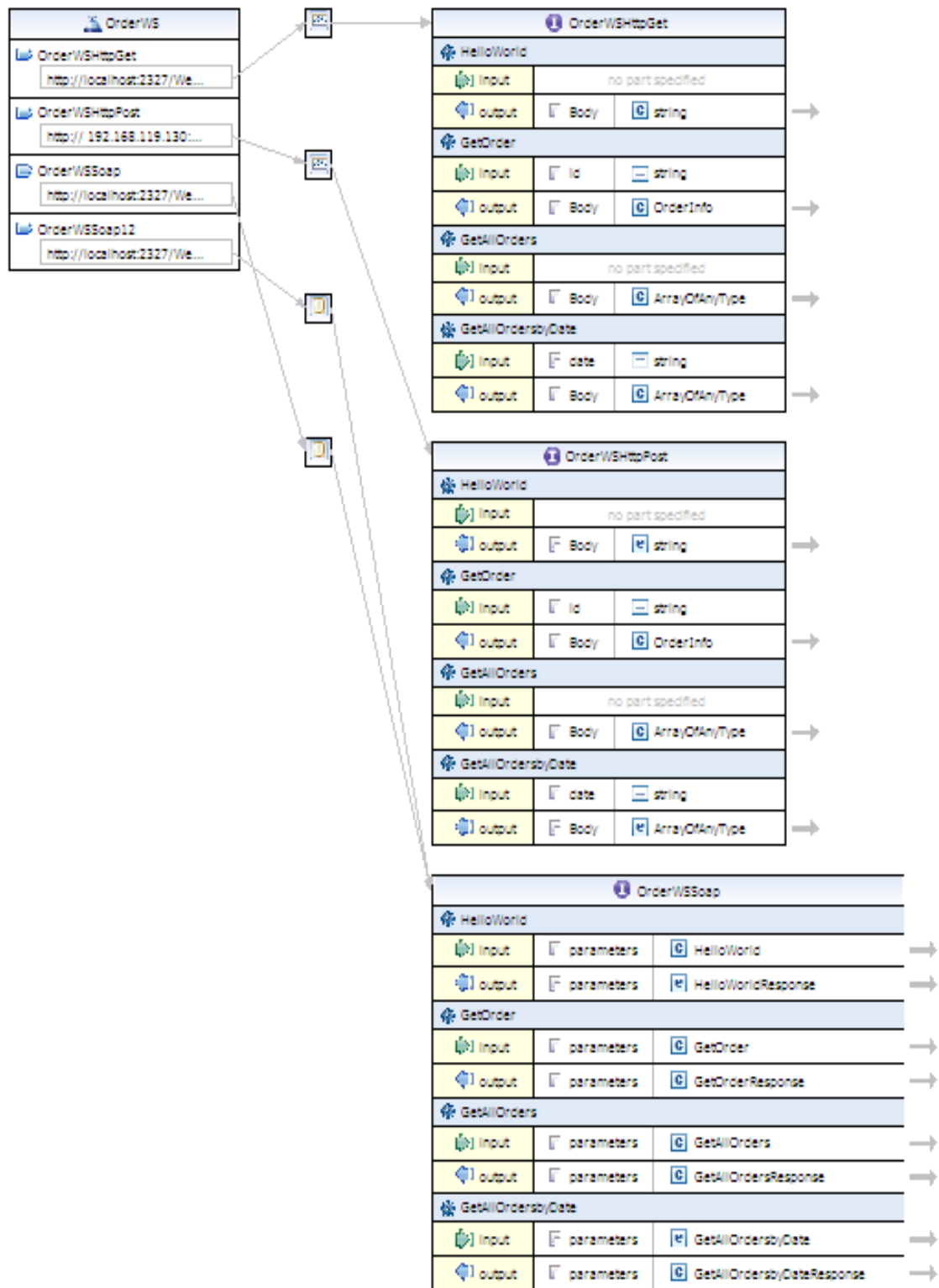
Gráfico:

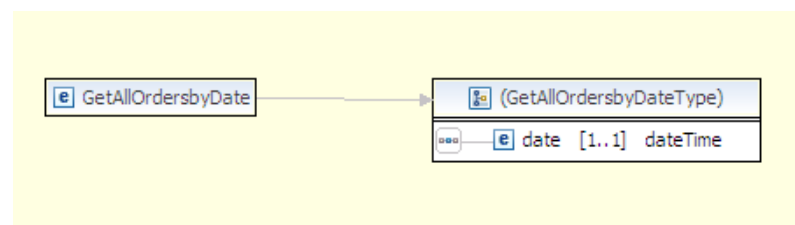
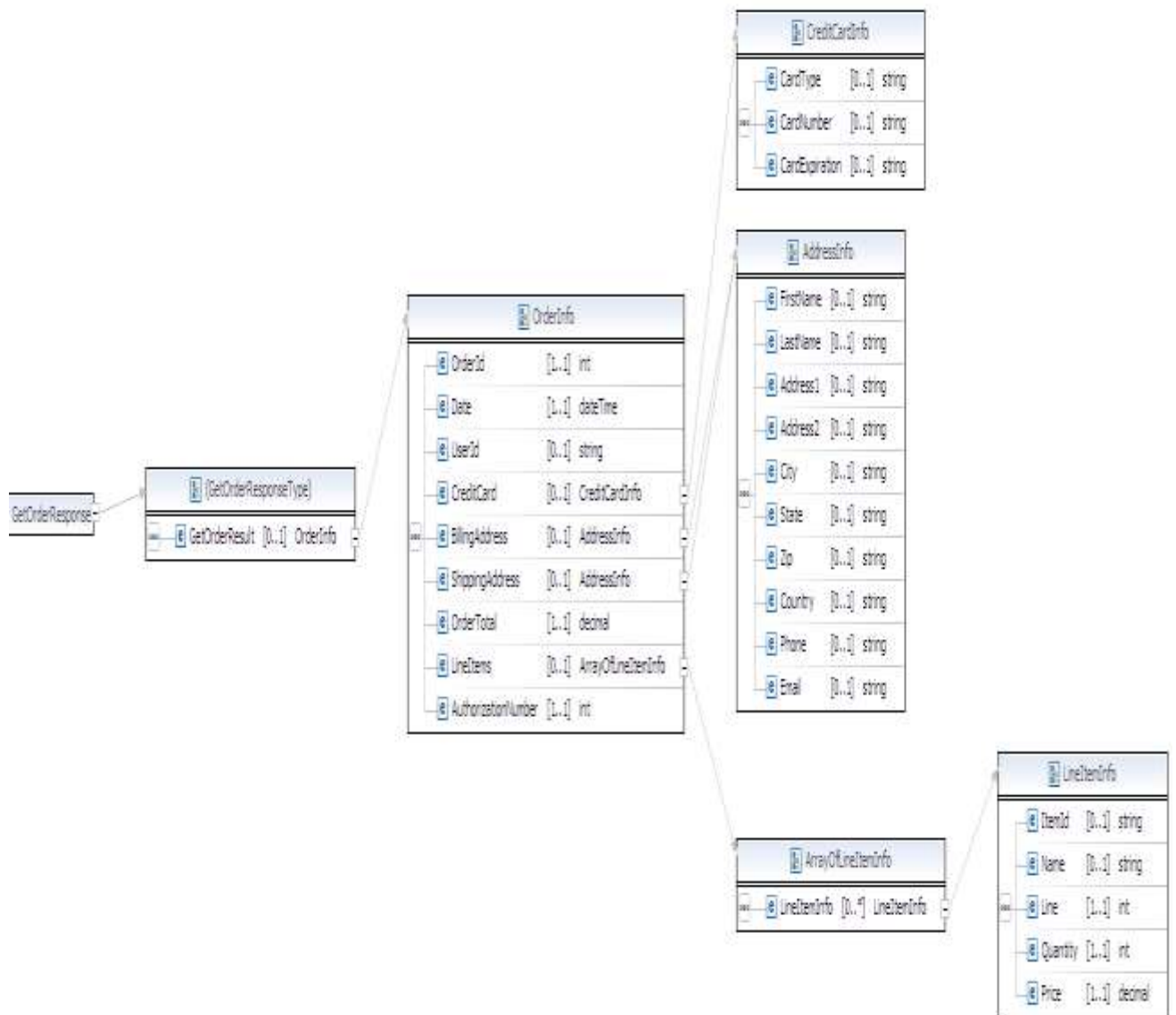
A continuación se muestra el gráfico de la interfase y su objeto relacionado “Order”.

Figura A.2.1.3

Diagrama del WSDL de pedidos

² Ver Tabla A.1.3.1-Catálogo de Interfa





WSDL de Pedidos:

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:tns="http://tempuri.org/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://tempuri.org">

  <wsdl:types>

    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org">

      <s:element name="HelloWorld">

        <s:complexType/>

      </s:element>

      <s:element name="HelloWorldResponse">

        <s:complexType>

          <s:sequence>

            <s:element maxOccurs="1" minOccurs="0" name="HelloWorldResult"
type="s:string"/>

          </s:sequence>

        </s:complexType>

      </s:element>

    </s:schema>

  </wsdl:types>

</wsdl:definitions>
```

```

<s:element name="obtenerOrderInfo">
  <s:complexType/>
</s:element>
<s:element name="obtenerOrderInfoResponse">
  <s:complexType>
    <s:sequence>
      <s:element maxOccurs="1" minOccurs="0" name="obtenerOrderInfoResult"
type="tns:ArrayOfOrderInfo"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfOrderInfo">
  <s:sequence>
    <s:element maxOccurs="unbounded" minOccurs="0" name="OrderInfo"
nillable="true" type="tns:OrderInfo"/>
  </s:sequence>
</s:complexType>
<s:complexType name="OrderInfo">
  <s:sequence>
    <s:element maxOccurs="1" minOccurs="1" name="OrderId" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="Date" type="s:dateTime"/>
    <s:element maxOccurs="1" minOccurs="0" name="UserId" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="0" name="CreditCard"
type="tns:CreditCardInfo"/>
  </s:sequence>
</s:complexType>

```

```

    <s:element maxOccurs="1" minOccurs="0" name="BillingAddress"
type="tns:AddressInfo"/>

    <s:element maxOccurs="1" minOccurs="0" name="ShippingAddress"
type="tns:AddressInfo"/>

    <s:element maxOccurs="1" minOccurs="1" name="OrderTotal"
type="s:decimal"/>

    <s:element maxOccurs="1" minOccurs="0" name="LineItems"
type="tns:ArrayOfLineItemInfo"/>

    <s:element maxOccurs="1" minOccurs="1" name="AuthorizationNumber"
nillable="true" type="s:int"/>

</s:sequence>
</s:complexType>
<s:complexType name="CreditCardInfo">
  <s:sequence>
    <s:element maxOccurs="1" minOccurs="0" name="CardType" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="0" name="CardNumber"
type="s:string"/>
    <s:element maxOccurs="1" minOccurs="0" name="CardExpiration"
type="s:string"/>
  </s:sequence>
</s:complexType>
<s:complexType name="AddressInfo">
  <s:sequence>
    <s:element maxOccurs="1" minOccurs="0" name="FirstName" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="0" name="LastName" type="s:string"/>

```

```

<s:element maxOccurs="1" minOccurs="0" name="Address1" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="Address2" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="City" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="State" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="Zip" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="Country" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="Phone" type="s:string"/>
<s:element maxOccurs="1" minOccurs="0" name="Email" type="s:string"/>
</s:sequence>
</s:complexType>
<s:complexType name="ArrayOfLineItemInfo">
  <s:sequence>
    <s:element maxOccurs="unbounded" minOccurs="0" name="LineItemInfo"
nillable="true" type="tns:LineItemInfo"/>
  </s:sequence>
</s:complexType>
<s:complexType name="LineItemInfo">
  <s:sequence>
    <s:element maxOccurs="1" minOccurs="0" name="ItemId" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="0" name="Name" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="Line" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="Quantity" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="Price" type="s:decimal"/>
  </s:sequence>
</s:complexType>

```

```

    </s:sequence>
</s:complexType>
<s:element name="obtenerOrderInfoByDate">
  <s:complexType>
    <s:sequence>
      <s:element maxOccurs="1" minOccurs="1" name="date" type="s:string"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="obtenerOrderInfoByDateResponse">
  <s:complexType>
    <s:sequence>
      <s:element
        maxOccurs="1"
        minOccurs="0"
        name="obtenerOrderInfoByDateResult" type="tns:ArrayOfOrderInfo"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="string" nillable="true" type="s:string"/>
<s:element
  name="ArrayOfOrderInfo"
  nillable="true"
  type="tns:ArrayOfOrderInfo"/>
</s:schema>
</wsdl:types>
<wsdl:message name="HelloWorldSoapIn">
  <wsdl:part element="tns:HelloWorld" name="parameters"/>

```



```
</wsdl:message>
<wsdl:message name="HelloWorldSoapOut">
  <wsdl:part element="tns:HelloWorldResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="obtenerOrderInfoSoapIn">
  <wsdl:part element="tns:obtenerOrderInfo" name="parameters"/>
</wsdl:message>
<wsdl:message name="obtenerOrderInfoSoapOut">
  <wsdl:part element="tns:obtenerOrderInfoResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="obtenerOrderInfobyDateSoapIn">
  <wsdl:part element="tns:obtenerOrderInfobyDate" name="parameters"/>
</wsdl:message>
<wsdl:message name="obtenerOrderInfobyDateSoapOut">
  <wsdl:part element="tns:obtenerOrderInfobyDateResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="HelloWorldHttpGetIn"/>
<wsdl:message name="HelloWorldHttpGetOut">
  <wsdl:part element="tns:string" name="Body"/>
</wsdl:message>
<wsdl:message name="obtenerOrderInfoHttpGetIn"/>
<wsdl:message name="obtenerOrderInfoHttpGetOut">
  <wsdl:part element="tns:ArrayOfOrderInfo" name="Body"/>
</wsdl:message>
```

```

</wsdl:message>

<wsdl:message name="obtenerOrderInfobyDateHttpGetIn">
  <wsdl:part name="date" type="s:string"/>
</wsdl:message>

<wsdl:message name="obtenerOrderInfobyDateHttpGetOut">
  <wsdl:part element="tns:ArrayOfOrderInfo" name="Body"/>
</wsdl:message>

<wsdl:message name="HelloWorldHttpPostIn"/>
<wsdl:message name="HelloWorldHttpPostOut">
  <wsdl:part element="tns:string" name="Body"/>
</wsdl:message>

<wsdl:message name="obtenerOrderInfoHttpPostIn"/>
<wsdl:message name="obtenerOrderInfoHttpPostOut">
  <wsdl:part element="tns:ArrayOfOrderInfo" name="Body"/>
</wsdl:message>

<wsdl:message name="obtenerOrderInfobyDateHttpPostIn">
  <wsdl:part name="date" type="s:string"/>
</wsdl:message>

<wsdl:message name="obtenerOrderInfobyDateHttpPostOut">
  <wsdl:part element="tns:ArrayOfOrderInfo" name="Body"/>
</wsdl:message>

<wsdl:portType name="OrderWSSoap">
  <wsdl:operation name="HelloWorld">

```

```

<wsdl:input message="tns:HelloWorldSoapIn"/>
<wsdl:output message="tns:HelloWorldSoapOut"/>
</wsdl:operation>
<wsdl:operation name="obtenerOrderInfo">
  <wsdl:input message="tns:obtenerOrderInfoSoapIn"/>
  <wsdl:output message="tns:obtenerOrderInfoSoapOut"/>
</wsdl:operation>
<wsdl:operation name="obtenerOrderInfobyDate">
  <wsdl:input message="tns:obtenerOrderInfobyDateSoapIn"/>
  <wsdl:output message="tns:obtenerOrderInfobyDateSoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:portType name="OrderWSHttpGet">
  <wsdl:operation name="HelloWorld">
    <wsdl:input message="tns:HelloWorldHttpGetIn"/>
    <wsdl:output message="tns:HelloWorldHttpGetOut"/>
  </wsdl:operation>
  <wsdl:operation name="obtenerOrderInfo">
    <wsdl:input message="tns:obtenerOrderInfoHttpGetIn"/>
    <wsdl:output message="tns:obtenerOrderInfoHttpGetOut"/>
  </wsdl:operation>
  <wsdl:operation name="obtenerOrderInfobyDate">
    <wsdl:input message="tns:obtenerOrderInfobyDateHttpGetIn"/>

```

```

    <wsdl:output message="tns:obtenerOrderInfobyDateHttpGetOut"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="OrderWSHttpPost">
  <wsdl:operation name="HelloWorld">
    <wsdl:input message="tns:HelloWorldHttpPostIn"/>
    <wsdl:output message="tns:HelloWorldHttpPostOut"/>
  </wsdl:operation>
  <wsdl:operation name="obtenerOrderInfo">
    <wsdl:input message="tns:obtenerOrderInfoHttpPostIn"/>
    <wsdl:output message="tns:obtenerOrderInfoHttpPostOut"/>
  </wsdl:operation>
  <wsdl:operation name="obtenerOrderInfobyDate">
    <wsdl:input message="tns:obtenerOrderInfobyDateHttpPostIn"/>
    <wsdl:output message="tns:obtenerOrderInfobyDateHttpPostOut"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="OrderWSSoap" type="tns:OrderWSSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="HelloWorld">
    <soap:operation soapAction="http://tempuri.org/HelloWorld" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>

```

```

</wsdl:input>
<wsdl:output>
  <soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="obtenerOrderInfo">
  <soap:operation
    style="document"
    soapAction="http://tempuri.org/obtenerOrderInfo"
  >
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
<wsdl:operation name="obtenerOrderInfofobyDate">
  <soap:operation
    style="document"
    soapAction="http://tempuri.org/obtenerOrderInfofobyDate"
  >
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>

```

```

</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="OrderWSSoap12" type="tns:OrderWSSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="HelloWorld">
    <soap12:operation soapAction="http://tempuri.org/HelloWorld"
style="document"/>
    <wsdl:input>
      <soap12:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="obtenerOrderInfo">
    <soap12:operation soapAction="http://tempuri.org/obtenerOrderInfo"
style="document"/>
    <wsdl:input>
      <soap12:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>

```

```

<wsdl:operation name="obtenerOrderInfobyDate">
  <soap12:operation      soapAction="http://tempuri.org/obtenerOrderInfobyDate"
style="document"/>
  <wsdl:input>
    <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap12:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="OrderWSHttpGet" type="tns:OrderWSHttpGet">
  <http:binding verb="GET"/>
  <wsdl:operation name="HelloWorld">
    <http:operation location="/HelloWorld"/>
    <wsdl:input>
      <http:urlEncoded/>
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body"/>
    </wsdl:output>
  </wsdl:operation>
<wsdl:operation name="obtenerOrderInfo">

```

```
<http:operation location="/obtenerOrderInfo"/>
<wsdl:input>
  <http:urlEncoded/>
</wsdl:input>
<wsdl:output>
  <mime:mimeXml part="Body"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="obtenerOrderInfobyDate">
  <http:operation location="/obtenerOrderInfobyDate"/>
  <wsdl:input>
    <http:urlEncoded/>
  </wsdl:input>
  <wsdl:output>
    <mime:mimeXml part="Body"/>
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="OrderWSHttpPost" type="tns:OrderWSHttpPost">
  <http:binding verb="POST"/>
  <wsdl:operation name="HelloWorld">
    <http:operation location="/HelloWorld"/>
    <wsdl:input>
```



```
<mime:content type="application/x-www-form-urlencoded"/>
</wsdl:input>
<wsdl:output>
  <mime:mimeXml part="Body"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="obtenerOrderInfo">
  <http:operation location="/obtenerOrderInfo"/>
  <wsdl:input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </wsdl:input>
  <wsdl:output>
    <mime:mimeXml part="Body"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="obtenerOrderInfobyDate">
  <http:operation location="/obtenerOrderInfobyDate"/>
  <wsdl:input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </wsdl:input>
  <wsdl:output>
    <mime:mimeXml part="Body"/>
  </wsdl:output>
```

```

</wsdl:operation>
</wsdl:binding>
<wsdl:service name="OrderWS">
  <wsdl:port binding="tns:OrderWSSoap" name="OrderWSSoap">
    <soap:address location="http://100.100.100.1:100/OrderWS.asmx"/>
  </wsdl:port>
  <wsdl:port binding="tns:OrderWSSoap12" name="OrderWSSoap12">
    <soap12:address location="http://100.100.100.1:100/OrderWS.asmx"/>
  </wsdl:port>
  <wsdl:port binding="tns:OrderWSHttpGet" name="OrderWSHttpGet">
    <http:address location="http://100.100.100.1:100/OrderWS.asmx"/>
  </wsdl:port>
  <wsdl:port binding="tns:OrderWSHttpPost" name="OrderWSHttpPost">
    <http:address location="http://100.100.100.1:100/OrderWS.asmx"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

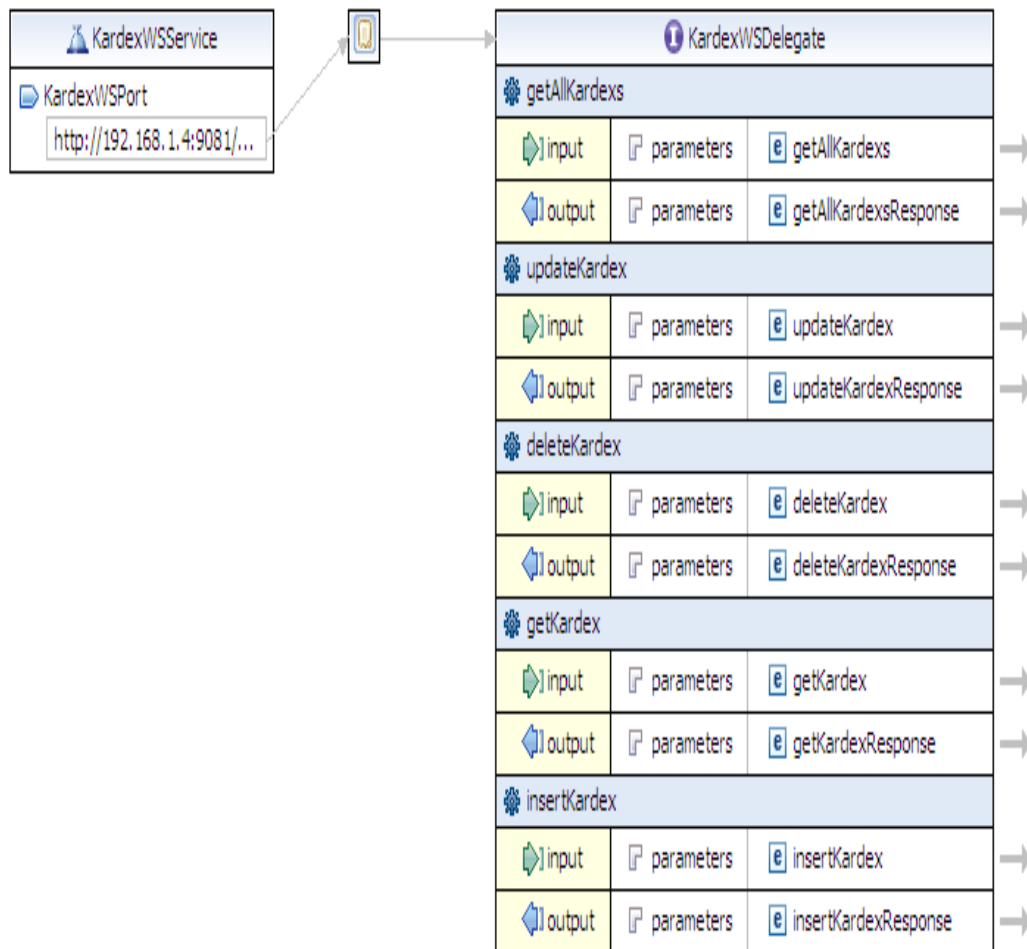
SCON-001-KARDEX y SCON-001-ASIENTO

Gráfico

En el siguiente gráfico se muestran ambas interfaces y su objeto relacionado “kardex” y “asiento”, cabe resaltar que se implementaron ambas interfaces mediante el mismo servicio web y por tanto WSDL.

Figura A.2.1.4

Diagrama del WSDL de Kardex



UNIVERSITAT DE VALÈNCIA
SCIENTIA ET PRAXIS

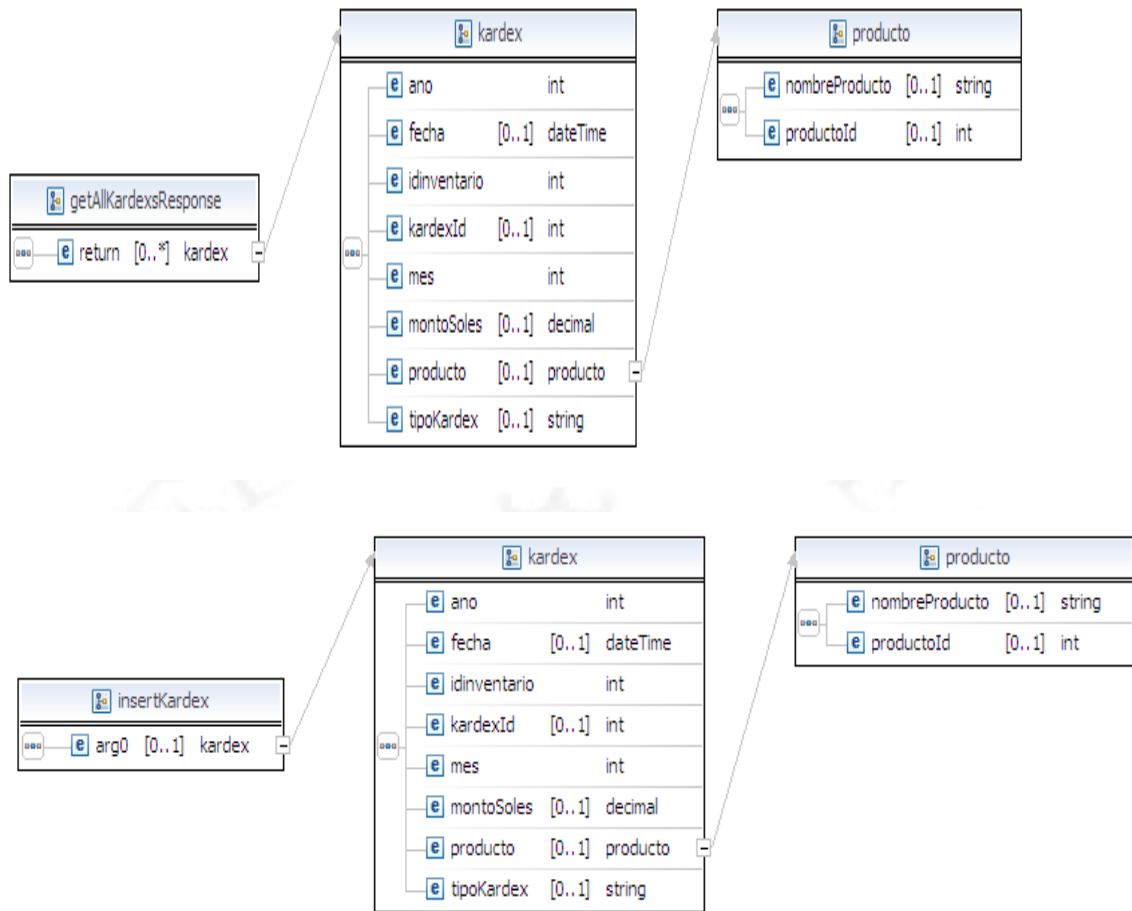
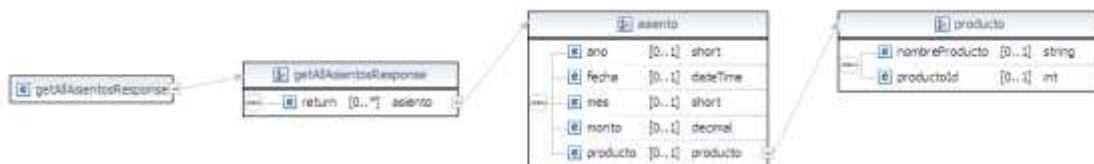
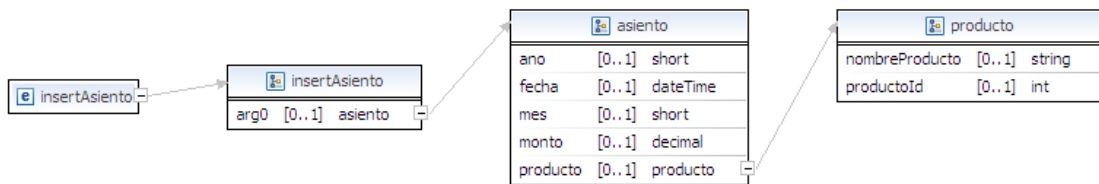
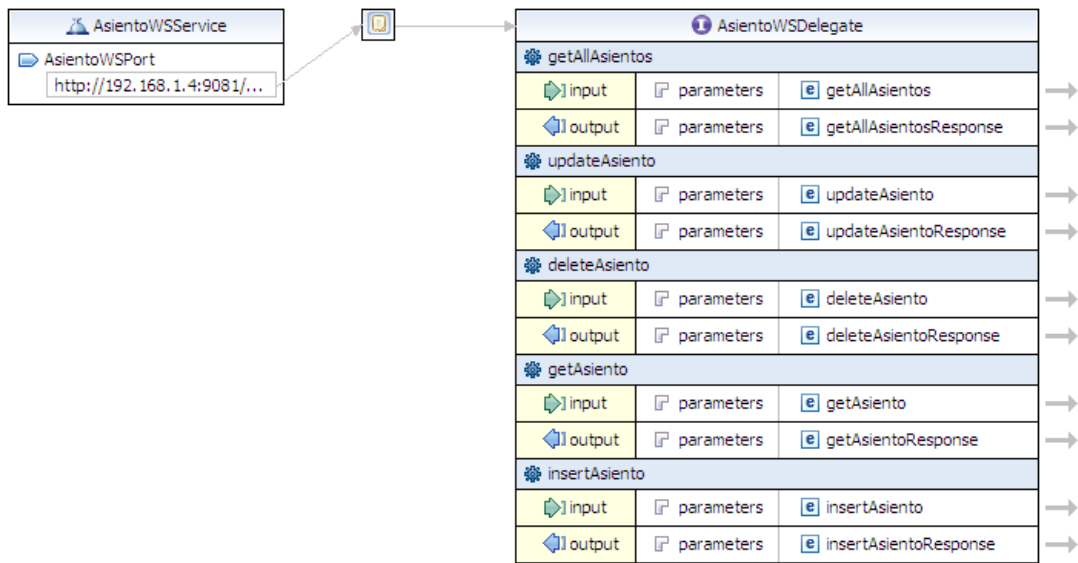


Figura A.2.1.5

Diagrama del WSDL de Asiento



WSDL de Asiento

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<definitions name="ProxyWSService" targetNamespace="http://ws.scontable/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://ws.scontable/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<types>

  <xsd:schema>

    <xsd:import
      namespace="http://ws.scontable/"
      schemaLocation="ProxyWSService_schema1.xsd"/>

  </xsd:schema>

</types>

<message name="deleteAsientoResponse">

  <part name="parameters" element="tns:deleteAsientoResponse">

  </part>

</message>

<message name="updateAsiento">

  <part name="parameters" element="tns:updateAsiento">

  </part>

</message>

<message name="getKardexResponse">

  <part name="parameters" element="tns:getKardexResponse">

  </part>

</message>

<message name="insertKardex">

  <part name="parameters" element="tns:insertKardex">

  </part>

</message>
```

```
<message name="getProducto">
  <part name="parameters" element="tns:getProducto">
  </part>
</message>

<message name="insertProductoResponse">
  <part name="parameters" element="tns:insertProductoResponse">
  </part>
</message>

<message name="insertAsientoResponse">
  <part name="parameters" element="tns:insertAsientoResponse">
  </part>
</message>

<message name="getAllAsientos">
  <part name="parameters" element="tns:getAllAsientos">
  </part>
</message>

<message name="updateKardexResponse">
  <part name="parameters" element="tns:updateKardexResponse">
  </part>
</message>

<message name="getAsientoResponse">
  <part name="parameters" element="tns:getAsientoResponse">
  </part>
```

```
</message>
<message name="deleteKardex">
  <part name="parameters" element="tns:deleteKardex">
  </part>
</message>
<message name="getAllProductosResponse">
  <part name="parameters" element="tns:getAllProductosResponse">
  </part>
</message>
<message name="updateProducto">
  <part name="parameters" element="tns:updateProducto">
  </part>
</message>
<message name="deleteProductoResponse">
  <part name="parameters" element="tns:deleteProductoResponse">
  </part>
</message>
<message name="getAllKardexsResponse">
  <part name="parameters" element="tns:getAllKardexsResponse">
  </part>
</message>
<message name="deleteAsiento">
  <part name="parameters" element="tns:deleteAsiento">
```



```
</part>
</message>
<message name="getKardex">
  <part name="parameters" element="tns:getKardex">
    </part>
  </message>
<message name="updateAsientoResponse">
  <part name="parameters" element="tns:updateAsientoResponse">
    </part>
  </message>
<message name="insertKardexResponse">
  <part name="parameters" element="tns:insertKardexResponse">
    </part>
  </message>
<message name="getProductoResponse">
  <part name="parameters" element="tns:getProductoResponse">
    </part>
  </message>
<message name="insertProducto">
  <part name="parameters" element="tns:insertProducto">
    </part>
  </message>
<message name="insertAsiento">
```

```
<part name="parameters" element="tns:insertAsiento">
</part>
</message>
<message name="getAllAsientosResponse">
  <part name="parameters" element="tns:getAllAsientosResponse">
  </part>
</message>
<message name="updateKardex">
  <part name="parameters" element="tns:updateKardex">
  </part>
</message>
<message name="getAsiento">
  <part name="parameters" element="tns:getAsiento">
  </part>
</message>
<message name="deleteKardexResponse">
  <part name="parameters" element="tns:deleteKardexResponse">
  </part>
</message>
<message name="getAllProductos">
  <part name="parameters" element="tns:getAllProductos">
  </part>
</message>
```

```
<message name="updateProductoResponse">
  <part name="parameters" element="tns:updateProductoResponse">
  </part>
</message>

<message name="deleteProducto">
  <part name="parameters" element="tns:deleteProducto">
  </part>
</message>

<message name="getAllKardexs">
  <part name="parameters" element="tns:getAllKardexs">
  </part>
</message>

<portType name="ProxyWSDelegate">
  <operation name="getAllAsientos">
    <input message="tns:getAllAsientos">
    </input>
    <output message="tns:getAllAsientosResponse">
    </output>
  </operation>
  <operation name="updateAsiento">
    <input message="tns:updateAsiento">
    </input>
    <output message="tns:updateAsientoResponse">

```

```
</output>
</operation>
<operation name="deleteAsiento">
  <input message="tns:deleteAsiento">
</input>
  <output message="tns:deleteAsientoResponse">
</output>
</operation>
<operation name="getAsiento">
  <input message="tns:getAsiento">
</input>
  <output message="tns:getAsientoResponse">
</output>
</operation>
<operation name="insertAsiento">
  <input message="tns:insertAsiento">
</input>
  <output message="tns:insertAsientoResponse">
</output>
</operation>
<operation name="getAllKardexs">
  <input message="tns:getAllKardexs">
</input>
```

```
<output message="tns:getAllKardexsResponse">
</output>
</operation>
<operation name="updateKardex">
  <input message="tns:updateKardex">
</input>
  <output message="tns:updateKardexResponse">
</output>
</operation>
<operation name="deleteKardex">
  <input message="tns:deleteKardex">
</input>
  <output message="tns:deleteKardexResponse">
</output>
</operation>
<operation name="getKardex">
  <input message="tns:getKardex">
</input>
  <output message="tns:getKardexResponse">
</output>
</operation>
<operation name="insertKardex">
  <input message="tns:insertKardex">
```

```
</input>
  <output message="tns:insertKardexResponse">
</output>
</operation>
<operation name="getProducto">
  <input message="tns:getProducto">
</input>
  <output message="tns:getProductoResponse">
</output>
</operation>
<operation name="insertProducto">
  <input message="tns:insertProducto">
</input>
  <output message="tns:insertProductoResponse">
</output>
</operation>
<operation name="deleteProducto">
  <input message="tns:deleteProducto">
</input>
  <output message="tns:deleteProductoResponse">
</output>
</operation>
<operation name="updateProducto">
```

```
<input message="tns:updateProducto">
</input>
<output message="tns:updateProductoResponse">
</output>
</operation>
<operation name="getAllProductos">
  <input message="tns:getAllProductos">
</input>
  <output message="tns:getAllProductosResponse">
</output>
</operation>
</portType>
<binding name="ProxyWSPortBinding" type="tns:ProxyWSDelegate">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getAllAsientos">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
```

```
<operation name="updateAsiento">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="deleteAsiento">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="getAsiento">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
```



```
<output>
  <soap:body use="literal"/>
</output>
</operation>
<operation name="insertAsiento">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="getAllKardexs">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="updateKardex">
```

```
<soap:operation soapAction=""/>
<input>
  <soap:body use="literal"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
<operation name="deleteKardex">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="getKardex">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
```

```
<soap:body use="literal"/>
</output>
</operation>
<operation name="insertKardex">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="getProducto">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="insertProducto">
  <soap:operation soapAction=""/>
```

```
<input>
  <soap:body use="literal"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
<operation name="deleteProducto">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="updateProducto">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
```

```
</output>
</operation>
<operation name="getAllProductos">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>
<service name="ProxyWSService">
  <port name="ProxyWSPort" binding="tns:ProxyWSPortBinding">
    <soap:address location="http://100.100.100.1:9081/SContable/ProxyWSService"/>
  </port>
</service>
</definitions>
```

Anexo 5: A.2.2 - Documento de Arquitectura (Final)



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 21- Noviembre-2009 - jcabrera

Modificaciones:

5 Diciembre 2009 – jcabrera

Principios de Arquitectura:

Los principios de arquitectura contemplados en la siguiente arquitectura son:

1. Escalabilidad: Las aplicaciones deben de conservar o mejorar su nivel de escalabilidad en lo referente a su capacidad transaccional.

Justificación: Dada la naturaleza y expectativas de crecimiento de la empresa, es necesario que en la medida de lo posible aumentar las capacidades transaccionales de la plataforma mediante el aumento de (capacidad computacional y/o nodos adicionales) no requiera modificaciones sobre las aplicaciones.

2. Interoperabilidad: Las aplicaciones deben de mantener un nivel adecuado de interoperabilidad entre si mediante tecnologías estándares.

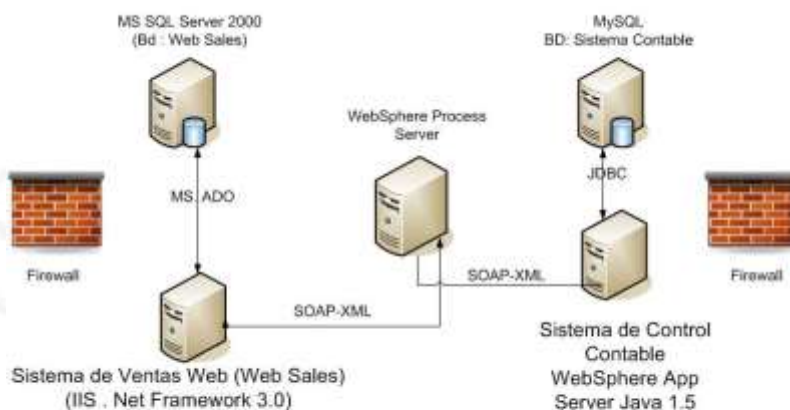
Justificación: Tomando en cuenta la diversidad de plataformas actuales y reconociendo la necesidad del intercambio de datos entre dichas aplicaciones, se requiere la utilización de interfaces con un nivel alto de interoperabilidad y que sigan, en la medida de lo posible, estándares abiertos soportados por la mayoría o todas las aplicaciones y/o plataformas de la empresa.

Arquitectura Propuesta:

A continuación se muestra un gráfico con la arquitectura propuesta para la integración de ambas aplicaciones.

Figura A.2.2.1

Diagrama de arquitectura propuesta



Descripción:

Como se puede observar los servidores se encuentran dentro de una DMZ (zona desmilitarizada) dentro de la cual existe comunicación libre y de confianza entre los servidores. Hacia fuera de la DMZ solo se permite el tráfico http, los nodos relacionados al proyecto son:

1. Servidor IIS: Donde se encuentra la aplicación de Ventas Web (Web Sales), la cual está desarrollada con Servidor MS
2. SQL Server: Nodo donde se encuentra la base de datos del Sistema de Ventas Web
3. Web Sphere App Server: Nodo donde se encuentra el Sistema de Gestión Contable que está siendo desarrollado.

4. MySQL: Nodo que contiene la base de Datos MySQL que aloja las tablas de el Sistema Contable.

5. IBM Process Server: Plataforma de Integración.

Decisiones de Arquitectura

A continuación se presenta y justifican las decisiones de arquitectura:

1. Utilización de servicios Web y SOI: La finalidad de esta decisión de arquitectura es lograr las ventajas que SOI plantea tales como rehúso de interfaces y reducción del tiempo de desarrollo.
2. Utilización del estándar SOAP: Esta decisión se sustenta en la capacidad de ambas plataformas para soportar dicho protocolo para el intercambio de datos.
3. Utilización de un BUS de integración: Esta decisión busca poder reutilizar las interfaces desarrolladas y se toma previendo que los sistemas involucrados deberán de intercambiar información con otros sistemas y entre si en un futuro próximo.
4. Proceso en Batch para transferencia de Órdenes: Con la finalidad de afectar lo menos posible la performance sobre la actual aplicación de ventas por Internet, la trasferencia de órdenes se hará una vez al día durante la hora de menos uso del sistema.
5. Se decidió mapear el componente BUS de integración al software IBM Process Server dado que cuenta con las capacidades necesarias para el presente proyecto y se ha definido con plataforma de integración de la empresa.

Diagrama de Interfaces

Las interfaces identificadas coinciden con las detalladas en la Figura A.1.2.2- Diagrama de Interfaces

Para las interfaces se tomó la siguiente convención:

- ABCDE-9999-NOMBRE: donde ABC son las siglas de el sistema, “9999” el número de interfaz relativa al sistema y “NOMBRE” el nombre de dicha interfaz.



Anexo 6: A.3.1 – Código Fuente



A continuación se presenta el código fuente de los métodos a partir de los cuales se generaron los Servicios Web.

Cabe señalar que ambas plataformas la generación de los servicios fue automática.

Código Fuente Aplicación: Sistema de Ventas Web

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
using PetShop.BLL;
using PetShop.Model;

/// <summary>
/// Clase para el manejo de los servicios web relacionados al Objeto /// Order
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class OrderWS : System.Web.Services.WebService
{
```

```
Order order = new Order();
```

```
public OrderWS()
```

```
{
```

```
    //Constructor de la clase
```

```
}
```

```
//Metodo de Prueba para el servicio
```

```
[WebMethod] //Indica que es un Servicio Web
```

```
public string HelloWorld()
```

```
{
```

```
    return "Hello World";
```

```
}
```

```
//Obtiene todas las ordenes del dia en curso
```

```
[WebMethod] //Indica que es un Servicio Web
```

```
public OrderInfo[] obtenerOrderInfo()
```

```
{
```

```
    Order order = new Order();
```

```
    ArrayList a = order.GetAllOrders();
```

```
    OrderInfo[] ass = new OrderInfo[a.Count];
```

```
    int i = 0;
```

```
    foreach (OrderInfo orden in a)
```

```
{  
    ass[i] = orden;  
    i++;  
}  
return ass;  
}
```

//Obtiene todas las ordenes del día enviando como parámetro

[WebMethod] //Indica que es un Servicio Web

public OrderInfo[] obtenerOrderInfofyDate(DateTime date)

```
{  
    Order order = new Order();  
    ArrayList a = order.GetAllOrdersbyDate(date);  
    OrderInfo[] ass = new OrderInfo[a.Count];  
    int i = 0;  
    foreach (OrderInfo orden in a)  
    {  
        ass[i] = orden;  
        i++;  
    }  
    return ass;  
}
```

Código Fuente Aplicación: Sistema Contable

```
package scontable.ws;

import java.util.List;

import scontable.persistence.KardexHibernateDao;

import scontable.vo.Kardex;

public class KardexWS {

    private KardexHibernateDao dao = new KardexHibernateDao();

    /**
     *
     * @return
     */
    public Kardex[] getAllKardexs() {
        List<Kardex> k = dao.getAllKardexs();

        Kardex[] kardex = new Kardex[k.size()];

        for(int i = 0; i < k.size() ; i++ ){

            kardex[i] = k.get(i);
        }
    }
}
```

```

    }

    return kardex;
}

/**
 *
 * @param kardex
 */
c public void updateKardex(Kardex kardex) {
    dao.update(kardex);
}

/**
 *
 * @param id
 */
public void deleteKardex(int id) {
    dao.delete((Integer)id);
}

/**
 *
 * @param id
 * @return
 */
public Kardex getKardex(int id) {

```



```
        return dao.getKardex((Integer)id);
    }

    /**
     *
     * @param kardex
     */
    public void insertKardex(Kardex kardex) {
        dao.insert(kardex);
    }
}
```

```
package scontable.ws;
```

```
import java.util.List;
```

```
import scontable.persistence.AsientoHibernateDAO;
```

```
import scontable.vo.Asiento;
```

```
public class AsientoWS {
```

```

private AsientoHibernateDAO dao = new AsientoHibernateDAO();

/**
 * Retorna una lista de todos los asientos creados
 * @return Asiento[]
 */
public Asiento[] getAllAsientos() {
    List<Asiento> a = dao.getAllAsientos();
    Asiento[] ass = new Asiento[a.size()];
    for(int i = 0; i < a.size() ; i++ ){
        ass[i] = a.get(i);
    }
    return ass;
}

/**
 * Actualize asiento
 * @param ass the Asiento objeto to be updated
 */
public void updateAsiento(Asiento ass) {
    dao.update(ass);
}

/**

```

```

* Delete an Asiento

* @param id int the id off the Asiento to be deleted

*/

public void deleteAsiento(int id) {

    dao.delete((Integer)id);

}

/**

* Returns an Asiento

* @param id int the id off the Asiento getted

* @return

*/

public Asiento getAsiento(int id) {

    return dao.getAsiento((Integer)id);

}

/**Creates a new Asiento

*

* @param ass Object Asiento to be created

*/

public void insertAsiento(Asiento ass) {

    dao.insert(ass);

}

```

```
}
```

```
package scontable.ws;
```

```
import java.util.List;
```

```
import scontable.persistence.AsientoHibernateDAO;
```

```
import scontable.persistence.KardexHibernateDao;
```

```
import scontable.persistence.ProductoDao;
```

```
import scontable.persistence.ProductoHibernateDao;
```

```
import scontable.vo.Asiento;
```

```
import scontable.vo.Kardex;
```

```
import scontable.vo.Producto;
```

```
public class ProxyWS {
```

```
    private AsientoHibernateDAO dao = new AsientoHibernateDAO();
```

```
    /**
```

```
     * Retorna una lista de todos los asientos creados
```

```
     * @return Asiento[]
```

```
     */
```

```
    public Asiento[] getAllAsientos() {
```

```

        List<Asiento> a = dao.getAllAsientos();

        Asiento[] ass = new Asiento[a.size()];

        for(int i = 0;i< a.size() ; i++ ){

            ass[i] = a.get(i);

        }

        return ass;
    }

    /**
     * Actualize asiento
     * @param ass the Asiento objeto to be updated
     */
    public void updateAsiento(Asiento ass) {

        dao.update(ass);

    }

    /**
     * Delete an Asiento
     * @param id int the id off the Asiento to be deleted
     */
    public void deleteAsiento(int id) {

        dao.delete((Integer)id);

    }

    /**

```

```

* Returns an Asiento

* @param id int the id off the Asiento getted

* @return

*/

public Asiento getAsiento(int id) {
    return dao.getAsiento((Integer)id);
}

/**Creates a new Asiento

*

* @param ass Object Asiento to be created

*/

public void insertAsiento(Asiento ass) {
    dao.insert(ass);
}

private KardexHibernateDao daoass = new KardexHibernateDao();

/**

*

* @return

*/

```

```

public Kardex[] getAllKardexs() {
    List<Kardex> k = daoass.getAllKardexs();
    Kardex[] kardex = new Kardex[k.size()];
    for(int i = 0;i< k.size() ; i++ ){
        kardex[i] = k.get(i);
    }
    return kardex;
}
/**
 *
 * @param kardex
 */
public void updateKardex(Kardex kardex) {
    daoass.update(kardex);
}
/**
 *
 * @param id
 */
public void deleteKardex(int id) {
    daoass.delete((Integer)id);
}
/**

```

```
*
* @param id
* @return
*/
public Kardex getKardex(int id) {
    return daoass.getKardex((Integer)id);
}
/**
*
* @param kardex
*/
public void insertKardex(Kardex kardex) {
    daoass.insert(kardex);
}

private ProductoDao daop = new ProductoHibernateDao();

public Producto getProducto(int id) {
    return daop.getProducto((Integer)id);
}
```



```
public void insertProducto(Producto emp) {  
    daop.insert(emp);  
}
```

```
public void deleteProducto(int id){  
    daop.delete((Integer)id);  
}
```

```
public void updateProducto(Producto prod){  
    daop.update(prod);  
}
```

```
public Producto[] getAllProductos() {  
    List<Producto> p = daop.getAllProductos();  
    Producto[] prod = new Producto[p.size()];  
    for(int i = 0; i < p.size() ; i++ ){  
        prod[i] = p.get(i);  
    }  
    return prod;  
}
```

```
}
```

```
package scontable.ws;

import java.util.List;

import scontable.persistence.AsientoHibernateDAO;

import scontable.persistence.KardexHibernateDao;

import scontable.persistence.ProductoDao;

import scontable.persistence.ProductoHibernateDao;

import scontable.vo.Asiento;

import scontable.vo.Kardex;

import scontable.vo.Producto;

@javax.jws.WebService (targetNamespace="http://ws.scontable/",
serviceName="ProxyWSService", portName="ProxyWSPort", wsdlLocation="WEB-INF/wsdl/ProxyWSService.wsdl")

public class ProxyWSDelegate{

    scontable.ws.ProxyWS _proxyWS = null;

    public Asiento[] getAllAsientos() {

        return _proxyWS.getAllAsientos();

    }

}
```

```
public void updateAsiento(Asiento ass) {
    _proxyWS.updateAsiento(ass);
}

public void deleteAsiento(int id) {
    _proxyWS.deleteAsiento(id);
}

public Asiento getAsiento(int id) {
    return _proxyWS.getAsiento(id);
}

public void insertAsiento(Asiento ass) {
    _proxyWS.insertAsiento(ass);
}

public Kardex[] getAllKardexs() {
    return _proxyWS.getAllKardexs();
}

public void updateKardex(Kardex kardex) {
    _proxyWS.updateKardex(kardex);
}
```

```
public void deleteKardex(int id) {  
    _proxyWS.deleteKardex(id);  
}
```

```
public Kardex getKardex(int id) {  
    return _proxyWS.getKardex(id);  
}
```

```
public void insertKardex(Kardex kardex) {  
    _proxyWS.insertKardex(kardex);  
}
```

```
public Producto getProducto(int id) {  
    return _proxyWS.getProducto(id);  
}
```

```
public void insertProducto(Producto emp) {  
    _proxyWS.insertProducto(emp);  
}
```

```
public void deleteProducto(int id) {  
    _proxyWS.deleteProducto(id);  
}
```

```
}

public void updateProducto(Producto prod) {
    _proxyWS.updateProducto(prod);
}

public Producto[] getAllProductos() {
    return _proxyWS.getAllProductos();
}

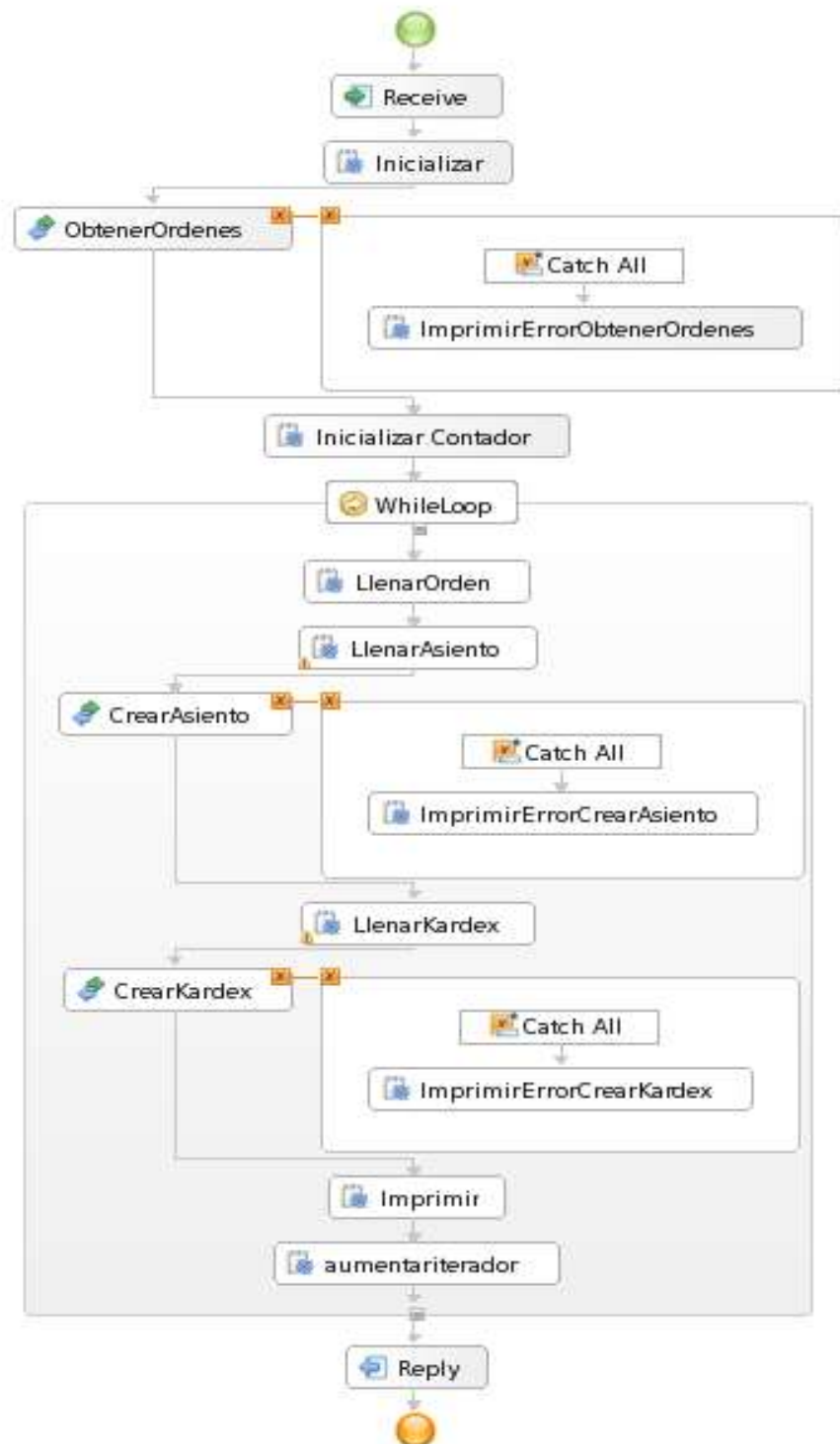
public ProxyWSDelegate() {
    _proxyWS = new scontable.ws.ProxyWS();
}
}
```

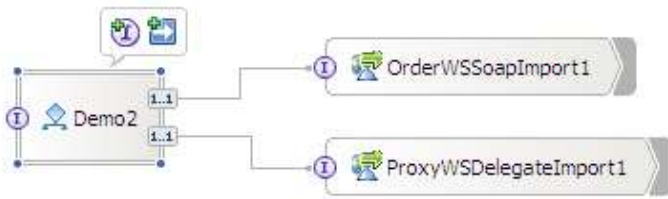
Código Fuente Integración (Process Server)

A continuación se presenta el modelo generado en Process Server para la integración de ambas aplicaciones. Este modelo debe ser ubicado en el componente de BUS de integración en este caso (process Server):

Figura A.3.1.1

Captura pantalla Modelo Process.





Anexo 7: A.4.2 – Diseño de Pruebas



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 05- Diciembre -2009 - jcabrera

Modificaciones:

Objetivo:

El objetivo del presente documento es el diseño de las pruebas para el proyecto de “Integración Web Sales/Sistema Contable”.

Dado que el proyecto de implementación de los sistemas a ser integrados (Proyecto Web Sales y Proyecto Sistema Contable) son los encargados de asegurar la verificación de los requerimientos y las funcionalidades del sistema, las pruebas consideradas se remiten a la validación de las funcionalidades y el grado en que han sido impactadas de ser el caso.

Tomando en cuenta la naturaleza del proyecto, las pruebas deben estar diseñadas a cumplir los siguientes objetivos:

- Validar que las funcionalidades no han sido afectadas funcionalmente.
- Validar que las interfaces funcionan adecuadamente
- Validar que los datos han sido correctamente trasladados de un sistema a otro

Tipos de Pruebas Diseñadas:

Teniendo en cuenta estos objetivos podemos alinear a ellos los siguientes tipos de pruebas:

- Prueba Funcional: Esta prueba consiste en ejecutar la funcionalidad utilizando un caso de prueba verificando el correcto funcionamiento de la funcionalidad contra el caso de prueba y verificando además los datos en la base de datos y/o reporte a fin.

- Prueba de Servicios (o Interfaces): Esta prueba consiste en ejecutar mediante herramientas de software las interfaces, en este caso los servicios web, la prueba consistirá en la ejecución del servicio y la verificación de los resultados contra la base de datos o un reporte a fin.

Recursos y Herramientas:

Para las pruebas se utilizarán las siguientes herramientas y recursos:

- Una PC para pruebas.
- El ambiente de pruebas.
- Navegador de Internet: Para ejecutar las funcionalidades web y/o las herramientas que puedan ser accedidas desde este medio.
- Microsoft Visual Studio 2005 o MS SQL Client : para la ejecución de las sentencias SQL
- MySQL WebAdmin (instalado en el ambiente de pruebas) para la verificación de datos en las tablas de dicha base de datos.

Automatización de Pruebas:

Para este caso no se considera la automatización de pruebas dado que estas no son necesarias para cumplir los objetivos de las pruebas. Esto en atención a que no se realizarán pruebas de performance y no se re-utilizarán las pruebas hechas.

Anexo 8: A.4.3 – Plan de Pruebas



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 21- Noviembre-2009 - jcabrera

Modificaciones:

Objetivos:

El objetivo del presente documento es el plan para la realización de las pruebas diseñadas (ver documento A.4.2- Diseño de Pruebas, Anexo 7), para el proyecto de “Integración Web Sales/Sistema Contable”.

Alcance:

El alcance definido para el presente plan consiste en:

1. Pruebas Funcionales de sobre los casos de uso de :
 - Prueba realizar pedido.
 - Prueba enviar pedidos.
2. Pruebas de Servicios sobre los servicios de :
 - WSAL-001-CPEDIDO (GetAllOrdersbyDate)
 - SCON-001-KARDEX (insertKardex)
 - SCON-001-ASIENTO (insertAsiento)

Recursos:

Para la realización del presente plan se toman en cuenta los siguientes recursos:

- Analista de Pruebas : Jorge Cabrera
- Software: Las herramientas detalladas en el documento de “Diseño de pruebas A.4.2”.
- Equipos : Computadora Personal (1Giga de RAM, CPU de 2.0 , 50 Gigas de HD libres)

- Software:
 - ✓ Navegador de Internet
 - ✓ Rational Software Architect 7.1
 - ✓ Microsot Office
 - ✓ Visual Studio 2005
- Tiempo Estimado 8 horas

Cronograma de Pruebas:

Tabla A.4.3.1 - Cronograma de Pruebas

Tarea	Etapa	Responsable
Elaboración de Casos de Pruebas	Construcción	Jorge Cabrera
Pruebas Funcionales	Pruebas	Jorge Cabrera
Pruebas de Servicios	Pruebas	Jorge Cabrera
Elaboración Informe de Pruebas	Pruebas	Jorge Cabrera

Descripción de Actividades:

Elaboración de Casos de Prueba : Consiste en la construcción de los casos de pruebas en base a los casos de uso

1. Pruebas Funcionales: Ejecución en si de las pruebas sobre los casos de uso.
2. Pruebas de Servicios: Ejecución en si de las pruebas sobre los servicios.
3. Elaboración Informe de de Pruebas: Elaboración del entregable detallando la ejecución y resultados de las pruebas realizadas.

Anexo 9: A.4.4 – Casos de Prueba



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 05- Diciembre -2009 - jcabrera

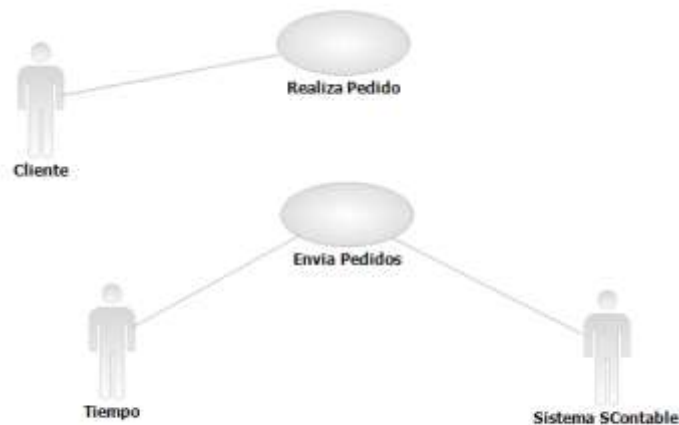
Modificaciones:

Funcionalidades Identificadas:

A continuación se muestran los casos de uso relacionados al proyecto.

Figura A.4.4.1

Diagrama de Casos de Uso



En donde el caso de uso “Realizar Pedido” pertenece al Sistema de Ventas Web, y a su vez el caso de uso “Envía Pedidos” pertenece al Sistema de Integración.

Casos de Prueba:

Se realizarán pruebas sobre los servicios web para verificar su correcto funcionamiento utilizando las herramientas provistas por las herramientas:

- Microsoft Visual Studio 2005.
- Rational Software Architect v7.5.

Caso de Prueba: Prueba realiza pedido

Si bien este caso no ha sido aparentemente afectado se reconoce la importancia de realizar pruebas sobre el mismo, por lo cual se utilizará para crear de productos aleatorios.

Script de Prueba

1. Ingresar 3 pedidos con diversos Ítems cantidades, con los usuarios: demo, AdamBarr, KimAbercrombie, RobYoung y la contraseña pass@word1.
2. Verificar y almacenar los datos mediante la siguiente sentencia SQL:

```
SELECT o.OrderId , o.OrderDate, o.UserId, o.BillToFirstName, o.BillToLastName,  
o.BillAddr1, o.BillAddr2, o.BillCity, o.BillState, BillZip, o.BillCountry,  
o.ShipToFirstName, o.ShipToLastName, o.ShipAddr1, o.ShipAddr2, o.ShipCity,  
o.ShipState, o.ShipZip, o.ShipCountry, o.TotalPrice, l.ItemId, l.LineNum, l.Quantity,  
l.UnitPrice
```

```
FROM Orders as o, lineitem as l
```

```
WHERE (o.OrderDate BETWEEN '2009-12-20' AND '2009-12-21')
```

3. Almacenar los datos mediante una hoja de Excel.

Caso de Prueba: Prueba envía pedidos

Mediante este caso de uso se verificará la transferencia de datos sobre las ordenes y creación de los asientos contables y Kardex correspondientes.

Script de Prueba

1. Cambiar aleatoriamente las fechas a la fecha del día en curso.
2. Ejecutar el modelo de Integración.
3. Verificar la transferencia de datos, mediante el administrador web de MySQL y el Excel resultante de la prueba anterior.

Anexo 10: A.4.1 – Informe de Pruebas



Proyecto: Integración Web Sales/Sistema Contable

Encargado: Jorge Cabrera

Creación: 05- Diciembre -2009 - jcabrera

Modificaciones:

Pruebas Ejecutadas

Se ejecutaron las siguientes pruebas de acuerdo a los siguientes casos de prueba:

1. Pruebas sobre servicios.
2. Prueba realiza pedido.
3. Prueba envía pedidos.

El resultado de las pruebas fue correcto para el caso de prueba definidos, a continuación se muestran los sustentos sobre las pruebas.

Pruebas sobre servicios

Se ejecutó la consulta de Órdenes sobre la aplicación Web Sales, se muestran a continuación las pantallas y wsdl resultado de ejecutar el servicio.

Figura A.4.1.

Captura de Pantalla I



<?xml version="1.0" encoding="utf-8" ?>

= <ArrayOfAnyType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org/">

= <anyType xsi:type="OrderInfo">

<OrderId>1</OrderId>

<Date>2009-12-11T11:24:01</Date>

<UserId>demo</UserId>

= <BillingAddress>

<FirstName>Adam</FirstName>

<LastName>Barr</LastName>

<Address1>Vertigo Software, Inc.</Address1>

<Address2>503A Canal Blvd.</Address2>

<City>Point Richmond</City>

<State>CA</State>

<Zip>94804</Zip>

<Country>USA</Country>

<Email>email</Email>

</BillingAddress>

= <ShippingAddress>

<FirstName>Jorge</FirstName>

<LastName>Luis</LastName>

<Address1>Manuel Aguila Durand</Address1>

<Address2 />

<City>Lima</City>

<State>NY</State>

<Zip>08540</Zip>

<Country>USA</Country>

<Email>email</Email>

</ShippingAddress>

<OrderTotal>120.95</OrderTotal>

= <LineItems>

= <LineItemInfo>

<ItemId>EST-24</ItemId>

```
<Name />
<Line>1</Line>
<Quantity>1</Quantity>
<Price>120.95</Price>
</LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-24</ItemId>
  <Name />
  <Line>2</Line>
  <Quantity>1</Quantity>
  <Price>120.95</Price>
  </LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-44</ItemId>
  <Name />
  <Line>1</Line>
  <Quantity>1</Quantity>
  <Price>41.95</Price>
  </LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-51</ItemId>
  <Name />
  <Line>2</Line>
```

<Quantity>1</Quantity>

<Price>85.99</Price>

</LineItemInfo>

=<LineItemInfo>

<ItemId>EST-50</ItemId>

<Name />

<Line>3</Line>

<Quantity>1</Quantity>

<Price>80.99</Price>

</LineItemInfo>

=<LineItemInfo>

<ItemId>EST-76</ItemId>

<Name />

<Line>4</Line>

<Quantity>1</Quantity>

<Price>349.00</Price>

</LineItemInfo>

=<LineItemInfo>

<ItemId>EST-54</ItemId>

<Name />

<Line>1</Line>

<Quantity>200</Quantity>

<Price>0.25</Price>

```
</LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-26</ItemId>
  <Name />
  <Line>2</Line>
  <Quantity>4</Quantity>
  <Price>14.95</Price>
</LineItemInfo>
</LineItems>
<AuthorizationNumber xsi:nil="true" />
</anyType>
- <anyType xsi:type="OrderInfo">
  <OrderId>2</OrderId>
  <Date>2009-12-11T16:38:31.733</Date>
  <UserId>AdamBarr</UserId>
- <BillingAddress>
  <FirstName>Adam</FirstName>
  <LastName>Barr</LastName>
  <Address1>Vertigo Software, Inc.</Address1>
  <Address2>503A Canal Blvd.</Address2>
  <City>Point Richmond</City>
  <State>CA</State>
  <Zip>94804</Zip>
```

```
<Country>USA</Country>
<Email>email</Email>
</BillingAddress>
=<ShippingAddress>
  <FirstName>Adam</FirstName>
  <LastName>Barr</LastName>
  <Address1>Vertigo Software, Inc.</Address1>
  <Address2>503A Canal Blvd.</Address2>
  <City>Point Richmond</City>
  <State>CA</State>
  <Zip>94804</Zip>
  <Country>USA</Country>
  <Email>email</Email>
</ShippingAddress>
<OrderTotal>557.93</OrderTotal>
=<LineItems>
=<LineItemInfo>
  <ItemId>EST-24</ItemId>
  <Name />
  <Line>1</Line>
  <Quantity>1</Quantity>
  <Price>120.95</Price>
</LineItemInfo>
```



```
-<LineItemInfo>  
  <ItemId>EST-24</ItemId>  
  <Name />  
  <Line>2</Line>  
  <Quantity>1</Quantity>  
  <Price>120.95</Price>  
</LineItemInfo>
```

```
-<LineItemInfo>  
  <ItemId>EST-44</ItemId>  
  <Name />  
  <Line>1</Line>  
  <Quantity>1</Quantity>  
  <Price>41.95</Price>  
</LineItemInfo>
```

```
-<LineItemInfo>  
  <ItemId>EST-51</ItemId>  
  <Name />  
  <Line>2</Line>  
  <Quantity>1</Quantity>  
  <Price>85.99</Price>  
</LineItemInfo>
```

```
-<LineItemInfo>  
  <ItemId>EST-50</ItemId>
```

```
<Name />
<Line>3</Line>
<Quantity>1</Quantity>
<Price>80.99</Price>
</LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-76</ItemId>
  <Name />
  <Line>4</Line>
  <Quantity>1</Quantity>
  <Price>349.00</Price>
  </LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-54</ItemId>
  <Name />
  <Line>1</Line>
  <Quantity>200</Quantity>
  <Price>0.25</Price>
  </LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-26</ItemId>
  <Name />
  <Line>2</Line>
```

```
<Quantity>4</Quantity>
<Price>14.95</Price>
</LineItemInfo>
</LineItems>
<AuthorizationNumber xsi:nil="true" />
</anyType>
- <anyType xsi:type="OrderInfo">
  <OrderId>3</OrderId>
  <Date>2009-12-11T16:45:41.717</Date>
  <UserId>RobYoung</UserId>
- <BillingAddress>
  <FirstName>Rob</FirstName>
  <LastName>Young</LastName>
  <Address1>Vertigo Software, Inc.</Address1>
  <Address2>503A Canal Blvd.</Address2>
  <City>Point Richmond</City>
  <State>CA</State>
  <Zip>94804</Zip>
  <Country>USA</Country>
  <Email>email</Email>
</BillingAddress>
- <ShippingAddress>
  <FirstName>Rob</FirstName>
```

```
<LastName>Young</LastName>
<Address1>Vertigo Software, Inc.</Address1>
<Address2>503A Canal Blvd.</Address2>
<City>Point Richmond</City>
<State>CA</State>
<Zip>94804</Zip>
<Country>USA</Country>
<Email>email</Email>
</ShippingAddress>
<OrderTotal>109.80</OrderTotal>
- <LineItems>
- <LineItemInfo>
  <ItemId>EST-24</ItemId>
  <Name />
  <Line>1</Line>
  <Quantity>1</Quantity>
  <Price>120.95</Price>
</LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-24</ItemId>
  <Name />
  <Line>2</Line>
  <Quantity>1</Quantity>
```

<Price>120.95</Price>

</LineItemInfo>

=<LineItemInfo>

<ItemId>EST-44</ItemId>

<Name />

<Line>1</Line>

<Quantity>1</Quantity>

<Price>41.95</Price>

</LineItemInfo>

=<LineItemInfo>

<ItemId>EST-51</ItemId>

<Name />

<Line>2</Line>

<Quantity>1</Quantity>

<Price>85.99</Price>

</LineItemInfo>

=<LineItemInfo>

<ItemId>EST-50</ItemId>

<Name />

<Line>3</Line>

<Quantity>1</Quantity>

<Price>80.99</Price>

</LineItemInfo>

```
-<LineItemInfo>
  <ItemId>EST-76</ItemId>
  <Name />
  <Line>4</Line>
  <Quantity>1</Quantity>
  <Price>349.00</Price>
</LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-54</ItemId>
  <Name />
  <Line>1</Line>
  <Quantity>200</Quantity>
  <Price>0.25</Price>
</LineItemInfo>
- <LineItemInfo>
  <ItemId>EST-26</ItemId>
  <Name />
  <Line>2</Line>
  <Quantity>4</Quantity>
  <Price>14.95</Price>
</LineItemInfo>
</LineItems>
<AuthorizationNumber xsi:nil="true" />
```

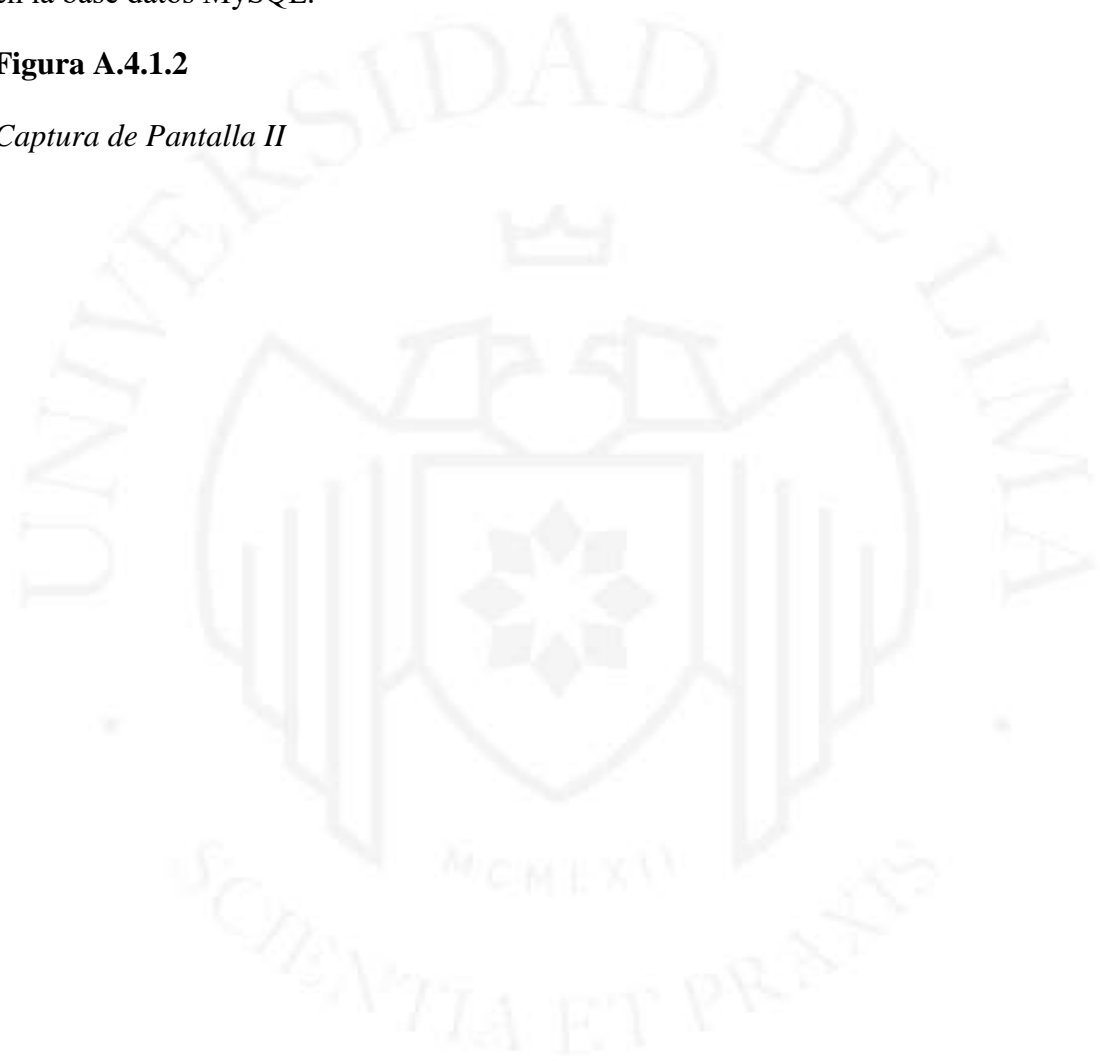
</anyType>

</ArrayOfAnyType>

A su vez, se probaron los servicios del Sistema Contable y se verificaron las inserciones en la base datos MySQL.

Figura A.4.1.2

Captura de Pantalla II



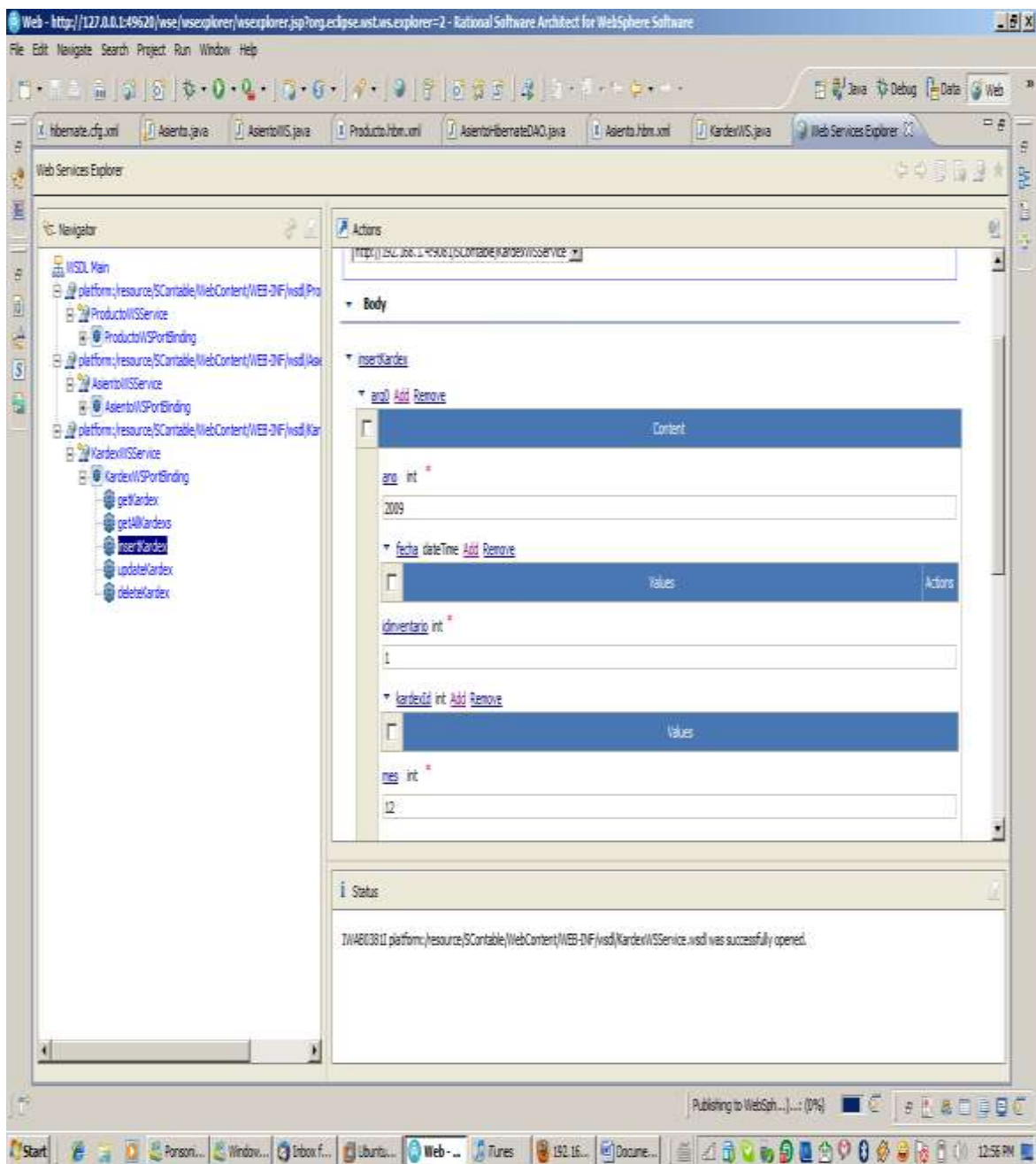


Figura A.4.1.3

Captura de Pantalla III

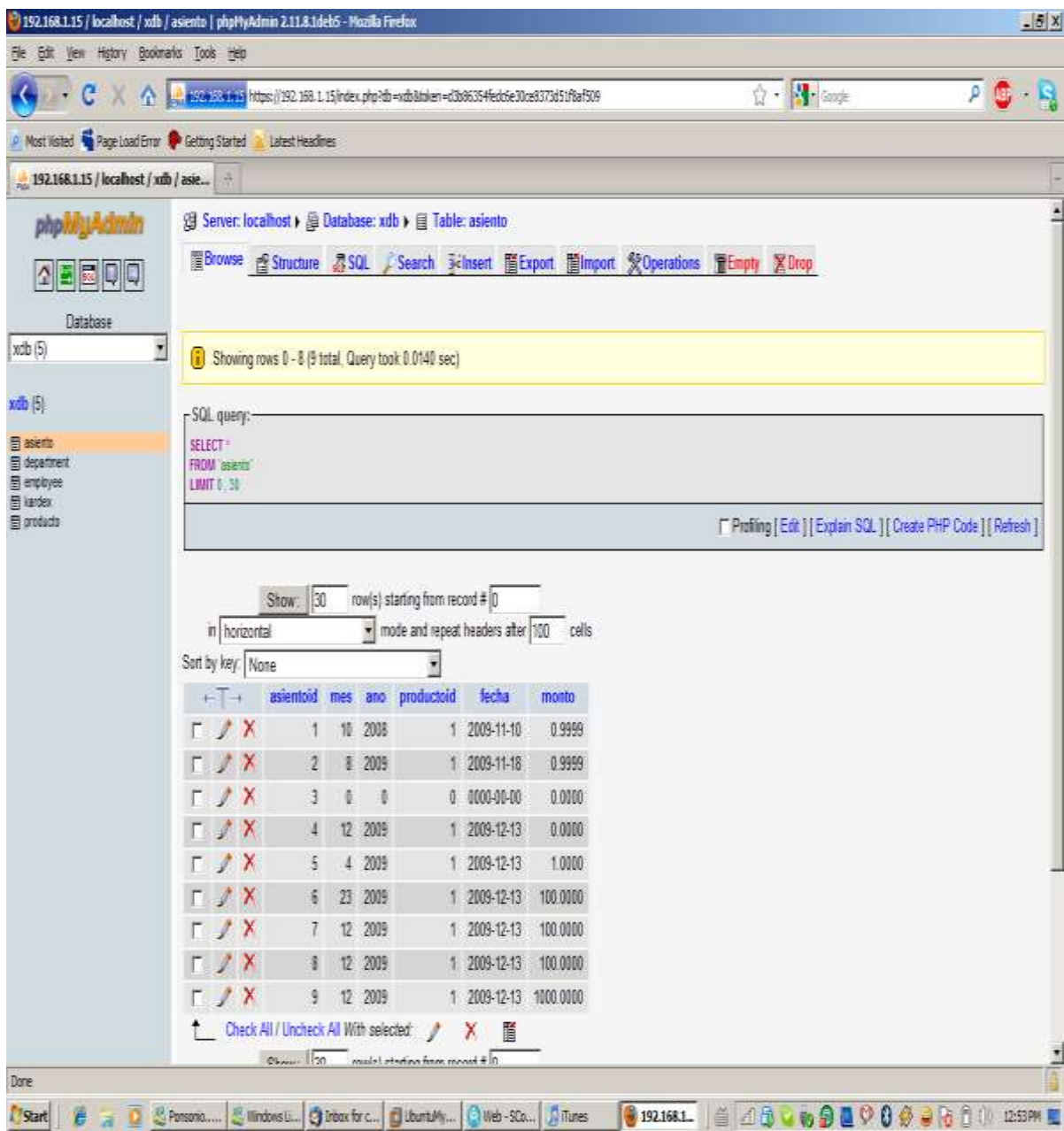
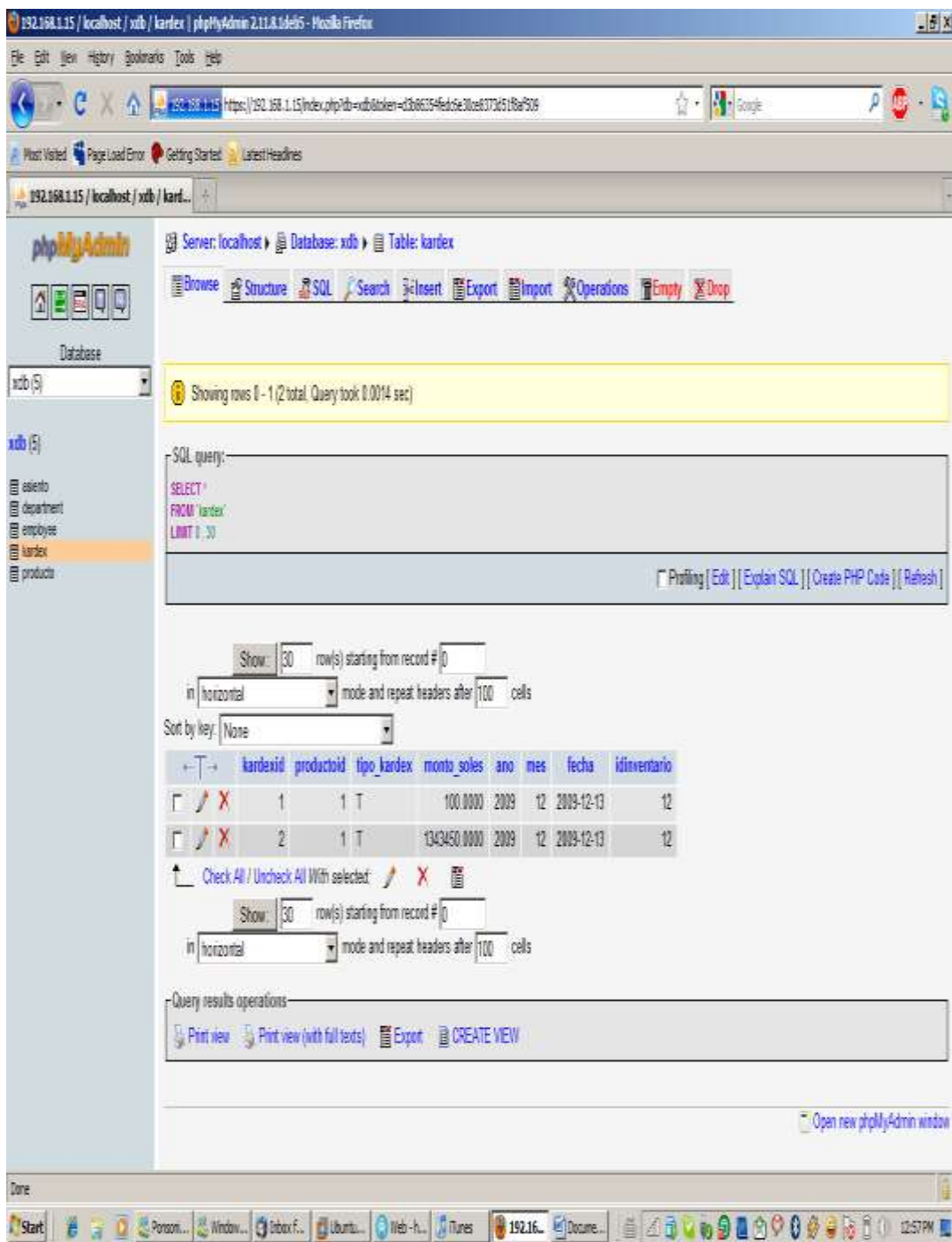


Figura A.4.1.4

Captura de Pantalla IV



Caso de Prueba: Prueba realiza pedido:

Se ejecutó el caso de uso exitosamente y se almacenaron los siguientes datos correspondientes a los ingresados:

Tabla A.4.1.1

Prueba Realizar Pedido

o.OrderId	o.OrderDate	o.UserId	o.BillToFirstName	o.BillToLastName	o.TotalPrice	l.ItemId	l.LineNum	l.Quantity	l.UnitPrice
1	30/11/2009	demo	Adam	Barr	120.95	EST-24	1	1	120.95
1	30/11/2009	demo	Adam	Barr	120.95	EST-24	2	1	120.95
1	30/11/2009	demo	Adam	Barr	120.95	EST-44	1	1	41.95
1	30/11/2009	demo	Adam	Barr	120.95	EST-51	2	1	85.99
1	30/11/2009	demo	Adam	Barr	120.95	EST-50	3	1	80.99
1	30/11/2009	demo	Adam	Barr	120.95	EST-76	4	1	349
1	30/11/2009	demo	Adam	Barr	120.95	EST-54	1	200	0.25
1	30/11/2009	demo	Adam	Barr	120.95	EST-26	2	4	14.95
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-24	1	1	120.95
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-24	2	1	120.95
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-44	1	1	41.95
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-51	2	1	85.99

2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-50	3	1	80.99
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-76	4	1	349
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-54	1	200	0.25
2	30/11/2009	AdamBarr	Adam	Barr	557.93	EST-26	2	4	14.95
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-24	1	1	120.95
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-24	2	1	120.95
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-44	1	1	41.95
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-51	2	1	85.99
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-50	3	1	80.99
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-76	4	1	349
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-54	1	200	0.25
3	30/11/2009	RobYoung	Rob	Young	109.8	EST-26	2	4	14.95

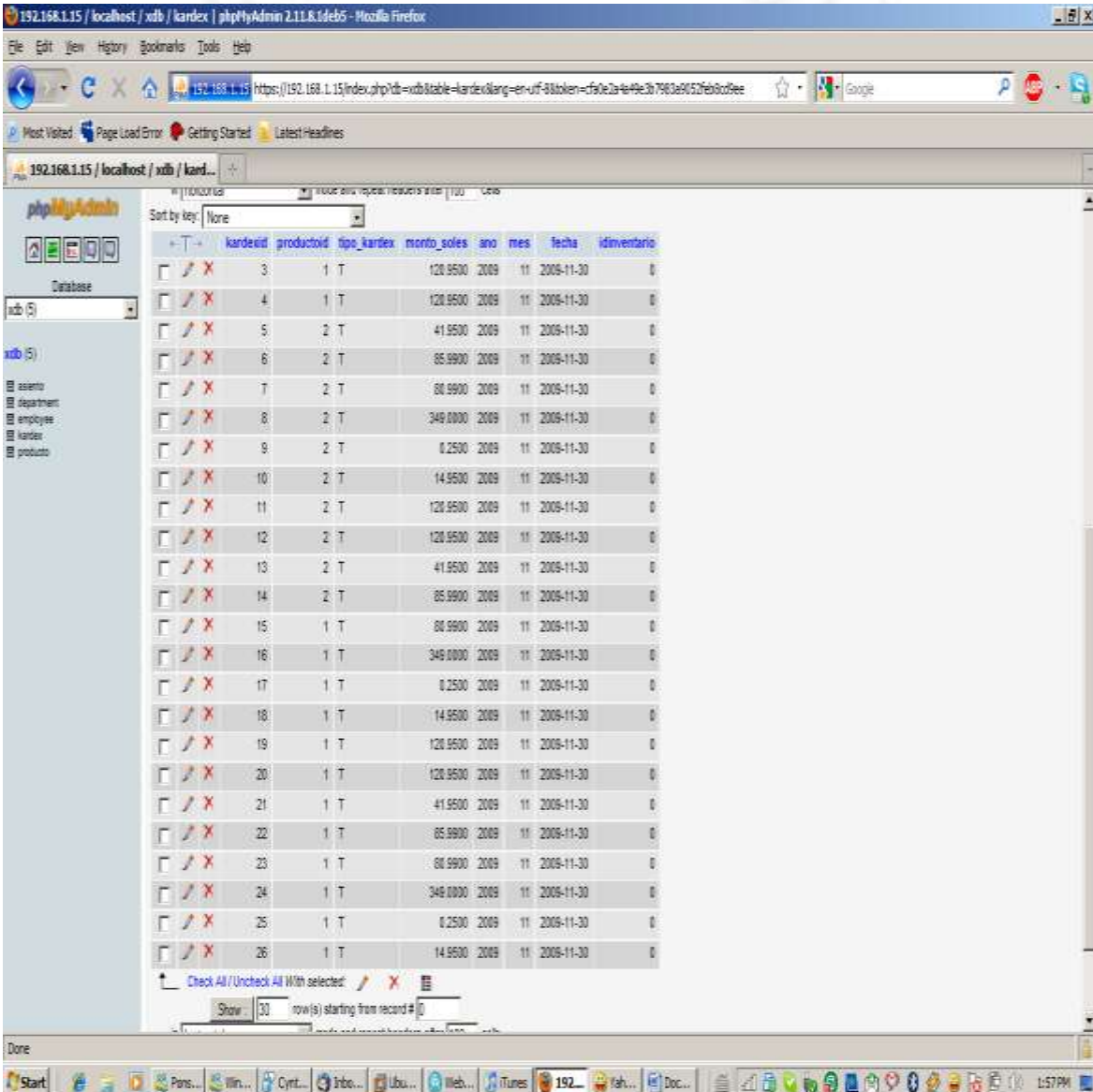
Caso de uso: Prueba envía pedidos

Se ejecutó el modelo de integración y se verificó la creación de los asientos y kárdex correspondientes, a continuación se muestra el detalle de los datos creados en el Sistema Contable:

Para la tabla Kardex:

Figura A.4.1.5

Captura de Pantalla V



The screenshot shows the phpMyAdmin interface for a database named 'xib'. The 'Kardex' table is selected, and its data is displayed in a table format. The table has columns for 'kardexid', 'productoid', 'tipo_kardex', 'monto_soles', 'ano', 'mes', 'fecha', and 'idinventario'. The data consists of 24 rows, each representing a different kardex entry. The 'ano' and 'mes' columns are consistently set to 2009 and 11, respectively. The 'fecha' column shows dates from 2009-11-30. The 'idinventario' column is consistently set to 0. The 'monto_soles' column varies across rows, with values such as 120,950, 41,950, 80,990, 349,000, and 0,250. The 'tipo_kardex' column has values of 1 or 2. The 'productoid' column has values of 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, and 25. The 'kardexid' column has values from 3 to 26.

kardexid	productoid	tipo_kardex	monto_soles	ano	mes	fecha	idinventario
3	1	T	120.950	2009	11	2009-11-30	0
4	1	T	120.950	2009	11	2009-11-30	0
5	2	T	41.950	2009	11	2009-11-30	0
6	2	T	80.990	2009	11	2009-11-30	0
7	2	T	80.990	2009	11	2009-11-30	0
8	2	T	349.000	2009	11	2009-11-30	0
9	2	T	0.250	2009	11	2009-11-30	0
10	2	T	14.950	2009	11	2009-11-30	0
11	2	T	120.950	2009	11	2009-11-30	0
12	2	T	120.950	2009	11	2009-11-30	0
13	2	T	41.950	2009	11	2009-11-30	0
14	2	T	80.990	2009	11	2009-11-30	0
15	1	T	80.990	2009	11	2009-11-30	0
16	1	T	349.000	2009	11	2009-11-30	0
17	1	T	0.250	2009	11	2009-11-30	0
18	1	T	14.950	2009	11	2009-11-30	0
19	1	T	120.950	2009	11	2009-11-30	0
20	1	T	120.950	2009	11	2009-11-30	0
21	1	T	41.950	2009	11	2009-11-30	0
22	1	T	80.990	2009	11	2009-11-30	0
23	1	T	80.990	2009	11	2009-11-30	0
24	1	T	349.000	2009	11	2009-11-30	0
25	1	T	0.250	2009	11	2009-11-30	0
26	1	T	14.950	2009	11	2009-11-30	0

Adicionalmente se muestran los datos en formato xml:

```
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
- <soapenv:Body>
- <getAllKardexsResponse xmlns:ns2="http://ws.scontable/">
- <return>
  <ano>2009</ano>
  <fecha>2009-11-30T00:00:00-05:00</fecha>
  <idinventario>0</idinventario>
  <kardexId>3</kardexId>
  <mes>11</mes>
  <montoSoles>120.9500</montoSoles>
- <producto>
  <nombreProducto>Setter Irlandes</nombreProducto>
  <productoId>1</productoId>
</producto>
  <tipoKardex>T</tipoKardex>
</return>
- <return>
  <ano>2009</ano>
  <fecha>2009-11-30T00:00:00-05:00</fecha>
  <idinventario>0</idinventario>
  <kardexId>4</kardexId>
  <mes>11</mes>
```

<montoSoles>120.9500</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>5</kardexId>

<mes>11</mes>

<montoSoles>41.9500</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>6</kardexId>

<mes>11</mes>

<montoSoles>85.9900</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>7</kardexId>

<mes>11</mes>

<montoSoles>80.9900</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>8</kardexId>

<mes>11</mes>

<montoSoles>349.0000</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>9</kardexId>

<mes>11</mes>

<montoSoles>0.2500</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

```
</producto>
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>10</kardexId>
<mes>11</mes>
<montoSoles>14.9500</montoSoles>
=<producto>
<nombreProducto>Pastor Aleman</nombreProducto>
<productoId>2</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>11</kardexId>
<mes>11</mes>
<montoSoles>120.9500</montoSoles>
```

```
= <producto>
  <nombreProducto>Pastor Aleman</nombreProducto>
  <productoId>2</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
```

```
= <return>
  <ano>2009</ano>
  <fecha>2009-11-30T00:00:00-05:00</fecha>
  <idinventario>0</idinventario>
  <kardexId>12</kardexId>
  <mes>11</mes>
  <montoSoles>120.9500</montoSoles>
```

```
= <producto>
  <nombreProducto>Pastor Aleman</nombreProducto>
  <productoId>2</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
```

```
= <return>
  <ano>2009</ano>
  <fecha>2009-11-30T00:00:00-05:00</fecha>
  <idinventario>0</idinventario>
```

<kardexId>13</kardexId>

<mes>11</mes>

<montoSoles>41.9500</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>14</kardexId>

<mes>11</mes>

<montoSoles>85.9900</montoSoles>

= <producto>

<nombreProducto>Pastor Aleman</nombreProducto>

<productoId>2</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>15</kardexId>

<mes>11</mes>

<montoSoles>80.9900</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>16</kardexId>

<mes>11</mes>

<montoSoles>349.0000</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

```
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>17</kardexId>
<mes>11</mes>
<montoSoles>0.2500</montoSoles>
=<producto>
<nombreProducto>Setter Irlandes</nombreProducto>
<productoId>1</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>18</kardexId>
<mes>11</mes>
<montoSoles>14.9500</montoSoles>
=<producto>
```

```
<nombreProducto>Setter Irlandes</nombreProducto>
<productoId>1</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>19</kardexId>
<mes>11</mes>
<montoSoles>120.9500</montoSoles>
=<producto>
<nombreProducto>Setter Irlandes</nombreProducto>
<productoId>1</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>20</kardexId>
```

<mes>11</mes>

<montoSoles>120.9500</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>21</kardexId>

<mes>11</mes>

<montoSoles>41.9500</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>22</kardexId>

<mes>11</mes>

<montoSoles>85.9900</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>23</kardexId>

<mes>11</mes>

<montoSoles>80.9900</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>24</kardexId>

<mes>11</mes>

<montoSoles>349.0000</montoSoles>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

<tipoKardex>T</tipoKardex>

</return>

= <return>

<ano>2009</ano>

<fecha>2009-11-30T00:00:00-05:00</fecha>

<idinventario>0</idinventario>

<kardexId>25</kardexId>

<mes>11</mes>

<montoSoles>0.2500</montoSoles>

= <producto>

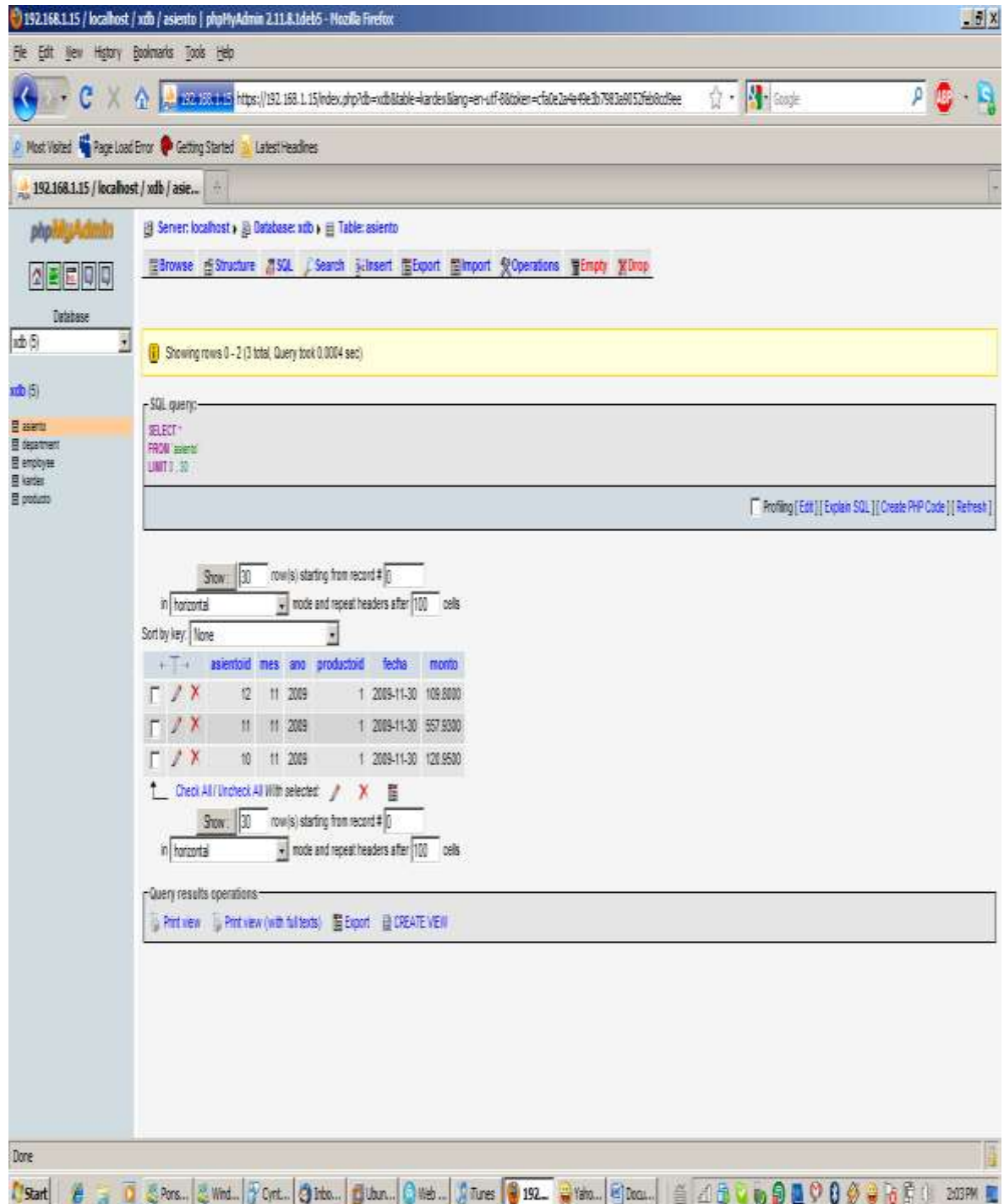
<nombreProducto>Setter Irlandes</nombreProducto>

```
<productoId>1</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
=<return>
<ano>2009</ano>
<fecha>2009-11-30T00:00:00-05:00</fecha>
<idinventario>0</idinventario>
<kardexId>26</kardexId>
<mes>11</mes>
<montoSoles>14.9500</montoSoles>
=<producto>
<nombreProducto>Setter Irlandes</nombreProducto>
<productoId>1</productoId>
</producto>
<tipoKardex>T</tipoKardex>
</return>
</getAllKardexsResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Para los asientos contables:

Figura A.4.1.6

Captura de Pantalla VI



Y los resultados en formato XML

```
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
- <soapenv:Body>
- <getAllAsientosResponse xmlns:ns2="http://ws.scontable/">
- <return>
  <ano>2009</ano>
  <fecha>2009-11-30T00:00:00-05:00</fecha>
  <mes>11</mes>
  <monto>109.8000</monto>
- <producto>
  <nombreProducto>Setter Irlandes</nombreProducto>
  <productoId>1</productoId>
</producto>
</return>
- <return>
  <ano>2009</ano>
  <fecha>2009-11-30T00:00:00-05:00</fecha>
  <mes>11</mes>
  <monto>557.9300</monto>
- <producto>
  <nombreProducto>Setter Irlandes</nombreProducto>
  <productoId>1</productoId>
</producto>
</return>
- <return>
  <ano>2009</ano>
```

<fecha>2009-11-30T00:00:00-05:00</fecha>

<mes>11</mes>

<monto>120.9500</monto>

= <producto>

<nombreProducto>Setter Irlandes</nombreProducto>

<productoId>1</productoId>

</producto>

</return>

</getAllAsientosResponse>

</soapenv:Body>

</soapenv:Envelope>

