APLICACIÓN DE LOS SERVICIOS WEB XML Y EL ESTÁNDAR BPEL AL COMERCIO ELECTRÓNICO

GILBERTO GÓMEZ GUALDRÓN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2010

APLICACIÓN DE LOS SERVICIOS WEB XML Y EL ESTÁNDAR BPEL AL COMERCIO ELECTRÓNICO

GILBERTO GÓMEZ GUALDRÓN

Trabajo de Grado para optar por el título de
Ingeniero de Sistemas

Director
SERGIO FERNANDO CASTILLO CASTELBLANCO
Ingeniero de Sistemas, Ph.D

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2010

Para mi familia: mis padres Imelda y Ricardo,
mi hermanita Janeth Gissella, y tú amor mío;
y por supuesto... para Dios.

AGRADECIMIENTOS

El autor agradece muy especialmente a:

DIOS, por lo que fui, soy y seré: por hacerme la persona más feliz del mundo.

Dr. Sergio F. Castillo C., director de esta investigación, por su constante apoyo y preocupación, y sus consejos durante mi estudio y el trabajo en la tesis.

Mi familia, especialmente mis padres Imelda y Ricardo a quienes les debo gran parte de lo que soy, por su apoyo y el ejemplo de vida y esfuerzo que me han dado, y mi hermanita adorada Janeth Gissella, quien me ha guiado, cuidado y educado siempre. También a Diego Armando, por su cariño, apoyo, y constancia.

Mis grandes amigos Hoober Alexis, Julián Enrique y Cristhian Ricardo. Asimismo a Andrés Felipe, Mao, Jesús, Natalí; Henry, Fernando, Nelly, Viviana, Diana V, Sandra, Diana Marcela, Angélica, Liseth, Sergio, Freddy, Carlos, Jhon, Jorge, Omar, Armando y Javier, por ser mis entrañables "*poké-amigos*" con quienes compartí momentos mágicos de la vida.

Todos los profesionales y amigos, especialmente Luis Antonio, cuyos consejos y experiencia me han servido de guía profesional y personal. Al igual que a mis maestros y profesores principalmente Elisa, Claudia, Sara Cecilia, Rubilma, Rubén Darío (Q.E.P.D) y José de Jesús.

Mis amigos y personas a quienes estimo y valoro, y han contribuido de alguna forma en mi vida.

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

LISTA DE ANEXOS

pág.

RESUMEN

TITULO:
APLICACIÓN DE LOS SERVICIOS WEB XML Y EL ESTÁNDAR BPEL AL COMERCIO ELECTRÓNICO *

AUTOR:
Gilberto Gómez Gualdrón **

PALABRAS CLAVES:
Servicios Web, Procesos de Negocio, Comercio Electrónico, XML, WSDL, BPEL

DESCRIPCIÓN:
Este trabajo de investigación plantea el diseño de una agencia de viajes electrónica (como un escenario ejemplo representativo del comercio electrónico), utilizando sistemas distribuidos bajo la arquitectura orientada a servicios SOA, aplicando servicios Web XML y el lenguaje de ejecución de procesos de negocio BPEL. Su principal aporte es la incorporación de estas nuevas tecnologías en las líneas de investigación sobre computación distribuida y comercio electrónico en la Universidad Industrial de Santander.

El escenario se seleccionó y definió con los fines de exponer y tratar la complejidad inherente a los procesos y transacciones electrónicas, y de plantear las consideraciones técnicas necesarias para brindar seguridad, interoperabilidad, estandarización y escalabilidad al desarrollo.

La metodología de desarrollo incremental dirigió el proceso de análisis e implementación, facilitando la reutilización y la realimentación en cada fase definida. El análisis preliminar del escenario se realizó utilizando lenguaje unificado de modelado UML, planteando los diversos actores involucrados en las transacciones y definiendo los procesos principales mediante flujos de actividades entre cada uno. Las fases de desarrollo planteadas cubrieron cada proceso interno a la agencia de viajes, identificado en el análisis.

Como resultado del desarrollo, se cuenta con un prototipo de agencia de viajes electrónica implementada mediante lenguaje BPEL y que coordina los servicios web definidos entre sus clientes y los socios comerciales. La implementación utilizó las tecnologías: Apache Web Services con sus proyectos AXIS y WSS4J como la infraestructura de servicios Web; el motor ActiveBPEL Engine para desplegar los procesos de negocio BPEL sobre un servidor Apache Tomcar; el lenguaje de programación Java para construir los componentes especializados y la interfaz del cliente; y el motor de base de datos MySQL para la persistencia y almacenamiento de datos de cada componente.

---

* Proyecto de Grado en la Modalidad Investigación.

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Sergio F. Castillo C., Ph.D

ABSTRACT

TITLE:
APPLICATION OF XML WEB SERVICES AND THE BPEL SPECIFICATION TO THE E-COMMERCE *

AUTHOR:
Gilberto Gómez Gualdrón **

KEYWORDS:
Web Services, Business Processes, E-Commerce, XML, WSDL, BPEL

DESCRIPTION:
This research proposes the design of an electronic travel agency (as a representative example scenario of e-commerce), using distributed systems under the service oriented architecture SOA, applying XML Web services and the Business Process Execution Language BPEL. Its main contribution is the incorporation of these new technologies in the research on distributed computing and e-commerce of the Industrial University of Santander.

The scenario was selected and defined to show and to apply the inherent complexity of the electronic processes and transactions, and to propose the necessary technical considerations to provide security, interoperability, standardization and scalability to the development.

The analysis and implementation process was led by the incremental development methodology, which made possible to take advantage of what was learned and developed in previous and incremental phases. Preliminary analysis of the scenario was made with Unified Modeling Language UML, considering the various actors involved in the transactions, and defining the main processes through the internal activity flows. The proposed development phases covered each internal process to the travel agency, previously indentified in the analysis.

The result of the development was an electronic travel agency prototype, implemented with BPEL language, which coordinates the web services defined between its clients and business partners. The implementation used the following technologies: Apache Web Services with AXIS and WSS4J projects as the web services infrastructure; the ActiveBPEL Engine to deploy the BPEL business processes over an Apache Tomcat server; the Java programming language to build the specialized components and the user interface; and the MySQL database server to provide persistency and data storage to each component.

---

* Project degree in Research category.

** Physical – Mechanical Engineering's Faculty. Systems and Informatics Engineering School. Director: Sergio F. Castillo C., Ph.D

INTRODUCCIÓN

El desarrollo, difusión y gran crecimiento de las redes y la Web, han permitido el surgimiento de la computación extendida y el uso en gran medida de sistemas distribuidos alrededor del mundo. La interoperabilidad y la integración en diversas plataformas de aplicaciones desarrolladas en diferentes lenguajes de programación han sido un foco importante de la investigación actual y los desarrollos en el campo de Internet, dando origen a variados estándares de comunicación, representación y seguridad de la información.

Los recientes trabajos y desarrollos han ido revolucionando la forma como los actores (entidades o usuarios) interactúan entre sí en el ámbito virtual. Millones de organizaciones están adaptando sus sistemas y su estructura para permitir que sus principales operaciones puedan ser realizadas a través de la Web y así tomar ventaja del gran potencial que ofrece la automatización, la eficiencia de los procesos de negocio y la visibilidad global que brinda Internet. De esta forma, la interacción entre los usuarios y las entidades se realiza mediante transacciones electrónicas e intercambiando información, productos y servicios a través de Internet, proceso conocido como comercio electrónico o *e-commerce*.

Los sistemas informáticos y aplicaciones Web, encargadas de implementar y desplegar la lógica de los procesos del comercio electrónico, utilizan diferentes paradigmas para su funcionamiento, entre ellos la orientación a objetos, el modelo de agentes móviles, y el modelo de computación centrada en la oferta y demanda de servicios. Este último, conocido también como el paradigma de la Computación Orientada a Servicios, se basa en la exposición de las funcionalidades disponibles de un componente software o aplicación informática, haciéndolas accesibles a través de una interfaz pública, adquiriendo la denominación de servicio.

Debido a la complejidad cada vez mayor de los procesos de comercio electrónico, se ha venido profundizando en la investigación y el desarrollo de la composición de servicios, una necesidad inherente del modelo centrado en servicios. Componer servicios consiste en coordinar y secuenciar su ejecución, de acuerdo con los flujos de información de la lógica a implementar. El estándar BPEL4WS provee un lenguaje basado en XML para definir la forma de ejecutar servicios Web XML, reutilizándolos y coordinándolos adecuadamente.

Con el desarrollo del proyecto “Aplicación de los Servicios Web XML y el estándar BPEL al comercio electrónico” se provee el diseño de los procesos inherentes a un negocio representativo del comercio electrónico, bajo la arquitectura de orientación a servicios; y además se ofrece un prototipo software adecuado para el escenario, desarrollado utilizando Servicios Web XML para las interfaces de las funcionalidades, implementadas con el lenguaje de programación JAVA, y aplicando el estándar BPEL para coordinar y componer su lógica de negocio.

Estructura del Documento

En el primer capítulo se describe el proyecto realizado, especificando la problemática definida y formulada, los objetivos y alcances establecidos y el impacto esperado de la investigación.

El segundo capítulo aborda la problemática escogida para la investigación, describiendo el escenario de investigación y describiendo brevemente las opciones tecnológicas alternas aplicables en el contexto.

El tercer capítulo describe el marco conceptual actual relacionado con el ámbito del proyecto; y en forma similar, el capítulo cuatro se ocupa de explicar brevemente las tecnologías y estándares seleccionados y empleados en el desarrollo.

En el quinto capítulo se detalla cada una de las etapas del desarrollo del proyecto, iniciando con la metodología utilizada e incluyendo aspectos relevantes como la descripción del sistema analizado con base en el escenario propuesto, el análisis y diseño general del sistema, las fases de implementación y sus respectivas pruebas realizadas.

El sexto capítulo muestra un ejemplo visual de uso del sistema desarrollado, detallando la forma de operación y trabajo con el mismo.

Finalmente se enuncian las *conclusiones* obtenidas con la presente investigación y se exponen diversos aspectos importantes que deben tenerse en cuenta para *trabajos futuros* en esta línea de estudio. También se detallan las *referencias* y *bibliografía* usadas como fuente de información en la investigación.

Adicionalmente se presentan Anexos que contienen resúmenes de parte de los desarrollos y presentan ordenadamente algunas definiciones realizadas de tipos de datos y estructura de la información. También se presentan los diseños de las pruebas que se aplicaron al desarrollo por cada fase de implementación.

1. DESCRIPCIÓN DEL PROYECTO

En este capítulo se presenta el proyecto: se describe la oportunidad que da origen a su desarrollo, los objetivos planteados, su alcance y las repercusiones a corto y largo plazo en el ámbito investigativo.

1.1 DEFINICIÓN

En el creciente mundo de las transacciones electrónicas comerciales (comercio electrónico), las entidades participantes realizan intercambio de información de acuerdo una lógica específica, propia del negocio que se está efectuando. Los flujos de datos son cada vez más estructurados y siguen algoritmos que involucran muchas más entidades de diversos tipos, generalmente con una mínima intervención de humanos, según sea el servicio que prestan. En las tiendas electrónicas, por ejemplo, los usuarios realizan búsquedas y compras de productos, proceso que involucra a la tienda, el usuario, los proveedores de los productos, bancos o entidades encargadas del pago, entre otras. Cada transacción realizada hace parte de un flujo complejo de información.

Las características de funcionamiento de los negocios considerados como electrónicos agregan necesidades significativas a los sistemas informáticos que habiliten su despliegue en la Web. Se requiere de la tecnología apropiada para brindar todas las garantías posibles, la información debe ser intercambiada en forma ágil, segura y con sencillez para los actores de los procesos, ocultando la complejidad interna que en la mayoría de los casos es mucho mayor. La agilidad y velocidad de transmisión puede lograrse utilizando la Web. La seguridad puede brindarse en un alto grado, utilizando estándares de cifrado de los datos y reconocimiento mutuo de los actores. Por su parte, la sencillez para las entidades involucradas puede darse permitiendo el acceso desde diferentes plataformas y por diversos medios; y la complejidad interna puede abordarse reutilizando

procesos ya implementados y coordinando los ya existentes para llevar a cabo de forma exitosa las tareas requeridas.

La computación orientada a servicios es actualmente una muy buena elección para el diseño de aplicaciones en Internet, gracias a las ventajas que brinda como la interoperabilidad, la fácil adaptación, y la posibilidad de composición Las tecnologías de desarrollo de aplicaciones Web más recientes y en evolución, permiten el uso de este paradigma, como es el caso de Java Platform, Enterprise Edition 6 (Java EE 6)[1] de la compañía Sun Microsystems, y de la plataforma .NET de Microsoft.

Bajo este enfoque, cada entidad expone en la Web un servicio de lo que ofrece, haciéndolo visible y asequible por otras organizaciones, sistemas, o usuarios humanos quienes pueden solicitarlo. Para realizar la búsqueda, solicitud e interacción con los servicios Web se utilizan los estándares XML (Extensible Markup Language), y WSDL (Web Services Description Language) que es un lenguaje basado en XML que describe un servicio en términos de sus operaciones y el formato que tienen. Los sistemas desarrollados utilizando las anteriores especificaciones adquieren características como heterogeneidad e interoperabilidad, lo que permite adaptarse a la tendencia actual de la computación distribuida, donde una máquina puede comunicarse con otra sin la necesidad de tener un ser humano interviniendo en la comunicación. Además habilita el uso de otras arquitecturas y tecnologías ya existentes simplemente realizando interfaces de comunicación entre ellas. Todo lo anterior brinda una *plataforma común* para el desarrollo de aplicaciones distribuidas, haciendo uso de software que se ejecuta en *cualquier sistema operativo y/o dispositivo*, y que puede ser desarrollado en diversos lenguajes o herramientas.

[1] Java EE 6 es la versión mas reciente de la plataforma de desarrollo empresarial de Java, para la nueva generación de aplicaciones Web y la Arqutectura Orientada a Servicios. http://java.sun.com/javaee/

El problema de la complejidad propia de las transacciones radica en que deben ser coordinadas y secuenciadas de acuerdo con los flujos de información presentes en el proceso, involucrando ciclos, realimentación de resultados intermedios, esperas asíncronas, etc. Para abordar esta problemática existen estándares adicionales que permitieran *componer* y *combinar* la funcionalidad de múltiples servicios Web, reutilizándolos y coordinándolos adecuadamente para realizar las tareas complejas. El estándar de coreografía de servicios Web BPEL *Business Process Execution Language* (Lenguaje de Ejecución de Procesos de Negocio) proporciona una infraestructura potente y eficiente que facilita la implementación de procesos complejos, ya que permite definirlos como composiciones de servicios Web y ejecutarlos automáticamente siguiendo una secuencia determinada.

El uso de los servicios Web se está extendiendo rápidamente y no se limita a la comunicación entre entidades, también se ha convertido en la forma preferida por empresas y gobiernos a la hora de diseñar e implementar sus infraestructuras de comunicación interna. Los trabajos de investigación y desarrollo en el mundo se centran en la adecuada descripción de los Servicios Web XML, y en la importante inquietud para los investigadores relacionada con la sintaxis del flujo de actividades y la forma en que los Servicios Web XML deben ser ejecutados con un nivel de seguridad adecuado.

El presente trabajo de investigación busca entonces proporcionar un punto inicial para la incorporación de la tecnología y el apropiamiento del conocimiento en torno a la computación orientada a servicios, los servicios Web en general, y los lenguajes de coordinación de servicios Web como BPEL, mediante su aplicación en el análisis, diseño y desarrollo de un prototipo de negocio electrónico.

1.2 OBJETIVOS

1.2.1 Objetivo General

Desarrollar un prototipo software como aplicación de la tecnología de Servicios Web XML y el estándar de procesos de negocio BPEL (*Business Process Execution Language*) al comercio electrónico, tomando como referencia y ejemplo los principales procesos inherentes al funcionamiento de un negocio electrónico representativo y utilizando el lenguaje de programación JAVA.

1.2.2 Objetivos Específicos

- Diseñar mediante el Lenguaje Unificado de Modelado (UML) la estructura y funcionamiento de los procesos inherentes a un negocio representativo del comercio electrónico, como es una agencia de viajes electrónica, utilizando los diagramas más apropiados: clases, despliegue, casos de uso, actividades y secuencia.

- Diseñar utilizando Servicios Web XML, los componentes Web requeridos para la implementación de los procesos de negocio propuestos previamente.

- Aplicar el estándar de ejecución de procesos de negocio BPEL, para diseñar e implementar la lógica y funcionamiento interno de los procesos descritos.

- Implementar el prototipo de software de los diseños realizados, haciendo uso del lenguaje de programación JAVA y AXIS, el proyecto SOAP de Apache.

- Diseñar y ejecutar un plan de pruebas a los prototipos, para que los resultados y las principales métricas de dicho software, permitan determinar las ventajas y desventajas de la aplicación de esas tecnologías mencionadas en el diseño.

1.3 ALCANCE

La presente investigación y sus resultados son parte de los trabajos sobre Comercio Electrónico y Servicios Web que ha venido realizando el grupo de Investigación en Ingeniería Telemática y Sistemas Inteligentes (GITSI) de la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander, y pretende marcar un hito importante en dicha línea de investigación, sirviendo como base de trabajos futuros en las mismas áreas.

El desarrollo planteado está concebido como un sistema académico que abarca tecnologías y estándares nuevos en áreas como el desarrollo para Internet y el Comercio Electrónico, y expone sus ventajas y falencias. El prototipo propuesto se aplica al escenario de las agencias de viajes, sin embargo el análisis y modelado empleado durante el diseño y los principios y metodologías aplicadas en el desarrollo son extensibles a muchas otras situaciones.

Las tecnologías utilizadas en el proyecto están en desarrollo y constante evolución por lo que la continuidad del proyecto es necesaria. Cada vez se introducen nuevas mejoras en diferentes aspectos como seguridad e integridad de los datos, especificaciones de los lenguajes, e interfaces de diseño y desarrollo computarizado, brindando una mayor gama de posibilidades para el desarrollo y aplicación de las tecnologías en el mundo de la información.

1.4 IMPACTO DE LA INVESTIGACIÓN

El desarrollo de esta investigación traerá grandes beneficios en el ámbito científico y tecnológico, puesto que:

- Servirá como refuerzo de la investigación y los desarrollos realizados en esta área los diferentes grupos de investigación adscritos a la Escuela de Ingeniería de Sistemas e Informática de la Universidad, y además actuará como punto de

partida de las exploraciones en áreas especializadas, como la aplicación de nuevos estándares como los relacionados con los servicios Web y su coordinación.

- Servirá de base académica y experimental a la comunidad UIS, con el fin de conocer e iniciar investigaciones de nuevas tecnologías para el intercambio de información en la Web, de representación de datos y de implementación de sistemas interempresariales muy exitosos en el ámbito corporativo y que comienzan también a implementarse como estándares de comunicación a nivel gubernamental en nuestro país, que es el caso de los servicios Web.

- Igualmente colaborará fomentando el desarrollo investigativo, tecnológico y económico de las empresas regionales involucradas en el ámbito del proyecto, al servir de referente de la aplicación de estándares de comunicación y tratamiento de los procesos de comercio electrónico, facilitando el diseño e implementación de nuevos y proyectos y reduciendo tiempos de desarrollo.

En resumen, este trabajo de grado es una puerta de entrada a una línea de investigación, que a largo plazo puede mejorar la situación tecnológica actual en esta área, tanto en la Universidad Industrial de Santander como en Colombia.

2. PLANTEAMIENTO DEL PROBLEMA

El presente capítulo describe la problemática abordada en este proyecto de investigación, y enuncia brevemente diferentes formas existentes de abordarlo desde el punto de vista tecnológico para lograr implementaciones exitosas.

2.1 EL COMERCIO ELECTRÓNICO Y SU ESCENARIO PLANTEADO

El auge actual de Internet y los grandes avances tecnológicos tanto en dispositivos como en medios de conexión han creado el ambiente propicio para el crecimiento y aceptación del comercio electrónico; cada vez los usuarios son menos reacios a realizar compras y ventas de productos o de servicios a través de medios electrónicos mediante transacciones a través de Internet u otras redes de computadores. Lo anterior estimula la creación de innovaciones tecnológicas, trayendo muchos desafíos al mundo del software relacionados con el diseño e implementación de las aplicaciones usadas: flexibilidad, interoperabilidad, sencillez al usuario, seguridad, movilidad, entre muchos otros.

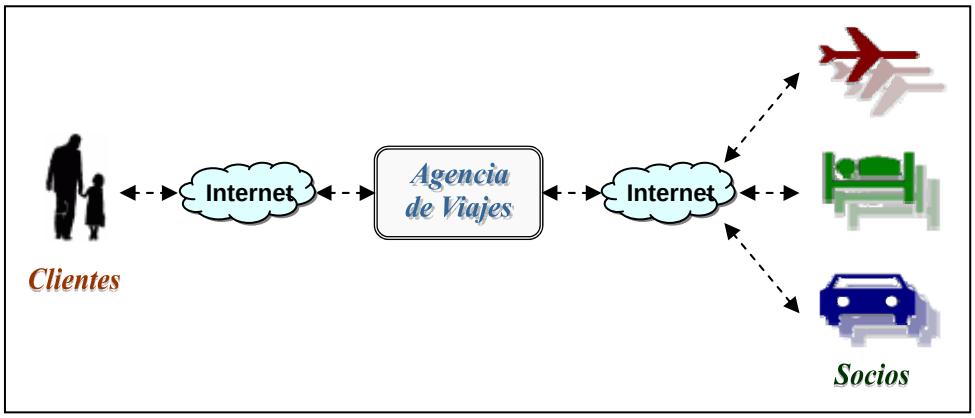
El comercio electrónico tiene un gran entorno de aplicación, y se relaciona básicamente con el intercambio de documentos entre entidades. Entre las aplicaciones principales se encuentran las compras o adquisiciones, finanzas, industria, transporte, salud y ambiente gubernamental. Actualmente sobresalen usos como el soporte técnico ininterrumpido, y el acceso interactivo a catálogos de productos y su venta directa a los clientes, conocido como tiendas electrónicas. Además un tipo de actividad notablemente beneficiada por el comercio electrónico son los sistemas de reserva y venta de servicios, como las agencias de viajes que poseen una gran base de datos de clientes y proveedores y sirven de intermediarios acordando transacciones entre ellos.

2.1.1 Agencia de Viajes Electrónica

La *agencia de viajes*, uno de los ejemplos más representativos del comercio electrónico, es una entidad cuyo principal fin es ser intermediaria entre usuarios, interesados en organizar sus viajes o planes vacacionales, y diferentes proveedores de servicios relacionados con viajes, hoteles, transporte, sitios turísticos, entre otros. En una agencia de viajes no electrónica, sus empleados o agentes de viaje realizan los procesos de búsqueda, reserva, cancelación y pagos de los servicios en representación del cliente, según las exigencias o condiciones que éste prefiera.

La dinámica sistémica de una agencia de viajes es compleja; en su estructura se aprecian diferentes *procesos* internos que interactúan con entidades externas, y que es necesario organizarlos y ejecutarlos coordinadamente, según una secuencia definida por el flujo de información inherente, para conseguir los resultados esperados y así cumplir con el servicio para el que fueron creados.

Figura 1. Caso Representativo del Comercio Electrónico: Agencia de Viajes



La agencia de viajes (electrónica) debe permitir a sus clientes acceder a sus servicios vía Internet para planear sus viajes vacacionales. De la misma manera, la agencia contactará a sus socios comerciales que proveen en individualmente

cada servicio incluido en los planes ella ofrece (transporte, alojamiento, visitas, etc.). La figura 1 ilustra gráficamente las relaciones entre cliente, agencia de viajes y socios comerciales, en el mundo de las transacciones electrónicas.

Este es el escenario escogido para el desarrollo de la presente investigación. Es necesario realizar el análisis del problema y el planteamiento de cómo abordar la complejidad y plasmarla en una solución software bajo perspectiva de orientación a servicios y su composición.

2.1.2 Implementación Informática

Diseñar e implementar una solución software para el escenario de una agencia de viajes implica gran trabajo de ingeniería. Basta con observar las tareas que realiza un agente humano: entrevistar a los usuarios y tomar nota de sus requerimientos y preferencias, contactar a los proveedores en busca de los servicios, hacer las reservaciones, realizar pagos, entre otras; para apreciar los retos y dificultades que se afrontan ya que se debe coordinar de forma automatizada las operaciones que ejecutarían, todo en una secuencia específica de acuerdo con un orden preestablecido y además dependiendo continuamente de los resultados obtenidos en cada paso previo y de criterios adicionales de decisión.

Resumiendo, los aspectos sobresalientes del escenario escogido son: (a) el acceso de clientes a la agencia y la comunicación entre ella y sus socios se hace a través de Internet y (b) los socios comerciales de la agencia son autónomos e independientes en su funcionamiento interno el cual no es necesario conocer entre ellos ni con la agencia. Estas características son requisitos cruciales que debe cumplir la aplicación informática y por tanto es necesario analizar qué posibilidades existen para su diseño e implementación.

2.2 OPCIONES DE DESARROLLO

La computación distribuida es un enfoque que propone sistemas cuyos componentes hardware y software localizados en computadores conectados en red, se comunican y coordinan sus acciones a través de pasos de mensajes siguiendo un protocolo preestablecido, logrando así un objetivo común.

El modelo de los sistemas distribuidos cumple los requisitos del escenario de la investigación, por lo tanto se analizará brevemente cada forma alterna de implementación desde sus diferentes enfoques y tecnologías existentes actualmente.

2.2.1 Arquitecturas de Objetos Distribuidos (RMI, DCOM, CORBA y .NET)

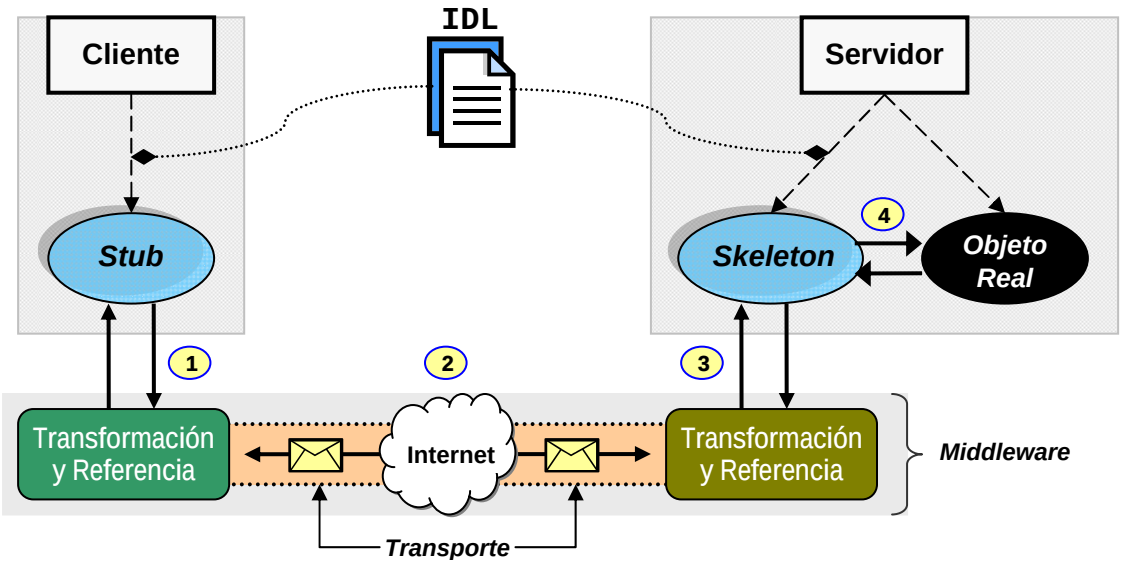
Uno de los enfoques de desarrollo de sistemas y aplicaciones distribuidos aplica el paradigma de la orientación a objetos, creando así el concepto de *objeto distribuido*. Bajo el contexto actual, un objeto distribuido es aquel que está gestionado por un servidor y sus clientes invocan los métodos que expone (invocación remota – RPC). El cliente realiza la invocación de los métodos de un objeto remoto en el servidor mediante el envío de un mensaje a este último a través de un protocolo común de comunicación. El método del objeto en el servidor se ejecuta y el resultado de ejecución se devuelve al cliente en forma de otro mensaje, es decir el mensaje es de petición - respuesta.

Para el caso de la agencia de viajes, la agencia actuaría como servidor a sus clientes, y de forma análoga tendría el rol de cliente ante sus proveedores comerciales (aerolínea, cadena hotelera, etc.) que le ofrecen los servicios que incluye en sus planes de viaje.

Estas arquitecturas utilizan típicamente un Lenguaje de Definición de Interfaces (IDL) para especificar las interfaces con los métodos que los objetos ofrecerán,

convirtiendo tipos de datos si es necesario y describiendo deben ser utilizados en las implementaciones del cliente y del servidor. Una interfaz IDL se compila para obtener código para el cliente y para el servidor (que implementará el objeto). El código del cliente, típicamente llamado *stub*, sirve para poder realizar las llamadas a métodos remotos mediante el *proxy* o representante del objeto remoto en el lado del cliente. El código generado para el servidor consiste en unos *skeletons* o esqueletos que el desarrollador implementa en los métodos a exponer del objeto.

Figura 2. Arquitectura típica de Objetos Distribuidos



La figura 2 ilustra gráficamente la arquitectura típica orientada a objetos distribuidos. El cliente, usando el *stub* (generado usando IDL), realiza la invocación del método del objeto remoto (1). Dicho llamado es recibido por un componente de *transformación y referencia* encargado de convertir (si es necesario) la información y enviarla hacia el objetivo localizado mediante un sistema de referencias con el que cuenta el middleware. El transporte se hace a través de Internet o la red, hacia el servidor (2). Ya en lado del servidor, un componente análogo de transformación y referencia (puede ser de otra clase) escucha y acepta el mensaje entrante, convierte los datos y los transfiere al

skeleton apropiado (3). El skeleton está ligado al objeto real que gestiona el servidor y en él se realiza la invocación y ejecución del método adecuado (4). La respuesta es enviada de forma similar hacia el cliente, utilizando el middleware. Existen diferentes opciones tecnológicas que utilizan este paradigma, entre las más importantes y populares se encuentran:

Java de Sun Microsystems dispone de su framework RMI (Remote Method Invocation) el cual permite la creación de aplicaciones Java distribuidas. Este sistema permite a un objeto que se está ejecutando en una Máquina Virtual Java (cliente), llamar a métodos de otro objeto que está en una JVM diferente (servidor); las máquinas virtuales entre sí se comunican a través de una red (middleware). La infraestructura RMI se encarga del procesamiento, transformación y transporte físico de los mensajes a través de la red. Esta tecnología está ligada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.[2]

Microsoft por su parte, dispuso del Modelo de Objetos de Componentes Distribuidos (DCOM) que permitía implementar componentes software distribuidos sobre varios ordenadores comunicados entre sí, haciendo uso de las API's de Microsoft. Pero luego desarrolló la plataforma de desarrollo .NET que dispone de mayores capacidades de desarrollo de implementación de aplicaciones sobre el sistema operativo Windows y que cuenta con la tecnología .NET Remoting para el desarrollo rápido de aplicaciones distribuidas. .NET también hace posible la creación de aplicaciones Web potentes.[3]

CORBA (*Common Object Request Broker Architecture*) es una tecnología introducida por el Grupo de Administración de Objetos (OMG), creada para establecer una plataforma para la gestión de objetos remotos independiente del

[2] Más información online sobre Java RMI: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

[3] Más información online sobre .NET Remoting : http://www.microsoft.com/learning/en/us/Books/6172.aspx

lenguaje de programación. En sentido general, CORBA toma el código escrito en otro lenguaje y lo "envuelve" en un paquete que contiene información adicional sobre las capacidades del código interior y sobre cómo llamar a sus métodos. Los objetos que resultan, pueden entonces ser invocados desde otro programa, u objeto CORBA, a través de la red. En esta arquitectura, existen intermediarios de solicitudes entre objetos (ORB) que forman parte de un bus de objetos donde residen las especificaciones de referencias y transporte, usadas en la invocación. Esta tecnología permite comunicación entre objetos heterogéneos: sin depender de un lenguaje de programación específico, ni del sistema operativo donde residan.[4]

2.2.2 Arquitectura de Agentes Móviles

Los Agentes Móviles son entidades software o programas informáticos que encapsulan las funcionalidades y viajan a través de la red para visitar otros dispositivos útiles para, de forma local en donde arriben, ejecutar tareas y reunir información en nombre de los propietarios, para luego regresar al punto de inicio.

Un agente móvil se compone del código con el cual define su comportamiento, el estado para poder continuar sus actividades luego de haberse movido, y sus propiedades o atributos. Se caracteriza por su autonomía (operación por sí mismo), sociabilidad (interactúa con otros agentes), reactividad (percibe el entorno y responde a él), proactividad (orientarse a metas u objetivos, iniciativa), y movilidad (suspender temporalmente su ejecución y trasladarse a otro entorno).[7]

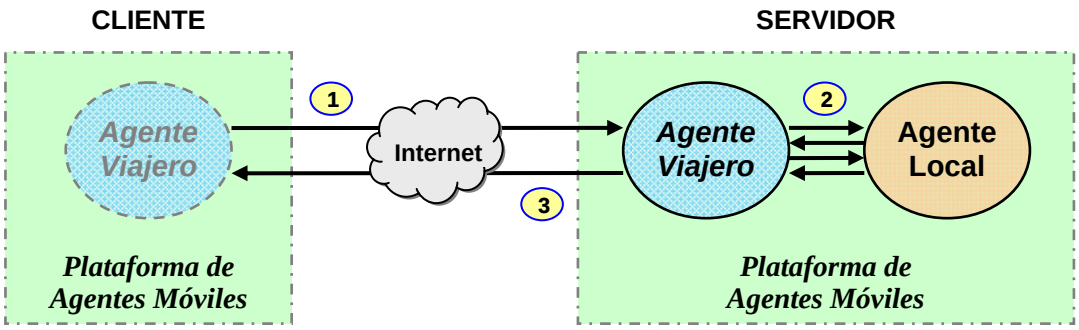
La filosofía de esta arquitectura es llevar el procesamiento a los datos, en lugar de los datos al procesamiento, reduciendo así el tráfico por la red. La figura 3 muestra el concepto de los agentes móviles y su funcionamiento típico. Para poder operar,

[4] Más información online sobre CORBA : http://www.omg.org/technology/documents/spec_catalog.htm

un agente móvil necesita de una plataforma de agentes sobre la cual se ejecutan, y contiene todos los agentes estáticos que interactúan con los viajeros.

El cliente crea un agente viajero y le da la orden de iniciar su operación; de forma autónoma (o como se programe), el agente decide viajar a un sitio determinado (servidor) y entonces se empaqueta y viaja a través de la red (1). Una vez en el servidor, es recibido por la plataforma de Agentes en la cual él se convierte de nuevo en Agente y es escuchado por un Agente Local con ese fin. El agente local al servidor tiene acceso a los programas e información interna y es él quien decide qué entrega al agente viajero. Estos dos agentes interactúan según las tareas que les asignaron (2). Cuando ya el agente viajero complete su tarea, decide viajar de nuevo hacia su origen o hacia otra máquina según sea su itinerario (3).

Figura 3. Arquitectura de Agentes Móviles



Son diversas las ventajas de los agentes móviles; entre ellas se destacan: la reducción del tráfico o carga de la red, la superación de los inconvenientes por tiempos de latencia, la ejecución asíncrona y autónoma, y la robustez y tolerancia a fallos que poseen. Entre sus principales desventajas sobresalen la existencia agentes hostiles que realicen acciones no autorizadas y nocivas para el huésped, y la posible modificación de la información del agente viajero.[7]

Actualmente existen diferentes plataformas de agentes móviles tales como Aglets originalmente de IBM Japón y ahora parte de SourceForge[5], Voyager ORB de ObjectSpace[6], y el proyecto opensource JADE (*Java Agent Development Framework*) distribuido por Telecom Italia[7]. La gran mayoría de las plataformas actuales están desarrolladas en el lenguaje de programación Java.

2.2.3 Arquitectura Orientada a Servicios

La orientación a servicios surge como extensión de la funcionalidad original de la Web (ofrecer contenidos multimedia) para permitir la oferta y demanda de servicios. Esta arquitectura forma parte de la computación distribuida y puede ser comparada con la orientada a objetos distribuidos; las diferencias principales son el uso de estándares basados en XML para definición los datos, operaciones y referencias (a modo de un IDL) y protocolos típicos de transporte para la comunicación (SOAP sobre HTTP, SMTP, FTP).

Su diseño está enfocado en garantizar la interoperabilidad entre distintas plataformas, sistemas operativos y lenguajes de programación. Y permite gran extensibilidad pues han sido desarrollados diferentes estándares para coordinar y secuenciar la ejecución de los servicios, resolviendo la complejidad de los procesos y flujos de información actuales.

Esta es la arquitectura seleccionada para esta investigación y se estudia a fondo en el capítulo 3.

---

[5] Más información online sobre Aglets: http://www.trl.ibm.com/aglets/ y http://aglets.sourceforge.net/

[6] Más información online sobre Voyager: http://www.inf.fu-berlin.de/lehre/WS99/VS/Misc/Voyager/API/

[7] Más información online sobre JADE: http://jade.tilab.com/

3. MARCO TEÓRICO Y ESTADO ACTUAL

Este capítulo muestra una selección suficiente del estado del arte y la técnica requeridos para el desarrollo de esta investigación. Inicia ilustrando las Arquitecturas Orientadas a Servicios, sus orígenes, el cambio de paradigma y los nuevos roles existentes bajo dicha concepción; luego presenta los Servicios Web XML destacando de ellos su definición, uso y principales ventajas; continúa con la Composición de Servicios Web incluyendo el lenguaje BPEL; y finaliza con el Comercio Electrónico destacando la relación entre éste y los procesos de negocio.

3.1 ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

3.1.1 El Paradigma de la Orientación a Servicios

La orientación a servicios es un paradigma para la construcción de aplicaciones distribuidas en los cuales los participantes de las transacciones son servicios, y un servicio se aprecia como un ente software que lleva a cabo una operación determinada y que puede ser invocado desde fuera del contexto de una gran aplicación. Es un concepto sencillo que presenta las siguientes características:

- Es primordial que todas las tareas u operaciones que lleva a cabo cada entidad y que deban ser visibles a las demás, sean expuestas como servicios. Es decir que las funcionalidades que preste u ofrezca cada entidad, construidas como componentes software, sean invocables y utilizables dentro de una red.

- El diseño se enfoca en la interfaz del servicio, es decir en la conversación; por tanto el diseñador se centra en los mensajes intercambiados entre los participantes de la comunicación, mas no en los participantes en sí. Estos participantes son los agentes en la conversación, y pueden ser programas o unidades funcionales pertenecientes a una organización o a un humano. Así se

logra el desarrollo de un servicio con una interfaz bien definida y que puede ser potencialmente usado en diversos contextos.

- La integración de las aplicaciones se da en el nivel de interfaz, no en el de implementación, logrando un acoplamiento débil y por ende mayor flexibilidad, pues se construyen para que puedan funcionar con cualquier servicio, dinámicamente escogido basándose en criterios preestablecidos.

- Poseen metadatos o descripciones acerca de la forma y el tipo de los elementos que transportan los mensajes, el orden y significado de los mensajes, etc. Esta información permite realizar validaciones en forma sencilla, y además brindan un mayor grado de escalabilidad a los desarrollos.

3.1.2 Roles y Operaciones

Toda arquitectura orientada a servicios presenta tres roles para sus participantes: solicitante o cliente del servicio, proveedor del servicio, y registro del servicio. Cada rol puede ser desempeñado por cualquier programa, ente o nodo en la red. Además un mismo programa puede desempeñar varios roles: por ejemplo se puede ser proveedor de un servicio, y a la vez cliente de otros servicios vitales para realizar las operaciones.

- *Proveedor del servicio (service provider)*: es la entidad responsable de la creación de la descripción del servicio, el despliegue del servicio en un entorno de ejecución que lo haga accesible en la red, la publicación de la descripción del servicio en un registro de servicios, y la recepción de los mensajes de invocación del servicio desde los clientes (solicitantes).

- *Cliente del servicio (service requestor)*: es el participante interesado en un servicio. Realiza la búsqueda de una descripción de servicio publicada en los

registros de servicios; con la descripción encontrada, el cliente establece un enlace con el proveedor oferente del servicio Web y realiza la invocación del mismo. Todo consumidor del servicio Web puede ser considerado cliente.

- *Registro de servicios (service registry)*: este actor es responsable de almacenar las descripciones de los servicios Web publicadas en él por los proveedores de servicios, y de anunciar a los clientes los servicios que contiene cuando realicen búsquedas. Se puede decir que el registro de servicios es aquel que presenta o reúne al cliente con el proveedor del servicio. Sólo se utiliza mientras se realiza la búsqueda del servicio y la publicación previa.

Los roles se relacionan entre sí mediante tres operaciones: publicación, búsqueda y enlace. En la figura 4 se aprecia visualmente los roles y sus relaciones.

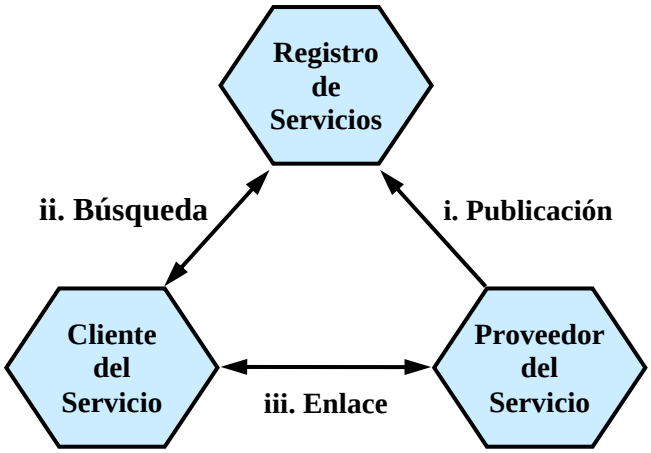
i. *Publicación (publish)*: consiste en registrar o anunciar el servicio. Se ejecuta entre el proveedor del servicio y el registro de servicios. Cuando un proveedor publica la descripción de su servicio Web a un registro, está anunciando los detalles de ese servicio Web a la comunidad de posibles clientes consumidores del mismo. La manera como interactúan durante la publicación (interfaz) es dada por la implementación del registro. El ejemplo más común es el estándar UDDI (*Universal Description, Discovery and Integration*) el cual define una implementación sofisticada de la operación de publicación.

ii. *Búsqueda (find)*: es la operación entre el cliente del servicio y el registro mediante el cual se consultan los servicios publicados previamente. El cliente define sus criterios de búsqueda como tipo de servicio, garantías de calidad, categorías, etc. y los transmite al registro mediante la operación “búsqueda”. Por su parte el registro se encarga de recibir la solicitud, comparar los criterios con las descripciones de los servicios Web publicados y devolver como

resultado la lista de descripciones de los servicios que cumplieron los criterios. Al respecto, UDDI provee potentes y amplias capacidades de búsqueda.

iii. *Enlace (bind)*: esta operación define la relación entre el consumidor y el proveedor del servicio. El consumidor o cliente contacta y solicita el servicio usando la información de descripción obtenida en la búsqueda, y el proveedor suministra como respuesta el resultado de la ejecución del servicio. Según su implementación, se clasifica como enlace estático o enlace dinámico. Es estático si el desarrollador codifica por sí mismo la forma como debe invocarse el servicio; por otro lado es dinámico, si la generación se hace basándose en la descripción del servicio y durante el instante mismo de su invocación.

Figura 4. Roles y Operaciones en SOA. Tomado de [1]



3.2 SERVICIOS WEB XML

3.2.1 Definición de Servicio Web

Los servicios Web han ganado un gran impulso y aceptación desde que fueron concebidos hace algunos años. Aún ahora parece existir una tendencia lenta hacia el entendimiento común del significado del término “Servicio Web”, es decir que no existe una definición única y universal adoptada. El grupo de trabajo de

Arquitectura de servicios Web del World Wide Web Consortium (W3C) desarrolló la siguiente definición para un servicio Web: [1]

> “Un servicio Web es un sistema de software diseñado para permitir comunicación interoperable máquina-a-máquina en una red. Posee una interfaz descrita en un formato procesable por la máquina (específicamente WSDL). Otros sistemas interactúan con el servicio Web en la forma como lo indique su descripción y usando mensajes SOAP típicamente transportados usando HTTP con serialización XML conjuntamente con otros estándares del Web.”

Existen varios aspectos relevantes sobre los servicios Web, entre ellos están:

- Aunque se denomine servicio Web, necesariamente no existe en la Web; puede residir en cualquier lugar de una red ya sea interna o externa. De igual forma puede ser utilizado internamente por una aplicación o bien, ser expuesto de forma externa en Internet por varias aplicaciones.

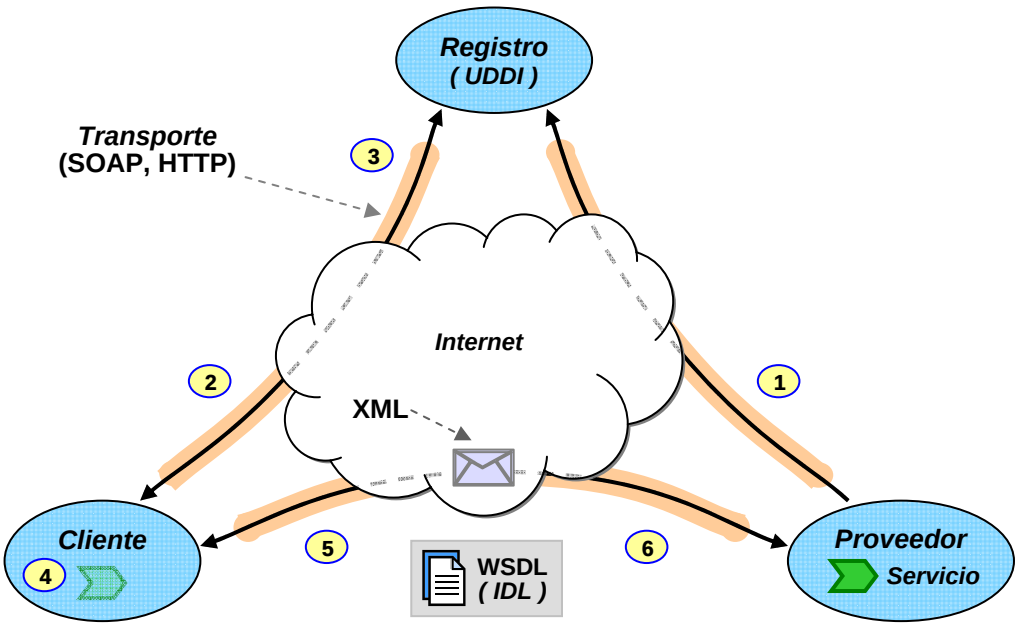
- Las plataformas de implementación y despliegue del servicio Web no son relevantes para el programa que lo invoca. Un servicio Web está disponible a través de su interfaz declarada y el mecanismo de invocación (protocolos de red, codificación de datos, etc.). Esto implica que los componentes son poco acoplados, permitiendo el funcionamiento de una serie de sistemas heterogéneos como un conjunto integrado.

- Al ser software, un servicio Web XML es una entidad programable que proporciona una funcionalidad determinada, como lógica de aplicación.

- No se centran en la portabilidad del código, sino que habilitan la interoperabilidad de datos y sistemas, utilizando mensajería basada en XML

para la comunicación de datos y así contribuir a reducir las diferencias existentes entre entornos, que utilizan distintos modelos de componentes, sistemas operativos y lenguajes de programación.

Figura 5. Ilustración gráfica del modelo de los servicios Web



La figura 5 muestra gráficamente el funcionamiento primario de cada componente (rol) del modelo de los servicios Web, y cómo interactúan entre sí a través de Internet, ampliando lo expuesto en 3.1.2. El *proveedor* expone una funcionalidad en forma de *servicio* y lo publica en el *registro* de servicios (1). Luego, un *cliente* interesado realiza búsquedas en este registro utilizando un lenguaje estándar como UDDI (2). El registro realiza la búsqueda según lo definido por el cliente y si encuentra servicios publicados en él con las características dadas, envía de regreso al usuario una respuesta satisfactoria conteniendo la información útil de conexión a los servicios (3), usando lenguaje WSDL, que contiene la descripción de cada servicio y la forma como debe ser accedido (equivalente al IDL de las arquitecturas enunciadas en 2.2). El cliente utiliza la descripción del servicio para realizar el enlace con él (4), y posteriormente lo invoca a través de la red (5). El

paso de mensajes se realiza utilizando protocolos de Internet como SOAP y HTTP, y en formato XML que es universalmente entendido y aceptado. Una vez el servicio recibe la petición, ejecuta el servicio y devuelve la respuesta al cliente (6).

Los servicios Web pueden ser vistos principalmente desde dos perspectivas según lo que representan y el uso que se les da, éstas son:

- *Perspectiva de Negocio*: los servicios Web permiten integrar fácilmente las funcionalidades de las aplicaciones internas y entre organizaciones similares. Esta unión brinda eficiencia en tiempo y costo. Además la combinación de servicios Web, permite la construcción rápida de sistemas de negocio.

- *Perspectiva Técnica*: un servicio Web es una colección de una o más operaciones relacionadas, accesibles en una red y definidas por una descripción de servicio. Los servicios Web se enfocan en la interfaz entre quien lo invoca y el servicio invocado, usando estándares simples y abiertos que permiten alcanzar sus capacidades en cualquier plataforma.

3.2.2 Ventajas de los Servicios Web

El desarrollo de sistemas distribuidos usando servicios Web tiene muchos beneficios, entre los cuales se destacan:

- Permite integrar aplicaciones con rapidez, eficiencia y bajo costo. Además en formas que difícilmente podrían darse con otras tecnologías; facilita la integración de sistemas con diversas tipos de dispositivos como teléfonos celulares y PDAs, con proveedores de servicios de cualquier forma y tamaño.

- Reduce los tiempos de implementación de sistemas que soportan procesos de negocios existentes, ya que permite extender la funcionalidad actual de un

componente difícil de integrar mediante una función simple a manera de "cubierta" que sirva de interfaz rápida y económica.

- El uso de estándares brinda una mayor gama de productos relacionados con esta área en el mercado, además reduce los costos de adquisición al evitar monopolios de tecnología de un solo fabricante, permitiendo que pequeñas y medianas organizaciones participen en el comercio con empresas grandes y poderosas.

- El desarrollo basado en interfaces y usando descripciones de servicios reduce el tiempo de integración de aplicaciones pues se necesita menos información para interactuar; cliente y proveedor sólo necesitan el conocimiento mutuo de las entradas, salidas y ubicación del servicio.

- Los programadores pueden crear aplicaciones que entrelacen servicios Web de orígenes diversos, de modo similar a como se utilizan tradicionalmente componentes en la creación de aplicaciones distribuidas.

3.2.3 Pila de Interoperabilidad de los Servicios Web

Los servicios Web han venido evolucionando desde finales de 1999, y pueden ser entendidos como un conjunto de tecnologías organizadas en un modelo de capas, aclarado que la combinación usada en las implementaciones puede variar. [1]

- Capa de Transporte

  Los servicios Web son básicamente un mecanismo de mensajería por lo cual las tecnologías de transporte de dichos mensajes están en la base de esta concepción. Se considera neutral pues puede ser implementada por diversos protocolos como HTTP, HTTPS, SMTP, TCP/IP, entre otros.

Figura 6. Pila de interoperabilidad de Servicios Web planteada en [1]

Composición	**BPEL4WS, WS-Notification**
Calidad de Servicio	**WS-Security, WS-ReliableMessaging, WS-Transactions, WS-ResourceLifetime**
Descripción	**WSDL, WS-Policy, UDDI, WS-ResourceProperties**
Mensajería	**XML, SOAP, WS-Addressing, JMS**
Transporte	**HTTP, HTTPS, SMTP, TCP/IP, ...**

- Capa de Mensajería

  En esta capa son fundamentales las tecnologías como SOAP, XML, JMS y WS-Addressing. Los servicios Web usan el estándar XML como el lenguaje básico para realizar su descripción. La especificación de SOAP define un mecanismo sencillo usado para el trasporte y la representación de los mensajes intercambiados por los servicios Web durante su funcionamiento.

- Capa de Descripción

  Las tecnologías usadas para definir los metadatos o las descripciones de los servicios Web incluyen WSDL, WS-Policy, UDDI y WS-ResourceProperties. El Lenguaje de Descripción de Servicios Web (WSDL) es el formato más importante utilizado para describir las características funcionales de los servicios Web incluyendo las operaciones que expone y los mensajes de entrada y salida asociados.

- Capa de Experiencia y Calidad del Servicio

  A esta capa pertenecen las especificaciones asociadas con la calidad de las soluciones que usan servicios Web, y su objetivo es aclarar ciertas

capacidades y requerimientos relacionados con las transacciones, seguridad y confiabilidad de los mensajes. WS-Security es una familia de especificaciones que definen la interacción segura de un servicio Web.

- Capa de Composición

  La finalidad de los estándares propuestos en esta capa es describir cómo combinar y componer servicios Web, extendiendo la funcionalidad de un servicio Web mediante la interacción automática con otros ya definidos. Aquí se encuentran especificaciones y trabajos como BPEL4WS, el estándar más aceptado en el mundo de los negocios para componer servicios Web.

3.2.4 Lenguaje de Descripción de Servicios Web: WSDL

Con el propósito de describir el formato, las opciones de transporte y los detalles necesarios para acceder e invocar un servicio Web se ha creado el Lenguaje de Descripción de Servicios Web (WSDL) basándose en XML. WSDL constituye un principio básico de la arquitectura de servicios Web al proveer un lenguaje común para la descripción de servicios y una plataforma para la integración automática de los mismos. En el contexto de la arquitectura, WSDL representa un convenio entre el proveedor y el cliente del servicio; el cliente puede localizar un servicio Web e invocar cualquiera de sus funciones publicadas. En esencia, una descripción WSDL presenta tres de sus propiedades fundamentales:

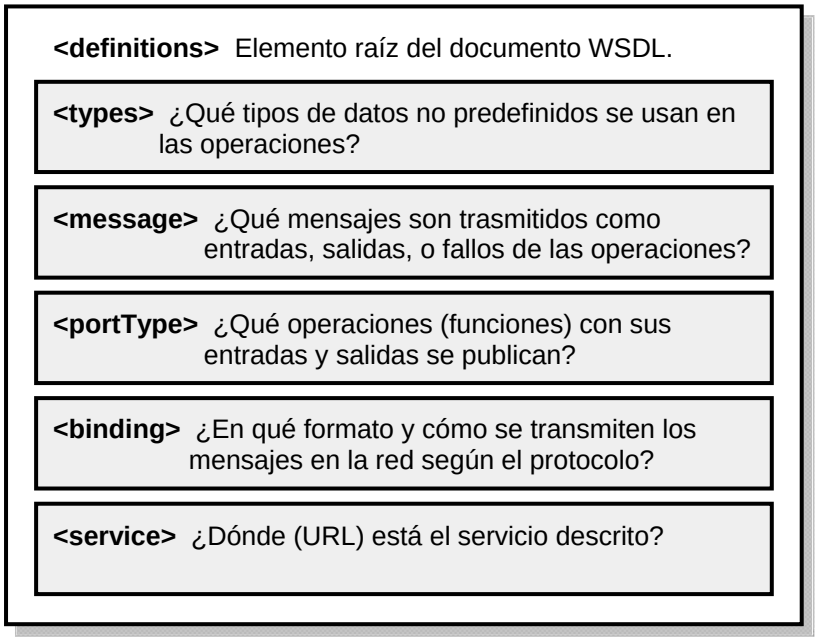
- *Qué hace el servicio*: las operaciones, métodos o funciones que provee el servicio (publicación), y los datos necesarios para invocarlos (argumentos de entrada y resultados).
- *Cómo se accede al servicio*: detalles de los formatos de datos, mensajes a transmitir y protocolos de transporte necesarios para acceder sus operaciones.
- *Dónde se localiza el servicio*: información relacionada con la ubicación del servicio en la red según su protocolo y acceso, por ejemplo la URL.

Una característica importante del modelo de información WSDL es su separación entre especificaciones abstractas e sus implementaciones concretas. Lo anterior se refleja en la división entre definiciones de interfaz de servicio (interfaces abstractas, *portTypes* en el documento) y las definiciones de implementación de servicio (implementación concreta como nodo de red, *bindings* en el documento).

3.2.5 Estructura de un documento WSDL

Todo documento WSDL es un archivo XML que cumple con el esquema WSDL versión 1.1 definido por IBM, Microsoft y otras compañías en Marzo de 2001 y propuesto para estandarización al W3C[8]. Dicha especificación propone un esquema dividido en seis elementos XML principales descritos a continuación. [3]

Figura 7. Significado de los elementos de un documento WSDL



[8] La especificación WSDL 1.1 es la más usada aunque en la actualidad la W3C ha desarrollado una estandarización llamada WSDL versión 2.0. Para más información ver la especificación 1.1 en http://www.w3.org/TR/wsdl y la 2.0 en http://www.w3.org/TR/wsdl20. El esquema XML de la versión 1.1 también está disponible en http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd.

- Definiciones *<definitions>*

  Este es el elemento raíz de todo documento WSDL: contiene a todos los demás. Posee atributos para especificar el nombre de las definiciones y los espacios de nombres XML (*namespaces*) usados en el resto del documento.

- Tipos de Datos *<types>*

  El elemento types describe los tipos de datos de la información transmitida por las operaciones definidas. Por defecto existen los definidos en la especificación XML Schema del W3C[9] (string, int, float, etc.). Es posible definir un esquema XML que contenga los tipos complejos o elementos personalizados necesarios.

- Mensajes *<message>*

  Este elemento hace referencia a mensajes unidireccionales que pueden actuar en peticiones como entradas, en respuestas como salidas, o en fallas (faults) como el contenido de las mismas. Está especificado por el nombre del mensaje y un conjunto de cero o más "partes" del mensaje (elemento *<part>*) que se refieren a parámetros en peticiones, o al valor de retorno en resultados.

- Interfaces *<portType>*

  Las interfaces declaradas con el elemento portType permiten la definición abstracta de operaciones. Cada operación es definida con el elemento *<operation>* combinando mensajes como entradas (elemento *<input>*), salidas (elemento *<output>*), y fallos o excepciones (elemento *<fault>*), formando invocaciones de operaciones con o sin argumentos y con o sin respuesta.

- Enlaces Concretos *<binding>*

  Este elemento permite implementar las interfaces portType definiendo la forma concreta como el servicio será formateado durante su transporte en la red.

[9] Para más información sobre el Esquema XML ver http://www.w3c.org/TR/xmlschema-0/

WSDL incluye extensiones para definir servicios SOAP sobre HTTP, SOAP sobre SMTP, HTTP GET/POST, entre otros.

- Servicio *<service>*

  Define el nombre del servicio publicado. Mediante el elemento *<port>* le asigna a cada enlace (binding) definido una dirección de red (típicamente URLs) que corresponde al sitio Web donde reside el servicio Web.

En síntesis, un documento WSDL presenta la anatomía mostrada en la figura 8, en la cual un asterisco (*) indica multiplicidad del elemento, y los puntos suspensivos (...) indican que requieren de atributos adicionales o contenido extra. La estructura completa se encuentra en la especificación del W3C.

Figura 8. Esqueleto de un documento WSDL

```
<definitions ...>
   <types>
      <xsd:schema ...>...</xsd:schema> *
   </types>
   <message> *
      <part ...></part> *
   </message>
   <portType> *
      <operation> *
         <input ...></input>
         <output ...></output>
         <fault ...></fault> *
      </operation>
   </portType>
   <binding ...> *
      <operation> *
         <input>...</input>
         <output>...</output>
         <fault>...</fault> *
      </operation>
   </binding>
   <service> *
      <port ...>...</port> *
   </service>
</definitions>
```

3.2.6 Garantía de Interoperabilidad de los Servicios Web

Aunque los servicios Web nacieron con el ánimo de ofrecer interoperabilidad entre servicios ejecutándose en diferentes plataformas e implementados con diferentes lenguajes de programación, esto no se ha dado fácilmente debido a problemas causados por la aplicación e implementación inadecuada de los estándares.

Diversos grupos en el mundo se han dedicado a estudiar la forma de garantizar la interoperabilidad de los servicios Web. Entre ellas destaca la Organización de Interoperabilidad de Servicios Web, WS-I Organization[10] que es un consorcio de industrias con el objetivo de promover la adopción de la tecnología de los servicios Web, y fue fundada por diversas compañías como Microsoft, IBM, BEA Systems, SAP, Oracle, Hewlett-Packard, entre otras.

El resultado de su trabajo se expresa en *profiles* o guías para un uso correcto de los estándares, *sample applications* o aplicaciones de ejemplo basadas en los profiles, y *test tools* o herramientas de verificación de conformidad con las guías. Las guías dadas en los profiles clarifican el uso de los estándares de servicios Web mediante la eliminación de la ambigüedad existente. Los más importantes usados en las arquitecturas de servicios Web actuales son:

- *WS-I Basic Profile 1.1* [19]: guía el uso correcto de servicios Web.
- *WS-I Attachments Profile 1.0* [20]: define el uso adecuado de datos adjuntos en mensajes SOAP.
- *WS-I Simple SOAP Binding Profile 1.0* [21]: define requerimientos para el uso de mensajes SOAP como único contenido de los mensajes HTTP.

[10] Más información de la WS-I Organization disponible en http://www.ws-i.org

3.3 COMPOSICIÓN DE SERVICIOS WEB

En la mayoría de los escenarios del mundo real, un proveedor de servicios Web no sólo expone servicios simples y estáticos, como por ejemplo la verificación de la temperatura. Los servicios Web se reutilizan y combinan en servicios más complejos que proporcionan funcionalidades más completas al usuario. Este proceso de combinación incluye la definición de la lógica de negocio en términos del orden de ejecución y las condiciones de invocación de los servicios Web coordinados. Los nuevos servicios Web son de mayor nivel aunque aún flexibles por su facilidad de adaptación a las necesidades cambiantes del entorno.

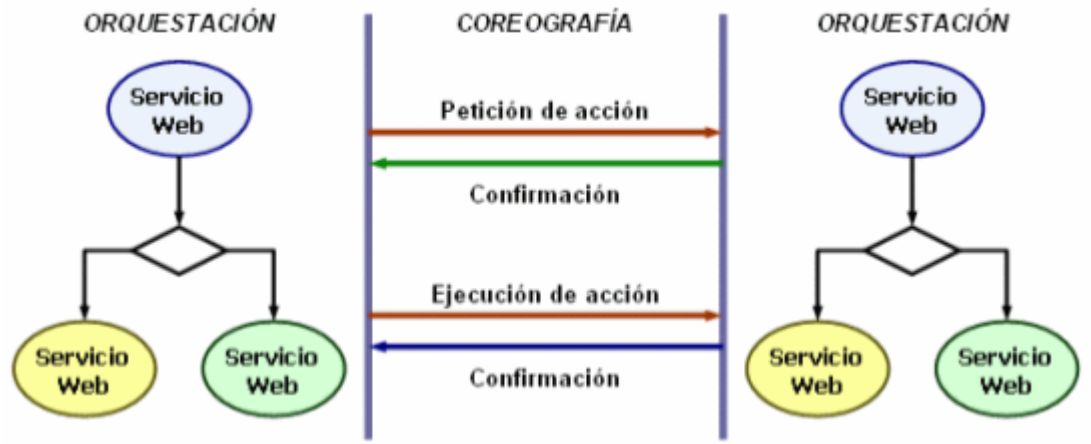
Como los servicios Web se están pasando a ser una tecnología clave para la implementación de aplicaciones distribuidas, han aparecido nuevos dominios de implementación que necesitan una infraestructura más compleja para el desarrollo de servicios; por esto los estándares básicos relacionados con servicios Web: WSDL, SOAP y UDDI no satisfacen todos las requerimientos, creando la necesidad de nuevos estándares a un nivel superior al de la descripción, que se encarguen de la definición de los nuevos procesos a implementar.

3.3.1 Orquestación y Coreografía de Servicios Web

La creación de procesos de negocio involucrando varios servicios Web, conlleva dos aspectos muy importantes denominados *orquestación y coreografía* [5].

La orquestación consiste en el diseño de procesos de negocio ejecutables que puedan interactuar con servicios Web, tanto internos como externos, a través de mensajes; siempre existe control del proceso de negocio desde la perspectiva de una de las partes involucradas en el mismo. La coreografía, que implica más colaboración, permite definir las posibles secuencias de mensajes que ocurren entre todas las partes involucradas en el proceso de negocio. No se centra en los mensajes que se intercambian desde y hacia una única parte.

Figura 9. Relación entre orquestación y coreografía de servicios Web



En la orquestación una de las partes actúa como “directora” y desde ella se aprecian los demás, en la coreografía se aprecia la totalidad, resaltando la forma en se relacionan entre sí todas las partes.

3.3.2 Lenguaje de Ejecución de Procesos de Negocio - BPEL

El Lenguaje de Ejecución de Procesos de Negocio *(Business Process Execution Languaje for Web Services – BPEL4WS*) o llamado simplemente BPEL, en un lenguaje XML que permite el modelado del comportamiento de los participantes en interacciones basadas en servicios Web. Esto incluye desde escenarios de integración de aplicaciones en el interior de una organización, hasta interacciones B2B[11]. Este lenguaje es usado para componer un conjunto de servicios Web en un proceso BPEL que puede ser publicado como otro servicio Web para ser utilizado por otras entidades.

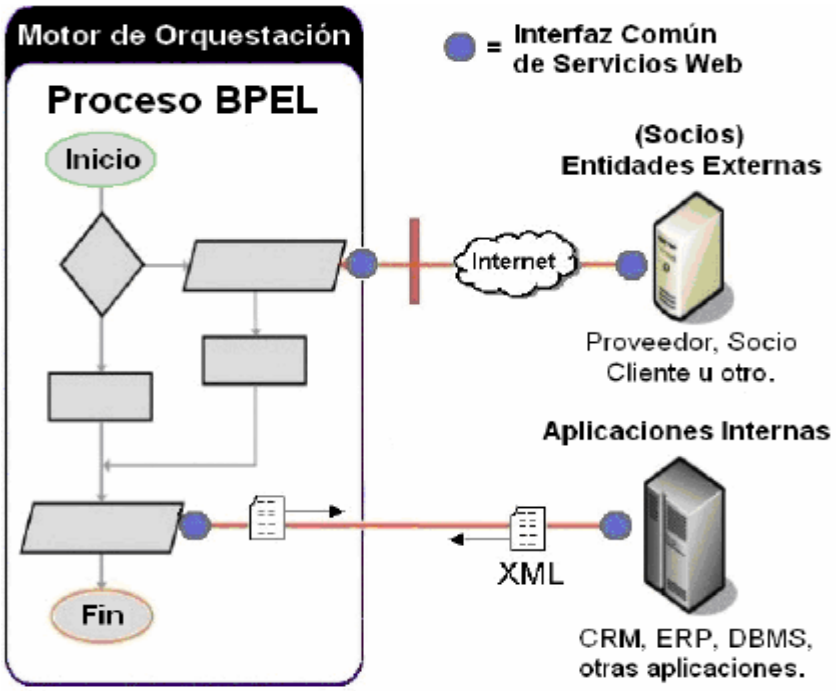
BPEL ha sido desarrollado mediante la combinación de los conceptos del Lenguaje de Flujo de Servicios Web (WSFL) y de XLANG propuestos por IBM y Microsoft respectivamente como lenguajes dentro de la “programación a gran

[11] B2B es la sigla como se conoce la interacción entre negocios o empresas a través de Internet. En estas interacciones, las aplicaciones se comunican sin requerir intervención humana.

escala" (programming in the large), que se refiere a algoritmos de la lógica de estados en un sistema a alto nivel. Esta lógica codifica la información concerniente al envío de mensajes, tiempos de espera de respuestas, contrarresto de transacciones fallidas, etc.

A finales del 2007, se terminó la especificación actual de BPEL llamada **WS-BPEL 2.0**[10], publicada por el OASIS Web Services BPEL Technical Committee[12] que es la entidad encargada de la formalización y mantenimiento del estándar, partiendo de la especificación inicial BPEL4WS 1.1 realizada en Mayo de 2003 por las compañías BEA Systems, IBM Corporation, Microsoft Corp., SAP AG y Siebel Systems[13].

Figura 10. Tecnología de integración BPEL. Tomado de [4]



[12] Para más información ver http://www.oasis-open.org/committees/wsbpel/

[13] La especificación inicial puede encontrarse en el sitio web de cada una de las compañías gestoras del estándar, por ejemplo IBM: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

3.3.3 Principales Objetivos de BPEL

El desarrollo del lenguaje BPEL siguió unos objetivos de diseño específicos y fundamentales [1], entre los cuales se destacan:

- Los procesos de negocio están descritos en un lenguaje basado en XML. La representación gráfica de los mismos está fuera del alcance de BPEL.

- El lenguaje define conceptos para vistas abstractas de protocolos de negocio y vistas internas de los procesos. Los procesos de negocio ejecutables modelan el *comportamiento real* de un participante en una interacción, mientras que las vistas abstractas especifican el comportamiento visible del intercambio de mensajes de cada actor del proceso, *sin revelar* su comportamiento interno.

- El lenguaje combina los beneficios de los lenguajes jerárquicos como XLANG, y de los orientados a grafos como WSFL.

- BPEL permite combinar actividades estructuradas en algoritmos de complejidad arbitraria que representan la implementación del proceso de negocio; los servicios Web son usados para la descomposición y el ensamblado de los procesos, tal como en un modelo basado en módulos.

- BPEL posee un conjunto limitado de términos, suficientes para permitir un manejo sencillo de datos, necesario en el control de flujo (coordinación de actividades) y en la definición de información relevante al proceso.

3.3.4 Extensiones WSDL para BPEL

BPEL es un lenguaje que permite escribir “programas” que usen servicios Web y a la vez sean un servicio Web; al implementar la lógica del programa se está

siguiendo el flujo de los procesos que involucra la secuencia de interacciones de los mismos. Su especificación extiende el lenguaje de descripción de servicios Web (WSDL), siendo en el fondo un esquema XML para representar los procesos y las actividades a realizar. Las extensiones del lenguaje WSDL que introduce BPEL son:

- Vínculos entre Socios: <*partnerLinkType*>

  Un proceso de negocio puede invocar servicios Web de otros "socios" (partners) y/o suministrar servicios Web para que sean accedidos por otros socios. Un proceso establece "conversaciones" con sus pares mediante el uso de servicios Web. Para describir la relación entre los dos servicios Web, BPEL introduce una extensión WSDL llamada <*partnerLinkType*>. En ella se definen los roles que juegan cada uno de los servicios en la conversación especificando la interfaz WSDL (portType) que provee cada servicio para recibir los mensajes en el contexto actual.

Figura 11. Ejemplo de declaración de Vínculo entre Socios

```
<partnerLinkType name="enlaceHotel">
   <role name="buscador">
      <portType name="htl:hotelPortType" />
   </role>
</partnerLinkType>
```

- Propiedades <*property*> y Alias de Propiedad <*propertyAlias*>

  Con el propósito de permitir la interacción de los servicios con procesos en ejecución, es necesario brindar un mecanismo con el cual los socios identifiquen cada proceso de forma única. Esta correlación entre las instancias de los procesos en ejecución y los mensajes que llegan se logra mediante campos identificadores llamados *propiedades*. Una propiedad se declara con el elemento <*property*> que es extendido de WSDL y posee nombre y tipo de

datos de su contenido. Para obtener el valor de la propiedad se necesita conocer la instancia del mensaje WSDL al que se asigna (messageType), la parte del mensaje a la que pertenece (part) y la cadena de selección dentro del elemento (query) que es una extensión XPath[14]. Esta información conforma el *alias de propiedad*, (asigna un alias o nombre al elemento del mensaje especificado) que se declara con el elemento <*propertyAlias*>.

Figura 12. Ejemplo de declaración de Propiedad y su correspondiente Alias

```
<property name="idCliente" type="xsd:ID" />
<propertyAlias propertyName="idCliente"
               messageType="htl:peticionBusqueda"
               part="datosEntrada"
               query="/infoCliente/id" />
```

3.3.5 Estructura general de la definición de un proceso BPEL

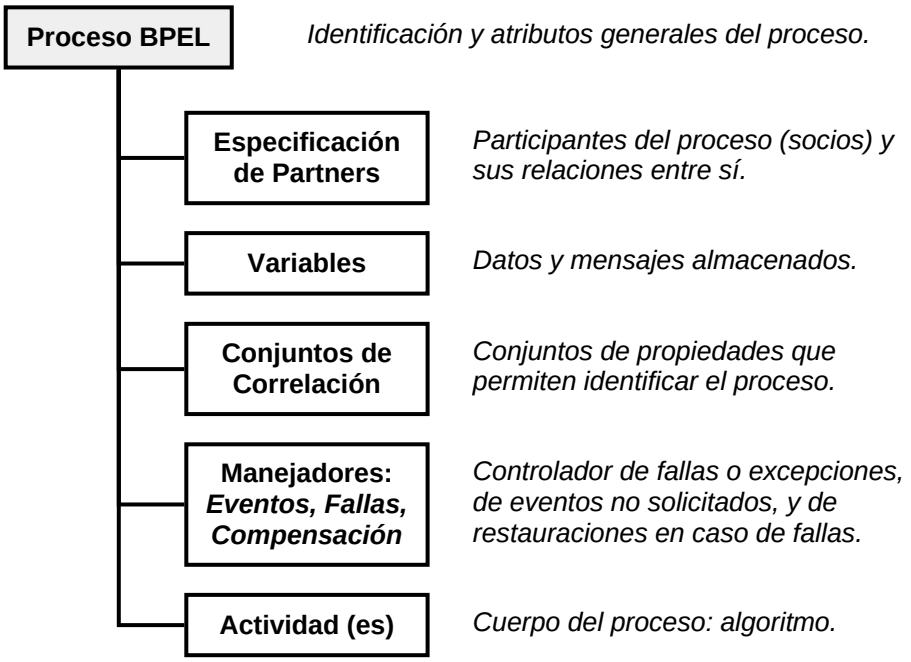
Un documento con la definición de un proceso BPEL (abstracto o ejecutable) es un archivo XML que cumple los esquemas declarados en la especificación WS-BPEL 2.0. La estructura general propuesta se aprecia en la figura 13 y se explica a continuación.

- Proceso BPEL <*process*>

  Es el elemento raíz de todo proceso BPEL y por ende contiene a todos los demás elementos. Mediante atributos se especifican el nombre, la configuración general como si es abstracto o no (*abstract*), y los espacios de nombres XML requeridos.

[14] XML Path Language (XPath) es un lenguaje basado en XML que permite buscar y seleccionar subconjuntos de un documento XML teniendo en cuenta su estructura jerárquica. Más información y la especificación en http://www.w3.org/TR/xpath

Figura 13. Estructura de un proceso de negocio BPEL.



- Especificación de Partners <*partnerLinks*> y <*partners*>

  La declaración interna a nivel del proceso de las relaciones o vínculos entre las entidades pares se realiza con el elemento <*partnerLink*> agrupados dentro de <*partnerLinks*>. Cada uno posee nombre, tipo de vínculo definido previamente (ver partnerLinkType en 3.3.4), rol desempeñado por el proceso y el rol que desempeña el socio (opcional). Además se introduce el elemento <*partner*> que concentra cada uno de los vínculos en los que participa cada entidad definiendo sus interacciones; todos se agrupan con el elemento <*partners*>.

- Variables <*variables*> y <*variable*>

  Las variables son la forma en que BPEL puede almacenar los mensajes intercambiados en las transacciones con los socios, y datos XML independientes de las interacciones que actúan como variables de programa. El elemento BPEL usado para declararlas es <*variable*> y en él se especifica el nombre y el mensaje del cual toma su valor o el tipo de datos que guarda (si es variable independiente). Todas se agrupan con el elemento <*variables*>.

- Conjuntos de Correlación <*correlationSets*> y <*correlationSet*>
  Estos conjuntos son grupos de propiedades (ver 3.3.4) usados para identificar un proceso y permitir la correspondencia entre mensajes entrantes y una instancia determinada. Cada conjunto se declara con el elemento <*correlationSet*> que posee su nombre y la lista de propiedades; todos los conjuntos se agrupan dentro del elemento <*correlationSets*>.

- Manejadores de Eventos <*eventHandler*>, <*onMessage*> y <*onAlarm*>
  Los eventos son sucesos no esperados en forma de mensajes entrantes que pueden ser recibidos y procesados en paralelo durante la ejecución de un proceso. BPEL define dos clases de eventos: *Mensajes* producto de invocaciones de los partners, y *Alarmas* a modo de temporizadores. Se pueden declarar globalmente dentro del elemento <*eventHandler*> usando los elementos <*onMessage*> y <*onAlarm*> respectivamente, y especificando dentro de ellos las acciones a realizar para procesar el evento según su tipo.

- Manejadores de Fallos <*faultHandlers*>, <*catch*> y <*catchAll*>
  Un fallo es una situación excepcional ocurrida durante invocaciones de servicios, causada por problemas de ejecución, o porque es parte de la lógica del proceso. El elemento <*catch*> permite capturar un fallo específico y realizar ciertas actividades que se definan. A su vez, <*catchAll*> intercepta cualquier tipo de fallo. Estos manejadores pueden definirse a nivel del proceso con <*faultHandlers*>, o dentro de la actividad de invocación de servicios <*invoke*>.

- Manejadores de Compensación <*compensationHandler*> y <*compensate*>
  La compensación es una característica de BPEL que permite deshacer actividades que ocurrieron previos a algún fallo y así corregir la situación. Esto se realiza mediante la ejecución de actividades que el algoritmo defina. Para declararlos se usa el elemento <*compensationHandler*> localizado en el ámbito donde puede ocurrir la falla, y dentro de él se especifican las actividades de

compensación. En el momento en que ocurra un fallo y se requiera, la compensación puede ser invocada con el elemento <*compensate*>.

- Actividades: Cuerpo del Proceso

  Las actividades son los elementos con que se implementa la lógica del negocio del proceso BPEL. Se distinguen *actividades básicas* que representan operaciones atómicas del proceso de negocio, y *actividades estructuradas* que permiten definir conjuntos de operaciones anidadas y el orden en que se establecen. En la tabla 1 se resumen las actividades disponibles en el estándar WS-BPEL 2.0 y su función. [1]

Tabla 1. Listado de actividades disponibles en un proceso BPEL

Actividad	Función
receive	Espera la recepción de un mensaje de petición de un partner, producida por la invocación del servicio.
reply	Proporciona respuesta a la invocación recibida por *receive* usando un mensaje de respuesta del partner.
invoke	Invoca una operación especificada en el portType WSDL de algún partner.
terminate	Termina la ejecución del proceso.
throw	Lanza un fallo desde dentro del proceso indicando una situación excepcional.
assign	Permite la manipulación de los datos de las variables copiando sus valores desde y hacia otras variables o expresiones.
wait	Espera por un periodo de tiempo o hasta un instante específico.
empty	Es una actividad nula o instrucción sin operatividad.
compensate	Inicia la compensación de un grupo de actividades.
if	Permite aplicar condicionales para seguir determinado flujo de actividades.
switch	Permite seleccionar un camino de ejecución entre varias posibilidades.
while	Permite especificar un ciclo de acciones con condicional al inicio.
repeatuntil	Permite realizar un ciclo de acciones con condicional al final.
foreach	Repite determinado número de veces las actividades contenidas.
pick	Permite ejecutar una de diversas opciones tan pronto como llegue un mensaje adecuado, o cuando se cumpla un tiempo establecido.
sequence	Especifica un grupo de actividades que se ejecutan en secuencia
flow	Define una colección de actividades para ejecución en paralelo.
scope	Crea un ámbito con su propio grupo de variables, conjuntos de correlación, y manejadores, y dentro del cual pueden realizarse actividades anidadas

Figura 14. Ejemplo del documento un proceso de negocio BPEL

```
<process name="procesoBusqueda" abstract="no" ...>
   <partnerLinks>
      <partnerLink name="hotel"
                   partnerLinkType="htl:enlaceHotel"
                   myRole="buscador" />
      ...
   </partnerLinks>
   <variables>
      <variable name="peticionBusqueda"
                messageType="htl:peticionBusqueda" />
      <variable name="codRespuesta"
                type="xsd:string" />
      ...
   </variables>
   <correlationSets>
      <correlationSet name="identificadorBusqueda"
                      properties="pro:idCliente pro:fecha" />
      ...
   </correlationSets>
   <faultHandlers>
      <catch faultName="htl:datosNoValidos">
         ...
      </catch>
      ...
   </faultHandlers>
   <messageHandlers>
      <onMessage partnerLink="hotel"
                 portType="htl:hotelPortType"
                 operation="cancelarBusqueda" .../>
         ...
      </onMessage>
      ...
   </messageHandlers>
   <!– Actividades – Logica/Algoritmo del Proceso de Negocio -->
</process>
```

3.3.6 Consideraciones Adicionales sobre BPEL

La definición de un proceso BPEL no tiene en cuenta ni la forma de enlace (*binding*) ni el sitio donde se encuentran los servicios con los que interactúa bien

sea internos o externos (partners). El estándar BPEL 1.1 maneja la noción de “referencia al sitio” (*endpoint reference*) para representar los datos que describen el sitio real donde se localiza cada servicio a acceder y así permitir establecerla dinámicamente durante la ejecución del proceso o estáticamente en una definición adicional independiente del proceso.

BPEL permite la definición de *protocolos de negocio* a través de la definición de *procesos abstractos*. Un protocolo de negocio se refiere a la definición de la forma como interactúan los partners por medio de mensajes y su posible secuencia: en otras palabras es conocer el orden en el cual deben “conversar” para obtener el propósito del negocio sin conocer los detalles que ocurren internamente. Los procesos abstractos no son ejecutables pues sólo brindan una visión de los procesos internos de los socios desde la perspectiva de la entidad principal. Su implementación con el lenguaje BPEL es usada sólo para modelarlos aunque proporcionan un entendimiento más completo de las interacciones entre las entidades. [1]

3.4 PROCESOS DE NEGOCIO Y COMERCIO ELECTRÓNICO

La enorme acogida de Internet alrededor del mundo ha casi estandarizado el uso de redes para realizar intercambios de información, bienes y servicios entre personas, empresas y organizaciones de todos los tipos. Uno de los ejemplos más claros de la influencia positiva del internet a nivel mundial es el comercio electrónico pues las transacciones comerciales vía Web se han convertido en determinante a la hora de competir. Las organizaciones y empresas son conscientes de la demanda de interactividad y facilidad de comunicación existente en el mercado, y por tanto necesitan implementar las tecnologías recientes de forma que reduzcan costos, y disminuyan tiempos y acorten distancias. Estos son beneficios que brinda el comercio electrónico, lo que hace que sea de vital importancia su uso y mejora en las organizaciones.

La dinámica de colaboración y participación existente entre organizaciones en el mundo del comercio expone la necesidad de automatizar los procesos de negocio inherentes. En este ámbito, se deben tener en cuenta tanto los procesos internos de cada organización (con los cuales funcionan como ente independiente) y los existentes a nivel externo, que se crean con el flujo de actividades presente entre diferentes tipos de organizaciones relacionadas o vinculadas para ejecutar operaciones de mayor dimensión y complejidad significativa.

A medida que se adopta el uso de internet y las nuevas tecnologías, es necesario la creación de estándares en la comunicación y la implementación de los sistemas de información que utilizan las empresas para intercambiar datos; estándares que son producto de investigación y desarrollo en elementos determinantes como la eficiencia y velocidad de las transacciones, y sobre todo la seguridad de la información, que proporciona confianza a los actores.

Ahora bien, el uso de los servicios Web y la gran facilidad de aplicación en esta clase de escenarios, hace necesaria más investigación al respecto, al igual que creatividad en el desarrollo de herramientas y soluciones que mejoren la experiencia de las organizaciones y los usuarios, y también requiere de profesionales capacitados y con experiencia en el área para que den soporte y planteen nuevos enfoques sobre esta arquitectura.

4. TECNOLOGÍAS APLICADAS AL PROYECTO

En este capítulo son presentadas y explicadas las tecnologías usadas para el desarrollo del proyecto, tales como son la plataforma de ejecución de procesos de negocio, la arquitectura de servicios Web, entre otros.

4.1 ENTORNO DE EJECUCIÓN DE PROCESOS BPEL: ACTIVEBPEL

Desde de la aparición del estándar del Lenguaje de Ejecución de Procesos de Negocio BPEL se han implementado diversos motores que permiten la ejecución de dichos procesos entre los cuales sobresalen:

- *Active Endpoints ActiveVOS 7.0*[15]: es una suite comercial de administración de procesos de negocio con los últimos estándares, desarrollado por la compañía Active Endpoints, Inc. Está orientado hacia el desarrollo en grandes empresas y por tanto tiene soporte para procesos de gran tamaño y complejidad.

- *IBM WebSphere Process Server*[16]: es el módulo de desarrollo BPEL muy completo y robusto, que hace parte de la familia de desarrollo WebSphere de IBM y funciona sobre el servidor de aplicaciones Java EE.

- *Microsoft BizTalk Server*[17]: es el servidor de administración de procesos compatible con BPEL, desarrollado por Microsoft y que está integrado con la tecnología .NET.

[15] Información sobre la suite en http://www.activevos.com/

[16] Más información en la pagina Web del producto: http://www-01.ibm.com/software/integration/wps/

[17] La Información sobre el producto está disponbile en http://www.microsoft.com/biztalk/default.mspx

- *ActiveBPEL Engine*: es un entorno de ejecución BPEL escrito en Java y distribuido como open-source (software libre) bajo GPL. Es la versión no comercial del motor empresarial desarrollado por Active Endpoints, Inc. [11]

Existen muchos otros desarrollos tanto open-source como comerciales, los cuales ofrecen entornos completos orientados a diferentes tipos aplicaciones y usos y desarrollados en diversos lenguajes de programación.

4.1.1 ActiveBPEL, El motor Open-source de BPEL

Para el desarrollo del proyecto se seleccionó el producto ActiveBPEL Engine. Este entorno ha sido desarrollado con la misma tecnología de los productos comerciales de la empresa Active Endpoints, Inc., y por tanto ofrece su confiabilidad y robustez necesaria y apreciada en la industria. Este producto surgió gracias a la creencia de la compañía que "*el modelo open-source es un medio efectivo para fomentar el interés de la comunidad, la educación y el desarrollo alrededor del estándar BPEL*".

Aparte de lo enunciado anteriormente, el motor ActiveBPEL:

- Es completo ya que el motor implementa en su totalidad la especificación BPEL4WS 1.1, y además incluye complementos para actividades, manejo de eventos, manejo de excepciones y administración de compensaciones, que habilita la ejecución de procesos bajo el estándar WS-BPEL 2.0.

- Tiene gran aceptación pues además incluye características avanzadas como despliegue de paquetes, persistencia de procesos, notificaciones de eventos y una consola para ejecución de funciones de su librería de interfaz de programación.

4.1.2 Arquitectura de ActiveBPEL

El motor ActiveBPEL ejecuta procesos escritos en BPEL de la siguiente forma: acepta las definiciones de los procesos (contenidas en archivos XML), crea las instancias necesarias basándose en su definición y posteriormente ejecuta la secuencia especificada.[18]

Existen tres áreas principales en la arquitectura: el *motor*, los *procesos* y las *actividades*. El motor coordina la ejecución de uno más procesos BPEL. Los procesos a están compuestos de actividades, las cuales a su vez pueden contener enlaces a otras actividades.

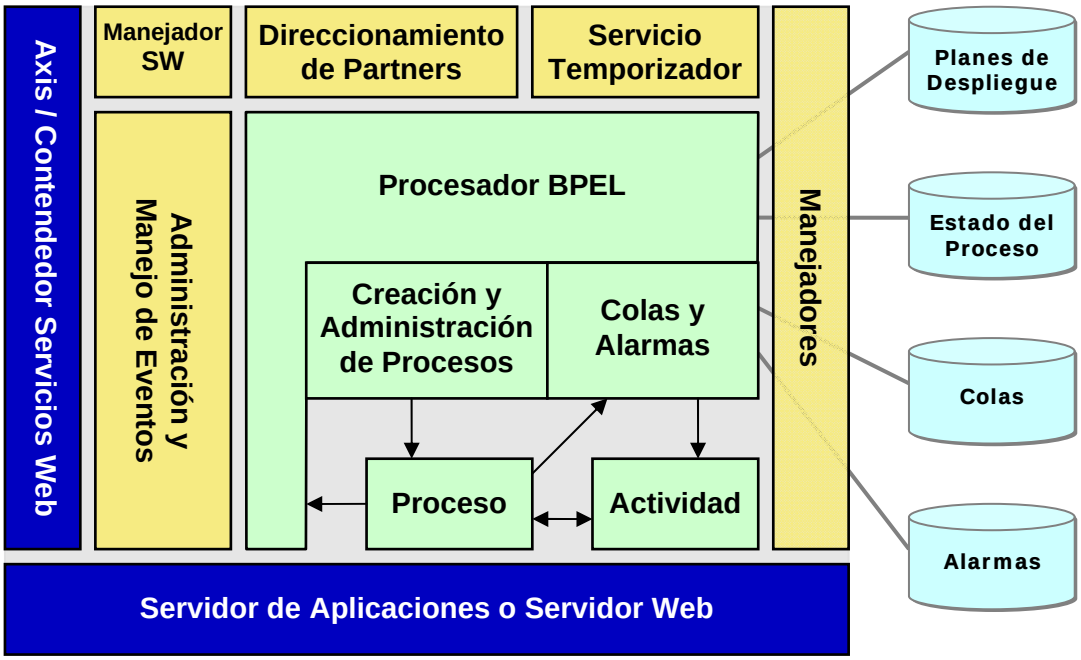
4.1.2.1 Estructura y componentes del motor

El Motor es el encargado de coordinar la ejecución de los procesos BPEL. Tal como se aprecia en la figura 15, los componentes del motor de ActiveBPEL están ubicados sobre un Servidor de Aplicaciones o un Servidor Web el cual es necesario para la comunicación vía Web (protocolo HTTP) o por mensajes entre las aplicaciones involucradas, y al mismo nivel que un Contenedor de Servicios Web como Axis, encargado de desplegar las referencias, realizar los enlaces y manejar las comunicaciones usando protocolo SOAP a los servicios tanto locales como externos y que son utilizados como partners en el proceso BPEL.

- El Procesador BPEL (*BPEL Processor*) es el núcleo del motor pues se encarga de crear, administrar los procesos y controlar su ejecución, también maneja las colas y alarmas que despliegan actividades y crean procesos. Adicionalmente y mediante diferentes manejadores, controla la persistencia de estados del sistema, de datos, y el almacenamiento en memoria y/o en disco.

[18] ActiveBPEL Engine Architecture, http://www.activebpel.org/docs/architecture.html. Última consulta en Febrero de 2007

Figura 15. Arquitectura del motor de ActiveBPEL



- La función del componente de Administración y Manejo de Eventos (*Admin and Event Manager*) es recibir, interpretar y responder los mensajes entrantes y salientes relacionados con los eventos ocurridos en el proceso y provocados por peticiones de los partners.

- El Manejador de Servicios Web (*Web Service Handler*) realiza la interfaz entre el contenedor de servicios Web y el procesador BPEL al traducir a los formatos establecidos internamente para los datos XML.

4.1.2.2 Funcionamiento del motor

Algunos aspectos relevantes del funcionamiento del motor de ActiveBPEL son:

- *Inicio y Carga*: Durante la creación del motor de ActiveBPEL, diferentes objetos se encargan de manejar los servicios de Administración de Colas, Alarmas y Temporizadores. El seudocódigo que describe la creación del motor y los servicios de soporte es:

```
motor = nuevo Motor( obtenerConfiguracionDeMotor(),
                     crearAdministradorDeColas(),
                     crearAdministradorDeEstadoProcesos() );
motor.definirAdministradorDePlan( crearProveedorDeDespliegueProcesos() );
crearAdministradorDeDespliegueProcesos();
crearAdministradorDeTrabajo();
crearServicioDeAlarmaYTemporizador();
```

La configuración del motor es tratada por un objeto que suministra valores por defecto y lee el archivo de configuración *aeEngineConfig.xml*, con varias opciones que lo hacen altamente configurable. El Proveedor de Despliegue de Procesos maneja la lectura de los archivos *Descriptores de Despliegue de Procesos* (pdd - Process Deployment Descriptor), y el Administrador de Despliegue de Procesos se encarga de la creación de los procesos. El administrador de trabajo realiza el control de las operaciones asíncronas.

- *Creación de los Procesos*. Un Nuevo proceso BPEL se crea cuando una de sus actividades de inicio es desencadenada por un mensaje entrante o por una alarma de Inicio de actividad. Cuando un mensaje entrante contiene datos de correlación, el motor localiza el proceso existente que corresponda con dicha información. En el momento que el motor lee la definición del proceso BPEL, crea objetos llamados Definición de Actividades los cuales modelan el proceso.

- *Entrada y Salida*. El motor ActiveBPEL no se encarga de la entrada y salida por sí mismo. Existen manejadores específicos según el protocolo de transferencia como el AeBpelRPCHandler para llamadas SOAP de tipo RPC, y el AeBpelDocumentHandler en el caso de llamadas SOAP de tipo Document; éstos manejadores traducen los datos a mensajes y viceversa.

- *Manejo de Datos*. Todas las variables definidas implementan la interfaz *IAeVariable*. Ésta le proporciona la habilidad de obtener la definición y el

contenido el cual difiere según esté declarada en el proceso como un tipo, un elemento o un mensaje.

- *Evaluación de Expresiones*. Todas las actividades y enlaces permiten el uso de expresiones para obtener los diversos atributos de los objetos, mediante las extensiones XPath definidas en la especificación BPEL.

- *Registro de Operaciones*. El motor de ActiveBPEL dispone de la opción de registrar los diferentes eventos que indican el progreso de la ejecución de los procesos. Esto se logra con una instancia de la clase AEEngineLogger la cual escucha y registra en archivos por cada proceso.

4.1.2.3 Despacho de mensajes de entrada

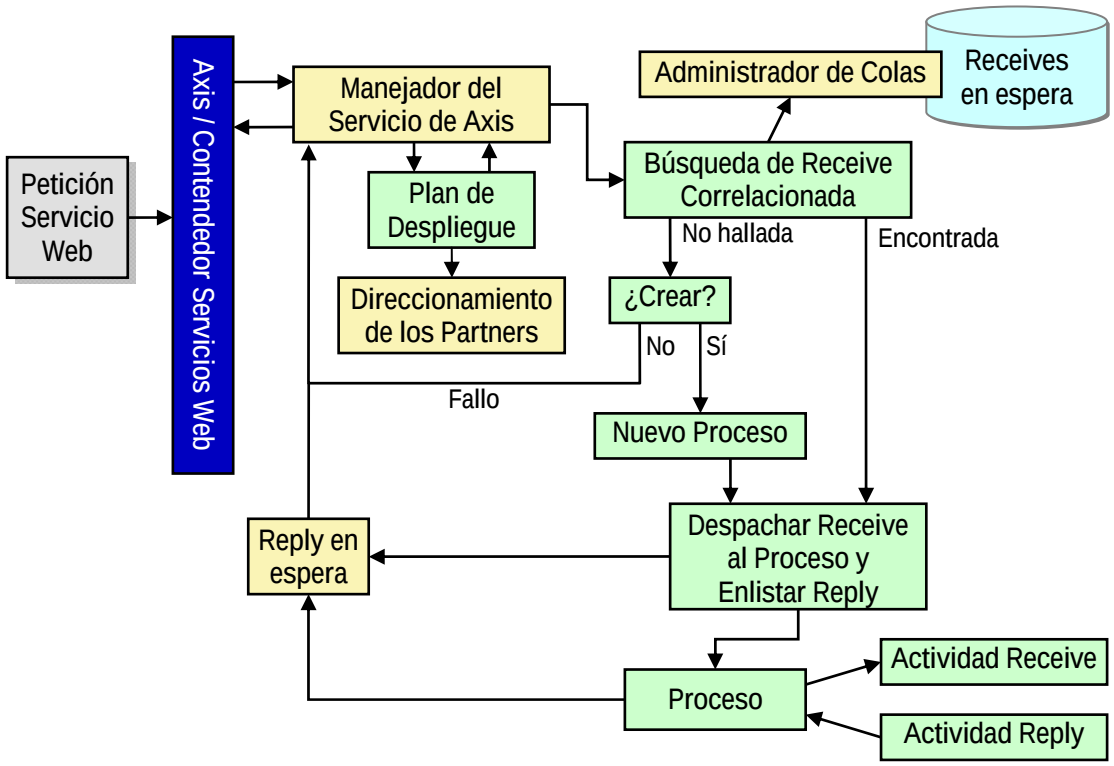
Un nuevo proceso BPEL se crea cuando una de sus actividades iniciales es accionada para ejecución, ya sea por un mensaje entrante o por una alarma en una actividad pick. La figura 16 muestra el diagrama de flujo con la secuencia que sigue un mensaje de petición al arribar al motor ActiveBPEL.

El motor envía los mensajes entrantes a la instancia de proceso correspondiente. Si existe información de correlación, el motor trata de encontrar la instancia correcta que concuerda con dicha información. Si no existe la información de correlación y la petición va a una actividad de inicio, se crea una nueva instancia de proceso.

La cola de receives (recepciones) es una lista de espera que contiene las actividades receive en ejecución en todas las instancias de proceso. Se dice que una actividad receive está en ejecución cuando ha sido puesta en espera por su actividad padre y aún no ha recibido el mensaje esperado desde el mundo exterior (fuera del proceso) para el cual se creó. Además esa lista contiene mensajes de entrada originados en el exterior que no han coincidido con ninguna actividad

receive en espera y que por sí solos no pueden crear nuevas instancias de proceso. Una recepción de datos sin concordancia puede suceder debido a la naturaleza asíncrona de algunos servicios Web.

Figura 16. Flujograma del despacho de mensajes de entrada en ActiveBPEL



4.2 ARQUITECTURA DE SERVICIOS WEB: APACHE WEB SERVICES

Existen diversas plataformas comerciales y open-source vigentes, enfocadas al desarrollo con servicios Web. Cada una dispone de recursos tecnológicos que utilizan sus propios lenguajes y aplicaciones de diseño, y brindan la posibilidad de implementar servicios Web de forma casi automática cumpliendo con las bases y reglas de la arquitectura, dadas en las especificaciones. Las principales plataformas son:

- *Microsoft .NET*[19]: la tecnología .NET de Microsoft Corporation permite el desarrollo integrado de servicios Web desde cualquiera de sus lenguajes, utilizando el Servidor de Información de Internet (IIS) y el lenguaje ASP.NET para su publicación. Cuenta un grupo extenso de herramientas para desarrollo con XML, diseño de productos, publicación de servicios, entre otros.

- *IBM WebSphere*[20]: la familia de desarrollo WebSphere de IBM es conjunto de productos comerciales muy completos y robustos para desarrollo de aplicaciones empresariales enfocados a la Web: integración comercial, comercio electrónico, administración de la información, etc. Ofrece una total compatibilidad con J2EE.

- *J2EE Web Services*[21]: Sun Microsystems desarrolló un completo grupo de herramientas y librerías para el desarrollo de servicios Web en la plataforma Java 2 Enterprise Edition.

- *Apache Web Services*: el Proyecto Web Services de Apache Software Foundation es un conjunto de desarrollos Java orientados las tecnologías y estándares de servicios Web. Está en continuo desarrollo aunque su estabilidad es apreciable, la realimentación que ofrece la comunidad de desarrolladores incrementa la calidad a medida que evoluciona. Es muy completo y de gran aceptación debido a que facilita la implementación de servicios al encapsular y automatizar gran parte del trabajo con XML y los demás estándares. [12]

---

[19] Todos los pormenores sobre esta tecnología en http://www.microsoft.com/latam/net/introduccion/ y en el Centro de Desarrollo de Microsoft: http://msdn.microsoft.com/webservices/

[20] Completa información y referencia en el sitio Web http://www-01.ibm.com/software/websphere/

[21] Para más información vea el sitio Web de Sun Microsystems http://java.sun.com/webservices/

Para el desarrollo del proyecto se utilizará el conjunto de proyectos de *Apache Web Services* pues brindan cualidades mínimas necesarias similares a las de productos comerciales (como .NET y WebSphere), un gran soporte y documentación técnica que posibilita el desarrollo, y porque ofrece grandes facilidades de implementación acordes con los objetivos del proyecto en los cuales no está contemplado el desarrollo pormenorizado de servicios Web (necesario al aplicar tecnologías como la de Sun Microsystems).

4.2.1 Contenedor de Servicios Web: AXIS

IBM y Apache Software Foundation crearon en 1999 una implementación Open Source del protocolo SOAP para el servidor Apache. Con el progreso de los estándares del Web como XML, esta implementación evolucionó hasta convertirse en AXIS (Apache eXtensible Interaction System) que, como sus antecesores, permite el manejo de SOAP, XML, y en sus versiones recientes integra las especificaciones de Sun Microsystems para servicios Web[22].

AXIS posee una API muy completa para el despliegue y hospedaje de servicios Web desde un servidor que soporte servlets (como Tomcat), o incluso un servidor independiente. Entre las ventajas más importantes de AXIS se encuentran: [1]

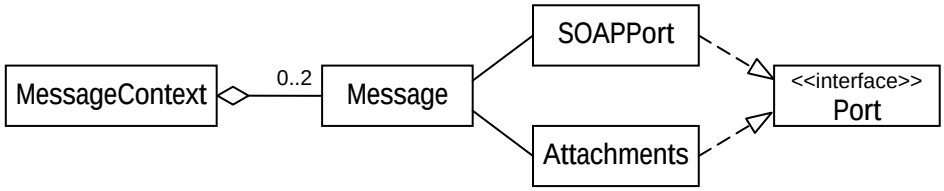
- Una completa API a nivel del cliente para la invocación de servicios Web SOAP (contando o no con la descripción WSDL).
- Un conjunto funcional para manipular los mensajes SOAP e incluso adicionar elementos adjuntos, al igual que una estructura de transporte que permite el uso de una gran variedad de mecanismos como JMS, correo electrónico, etc.
- Herramientas para traducir documentos WSDL en conjuntos de clases Java tanto para clientes como para proveedores de servicios.

[22] En el sitio web de Apache Axis http://ws.apache.org/axis/ se encuentra la información de descarga e instalación del producto y toda la documentación necesaria para clientes y desarroladores.

4.2.1.1 Funcionamiento de AXIS

El funcionamiento de Axis está basado en el procesamiento de mensajes. Durante la ejecución se invoca una serie de *Handlers* (manejadores) en orden, pasándoles como argumento un objeto denominado *MessageContext* (contexto del mensaje). Un MessageContext está formado por un mensaje *request* (petición) o un mensaje *response* (respuesta) y un conjunto de propiedades, y es usado para transmitir esos mensajes y ambientes asociados a través de la secuencia de Handlers. [13]

Figura 17. Estructura de un MessageContext en Axis



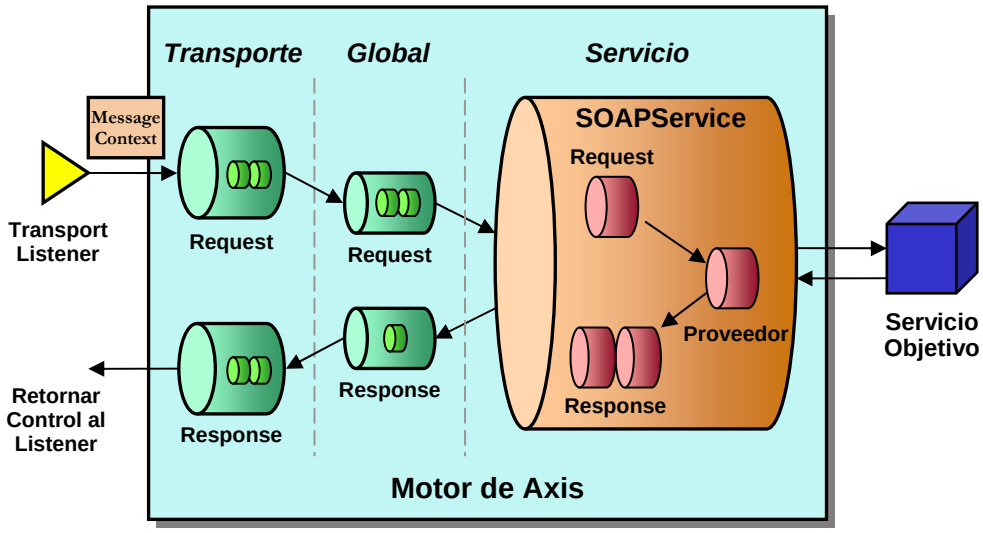
El trabajo del sistema Axis es llevar el objeto MessageContext creado a través de unos conjuntos de Handlers ordenados y ejecutados en serie llamados *Chains* (cadenas), los cuales tienen la posibilidad de realizar con el objeto todo aquello para lo que fueron diseñados. La manera como se crea y la trayectoria que sigue depende de la forma como Axis sea invocado: como servidor o como cliente.

- Servidor

  Axis es invocado por un Transport Listener que es un objeto que monitorea la red esperando peticiones en mensajes, trasmitidos según protocolos dados. El mensaje es recibido por el Transport Listener tal como muestra la figura 18.

  El trabajo del Transport Listener es empaquetar la información recibida (de acuerdo al protocolo) en un objeto de Mensaje (clase org.apache.axis.Message) y ponerlo dentro de un MessageContext junto con una serie de propiedades que lo describen. Una vez que el MessageContext está listo, el TransportListener lo entrega al motor.

Figura 18. Trayecto de un MessageContext en Axis como Servidor



Dentro del motor, el MessageContext se propaga por los Request Chain. Primero se localiza el de Transporte de acuerdo con la propiedad “nombre”, y se invoca con el método *invoke()* pasándole el MessageContext como argumento. Esto desencadena llamadas a los Handlers especificados en la configuración del Chain. El proceso continua en cascada por todos los Chain y Handlers hasta que llega al Service Chain típicamente representado por instancias de la clase SOAPService (org.apache.axis.handlers.soap.SOAPService) las cuales contienen un Handler llamado *Provider* (proveedor) responsable de implementar la lógica de llamada al servicio.

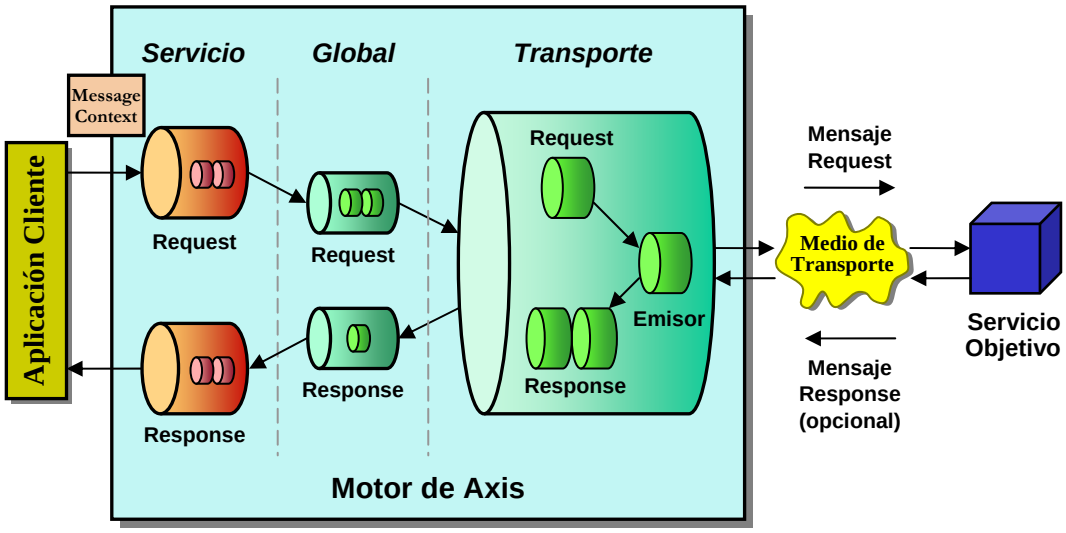
El resultado del servicio, entregado como mensaje de respuesta es almacenado en el MessageContext, que se devuelve a través de la secuencia de response Chains.

- Cliente

  Axis es invocado directamente por una aplicación (generalmente ayudada por el modelo de cliente de Axis) la cual crea el MessageContext pertinente. La ruta que sigue el mensaje en el lado del cliente de Axis es similar a la del servidor,

excepto por que el orden de las acciones se invierte, como se muestra en la figura 19.

Figura 19. Trayecto de un MessageContext en Axis como Cliente



Primero se llama al service Handler (manejador de servicio) debido a que no existe un proveedor local. Allí, el request Chain realiza cualquier procesamiento requerido por mensaje de petición, previo a su entrada al sistema; por su parte, el response Chain procesa el mensaje de respuesta que va de regreso.

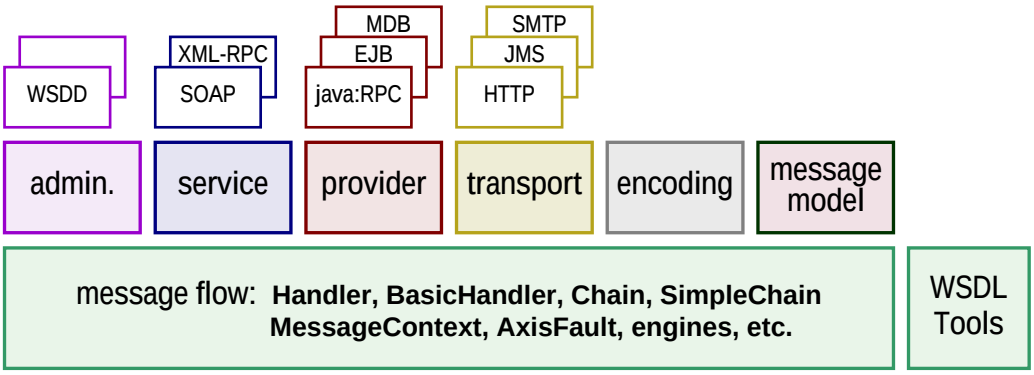
Al llegar al Transport Chain, un manejador especial llamado *Sender* (emisor) ejecuta las acciones necesarias para transmitir y recibir el mensaje al servidor SOAP. Si existe respuesta, es almacenada en el MessageContext y devuelta a través de la secuencia de response Chains.

4.2.1.2 Visión general de la arquitectura de AXIS

El diseño de AXIS plantea una división en varios subsistemas o módulos en modelo de capas, trabajando mancomunadamente pero separando actividades. Esto posibilita el uso de partes del sistema sin usar la totalidad del mismo. [13]

La figura 20 muestra un diagrama de capas con los subsistemas de Axis. Las capas son independientes entre sí, aunque las mostradas en pila son componentes independientes que pueden ser usados simultáneamente.

Figura 20. Estructura modular de Axis



- El subsistema *Message Flow* define todas las clases necesarias para representar los mensajes y controlar su ruta a través de los motores de Axis.

- La definición de clases y procesamiento relacionado con la configuración de los motores de Axis corre por cuenta del subsistema *Administration*.

- El subsistema *Encoding* se encarga de la transformación de datos entre los tipos de datos propios del lenguaje de programación y los de XML. Define clases que realizan la serialización XML de los objetos y tipos primitivos Java, y la de-serialización del código XML para obtener objetos y datos primitivos Java.

- Las clases relacionadas con la representación de los mensajes internos y los mensajes SOAP (envelope, header, body, etc.) son definidas en el subsistema *Message Model*.

- El subsistema *WSDL Tools* brinda facilidades de generación de código Java desde WSDL y viceversa. La herramienta *WSDL2Java* permite generar las

clases Java necesarias para acceder al servicio Web definido en un documento WSDL dado. Y *Java2WSDL* genera un documento WSDL con la definición de un servicio tomando como base operaciones definidas en una clase Java.

4.2.2 Seguridad de Servicios Web: Apache WSS4J

Apache WSS4J (Web Services Security for Java) es una implementación del estándar OASIS Web Services Security (WS-Security) del OASIS Web Services Security (WSS) Technical Committee[23]. [15]

En esencia el proyecto WSS4J es una librería Java que es usada para firmar y verificar mensajes SOAP con información de WS-Security. Es un proyecto reciente en constante evolución, construido sobre Apache Axis y los proyectos de Apache XML-Security, y su interoperabilidad con servidores y/o clientes basados en JAX-RPC y.NET está en desarrollo. Actualmente, WSS4J implementa la versión WS-Security 1.0 (2004), y dentro de ella las especificaciones:

- *Web Services Security: SOAP Message Security V1.0* la cual describe, entre otros aspectos, mejoras en la mensajería SOAP para proporcionar integridad y confidencialidad a los mensajes, mediante el uso de la amplia variedad de modelos de seguridad y tecnologías de encriptación existentes. [16]

- *Web Services Security: UsernameToken profile V1.0* que trata acerca de cómo usar UsernameTokens con la especificación Web Services Security (WSS). [17]

- *Web Services Security: X.509 Certificate Token Profile V1.0* que describe el uso de Certificados X.509 con la especificación Web Services Security (WSS). [18]

[23] El OASIS Web Services Security TC se dedica a estudiar y emitir estándares relacionados con la seguridad de los servicios Web, utilizando http://www.oasis-open.org/committees/wss/

4.2.3 Servidor Web: Apache Tomcat

Apache Tomcat (conocido anteriormente como Jakarta Tomcat) es el servidor Web de la familia Apache que funciona como contenedor de servlets y lo convierten en la Implementación de Referencia Oficial para las tecnologías Java Servlet[24] y JavaServer Pages[25] de Sun Microsystems. [14]

Tomcat es el servidor utilizado por los diferentes proyectos de desarrollo de servicios Web no sólo de The Apache Software Foundation sino de muchas otras compañías que lo consideran como un entorno fuerte, robusto y multiplataforma dado que está escrito en lenguaje de programación Java y funciona en cualquier sistema operativo que disponga de la máquina virtual.

4.3 LENGUAJE DE PROGRAMACIÓN: JAVA – SUN MICROSYSTEMS

El desarrollo de los componentes software del proyecto se realizó utilizando el lenguaje de programación Java de Sun Microsystems[26], más exactamente la plataforma Java, Standard Edition 6, versión 1.6.17. Las razones fundamentales que motivaron esta selección son:

- Algunas de sus características principales como seguridad, serialización de objetos, organización e independencia de la plataforma, las cuales facilitan la ejecución de código y el desarrollo de los componentes.

- La gran aceptación que ha tenido para el desarrollo de aplicaciones empresariales y comerciales debido a su estabilidad y experiencia.

[24] La tecnología Java Servlets extiende la funcionalidad de un servidor Web pues permite ejecutar aplicaciones en él y generar resultados HTML. Más información en http://java.sun.com/products/servlet

[25] JavaServer Pages es una tecnología que permite la creación dinámica, rápida y simplificada de contenido Web. Más información en http://java.sun.com/products/jsp

[26] Documentación e instaladores del producto Java disponibless en http://java.sun.com/javase/

- Es el lenguaje en el que están desarrollados los demás componentes tecnológicos usados en el proyecto como el motor ActiveBPEL. Esto garantiza el funcionamiento de los mismos, y brinda uniformidad al desarrollo.

4.4 MOTOR DE BASE DE DATOS: MySQL

La implementación y pruebas de las diferentes entidades software diseñadas en el proyecto implican el uso de información de prueba que, aunque pequeña en tamaño, es de fácil almacenamiento y manipulación si se diseña mediante bases de datos relacionales. Por tanto se seleccionó el sistema gestor de base de datos MySQL desarrollado y soportado por Sun Microsystems[27]. Las razones principales por las cuales que se realizó la selección son:

- No representa costos para el proyecto ya que es de libre distribución. Sin embargo posee gran cantidad documentación, tanto referencia como ejemplos, y su desempeño es rápido y consistente dando una alta confiabilidad.

- Su sencilla utilización desde aplicaciones realizadas en Java, gracias al controlador de base de datos MySql Connector/J proporcionado por la empresa MySQL AB, que sirve de interfaz entre JDBC[28] y el protocolo de red de MySQL.

- Funciona en más de 20 plataformas incluyendo LINUX y Windows, lo cual brinda una gran flexibilidad al desarrollo.

---

[27] Mayor información sobre el producto MySQL, documentación e instalación en http://www.mysql.com/.

[28] JDBC (Java Database Connectivity) es la biblioteca software estándar de la industria para la conectividad entre Java y un amplio número de bases de datos y orígenes de datos. Utiliza lenguaje de consulta SQL. http://java.sun.com/javase/technologies/database.jsp

5. DESARROLLO E IMPLEMENTACIÓN SOFTWARE

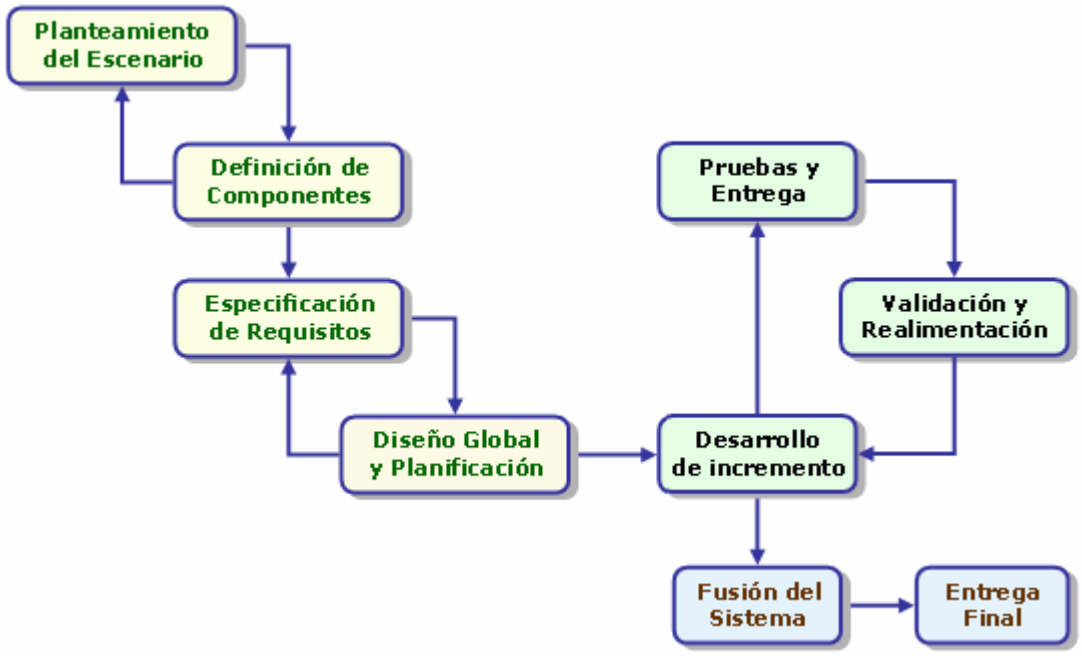
El presente capítulo expone el proceso de desarrollo del sistema e implementación del software del proyecto. Inicia explicando brevemente la metodología de desarrollo utilizada, luego se documenta cada una de las etapas del proceso como son el estudio preliminar, la estructuración del sistema –planteamiento del diseño–, los pasos de la implementación y las pruebas pertinentes realizadas al software.

5.1 METODOLOGÍA DE DESARROLLO

La metodología de desarrollo usada en este proyecto es el Modelo de Desarrollo Incremental [22], seleccionada debido a que se adapta en mayor grado a las condiciones de diseño y desarrollo de aplicaciones bajo esta arquitectura de servicios Web, brindando la capacidad de controlar cada fase o componente en desarrollo y de ir evolucionando gradualmente el software generado mediante la realimentación y la revisión y/o redefinición de los requerimientos. El modelo de desarrollo incremental adaptado al proyecto presenta las fases que se muestran en la figura 21.

Según la metodología, el desarrollo del proyecto se divide en varias etapas. Las dos etapas iniciales consisten en el Planteamiento del Escenario y la Definición de Componentes y van de la mano ya que mientras en la primera se va realizando un estudio del ámbito y la problemática del proyecto, en la segunda se va planteando al tiempo el concepto del desarrollo describiendo el *qué* se va a hacer, *cómo* se va a desarrollar y *cuáles* componentes se necesitan. Luego se pasa a la etapa de Especificación de Requisitos donde se estructura el sistema definiendo los actores del mismo y las funcionalidades de cada componente basándose en su descripción y en el objetivo para el que fue creado. Seguido de esto viene la etapa de Diseño Global y Planificación en la que se realiza el diseño de los elementos globales de los sistemas y se planifica cada uno de los incrementos a desarrollar.

Figura 21. Metodología de Desarrollo aplicada al Proyecto



Posteriormente se inicia el desarrollo efectivo del sistema, para cada incremento establecido en la fase anterior. Este proceso es cíclico y consta de la implementación (realizada con base en los requisitos), la realización de pruebas y el proceso de realimentación según los resultados de validación.

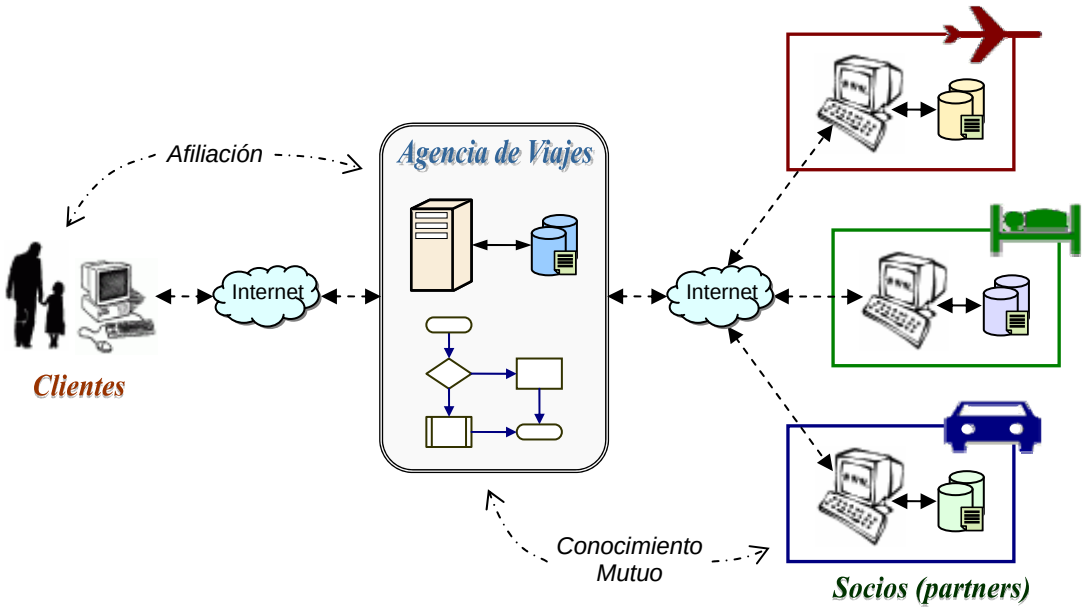
Para terminar, se realiza la etapa de Fusión del Sistema en la cual se lleva a cabo verifica la integración de los componentes, previamente terminados y se realizan los ajustes necesarios para producir la Entrega Final.

5.2 ESCENARIO DE DESARROLLO

Una Agencia de Viajes, concebida en el planteamiento del problema (*véase 2.1.1*) como uno de los escenarios reales más conocidos y representativos del comercio electrónico, dispone de clientes afiliados y les ofrece servicios de búsqueda y reserva en línea de planes de viaje que incluyen principalmente transporte aéreo,

alojamiento y transporte terrestre en el sitio donde se hospeda –es posible incluir muchos otros servicios como espectáculos, excursiones, etc. Para llevar a cabo sus funciones, la agencia de viajes mantiene relaciones comerciales *peer-to-peer* con otras entidades como aerolíneas, cadenas hoteleras y empresas de alquiler de autos, las cuales prestan un servicio de forma autónoma e independiente pero que la agencia utiliza coordinadamente para cumplir sus objetivos.

Figura 22. Escenario del Comercio Electrónico para el Proyecto



El funcionamiento interno de estas entidades no es relevante para la agencia y viceversa; cada uno sólo está interesado en la manera en que se comunican, es decir el protocolo de comunicación.

Bajo esta perspectiva, se han definido tres tipos de transacciones de la agencia que vinculan a sus clientes y socios comerciales: búsqueda de planes vacaciones según los criterios o preferencias del cliente, reserva de los mismos dada la información relevante contenida en los resultados de búsquedas anteriores, y la

confirmación y/o cancelación de las reservas previamente realizadas. El proceso de reserva es temporal y requiere que el cliente lo confirme cuando esté totalmente seguro de realizarlo o lo cancele en caso contrario, esto con el fin de que la agencia genere los cobros respectivos según un acuerdo preestablecido entre ellos en la afiliación.

En el mundo del comercio electrónico son muy importantes los aspectos de seguridad y confidencialidad de la información, por tanto las transacciones realizadas por la agencia con sus socios deben contar con métodos de protección y verificación de los datos, así como de autenticación para evitar suplantaciones identidad y denegaciones de los servicios.

5.3 CONCEPTO DEL DESARROLLO

El escenario escogido será implementado utilizando una infraestructura basada en Servicios Web XML, aprovechando así las ventajas de transmisión de datos en formato XML y las reglas definidas por los estándares relacionados. Además se utilizará el Lenguaje de Ejecución de Procesos de Negocio BPEL para implementar los procesos de negocio de la Agencia de Viajes con sus socios comerciales, relacionados con el flujo de actividades necesario para el correcto procesamiento de itinerarios (búsqueda / reserva) enviados por sus clientes.

Los elementos considerados en el diseño global del sistema se listan y explican en la tabla 2. De la misma forma, las características generales relacionadas con el funcionamiento del sistema aparecen en la tabla 3

Tabla 2. Consideraciones Generales de Diseño

Elemento	Descripción
Operaciones	El sistema de la Agencia de Viajes está diseñado para permitir a sus clientes la realización de cuatro operaciones vía servicios Web: ***Búsquedas con Posible Reserva Inmediata***, ***Reservas*** y ***Confirmaciones / Cancelaciones de Reservas*** de planes de viaje. Cada una de estas operaciones se realiza según los criterios especificados por el cliente. La reserva inmediata en las búsquedas se realiza con el mejor plan de viaje encontrado, seleccionado de acuerdo a criterios como su costo.
Planes de Viaje	Los Planes de Viaje de los clientes de la agencia incluyen tres servicios: ***Transporte Aéreo*** representados en tiquetes de vuelos, ***Alojamiento*** o habitaciones de hotel y ***Transporte Terrestre Local*** que son vehículos rentados por un tiempo dado. Es claro que puede ampliarse según las necesidades.
Partners (*Socios*)	Existen tres clases de socios comerciales o partners de la agencia de acuerdo a los tipos de servicios en los planes de viaje así: ***Aerolíneas*** que incluyen las empresas de transporte aéreo que disponen de vuelos entre diferentes lugares. ***Cadenas Hoteleras*** que son grupos empresariales o pequeños negocios de alojamiento y hospedaje en las ciudades ***Alquileres de Vehículos*** los cuales corresponden a empresas que permiten el préstamo o renta temporal de diferentes clases de transporte terrestre personal como autos.

Figura 23. Relación entre Servicios y Socios de la Agencia de Viajes

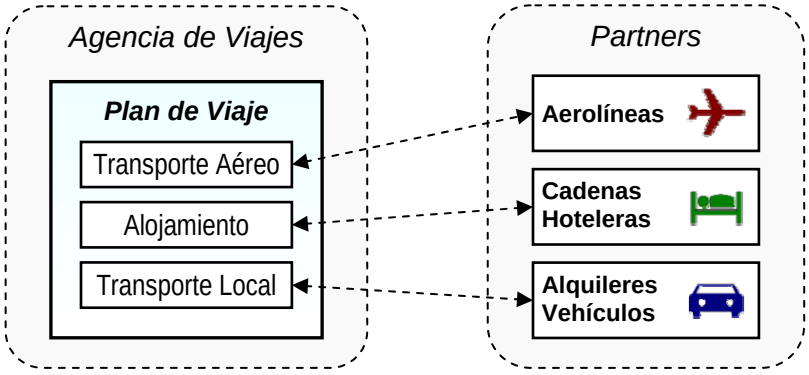


Tabla 3. Consideraciones Generales de Funcionamiento

Elemento	Descripción
Interfaces entre agencia, socios y cliente	Existen ***interfaces WSDL*** (servicios Web) que definen la sintaxis de comunicación entre la agencia, el sistema del cliente y cada tipo de socio. Se definen en la agencia y los socios las implementan para comunicarse con la agencia.
Procesamiento en la Agencia	El sistema de la agencia de viajes recopila los requerimientos del cliente y con base en ellos realiza el procesamiento *automático* de la solicitud, usando los servicios de sus socios y recogiendo los resultados individuales con el fin de generar un resultado general de la actividad. Los procesos de negocio desarrollados, implementan mecanismos de ***protección a fallos*** que los hace robustos y coordinados.
Conocimiento de los socios en la Agencia	Al interior de la agencia de viajes se tiene pleno conocimiento de quiénes son sus socios comerciales. Para ello dispone de un ***repositorio de socios*** con la información WSDL relacionada con la ubicación en la red de los servicios. Cada nueva entidad comercial que se asocie a la agencia podrá fácilmente agregarse a éste repositorio y será usada de inmediato en sus procesos.

Algunos de los elementos del funcionamiento mencionados pueden ser considerados como restricciones o limitaciones de los sistemas. Sin embargo, son aspectos necesarios para la seguridad de los mismos que es muy importante en el mundo del comercio electrónico. En la tabla 4 se enuncian los elementos definidos referentes a la seguridad de los sistemas.

Tabla 4. Consideraciones Generales de Seguridad

Elemento	Descripción
Seguridad en Servicios Web	La comunicación mediante servicios Web a través de Internet implementa los mecanismos de seguridad definidos en el estándar WS-Security. La información se codifica usando criptografía asimétrica –Llaves Públicas y Privadas– y se autentica mediante Firmas Digitales.
Seguridad Agencia / Socios	Tanto la agencia como sus socios poseen llaves públicas y privadas emitidas por una autoridad certificadora. Cada uno tiene conocimiento de su existencia desde el momento de asociarse. En todas las transacciones la información a enviar se codifica con la llave pública de la otra entidad y se firma con la llave privada propia para que la otra verifique su autenticidad.
Seguridad Cliente / Agencia	El software del cliente codifica la información a enviar a la agencia usando la llave pública de la agencia y para autenticarse ante ella, el cliente incluye los datos de identificación –id y contraseña–. Por otro lado, la agencia firma digitalmente la información enviada al cliente y el software verifica su autenticidad con la llave pública. No se codifica la información ya que no es de carácter sensible o secreta.

Figura 24. Seguridad en las Transacciones de los Sistemas

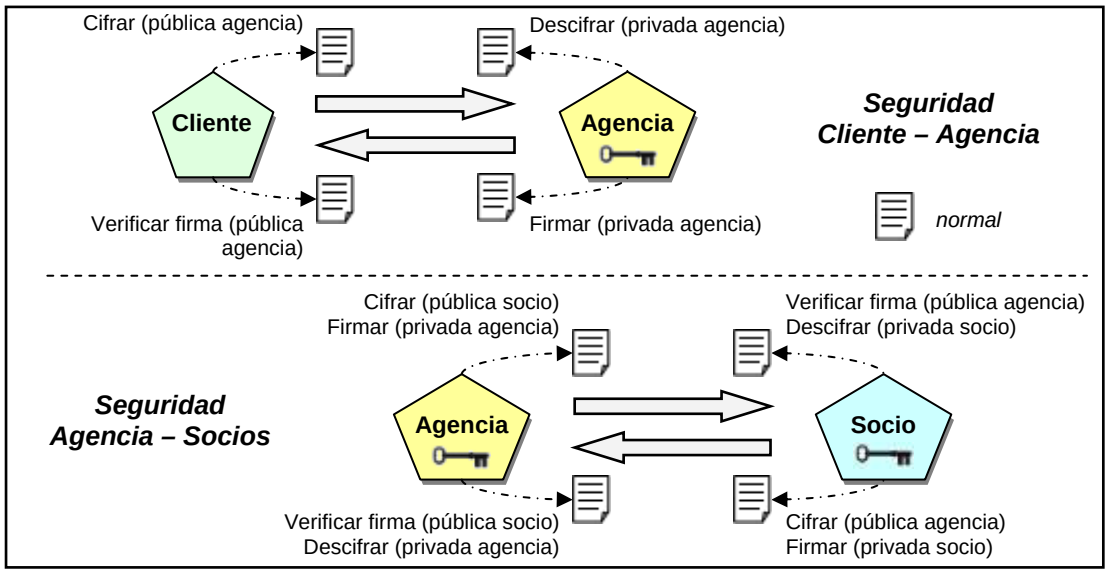


Tabla 5. Consideraciones Generales de Software

Elemento	Descripción
Agencia	El software del sistema de la agencia de viajes está constituido por los procesos de negocio en lenguaje BPEL, las interfaces WSDL para comunicación externa e interna y la implementación en Java de diferentes tareas especializadas necesarias en los procesos y que son accedidas como operaciones de un servicio Web local.
Socios	El sistema de los socios posee simplemente la implementación en Java de los algoritmos de procesamiento de solicitudes, accedidas mediante las interfaces WSDL especificadas.
Cliente	Con el fin de permitir el acceso al cliente a las funcionalidades de la agencia, el sistema del cliente dispone de una sencilla aplicación de escritorio desarrollada en Java. Se encarga de gestionar el acceso a los servicios Web que publica la agencia con todas las consideraciones de seguridad definidas.

Las implementaciones mencionadas pueden realizarse en diversos lenguajes de programación, bajo cualquier plataforma de desarrollo y sistema operativo, con diferentes clases de hardware, etc. ya que esa es la principal ventaja del uso de servicios Web como interfaz de comunicación.

Como aclaración final, los sistemas desarrollados son conforme con el alcance del proyecto (*véase 1.3*) y por tanto no pretende implantarse de manera inmediata en el mundo comercial. Aunque abarca muchos aspectos técnicos del entorno y características avanzadas de las tecnologías usadas, su escenario es muy limitado y se requiere más dinero e investigación para llevarlo a la realidad.

5.4 ANÁLISIS Y ESPECIFICACIONES GENERALES

Con base en el escenario planteado y las características generales de los sistemas definidas anteriormente, es posible analizar y definir algunos aspectos

generales muy importantes como son los actores y sus correspondientes tareas. También se plantean la estructura de la información intercambiada en los procesos, y finalmente se presenta la planificación del proceso de desarrollo, el cual se dividirá en etapas de acuerdo a lo planteado por la metodología adoptada.

El análisis general se hace previo a la implementación de cada componente, y para representarlos se usan diagramas UML.

5.4.1 Actores en el Sistema

En el escenario planteado se distinguen cuatro tipos de actores relacionados entre sí: la *agencia de viajes*, los *clientes* que utilizan sus servicios, el *sistema del cliente* que sirve de intermediario entre la agencia y el cliente, y las *entidades socias* o *partners* que prestan los servicios.

- *Agencia de Viajes*
  La agencia de viajes es un actor software representado por el sistema principal, cuyo fin es procesar las solicitudes que vienen del cliente ejecutando procesos de negocio y algoritmos internos, y coordinando los servicios de los socios.

- *Cliente de la Agencia*
  El actor cliente es la persona o sujeto humano que utiliza los servicios de la agencia para organizar sus planes de viaje. Interactúa directa y únicamente con el sistema del cliente. De él depende el ingreso de la información y las solicitudes de las operaciones.

- *Sistema del Cliente*
  Este sistema se considera un actor *Software* pues es la representación del cliente en la agencia. Su función es acceder a los servicios vía Web para realizar las peticiones de cada operación permitida como búsquedas y

reservas cuando el cliente lo indique. De él depende la recolección de los criterios de búsqueda y reserva, la invocación de las tareas del cliente en la agencia y la visualización de resultados.

- *Socio: aerolínea, cadenas hotelera y alquiler de vehículos*
  El actor socio representa a cada entidad relacionada con la agencia en la prestación de los servicios. Puede ser Aerolíneas, Cadenas Hoteleras, y Alquileres de Vehículos. Se considera un actor *Software* constituido por un sistema independiente. Su la función principal es procesar las solicitudes referentes a búsquedas, reservas y confirmación/cancelación de reservas de cada servicio del plan de viaje que hace la agencia: vuelos para la aerolínea, habitaciones para la cadena hotelera, y autos para el alquiler de vehículos.

5.4.2 Análisis de los Casos de Uso en los Sistemas

Un análisis general desde el punto de vista de cada actor permite establecer las acciones y tareas básicas en las cuales toman parte activa según las funcionalidades que los componentes desplieguen.

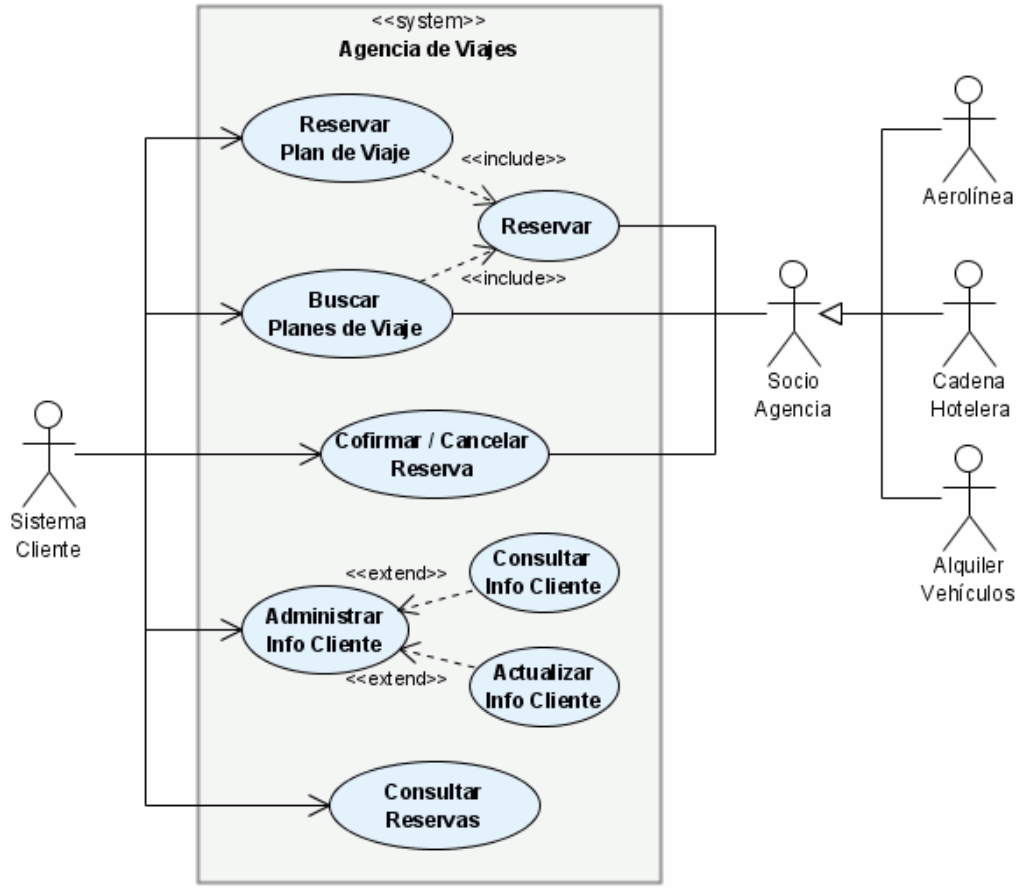
5.4.2.1 Agencia de Viajes

El sistema de la Agencia de Viajes tiene las siguientes funcionalidades:

- Búsqueda de planes de viaje con posible reserva automática
- Reserva de Planes de Viaje
- Confirmación / Cancelación de Reservas de Planes de Viaje
- Administración de la Información del Cliente
- Consulta de Reservas Previas

Con base en ellas, resulta el diagrama de casos de uso general mostrado en la figura 25.La especificación de cada caso de uso se describe después.

Figura 25. Diagrama de Casos de Uso de la Agencia de Viajes



- *Nombre*: **Buscar Planes de Viaje**

  Actores: Sistema del Cliente y Socios de la Agencia

  Descripción: Realiza la búsqueda de planes de viaje según las especificaciones del cliente, y la posible reserva automática del mejor según el criterio dado.

  Flujo de Eventos: Inicia cuando el sistema del cliente envía la petición de búsqueda con los datos adjuntos. El sistema contacta cada uno de los socios para realizar las búsquedas individuales enviando los respectivos criterios, mientras que recibe y acumula los resultados. Si se especifica reserva automática, se selecciona el mejor plan y realiza el proceso de reserva, obteniendo los resultados. Termina devolviendo dichas respuestas al sistema del cliente, bien sea de búsqueda o reserva.

- *Nombre*: **Reservar Planes de Viaje**

  Actores: Sistema del Cliente

  Descripción: Realiza reservas de planes de viajes que el cliente especifique en los datos adjuntos basándose en búsquedas previas

  Flujo de Eventos: Inicia con la llegada de la petición de reserva proveniente del sistema del cliente con la información adjunta. El sistema utiliza realiza la reserva con la información de entrada y recoge los resultados. Termina con el envío al sistema del cliente de los resultados.

- *Nombre*: **Reservar**

  Actores: Socios de la Agencia

  Descripción: Realiza el proceso de reserva de cada servicio del plan de viaje en los socios según las especificaciones dadas.

  Flujo de Eventos: Inicia cuando el sistema de la agencia procesa una reserva o una búsqueda con reserva automática; aquí se contacta cada socio y se realiza la reserva respectiva obteniendo los resultados que al final se agrupan y devuelven a sus casos de uso invocadores con lo cual termina su flujo.

- *Nombre*: **Confirmar / Cancelar Reserva**

  Actores: Sistema del Cliente y Socios de la Agencia

  Descripción: Realiza la confirmación o cancelación definitiva de una reserva realizada previamente.

  Flujo de Eventos: Inicia cuando el sistema del cliente lo requiere enviando la solicitud y la información requerida, luego el sistema procesa cada cancelación de servicio realizando la petición al socio y esperando el resultado. Al final recopila los resultados y los envía de regreso al sistema del cliente.

- *Nombre*: **Administrar Información del Cliente**

  Actores: Sistema del Cliente

Descripción: Realiza operaciones de gestión con la información personal del cliente almacenada localmente en el sistema de la agencia de viajes.
Extensiones: **Consultar Información del Cliente**: realiza sólo la obtención de la información del cliente. **Modificar Información del Cliente**: realiza actualizaciones de los datos almacenados del cliente.
Flujo de Eventos: Inicia con la llegada de la petición de gestión enviada desde el sistema del cliente. Luego el sistema utiliza la información adjunta para realizar la operación de administración, y termina con la confirmación del proceso enviada de vuelta al sistema del cliente.

- *Nombre*: **Consultar Reservas**
  Actores: Sistema del Cliente
  Descripción: Realiza una consulta de las reservas vigentes de un cliente dado, especificado en la información de entrada.
  Flujo de Eventos: El caso de uso inicia con la solicitud que llega desde el sistema del cliente, entonces el sistema consulta en sus registros las reservas vigentes del cliente especificado y realiza un listado. Termina devolviendo el listado respectivo al sistema del cliente.

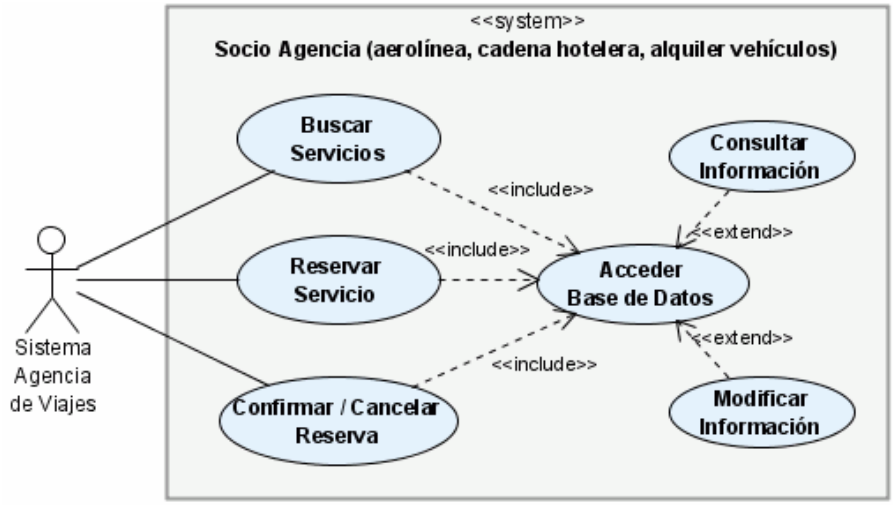
5.4.2.2 Socios: Aerolínea, Cadena Hotelera, Alquiler Vehículos

Los sistemas de los socios presentan funcionalidades similares desde una perspectiva global del procesamiento interno, aunque en detalle puede ser tan complejo como sea necesario. Esas funcionalidades son:

- Búsqueda de servicios
- Reserva del servicio
- Confirmación / Cancelación de reservas de servicio

En la figura 26 se muestra el diagrama de casos de uso general resultante de las funcionalidades indicadas. La especificación de cada caso de uso es:

Figura 26. Diagrama de Casos de Uso de los Socios de la Agencia



- *Nombre*: **Buscar Servicios**

  Actores: Sistema de la Agencia de Viajes

  Descripción: Realiza la búsqueda del servicio que presta según los criterios especificados como entrada.

  Flujo de Eventos: Inicia cuando el sistema de la agencia de viajes contacta al socio y le envía la solicitud de búsqueda del servicio determinado con ciertos criterios adjuntos, los cuales son procesados por el sistema mediante sus procesos de acceso a las bases de datos. Los resultados obtenidos son preparados y finalmente enviados de vuelta a la agencia como respuesta a la operación, así termina el caso de uso.

- *Nombre*: **Reservar Servicio**

  Actores: Sistema de la Agencia de Viajes

  Descripción: Realiza la reserva de un servicio según los datos de entrada.

  Flujo de Eventos: Inicia cuando el sistema de la agencia de viajes solicita la operación enviando la información requerida. El sistema del socio la procesa consultando y modificando la información en sus bases de datos para hacer efectiva la reserva. Termina enviando de vuelta a la agencia la confirmación.

- *Nombre*: **Confirmar / Cancelar Reserva**
  Actores: Sistema de la Agencia de Viajes
  Descripción: Realiza la confirmación o cancelación definitiva de la reserva del servicio previamente realizada por el cliente.
  Flujo de Eventos: Inicia con la solicitud realizada por la agencia de viajes con la información requerida, luego el sistema procesa los datos y determina la acción a realizar sobre la reserva consultada en sus bases de datos. Luego de realizar la acción, termina devolviendo el resultado al sistema de la agencia de viajes.

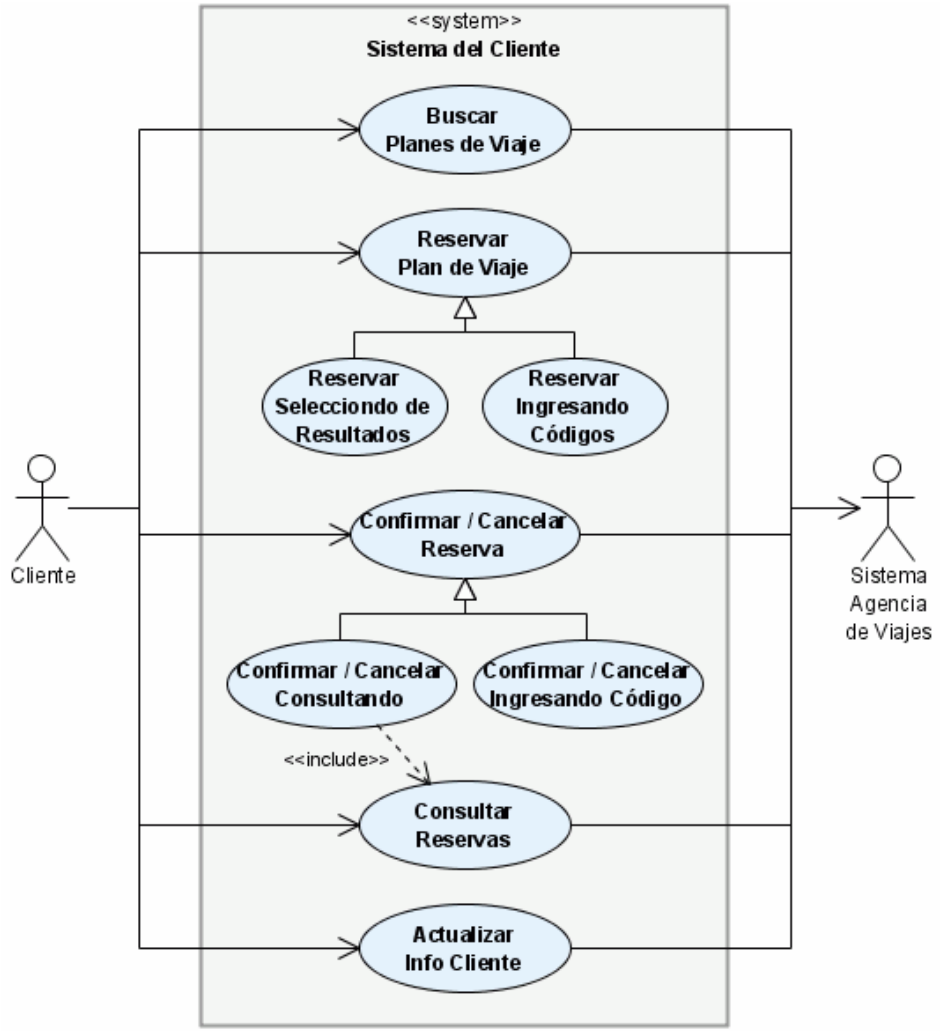
- *Nombre*: **Acceder a Base de Datos**
  Descripción: Habilita el acceso y operaciones sobre las bases de datos.
  Extensiones: **Consultar Información**: permite realizar consultas a las bases de datos. **Modificar Información**: permite realizar modificaciones de los datos almacenados en las bases de datos.

5.4.2.3 Software del Cliente

El sistema del cliente (software) está enfocado en servir de puente de comunicación amigable entre el cliente y los servicios Web de la agencia por lo tanto las tareas son las mismas del sistema de la agencia de viajes.
En la figura 27 se muestra el diagrama de casos de uso general resultante de las funcionalidades en el sistema del cliente. Las especificaciones son:

- *Nombre*: **Buscar Planes de Viaje**
  Actores: Cliente y Sistema de la Agencia de Viajes
  Descripción: Captura los criterios de búsqueda del cliente y realiza la interfaz cliente-agencia para permitir la búsqueda de planes de viaje.
  Flujo de Eventos: Inicia cuando el cliente solicita una nueva búsqueda. El sistema responde solicitándole los datos, que una vez diligenciados por el cliente son usados para realizar la petición al sistema de la agencia. Termina recibiendo los resultados presentándoselos al cliente.

Figura 27. Diagrama de Casos de Uso del Sistema del Cliente



- *Nombre*: **Reservar Plan de Viaje**

  Actores: Cliente, Sistema de la Agencia de Viajes

  Descripción: Permite el ingreso de la información de reserva y la realización del proceso de reserva mediante la interfaz cliente-agencia.

  Especializaciones: **Reservar Seleccionando de Resultados**: el cliente inicia la reserva desde resultados de búsquedas. **Reservar Ingresando Códigos**: el cliente inicia la reserva desde cero.

  Flujo de Eventos: Inicia con la solicitud que hace el cliente reservar introduciendo la información de los servicios o seleccionándolos de resultados

de búsquedas. Una vez el sistema conoce estos datos, crea la petición de reserva y la envía al sistema de la agencia. Luego de esperar y recibir el resultado, termina informándole al cliente la respuesta del proceso.

- *Nombre*: **Confirmar / Cancelar Reserva**
  Actores: Cliente, Sistema de la Agencia de Viajes
  Descripción: Permite al cliente realizar la confirmación o cancelación definitiva de una reserva previa a través de la interfaz cliente-agencia.
  Especializaciones: **Confirmar / Cancelar Consultando**: la reserva se selecciona entre las reservas vigentes. **Confirmar / Cancelar Ingresando Código**: la reserva la especifica manualmente el cliente.
  Flujo de Eventos: Inicia cuando el cliente solicita cancelar/confirmar una reserva ingresando su información o seleccionándola de las reservas vigentes, entonces el sistema elabora y envía la petición al sistema de la agencia y espera su respuesta. Cuando la recibe, le responde al cliente y termina.

- *Nombre*: **Consultar Reservas**
  Actores: Cliente, Sistema de la Agencia de Viajes
  Descripción: Permite al cliente consultar a la agencia sus reservas vigentes, mediante la interfaz cliente-agencia.
  Flujo de Eventos: Se inicia con la petición del cliente, a la cual el sistema local actúa invocando el servicio a la agencia y esperando la respuesta respectiva. Termina mostrando al cliente los resultados recibidos.

- *Nombre*: **Administrar Información del Cliente**
  Actores: Cliente, Sistema de la Agencia de Viajes
  Descripción: Permite al cliente la administración de su información personal registrada en la agencia mediante la interfaz cliente-agencia.
  Flujo de Eventos: Inicia con la petición de actualización del cliente. El sistema envía una petición de los datos actuales del cliente y al recibirlos se los

muestra al cliente para que realice modificaciones. Una vez hechas, el cliente se lo indica al sistema el cual responde tomando los datos modificados y solicitando su actualización a la agencia. Luego de esperar y recibir la confirmación del proceso, termina informándole la respuesta al cliente.

5.4.3 Análisis de la lógica de los procesos

El análisis de los casos de uso y los actores del escenario planteado permiten identificar tres procesos principales que involucran la totalidad de los actores y hacen uso de las funcionalidades especificadas en los casos de uso. Este análisis se concentra en la lógica de dichos procesos, representado mediante el flujo general de actividades que posee. Estos procesos son:

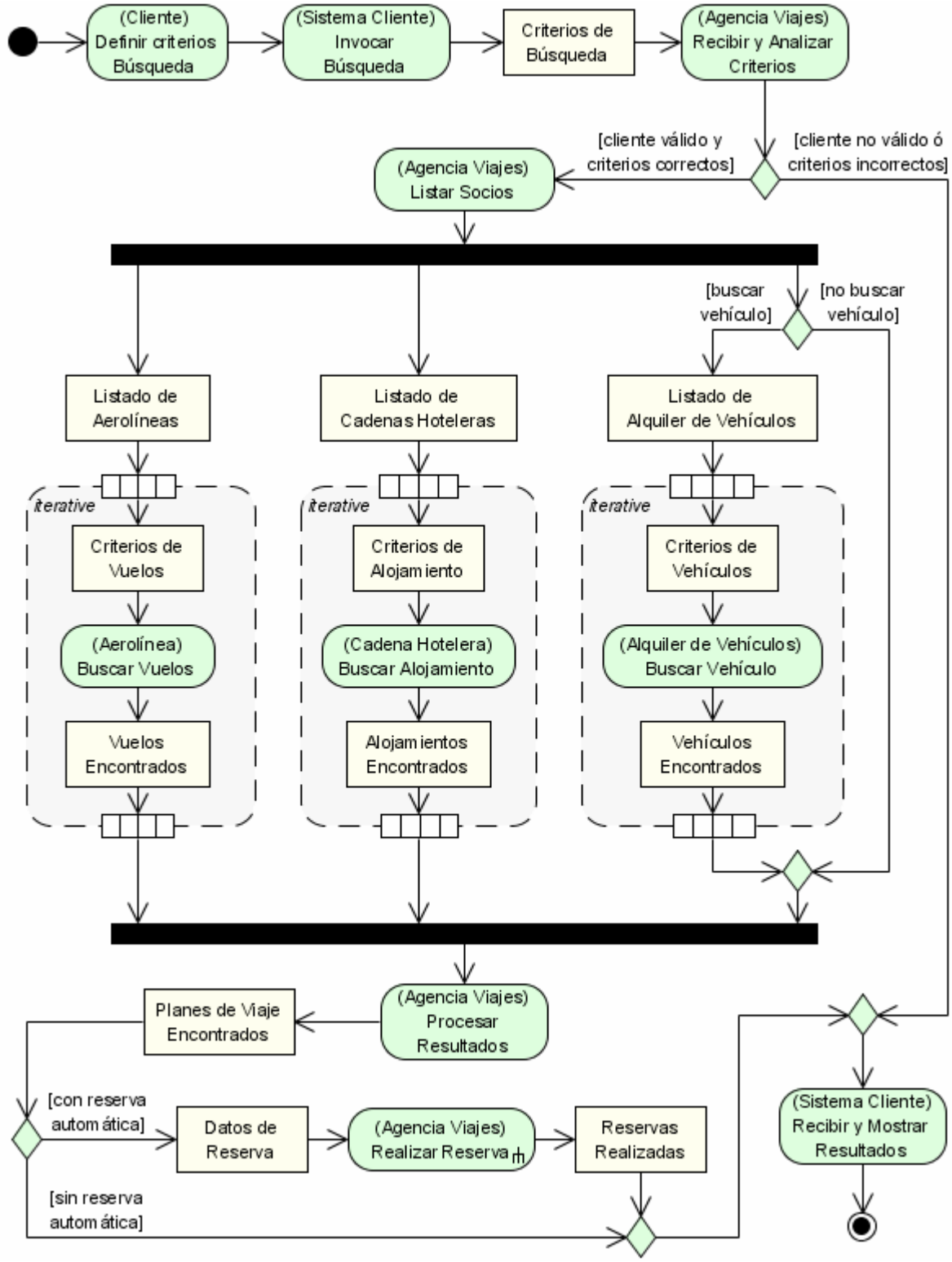
- Búsqueda de Planes de viaje
- Reserva de Planes de viaje
- Confirmación / Cancelación de Reservas de Planes de Viaje.

5.4.3.1 Proceso Búsqueda de Planes de Viaje

La búsqueda de planes de viaje es iniciada por el cliente de la agencia mediante una invocación al sistema de la agencia, usando la interfaz software; luego el sistema de la agencia se encarga de procesar los criterios y realizar las búsquedas por en cada socio que tenga registrado según su tipo (aerolínea, cadena hotelera o alquiler de vehículos).

Una vez obtenga respuesta de cada socio consultado, debe organizar y procesar los resultados teniendo en cuenta las opciones dadas por el cliente; si se ha especificado, debe invocar una reserva inmediata del mejor plan encontrado; y finalmente tiene que devolver al usuario los resultados mediante la interfaz, que se encarga de visualizarlos. En la figura 28 se aprecia este flujo de actividades mediante un diagrama UML.

Figura 28. Diagrama de Actividades, Búsqueda de Planes de Viaje



Con el fin de incluir la funcionalidad de realizar de forma inmediata la reserva del mejor plan de viaje encontrado en la búsqueda (*véase 5.3*), el flujo de actividades incluye una llamada a la actividad *Realizar Reserva*, el cual se especifica más adelante.

5.4.3.2 Proceso Reserva de Planes de Viaje

El proceso de reserva de planes de viaje se realiza cuando ya el cliente posee la información de los servicios a solicitar y mediante la interfaz software de la agencia ingresa los códigos o selecciona directamente de los resultados de la búsqueda. Con esta información el sistema de la agencia contacta los socios respectivos y realiza las reservas, y finalmente muestra los resultados al cliente.

En la figura 29 se aprecia el flujo de actividades general del proceso de reserva, destacándose la referencia a un subdiagrama con la actividad *Realizar Reserva* usado tanto en la búsqueda como en la reserva, y que se aprecia en la figura 30.

Figura 29. Diagrama de Actividades, Reserva de Planes de Viaje

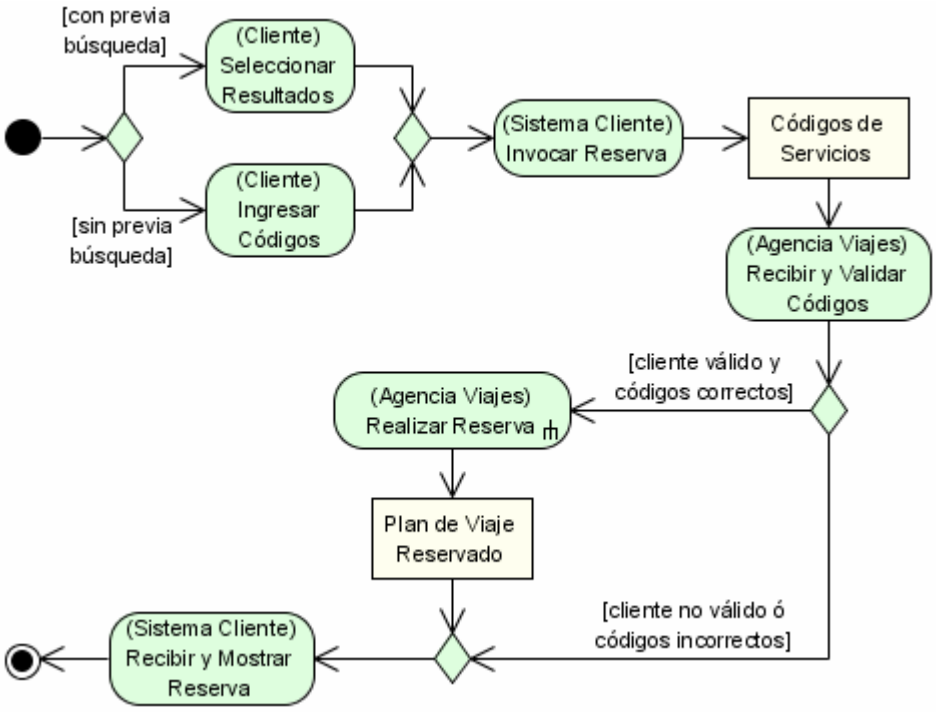
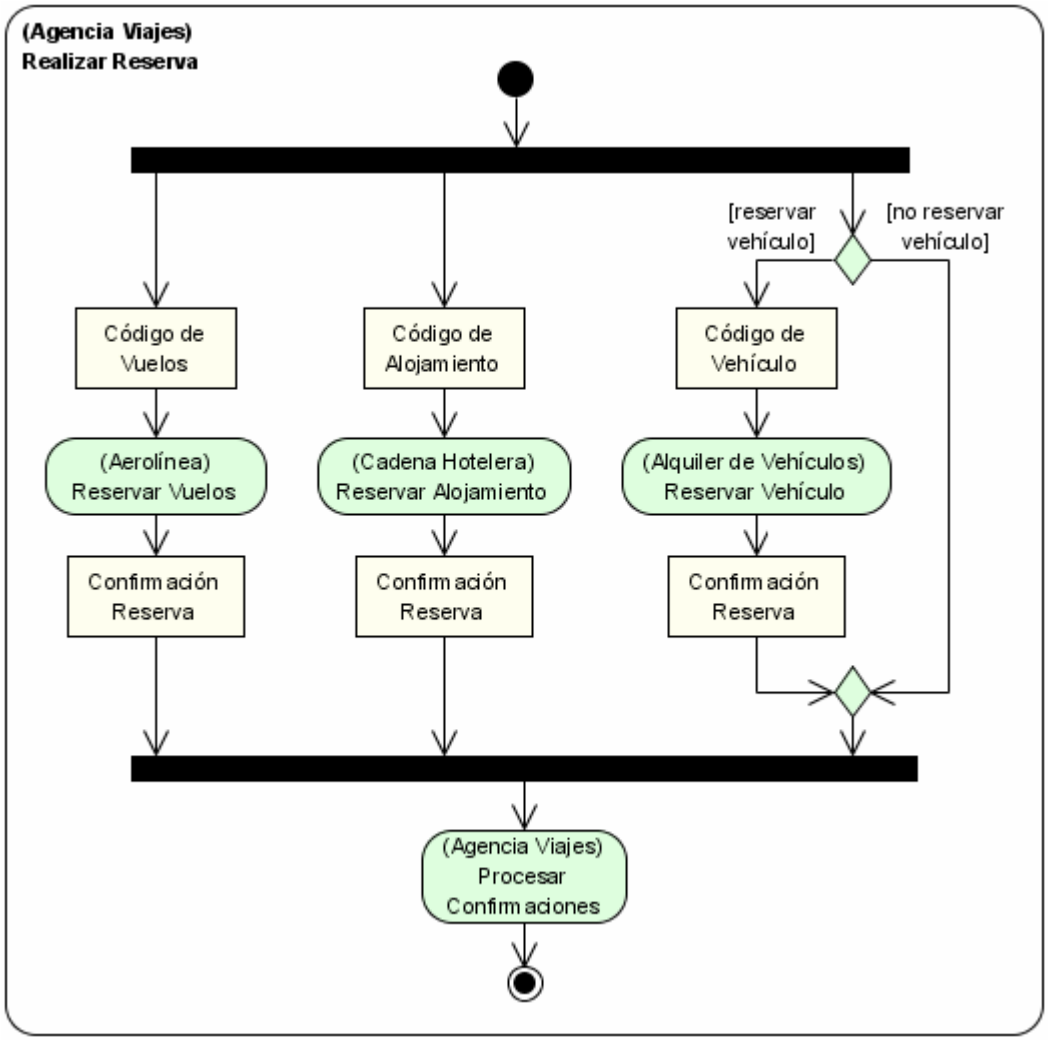


Figura 30. Diagrama de la Actividad Realizar Reserva



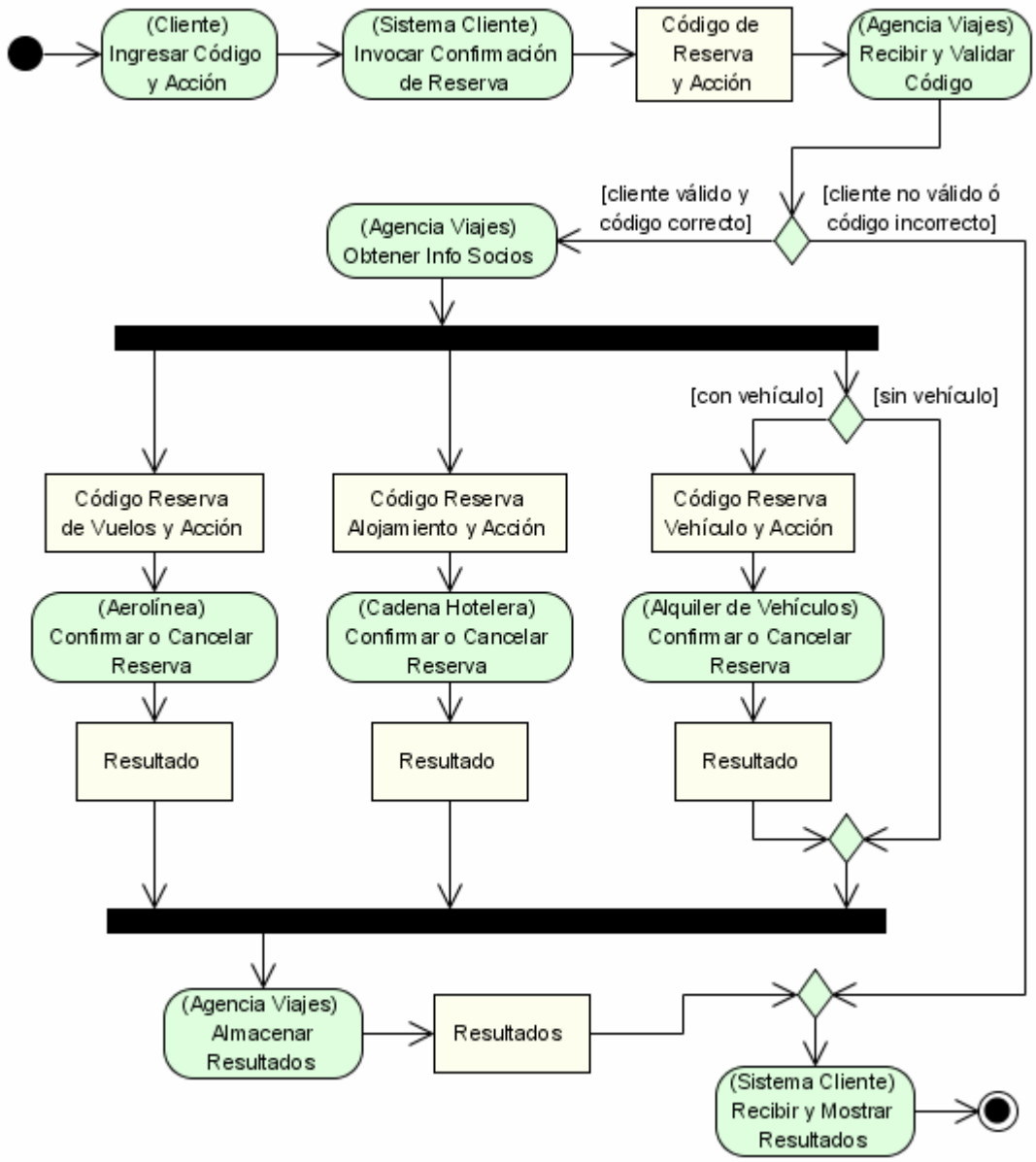
Esta actividad común consta en primer lugar de la realización del contacto con los socios respectivos según el tipo, para luego realizar reserva en cada uno paralelamente; de cada una obtendrá el respectivo resultado el cual procesará posteriormente de acuerdo al proceso al que pertenezca: búsqueda o reserva.

5.4.3.3 Proceso Confirmación o Cancelación de Reservas de Planes de Viaje

Una vez el cliente de la agencia ha realizado reservas en la agencia de viajes, puede confirmarlas o cancelarlas de acuerdo a su criterio. Por tanto la interfaz de cliente ofrece la opción de capturar la información de la reserva para realizar

dichos procesos. Los datos los recibe el sistema de la agencia y obtiene internamente la información de la reserva realizada previamente para determinar a qué socio corresponde y poder contactarlo y ejecutar la petición de confirmación o cancelación. La figura 31 presenta el diagrama UML de actividades del proceso descrito.

Figura 31. Diagrama de Actividades, Confirmación Reservas de Planes de Viaje



Finalmente cada socio envía una respuesta con la confirmación, o un mensaje de error; dichas respuestas son recibidas por el sistema de la agencia; posteriormente se procesan y retornan a la interfaz de usuario para ser expuestas al cliente como resultado del proceso.

5.4.4 Especificación de la Información Básica a Intercambiar

Los casos de uso mostrados en el análisis general (*véase 5.4.2*) se exponen como operaciones o transacciones entre las entidades y por ende necesitan intercambiar información tanto de entrada en las solicitudes, como de salida en las respuestas. Dicha información debe ser suficiente y estar correctamente estructurada para lograr una ejecución eficiente de los procesos.

A continuación se describen los datos transmitidos para cada interacción entre entidades del escenario planteado para el presente trabajo de grado.

5.4.4.1 Información Agencia – Aerolíneas

La aerolínea, así como los otros socios, expone operaciones de búsqueda, reserva y confirmación/cancelación de reservas de acuerdo a su servicio. Estas operaciones son usadas por el sistema de la agencia mediante invocación o llamado, transmitiendo la información que requiera cada operación.

La descripción de los datos implicados en cada transacción que expone la Aerolínea se presenta en las tablas a continuación, cada dato se presenta con el nombre del campo y la descripción y funcionamiento.

Tabla 6. Datos en Solicitud de Búsqueda de Vuelos

Búsqueda de Vuelos – Solicitud	
Campo	***Descripción***
Lugar Origen	Sitio de origen del vuelo de ida, o de destino del vuelo de regreso (si se especifica). Consta de *País* y *Ciudad*.
Lugar Destino	Sitio de destino del vuelo de ida, o de origen del vuelo de regreso (si se especifica). Consta de *País* y *Ciudad*.
Fecha Ida	Fecha en que se busca el vuelo de ida.
Hora Ida	Hora aproximada para el vuelo de ida, opcional.
Fecha Regreso	Fecha en que se busca el vuelo de regreso; si aplica.
Hora Regreso	Hora aproximada del vuelo de regreso, opcional, si aplica.
Clase	Tipo de clase de los tiquetes, opcional. Cuatro definidas: - Única - Turista - Ejecutiva - Primera Clase
Cantidad Personas	Número de personas que viajan.
Criterio de Orden	Para la ordenación de los resultados. Definidos tres: - Precio - Hora de Ida - Hora de Regreso (cercanía)

Tabla 7. Datos en Respuesta de Búsqueda de Vuelos

Búsqueda de Vuelos – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la búsqueda realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Vuelos Ida	Listado con los vuelos de Ida encontrados. Cada uno tiene: - Código del vuelo, usado para reservar - Nombre de los aeropuertos de origen y de destino - Fecha y Hora de salida - Duración completa estimada. - Descripción del avión: nombre y modelo - Clase de los tiquetes - Costo total
Vuelos Regreso	Listado de los vuelos de Regreso encontrados. Cada uno posee información similar a los vuelos de Ida.

Tabla 8. Datos en Solicitud de Reserva de Tiquetes

Reserva de Tiquetes – Solicitud	
Campo	***Descripción***
Fecha Ida	Fecha para el vuelo de ida especificado
Código Vuelo Ida	Código del vuelo de ida.
Fecha Regreso	Fecha para el vuelo de regreso. Opcional, si aplica.
Código Vuelo Regreso	Código del vuelo de regreso, si aplica.
Cantidad Personas	Número de personas a incluir en la reserva.
Información Cliente	Datos personales del cliente que reserva: identificación, apellidos, nombre, e-mail, dirección y teléfono.

Tabla 9. Datos en Respuesta de Reserva de Tiquetes

Reserva de Tiquetes – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la reserva realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Código Reserva	Código identificador de la reserva.
Lugar Origen	Sitio de origen del vuelo de ida, o de destino del vuelo de regreso (si se especifica). Consta de *País* y *Ciudad*.
Lugar Destino	Sitio de destino del vuelo de ida, o de origen del vuelo de regreso (si se especifica). Consta de *País* y *Ciudad*.
Vuelo Ida	Información del vuelo de Ida reservado: - Nombre de los aeropuertos de origen y de destino - Fecha y Hora de salida exacta de salida. - Duración total estimada - Descripción del avión: nombre y modelo - Clase de los tiquetes - Costo total - Listado de códigos de las sillas reservadas en el avión
Vuelos Regreso	Información del vuelo de Regreso reservado si se especificó en la solicitud. Contiene datos similares a los del vuelo de Ida.
Costo Total	Costo total de la reserva

Tabla 10. Datos en Solicitud de Confirmación de Reserva de Tiquetes

Confirmación de Reserva – Solicitud	
Campo	***Descripción***
Código Reserva	Código con el que se identifica la reserva.
Acción	Indica si la reserva se Confirma o Cancela definitivamente.
Identificación Cliente	Identificación del cliente propietario de la reserva.

Tabla 11. Datos en Respuesta de Confirmación de Reserva de Tiquetes

Confirmación de Reserva – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la confirmación de reserva realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.

5.4.4.2 Información Agencia – Cadenas Hoteleras

Las cadenas hoteleras también exponen las operaciones mencionadas, y los datos intercambiados de describen en las tablas a continuación.

Tabla 12. Datos en Solicitud de Búsqueda de Alojamientos

Búsqueda de Alojamientos – Solicitud	
Campo	***Descripción***
Lugar	Sitio donde buscar los hoteles: *País* y *Ciudad*.
Fecha Llegada	Fecha en que se llega al hotel (check-in).
Fecha Salida	Fecha de salida del hotel (check-out).
Cantidad Personas	Número de personas a incluir en el alojamiento.
Tipo Habitaciones	Tipo de habitación deseada, Opcional. Definidas 5: - Estándar Sencilla - Estándar Doble - Suite Preferente - Suite Superior - Suite Ejecutiva
Criterio de Orden	Para la ordenación de los resultados. Definidos dos: - Precio - Numero habitaciones requeridas

Tabla 13. Datos en Respuesta de Búsqueda de Alojamientos

Búsqueda de Alojamientos – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la búsqueda realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Alojamientos	Listado de los alojamientos encontrados. Cada uno con: - Código identificador, usado para reservar - Tipo de habitación seleccionada - Descripción de las habitaciones - Cantidad de habitaciones necesarias para las personas - Número de personas por habitación - Nombre del hotel en el que se ubica - Costo total de las habitaciones

Tabla 14. Datos en Solicitud de Reserva de Alojamiento

Reserva de Alojamiento – Solicitud	
Campo	***Descripción***
Código Alojamiento	Código con el que se identifica alojamiento
Fecha Llegada	Fecha en que se llega al hotel (check-in).
Hora Llegada	Hora aproximada de llegada al hotel, es opcional.
Fecha Salida	Fecha de salida del hotel (check-out).
Cantidad Personas	Número de personas a incluir en el alojamiento.
Información Cliente	Datos personales del cliente poseedor de la reserva. Incluye la identificación, los apellidos y nombre, el e-mail, la dirección y el teléfono.

Tabla 15. Datos en Solicitud de Confirmación de Reserva de Alojamiento

Confirmación de Reserva de Alojamiento – Solicitud	
Campo	***Descripción***
Código Reserva	Código con el que se identifica la reserva.
Acción	Indica si la reserva se Confirma o Cancela definitivamente.
Identificación Cliente	Identificación del cliente propietario de la reserva.

Tabla 16. Datos en Respuesta de Reserva de Alojamiento

Reserva de Alojamiento – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la reserva realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Código Reserva	Código identificador de la reserva.
Lugar	País y Ciudad de ubicación del hotel.
Alojamiento	Información del alojamiento reservado: - Tipo de habitación - Descripción - Cantidad de habitaciones reservadas - Nombre del hotel en el que se ubica - Costo total de las habitaciones - Listado de habitaciones reservadas, cada una tiene: * Código o número * Cantidad personas asignadas

Tabla 17. Datos en Respuesta de Confirmación de Reserva de Alojamiento

Confirmación de Reserva de Alojamiento – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la confirmación de reserva realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.

5.4.4.3 Información Agencia – Alquileres de Vehículos

Las entidades de renta de vehículos exponen las tres operaciones: búsqueda, reserva y confirmación/cancelación de reservas de forma similar a los otros socios de la agencia.

Los datos usados en las interacciones de describen en las tablas siguientes.

Tabla 18. Datos en Solicitud de Búsqueda de Vehículos

Búsqueda de Vehículos – Solicitud	
Campo	***Descripción***
Lugar	Sitio donde buscar los autos: *País* y *Ciudad*.
Fecha Entrega	Fecha de inicio del alquiler: entrega al cliente.
Fecha Devolución	Fecha de terminación del alquiler: devuelto por el cliente.
Especificación del Vehículo	Opcional pero relevante del vehículo a buscar. Incluye: - ¿Debe tener Aire acondicionado?, es opcional. - ¿Debe ser Automático?, es opcional. - ¿Capacidad Mínima que debe tener?, es opcional. - Tipo o categoría, obligatoria si se usa el campo. Definidas 3: * Económico * Full Equipo * Compacto
Criterio de Orden	Para la ordenación de los resultados. Definidas tres: - Precio - Tipo de Vehículo (*si no fue usado en la búsqueda)* - Capacidad de Personas del vehículo.

Tabla 19. Datos en Respuesta de Búsqueda de Vehículos

Búsqueda de Vehículos – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la búsqueda realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Vehículos	Listado de los vehículos encontrados. Cada uno posee: - Código identificador, usado para reservar - ¿Posee Aire? - ¿Es Automático? - Capacidad personas - Tipo - Descripción textual - Línea o marca - Sitio del alquiler - Costo total incluyendo adicionales

Tabla 20. Datos en Solicitud de Reserva de Vehículo

Reserva de Vehículo – Solicitud	
Campo	***Descripción***
Código Vehículo	Código con el que se identifica vehículo
Fecha Entrega	Fecha de inicio del alquiler: entrega al cliente.
Fecha Devolución	Fecha de fin del alquiler: devolución por parte del cliente.
Información Cliente	Datos personales del cliente poseedor de la reserva. Incluye identificación, apellidos, nombre, e-mail, dirección y teléfono.

Tabla 21. Datos en Respuesta de Reserva de Vehículo

Reserva de Vehículo – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la reserva realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Código Reserva	Código identificador de la reserva.
Lugar	Lugar donde se localiza la empresa alquiler.
Vehículo	Especificación del vehículo reservado. Los datos incluyen: - Tipo - ¿Posee Aire? - ¿Es Automático? - Capacidad de personas - Descripción textual - Línea o marca - Sitio del alquiler, dirección - Costo total incluyendo adicionales

Tabla 22. Datos en Solicitud de Confirmación de Reserva de Vehículo

Confirmación de Reserva de Vehículo – Solicitud	
Campo	***Descripción***
Código Reserva	Código con el que se identifica la reserva.
Acción	Indica si se Confirma o Cancela definitivamente la reserva.
Identificación Cliente	Identificación del cliente propietario de la reserva.

Tabla 23. Datos en Respuesta de Confirmación de Reserva de Vehículos

Confirmación de Reserva de Vehículo – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la confirmación de reserva realizada.
Mensaje	Texto descriptivo del resultado y las causas de éste.

5.4.4.4 Información Clientes – Agencia

Los datos usados en las operaciones de búsqueda, reserva, confirmación o cancelación y consulta de reservas, y actualización de datos, de la interfaz agencia-cliente se describen en las tablas siguientes.

Tabla 24. Datos en Solicitud de Búsqueda de Planes de Viaje

Búsqueda de Planes de Viaje – Solicitud	
Campo	***Descripción***
Lugar Origen	Sitio de origen del viaje. Consta de *país* y *ciudad*.
Lugar Destino	Sitio de destino del viaje. Consta de *país* y *ciudad*.
Fecha Ida	Fecha de inicio del viaje: ida al sitio.
Fecha Regreso	Fecha de terminación del viaje: regreso del sitio.
Cantidad Personas	Número de personas a viajar.
Criterios de Vuelos	Información del transporte aéreo (vuelos) a buscar. Incluye: - Clase de tiquetes, opcional. - Hora de Ida, - Hora de Regreso (opcionales). - Criterio de orden: *precio*, *hora de ida*, *hora de regreso*.
Criterios de Alojamiento	Información del hospedaje o alojamiento a buscar. Incluye: - Tipo de Habitaciones, opcional. - Criterio de orden: *precio*, *habitaciones requeridas*.
Criterios de Vehículo	Datos del transporte local (autos) a buscar. Opcional e Incluye: - ¿Aire?, - ¿Automático?, - Capacidad mínima (*opcionales*). - Tipo de auto. - Criterio de orden: *precio*, *tipo*, *capacidad*.
Reserva del Mejor	Indicador de reserva automática del mejor plan de viaje encontrado.
Validación Cliente	Datos para validar el cliente en la agencia. Incluye id y contraseña.

Tabla 25. Datos en Respuesta de Búsqueda de Planes de Viaje

Búsqueda de Planes de Viaje – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la búsqueda.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Lugar Origen	Sitio de origen del viaje, consta de *país* y *ciudad*.
Lugar Destino	Sitio de destino del viaje, consta de *país* y *ciudad*.
Período	Descripción del período: fechas y duración del viaje.
Cantidad Personas	Número de personas en el viaje.
Vuelos de Ida Encontrados	Listado de vuelos de ida encontrados, cada uno incluye: - Código identificador, usado para reservar - Nombre de los aeropuertos de origen y destino - Fecha y Hora de salida - Duración total estimada del trayecto - Descripción del avión: nombre y modelo - Clase de los tiquetes - Costo total - Nombre de la Aerolínea.
Vuelos de Regreso Encontrados	Listado de vuelos de regreso encontrados. Cada uno contiene información similar a la de los vuelos de Ida.
Alojamientos Encontrados	Listado de los alojamientos. Cada uno con: - Código identificador, usado para reservar - Tipo de habitación - Descripción de las habitaciones - Cantidad de habitaciones necesarias - Cantidad de personas por habitación - Nombre del hotel - Nombre de la Cadena Hotelera - Costo total
Vehículos Encontrados	Listado de los vehículos a alquilar. Cada uno posee: - Código identificador, usado para reservar - ¿Posee Aire? - ¿Es Automático? - Capacidad de personas - Tipo y Descripción textual - Línea o marca - Sitio del alquiler - Nombre del Alquiler de Vehículos - Costo total

Tabla 26. Datos en Solicitud de Reserva de Plan de Viaje

Reserva de Plan de Viaje – Solicitud	
Campo	***Descripción***
Fecha Ida	Fecha de inicio del viaje: ida al sitio.
Fecha Regreso	Fecha de terminación del viaje: regreso del sitio.
Cantidad Personas	Número de personas a viajar.
Código Vuelo de Ida	Código identificador del vuelo de ida a reservar.
Código Vuelo de Regreso	Código identificador del vuelo de regreso a reservar.
Código Alojamiento	Código identificador del alojamiento a reservar.
Código Vehículo	Código identificador del vehículo a reservar. Es opcional.
Validación Cliente	Datos para validar el cliente en la agencia. Id y contraseña.

Tabla 27. Datos en Respuesta de Reserva de Plan de Viaje

Reserva de Plan de Viaje – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la reserva.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Código Reserva	Código identificador de la reserva.
Lugar Origen	Sitio de origen del viaje, consta de *país* y *ciudad*.
Lugar Destino	Sitio de destino del viaje, consta de *país* y *ciudad*.
Período	Período de viaje dado por las fechas y la duración.
Cantidad Personas	Número de personas a viajar.
Vuelo de Ida	Especificación del vuelo de ida. Incluye: - Nombre de los aeropuertos de origen y destino - Fecha y Hora de la salida - Duración estimada - Descripción del avión: nombre y modelo - Clase de los lugares reservados - Costo de los tiquetes - Nombre de la Aerolínea - Listado de códigos de las sillas asignadas en el avión
Vuelo de Regreso	Especificación del vuelo de regreso si se ha solicitado. Incluye la información similar a la del vuelo de ida.

Alojamiento	Especificación del alojamiento reservado. Incluye: - Nombre del Hotel - Tipo de habitación - Descripción de las mismas - Cantidad de habitaciones necesarias - Costo - Listado de habitaciones y cantidad de personas asignadas a cada una de ellas - Nombre de la Cadena Hotelera
Vehículo Alquilado	Especificación del vehículo alquilado. Es opcional ya que la solicitud de reserva del servicio también lo es. Incluye: - ¿Posee Aire?, ¿Es Automático? - Capacidad personas - Tipo y Descripción textual - Línea o marca - Sitio del alquiler - Costo - Nombre del Alquiler de Vehículos
Costo Total	Costo total de la reserva

Tabla 28. Datos en Solicitud de Confirmación de Reserva de Plan de Viaje

Confirmación de Reserva de Plan de Viaje – Solicitud	
Campo	***Descripción***
Código Reserva	Código con el que se identifica la reserva.
Acción	Indica si se Confirma o se Cancela definitivamente la reserva.
Validación Cliente	Datos para validar el cliente en la agencia: Id y contraseña.

Tabla 29. Datos en Respuesta de Confirmación de Reserva de Plan de Viaje

Confirmación de Reserva de Plan de Viaje – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la confirmación de reserva.
Mensaje	Texto descriptivo del resultado y las causas de éste.

Tabla 30. Datos en Solicitud de Consulta de Reservas

Consulta de Reservas – Solicitud	
Campo	***Descripción***
Validación Cliente	Información del cliente en la agencia. Incluye id y contraseña.

Tabla 31. Datos en Respuesta de Consulta de Reservas

Consulta de Reservas – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la confirmación de reserva.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Reservas	Listado de las reservas vigentes del cliente. Cada una posee: - Código - Origen y Destino: *país* y *ciudad*. - Período: *fechas* y *duración*. - Cantidad de personas incluidas - Vuelo de Ida y Vuelo de Regreso (opcional), y contienen: * Nombre de los aeropuertos de origen y destino * Fecha y Hora de salida * Duración estimada * Descripción del avión: nombre y modelo * Clase de los tiquetes * Costo tiquetes * Listado de códigos de sillas asignadas en el avión * Nombre de la aerolínea - Alojamiento reservado, que contiene: * Nombre del hotel * Tipo de habitación * Descripción de las habitaciones * Cantidad de habitaciones necesarias * Listado de habitaciones y personas asignadas a ellas * Costo * Nombre de la cadena hotelera - Vehículo alquilado, opcional y contiene: * ¿Posee Aire? * ¿ Automático? * Capacidad personas * Tipo y su descripción textual * Línea o marca * Sitio del alquiler * Costo * Nombre del alquiler de vehículos - Costo Total

Tabla 32. Datos en Solicitud de Administración de la Información del Cliente

Administración de la Información Cliente – Solicitud	
Campo	***Descripción***
Acción	Indica se va a consultar o a actualizar la información del cliente.
Nueva Información	Nueva información del cliente a ser almacenada cuando se solicita actualización.
Validación Cliente	Información de validación del cliente en la agencia. Incluye identificación y contraseña.

Tabla 33. Datos en Respuesta de Administración de la Información del Cliente

Administración de la Información Cliente – Respuesta	
Campo	***Descripción***
Resultado	Indicador del éxito o fracaso de la confirmación de reserva.
Mensaje	Texto descriptivo del resultado y las causas de éste.
Información Cliente	Información actual del cliente cuando se ha consultado.

5.4.5 Planificación del Desarrollo

En cada fase o etapa en que se divide el desarrollo, según la metodología adoptada para el proyecto, se implementa un subconjunto de los objetivos o requerimientos del mismo. Las fases fueron delimitadas según la mejor forma de agrupación de las funcionalidades de los sistemas identificadas previamente y también según la reutilización algunas de ellas que presentan entre sí, de esta forma realizar el desarrollo eficientemente aprovechando lo desarrollado previamente.

- ***Fase 1***: *Realización de Búsquedas*

  Consiste en implementar todos los casos de uso relacionados con búsquedas tanto a nivel del cliente, como en la agencia y los tres socios.

- ***Fase 2**: Realización de Reservas*

  En esta fase se implementan los casos de uso de las operaciones de la reserva. El proceso de búsqueda puede hacer uso de la reserva por lo que el desarrollo de ésta última no es independiente.

- ***Fase 3**: Realización de Consulta de Reservas*

  Aquí se despliegan los casos de uso relacionados con la consulta de reservas de los clientes y por tanto involucran sólo al software del cliente con el sistema de la agencia de viajes. Debe ser realizada antes de la confirmación o cancelación de reservas pues incluye su funcionalidad.

- ***Fase 4**: Realización de Confirmación y Cancelación de Reservas*

  Consiste en implementar los casos de uso de confirmación y cancelación de reservas en todos los sistemas.

- ***Fase 5**: Realización de Administración de la Información del Cliente*

  Descripción: en esta etapa se realiza el desarrollo relacionado con la administración de la información de los clientes tanto en la agencia de viajes como en el software del cliente.

- ***Fase 6**: Implementación de Seguridad*

  Consiste en realizar los ajustes necesarios para implementar la seguridad requerida en los sistemas que no se haya considerado previamente, y además se implementan requisitos adicionales no cubiertos hasta el momento.

- ***Fase 7**: Despliegue Final*

  Consiste en describir la forma como cada sistema se acopla para dar vida al prototipo funcional planteado.

Una vista interior preliminar de las fases de desarrollo permitió establecer una secuencia de pasos, buscando una mejor interpretación del proceso seguido en la elaboración del presente trabajo de grado, las cuales se describen a continuación:

- *Desarrollo del proceso en la Agencia de Viajes*
  Especifica detalladamente el desarrollo de cada proceso de negocio BPEL que implementa la lógica de ejecución de las operaciones requeridas en la agencia de viajes. El punto de partida debe ser la definición de los esquemas XML de los datos a intercambiar y las interfaces WSDL involucradas en todo el proceso, comenzando por los socios, siguiendo con la interfaz software del cliente y finalmente con la agencia, la cual posee servicios Web internos que ejecutan las tareas especializadas que BPEL no alcanza a cubrir.

  Los esquemas se definen comenzando cada proceso para lograr un desarrollo ordenado, uniforme y eficiente. Las representaciones de los esquemas XML se harán mediante diagramas de clase UML, los cuales permiten apreciar más fácilmente características como herencia, multiplicidad y composición en paquetes (espacios de nombres) usados a la hora de definir los datos, y también favorecen la interpretación de la información.

  Cuando se implementan en código, los tipos de datos XML usados (en general los esquemas XSD), se convierten en clases, relacionando los elementos y atributos del esquema con tipos de datos propios del lenguaje usado en la implementación, en este caso java.

- *Implementación del Sistema de cada Socio*
  Muestra la implementación de las funcionalidades de los socios enlazando el software ejemplo de procesamiento de solicitudes usando la interfaz WSDL definida previamente.

- *Implementación del software del Cliente*
  Diseño y desarrollo de la interfaz software de la agencia de viajes con el cliente vía servicios Web.

- *Realización de Pruebas*
  Diseño y realización de las pruebas pertinentes a los desarrollos.

5.5 DESARROLLO

A continuación se describe el trabajo en las etapas del desarrollo descritas anteriormente. Cada fase se explica detalladamente sólo en los pasos realizados por primera vez y que presentan cambios significativos, ya que el trabajo realizado en cada fase presenta características similares.

5.5.1 Fase Preliminar. Configuración de las Definiciones WSDL

El primer paso del desarrollo consiste en precisar y especificar la configuración de las definiciones WSDL del proyecto y así establecer un parámetro guía para las etapas subsecuentes. Como el éxito de la comunicación está dado por la correcta definición de las interfaces WSDL, para el sistema se plantearon de conformidad con la especificación WSDL 1.1 y con las guías WS-I Basic Profile 1.1, WS-I Simple SOAP Binding Profile y WS-I Attachment Profile 1.0.

La configuración de las definiciones WSDL consiste en:

- Estilo de Invocación: ***SOAP - Document Literal***
  Todas las interfaces de operaciones *portType* se definen como mensajes SOAP. Además el estilo de los mensajes es Document-Literal por lo tanto su contenido es interpretado como XML puro, obligando a que las operaciones intercambien mensajes de entrada y salida que posean sólo un componente definido como *elemento* XML global.

- Uso de ***Excepciones***

  Para ilustrar el manejo de excepciones o fallos en las transacciones con servicios Web y en los procesos de negocio, se implementarán excepciones en la operación de reserva de la interfaz entre la agencia de viajes y el socio alquiler de vehículos y todos los procesos expuestos al cliente, capturando si el cliente no fue autorizado y en general los procesos no se realizaron satisfactoriamente. Así estas operaciones tendrán, además de su entrada y salida, la declaración de varias excepciones que encapsulen los resultados del proceso fallido y la información del error presentado, estas excepciones proporcionan robustez al desarrollo.

- ***Invocaciones Asíncronas***

  Las operaciones asíncronas permiten manejar mejor las posibles fallas de comunicación y además le da un carácter más real al uso de Servicios Web. Para tener en cuenta su comportamiento y tratamiento, las operaciones que la agencia de viajes realiza en los socios son invocadas de esta forma y por lo tanto necesitan de un mensaje de sólo petición y uno de respuesta especial definido en el otro lado de la interfaz. La operación de reserva en el socio alquiler de vehículos está exenta de esto pues utiliza excepciones, por otro lado la interfaz con el cliente no presenta invocaciones asíncronas pues la agencia no tiene forma de contactar directamente a sus clientes, sólo responde a las peticiones que ellos realicen: mensajes petición/respuesta totalmente síncronos en los cuales se implementan excepciones para capturar posibles fallos.

- Espacios de Nombres, ***Namespaces***

  Los espacios de nombres se asemejan a *paquetes* que agrupan las definiciones XML y logran una estructura entendible a los documentos. Se han definido los siguientes espacios de nombres con su respectivo prefijo, de acuerdo a los sistemas presentes y el uso que tienen:

Para los *esquemas XML* de los datos intercambiados en las operaciones de la agencia de viajes, clientes y socios:

- *Datos comunes o globales a todos los sistemas*
  `gls: http://agenciaviajes.com/esquemas/global`
- *Datos comunes intercambiados con los socios*
  `sos: http://agenciaviajes.com/esquemas/socios`
- *Datos de la interfaz con la aerolínea, usados también por ella*
  `aes: http://agenciaviajes.com/esquemas/aerolinea`
- *Datos de la interfaz con la cadena hotelera, usados por ella también*
  `hts: http://agenciaviajes.com/esquemas/cadenahotelera`
- *Datos de la interfaz con el alquiler de vehículos, usados por él también*
  `vhs: http://agenciaviajes.com/esquemas/alquilervehiculos`
- *Datos de la interfaz con el sistema del cliente*
  `cls: http://agenciaviajes.com/esquemas/cliente`
- *Datos de los procesos internos de la agencia*
  `avs: http://agenciaviajes.com/esquemas/agencia`

Para las *definiciones WSDL y procesos BPEL* de la agencia de viajes:

- *Interfaz de servicios internos de la agencia*
  `age: http://agenciaviajes.com/servicios/agencia`
- *Interfaz con el sistema del cliente*
  `acl: http://agenciaviajes.com/servicios/cliente`
- *Interfaz con las aerolíneas*
  `aae: http://agenciaviajes.com/servicios/aerolinea`
- *Interfaz con las cadenas hotelera*
  `aht: http://agenciaviajes.com/servicios/cadenahotelera`
- *Interfaz con los alquileres de vehículos*
  `avh: http://agenciaviajes.com/servicios/alquilervehiculos`
- *Propiedades de los procesos BPEL*
  `appa: http://agenciaviajes.com/procesos/agenciadeviajesPropiedades`

Para las definiciones WSDL propias de los socios:

- *Interfaz de la aerolínea con la agencia de viajes*
  `aer: http://aerolinea.com/servicios/agenciaviajes`
- *Interfaz de la cadena hotelera con la agencia de viajes*
  `htl: http://cadenahotelera.com/servicios/agenciaviajes`
- *Interfaz del alquiler de vehículos con la agencia de viajes*
  `vhc: http://alquilervehiculos.com/servicios/agenciaviajes`

5.5.2 Fase 1. Búsquedas de Planes de Viaje

La búsqueda de planes de viaje es la operación de partida del desarrollo. Según el análisis de su lógica (*véase 5.4.3.1*), implica tareas sencillas en los que interviene el cliente, el sistema de la agencia de viajes y los sistemas de los socios.

5.5.2.1 Desarrollo del Proceso en la Agencia de Viajes

El proceso de buscar de planes de viaje intercambia información con los socios y con el cliente, teniendo en cuenta lo definido previamente (*véase 5.4.4*), toda la información debe representarse primeramente mediante esquemas XML los cuales serán usados por las definiciones WSDL.

- ***Esquemas XML de la Información***

Los esquemas se organizaron en espacios de nombres (o paquetes UML) agrupados según su uso tal cual como se enunció en la fase preliminar. A continuación se presentan los diagramas de clases UML, explicados en tablas continuas, de la información intercambiada en las búsquedas en todos los socios.

Notación del Diagrama de Clases

Las clases representan tipos de datos, que según su definición XML pueden ser simples *simpleType* cuando representan sólo un valor de tipos nativos XML como int, string, o boolean (pero puede poseer atributos), y complejos *complexType*

cuando poseen elementos de cualquier tipo y además atributos. Para identificarlos se usan los estereotipos <<XSDsimpleType>> y <<XSDcomplexType>>. Por defecto se ha tomado que todo campo de las clases es de tipo *element* (etiqueta xml <element>); también existen campos de tipo *attribute* (etiqueta <attribute>) o *enumeration* (etiqueta <enumeration>) a los cuales se les añade el estereotipo <<XSDattribute>> o <<XSDenumeration>> respectivamente. Una clase en color blanco y sin atributos indica que ya fue presentada plenamente en un diagrama anterior.

Figura 32. Diagrama de Clases – Aerolínea, Entrada de Búsqueda

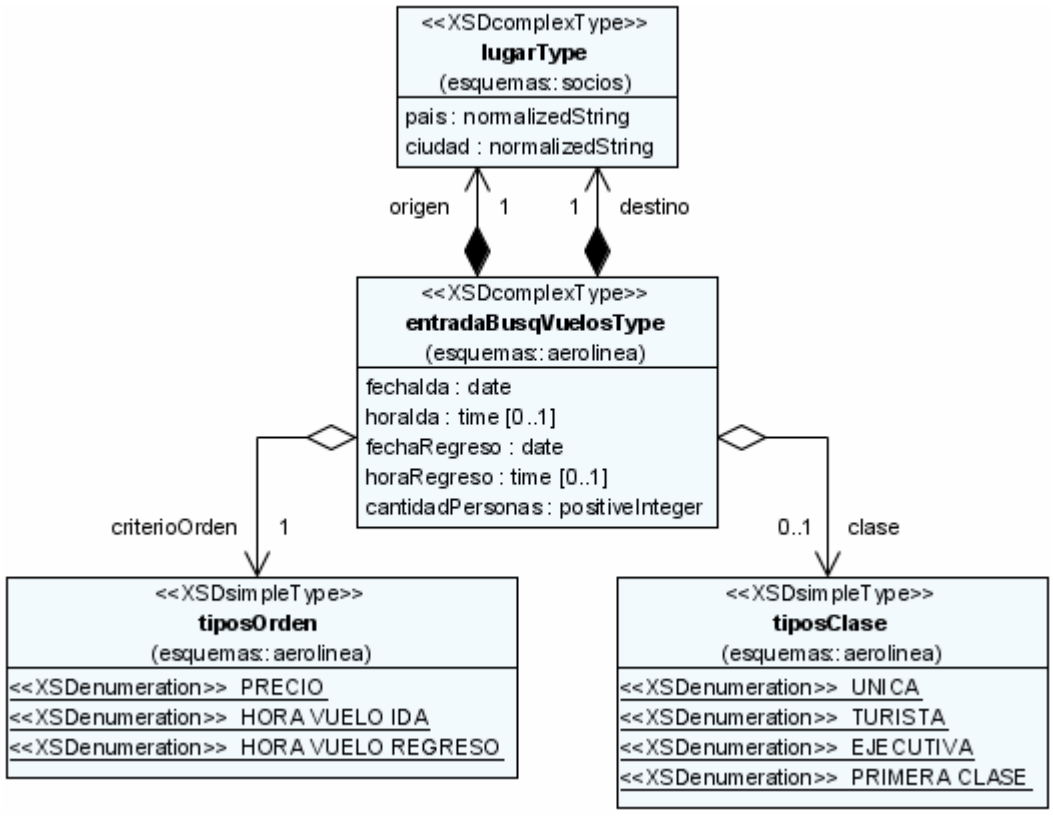


Tabla 34. Descripción de Clases – Aerolínea - Entrada de Búsqueda

Descripción Clases – Búsqueda en Aerolíneas, Entrada Información descrita en Tabla 6. Datos en Solicitud de Búsqueda de Vuelos.	
Clase y Paquete	***Descripción***
TiposOrden *esquemas.aerolínea*	Tipo simple String que enumera los posibles criterios de ordenación de los resultados de la búsqueda de vuelos.: PRECIO, HORA VUELO IDA, HORA VUELO REGRESO.
TiposClase *esquemas.aerolínea*	Tipo simple String que define las clases de tiquetes a buscar con las enumeraciones: UNICA, TURISTA, EJECUTIVA, PRIMERA CLASE.
LugarType *esquemas.socios*	Tipo complejo que encapsula los códigos del lugar o sitio (país y ciudad) incluido en las búsquedas en los socios.
EntradaBusqType *esquemas.aerolínea*	Tipo complejo que encapsula todos los criterios de la búsqueda de vuelos especificados en la tabla 6.

Figura 33. Diagrama de Clases – Aerolínea, Salida de Búsqueda

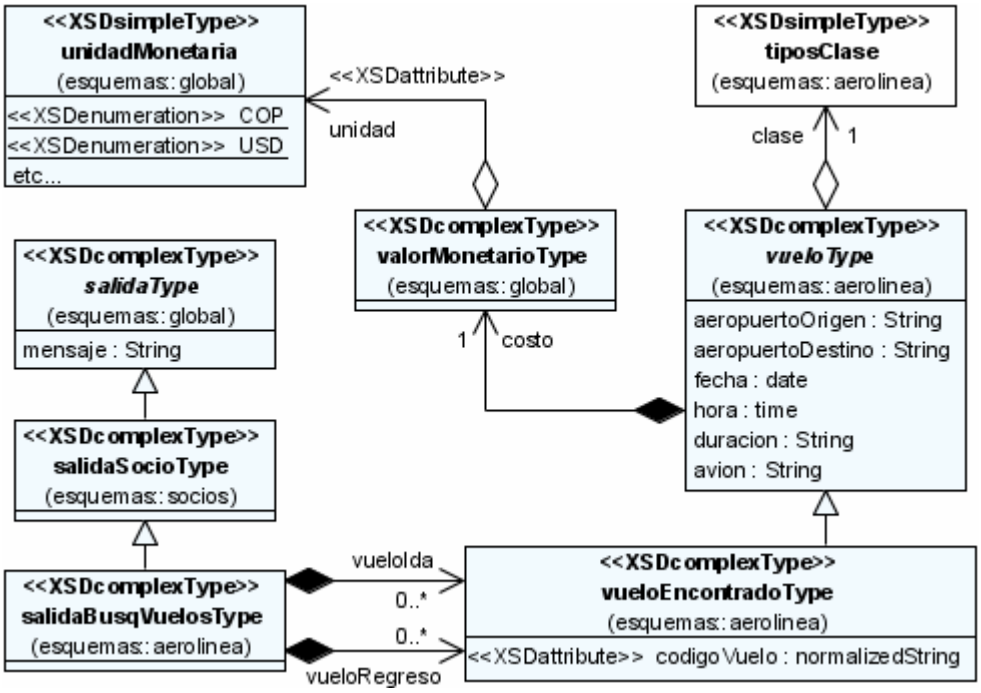


Tabla 35. Descripción de Clases – Aerolínea. Salida de Búsqueda

Descripción Clases –Búsquedas en Aerolíneas, Salida Información descrita en Tabla 7. Datos en Respuesta de Búsqueda de Vuelos.	
Clase y Paquete	***Descripción***
UnidadMonetaria *esquemas.global*	Tipo simple String que enumera los tipos de moneda o *currency*, según el estándar ISO 4217: COP, USD, etc.
ValorMonetarioType *esquemas.global*	Tipo complejo de contenido simple Double que encapsula un valor monetario con la respectiva unidad.
VueloType *esquemas.aerolinea*	Tipo complejo abstracto con la información básica de un vuelo, común tanto en la búsqueda como en la reserva.
VueloEncontradoType *esquemas.aerolinea*	Tipo complejo que encapsula la información de los vuelos encontrados. Extiende de VueloType.
SalidaType *esquemas.global*	Tipo complejo abstracto que define los datos de salida generales, es decir el mensaje descriptivo
SalidaSocioType *esquemas.socios*	Tipo complejo con la salida estándar de toda operación en los socios. Extiende de SalidaType.
SalidaBusqVuelosType *esquemas.aerolínea*	Tipo complejo para los resultados la búsqueda de vuelos, especificados en la tabla 7, extiende de SalidaSocioType.

Figura 34. Diagrama de Clases – Cadena Hotelera, Entrada de Búsqueda

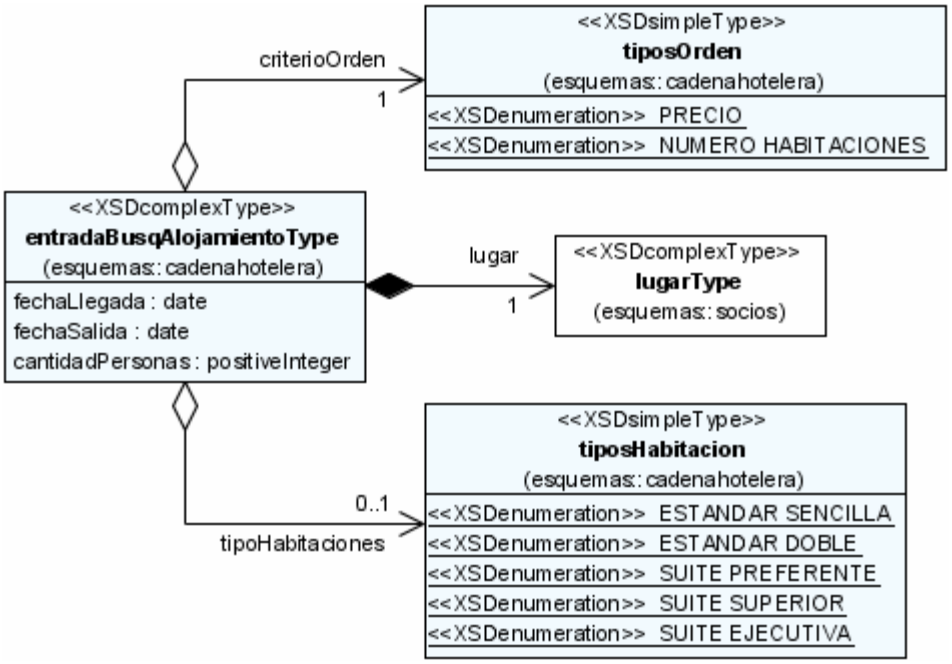


Tabla 36. Descripción de Clases – Cadena Hotelera, Entrada de Búsqueda

Descripción Clases – Búsqueda en Cadenas Hoteleras, Entrada Información descrita en Tabla 12. Datos en Solicitud de Búsqueda de Alojamientos.	
Clase y Paquete	***Descripción***
TiposHabitacion *esquemas.cadenahotelera*	Tipo simple String que define los tipos de habitación a buscar con las enumeraciones: ESTANDAR SENCILLA, ESTANDAR DOBLE, SUITE PREFERENTE, SUITE SUPERIOR, SUITE EJECUTIVA.
TiposOrden *esquemas.cadenahotelera*	Tipo simple String que enumera los criterios de ordenación de los resultados de la búsqueda de alojamientos: PRECIO, NUMERO HABITACIONES.
EntradaBusqAlojamientoType *esquemas.cadenahotelera*	Tipo complejo que encapsula todos los criterios de la búsqueda de alojamientos especificados en la tabla 12.

Figura 35. Diagrama de Clases – Cadena Hotelera, Salida de Búsqueda

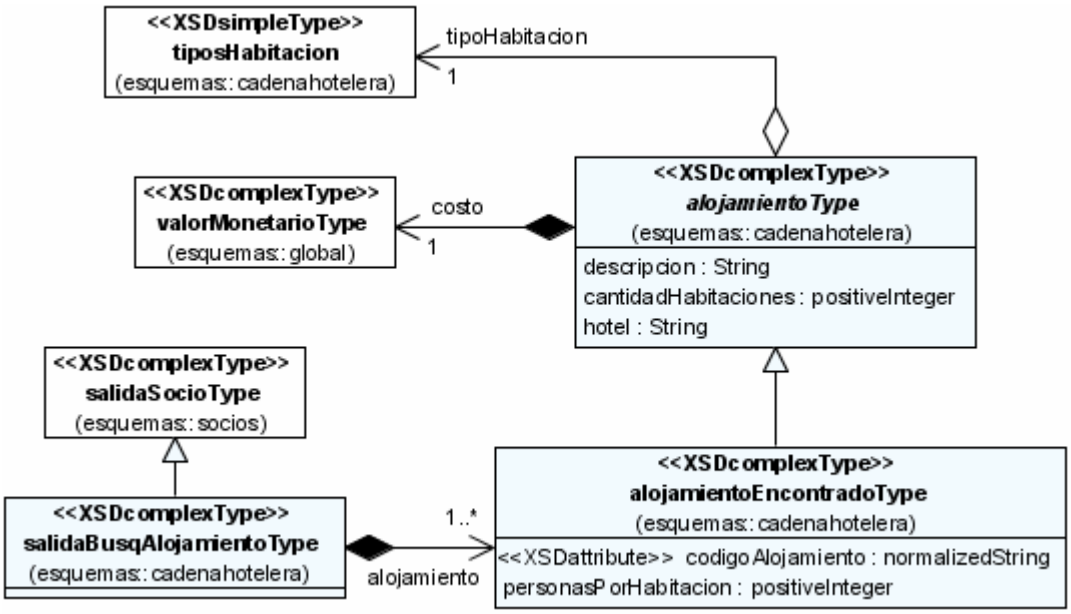


Tabla 37. Descripción de Clases – Cadena Hotelera, Salida de Búsqueda

Descripción Clases – Búsquedas en Cadenas Hoteleras, Salida Información descrita en Tabla 13. Datos en Respuesta de Búsqueda de Alojamientos.	
Clase y Paquete	***Descripción***
AlojamientoType *esquemas.cadenahotelera*	Tipo complejo abstracto con la información básica de un alojamiento, común búsquedas y reservas.
AlojamientoEncontradoType *esquemas.cadenahotelera*	Tipo complejo que encapsula la información de los alojamientos encontrados. Extiende del tipo AlojamientoType.
SalidaBusqAlojamientoType *esquemas.cadenahotelera*	Tipo complejo con los resultados de la búsqueda, especificados en la tabla 13. Extiende del tipo SalidaSocioType.

Figura 36. Diagrama de Clases – Alquiler Vehículos. Entrada de Búsqueda

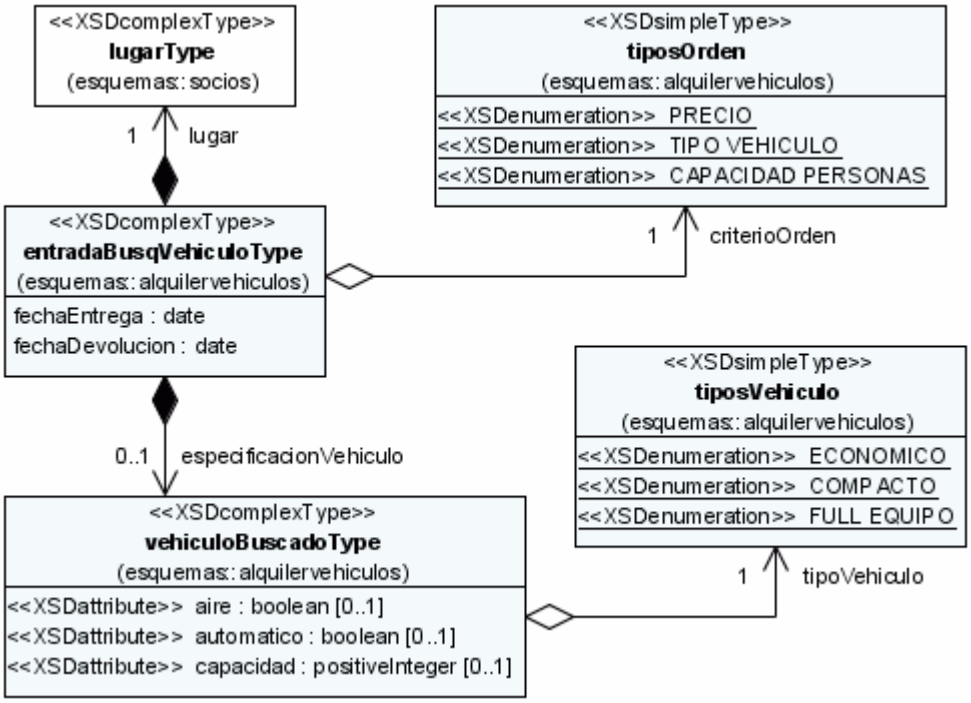


Tabla 38. Descripción de Clases – Alquiler Vehículos, Entrada de Búsqueda

Descripción Clases – Búsqueda en Alquileres de Vehículos, Entrada Información descrita en Tabla 18. Datos en Solicitud de Búsqueda de Vehículos.	
Clase y Paquete	***Descripción***
TiposOrden *esquemas.alquilervehiculos*	Tipo simple String que enumera las formas de orden de los resultados de la búsqueda de vehículos: PRECIO, TIPO VEHÍCULO, CANTIDAD PERSONAS
TiposVehiculo *esquemas.alquilervehiculos*	Tipo simple String que define los tipos de vehículos a buscar con las enumeraciones: ECONOMICO, COMPACTO, FULL EQUIPO.
VehiculoBuscadoType *esquemas.alquilervehiculos*	Tipo complejo que encapsula las especificaciones del vehículo buscado.
EntradaBusqVehiculoType *esquemas.alquilervehiculos*	Tipo complejo que encapsula todos los criterios de la búsqueda de vehículos especificados en la tabla 18.

Figura 37. Diagrama de Clases – Alquiler Vehículos, Salida de Búsqueda

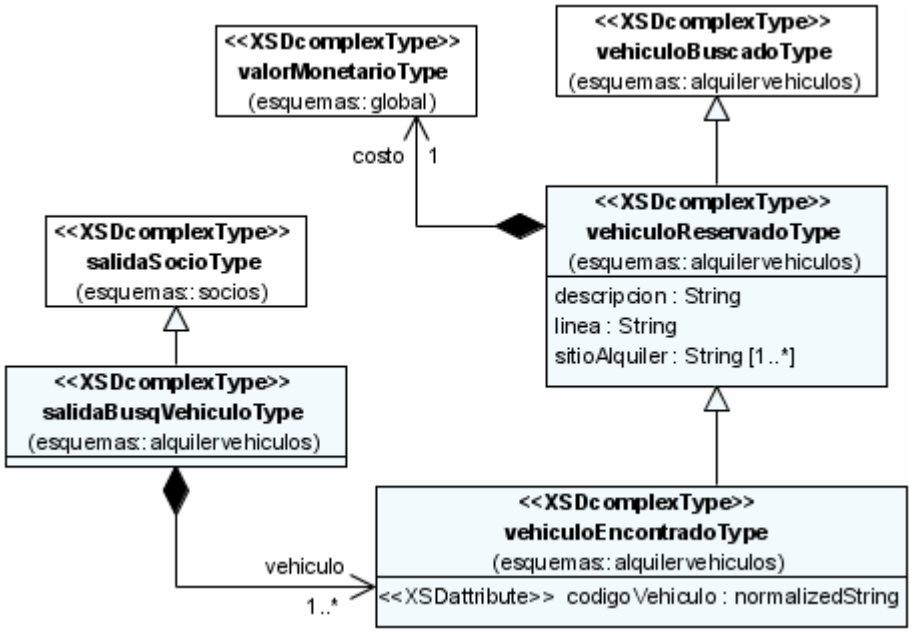


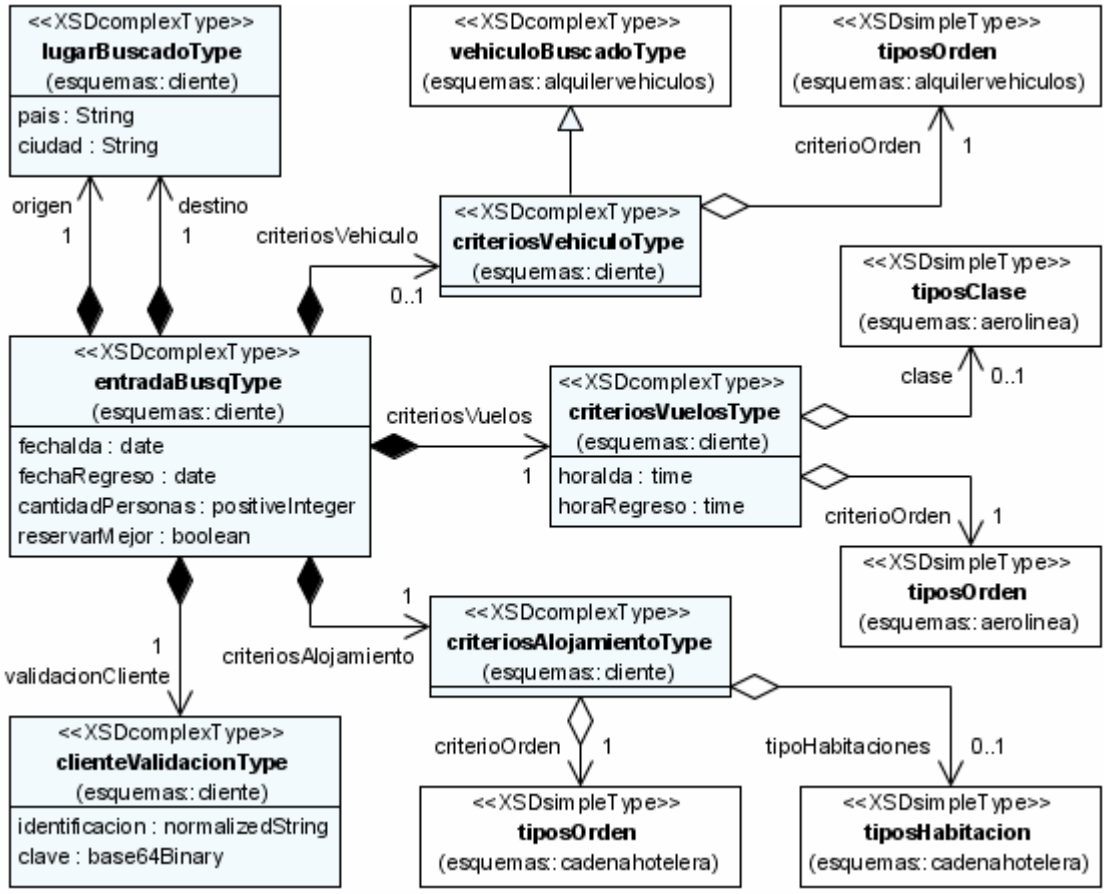
Tabla 39. Descripción de Clases – Alquiler Vehículos, Salida de Búsquedas

Descripción Clases – Búsquedas en Alquileres de Vehículos, Salida Información descrita en Tabla 19. Datos en Respuesta de Búsqueda de Vehículos.	
Clase y Paquete	***Descripción***
VehiculoReservadoType *esquemas.alquilervehiculos*	Tipo complejo con la información de un vehículo reservado y que se usa también como respuesta en una búsqueda.
VehiculoEncontradoType *esquemas.alquilervehiculos*	Tipo complejo que encapsula la información de los vehículos encontrados en la búsqueda. Extiende de VehiculoReservadoType.
SalidaBusqVehiculoType *esquemas.alquilervehiculos*	Tipo complejo para los resultados la búsqueda de vehículos a alquilar, especificados en la tabla 19. Extiende del tipo SalidaSocioType.

Tabla 40. Descripción de Clases – Cliente. Entrada de Búsqueda

Descripción Clases – Búsqueda para Clientes, Entrada Información descrita en Tabla 24. Datos en Solicitud de Búsqueda de Planes de Viaje.	
Clase y Paquete	***Descripción***
LugarBuscadoType *esquemas.cliente*	Tipo complejo que representa un lugar dado por los nombres de la ciudad y el país.
CriteriosVuelosType *esquemas.cliente*	Tipo complejo que encapsula la especificación de búsqueda de los vuelos, reutiliza la información ya definida para la aerolínea (otras clases).
CriteriosAlojamientoType *esquemas.cliente*	Tipo complejo con la especificación del alojamiento a buscar, utiliza clases definidas en la cadena hotelera.
CriteriosVehiculoType *esquemas.cliente*	Tipo complejo con los criterios de búsqueda del vehículo a alquilar, reutiliza tipos ya definidos para la interfaz con el alquiler de vehículos y extiende de VehiculoBuscadoType.
ClienteValidacionType *esquemas.cliente*	Tipo complejo que define la información de validación del cliente en la agencia: nombre y clave.
EntradaBusqType *esquemas.cliente*	Tipo complejo que encapsula toda la información referente a los criterios de la búsqueda de planes de viaje según lo definido en la tabla 24.

Figura 38. Diagrama de Clases – Cliente, Entrada de Búsqueda



Respecto a las definiciones de las clases para los resultados de las búsquedas es necesario hacer explícita la dependencia de la búsqueda con la reserva, descrita en la Planificación del Desarrollo (*véase 5.4.5*). Esto se aprecia en que los resultados de una búsqueda pueden ser los mismos que los de una reserva, que serán definidos posteriormente en la fase 2. También se destaca la presencia de los tipos de datos para el manejo de excepciones y la captura de su información.

En el diagrama siguiente se aprecia una clase en color *verde claro* que no está detalla porque su estudio y definición hace parte de la fase 2, sólo aparece enunciada. En el Anexo A se muestran todos los diagramas de clases UML

organizados por paquete (espacio de nombres), en ellos se puede apreciar la totalidad de los tipos de datos XML definidos y las relaciones que existen.

Figura 39. Diagrama de Clases – Cliente, Salida de Búsquedas

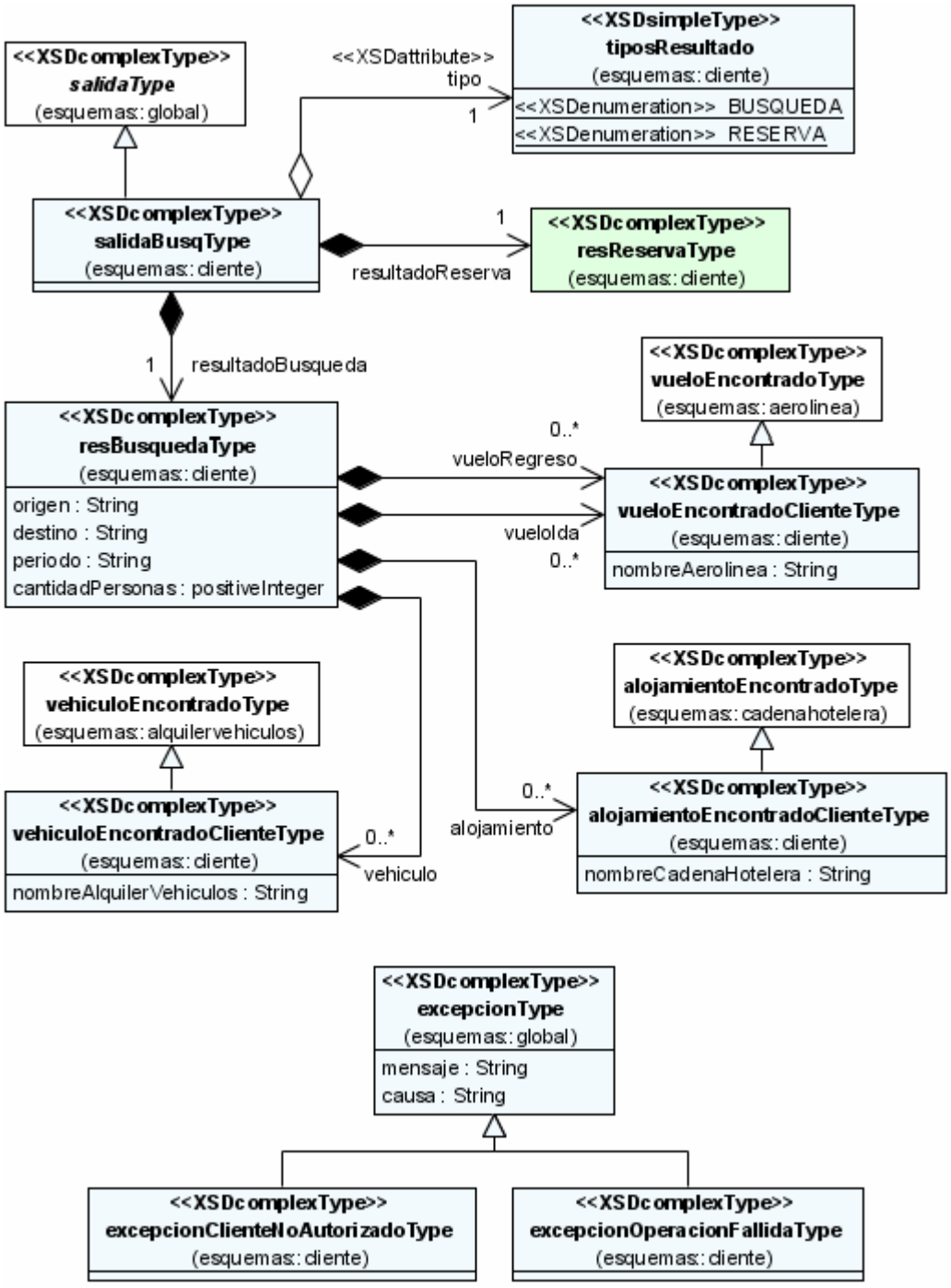


Tabla 41. Descripción de Clases – Cliente, Salida de Búsqueda

Descripción Clases – Búsqueda para Clientes, Salida Información descrita en Tabla 25. Datos en Respuesta de Búsqueda de Planes de Viaje.	
Clase y Paquete	***Descripción***
TiposResultado *esquemas.cliente*	Tipo simple String con los tipos de resultado del proceso de búsqueda con posible reserva, enumeraciones: BUSQUEDA, RESERVA.
SalidaBusqType *esquemas.cliente*	Tipo complejo que encapsula los resultados tanto de búsqueda como de reservas, y es usado como resultado real de la operación. Extiende de SalidaType.
ResBusquedaType *esquemas.cliente*	Tipo complejo que encapsula toda la información de los resultados de búsqueda de planes de viaje especificados en la tabla 25.
ResReservaType *esquemas.cliente*	Tipo complejo que encapsula la información resultado de las reservas de planes de viaje. Será analizada posteriormente.
VueloEncontrado_ ClienteType *esquemas.cliente*	Tipo complejo con la información de cada vuelo encontrado. Extiende de VueloEncontradoType, definido en la aerolínea.
AlojamientoEncontrado_ ClienteType *esquemas.cliente*	Tipo complejo con los datos de cada alojamiento encontrado. Extiende de AlojamientoEncontradoType definido en la cadena hotelera.
VehiculoEncontrado_ ClienteType *esquemas.cliente*	Tipo complejo con la información de cada vehículo encontrado. Extiende de VehiculoEncontradoType, definido en el alquiler de vehículos.
ExcepcionType *esquemas.global*	Tipo complejo abstracto y base de las excepciones que encapsula la información descriptiva. Posee el atributo *mensaje* que representa la descripción del fallo.
ExcepcionCliente_ NoAutorizadoType *esquemas.cliente*	Tipo complejo que representa la excepción debida a la no verificación del cliente que intenta realizar las operaciones definidas.
ExcepcionOperacion_ FallidaType *esquemas.cliente*	Tipo complejo que representa la excepción debida al fallo de la operación: no fue exitosa.

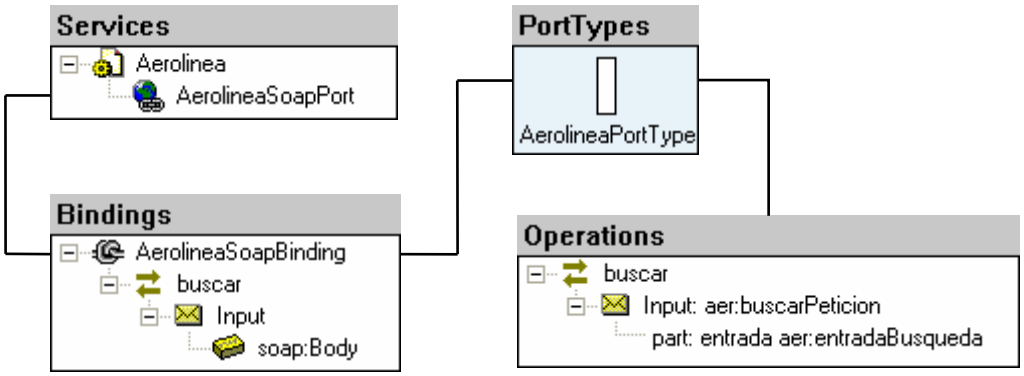
- ***Definición de Interfaces WSDL***

Para definir cada una de las interfaces externas WSDL se tiene en cuenta la configuración WSDL de la fase preliminar. Según el análisis preliminar de las funcionalidades (*véase 5.4.2*), la búsqueda de planes de viaje comprende la operación de búsqueda en la agencia y la invocación de la búsqueda de cada servicio en los socio respectivo.

Petición de la Búsqueda en el Socio Aerolínea

La interfaz propuesta, relacionada con la petición de la búsqueda que el socio aerolínea debe implementar para ofrecer sus servicios se muestra gráficamente en la figura 40.

Figura 40. Estructura WSDL de Aerolíneas: Búsqueda



El código WSDL que ilustra la sintaxis aparece en la figura 41 y se describe en la detalladamente tabla 42.

Figura 41. Documento WSDL de Aerolíneas: Búsqueda

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="aerolinea"
                  xmlns:aer="http://aerolinea.com/servicios/agenciaviajes"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  targetNamespace="http://aerolinea.com/servicios/agenciaviajes">

  <wsdl:types>
    <xsd:schema targetNamespace="http://aerolinea.com/servicios/agenciaviajes"
                 xmlns:aes="http://agenciaviajes.com/esquemas/aerolinea">
      <xsd:import namespace="http://agenciaviajes.com/esquemas/aerolinea"
                  schemaLocation="./aerolinea.xsd" />
      <xsd:element name="entradaBusqueda" type="aes:entradaBusqVuelosType" />
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="buscarPeticion">
    <wsdl:part name="entrada" element="aer:entradaBusqueda" />
  </wsdl:message>

  <wsdl:portType name="AerolineaPortType">
    <wsdl:operation name="buscar">
      <wsdl:input name="buscarIn" message="aer:buscarPeticion" />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="AerolineaSoapBinding" type="aer:AerolineaPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="buscar">
      <soap:operation soapAction="http://aerolinea.com/servicios/agenciaviajes/buscar" />
      <wsdl:input>
        <soap:body parts="entrada" use="literal" />
      </wsdl:input>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="AerolineaService">
    <wsdl:port name="AerolineaSoapPort" binding="aer:AerolineaSoapBinding">
      <soap:address location="http://www.aerolinea.com/services/AerolineaSoap" />
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

Tabla 42. Interfaz WSDL de Aerolíneas: Búsqueda

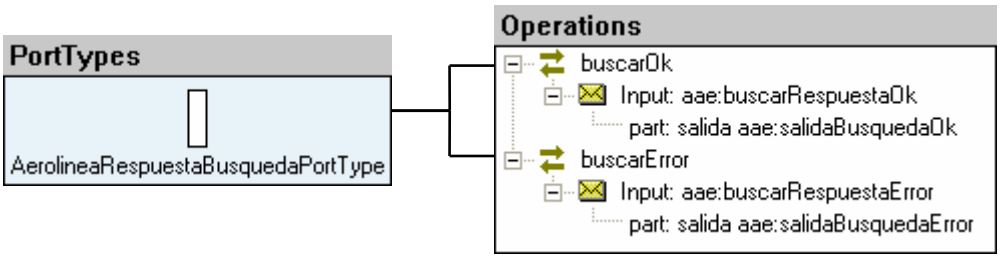
Elemento	Descripción del Contenido
Types	Contiene un esquema local con la declaración del elemento usado en el mensaje, además importar el esquema XML de las aerolíneas donde se encuentran definidos los tipos de datos enunciados previamente. - ***aer:entradaBusqueda***: elemento de tipo aes:entradaBusqVuelosType con la información necesaria en la búsqueda de vuelos, criterios.
Message	Define el mensaje a transmitir en la operación buscar: - ***buscarPeticion***: mensaje de la petición de la operación *buscar*, contiene cómo única parte, el elemento entradaBusqueda.
PortType	Define la interfaz **AerolineaPortType** con sus operaciones: - *buscar*: invoca la búsqueda, asigna como entrada el mensaje buscarPeticion. Sólo se asigna entrada pues es invocación asíncrona.
Binding	El enlace **AerolineaSoapBinding** establece estilo *document* y transporte SOAP. Se configura la operación *buscar* para que tenga contenido tipo *literal*.
Service	Define el servicio **AerolíneaService** que asocia el binding con la dirección física la red mediante el puerto *AerolineaSoapPort*.

En el Anexo B se puede apreciar el documento WSDL completo (con todas sus operaciones) del servicio expuestos por el socio alquiler de vehículos a manera de ejemplo. En el código fuente del desarrollo están disponibles todos los archivos relacionados.

Respuesta de la Búsqueda en el Socio Aerolínea

La aerolínea debe contactar la agencia para enviar los mensajes con la respuesta de la búsqueda (es una operación asíncrona); por tanto se debe declarar una interfaz entre la agencia y el socio aerolínea con las operaciones de respuesta, también asíncronas. Esta interfaz no define enlaces ni servicios, pues éstos son generados por el motor del sistema de la agencia en el momento de su publicación; los procesos BPEL se preocupan por el qué (la estructura) y no por el cómo. Gráficamente la interfaz definida se visualiza en la figura 42.

Figura 42. Estructura WSDL de Aerolíneas: Respuesta de Búsqueda



Allí se aprecian dos operaciones de respuesta, dado que se debe indicar si la búsqueda fue exitosa (buscarOk) o si por el contrario, fue fallida (buscarError). El código WSDL de esta interfaz, mostrado en la figura 43 y descrito en la tabla 43, es similar al de la petición de búsqueda pero no incluye definición de *bindings* ni *services*, además se aprecia la inclusión de las dos operaciones en el portType.

Tabla 43. Interfaz WSDL de Aerolíneas: Respuesta de Búsqueda

Elemento	Descripción del Contenido
Types	Contiene un esquema local con los elementos usados en el mensaje, importando los esquemas que definen los tipos de datos necesarios. - ***aer:salidaBusquedaOk***, elemento de tipo aes:salidaBusqVuelosType con los resultados de la búsqueda de vuelos. - ***aer:salidaBusquedaError***, elemento de tipo sos:salidaSocioType con la información del resultado de la búsqueda fallida.
Message	Define los mensajes a transmitir como resultados de la operación buscar: - ***buscarRespuestaOk***: mensaje de la respuesta exitosa de la operación *buscar*, contiene cómo única parte, el elemento salidaBusquedaOk. - ***buscarRespuestaError***: mensaje de respuesta fallida de la operación *buscar*, contiene cómo única parte, el elemento salidaBusquedaError.
PortType	Define la interfaz **AerolineaRespuestaBusquedaPortType** con sus operaciones: - *buscarOk*, responde el éxito, asigna como entrada el mensaje buscarRespuestaOk. - *buscarError*, responde el fallo, asigna como entrada el mensaje buscarRespuestaError.

Figura 43. Documento WSDL de Aerolíneas: Respuesta de Búsqueda

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="aerolinearespuesta"
                  xmlns:aae="http://agenciaviajes.com/servicios/aerolinea"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  targetNamespace="http://agenciaviajes.com/servicios/aerolinea" >

  <wsdl:types>
    <xsd:schema targetNamespace="http://agenciaviajes.com/servicios/aerolinea"
                 xmlns:aes="http://agenciaviajes.com/esquemas/aerolinea"
                 xmlns:sos="http://agenciaviajes.com/esquemas/socios">
      <xsd:import namespace="http://agenciaviajes.com/esquemas/socios"
                 schemaLocation="./socios.xsd"/>
      <xsd:import namespace="http://agenciaviajes.com/esquemas/aerolinea"
                 schemaLocation="./aerolinea.xsd"/>
      <xsd:element name="salidaBusquedaOk" type="aes:salidaBusqVuelosType"/>
      <xsd:element name="salidaBusquedaError" type="sos:salidaSocioType"/>
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="buscarRespuestaOk">
    <wsdl:part name="salida" element="aae:salidaBusquedaOk"/>
  </wsdl:message>
  <wsdl:message name="buscarRespuestaError">
    <wsdl:part name="salida" element="aae:salidaBusquedaError"/>
  </wsdl:message>

  <wsdl:portType name="AerolineaRespuestaBusquedaPortType">
    <wsdl:operation name="buscarOk">
      <wsdl:input name="buscarOkIn" message="aae:buscarRespuestaOk"/>
    </wsdl:operation>
    <wsdl:operation name="buscarError">
      <wsdl:input name="buscarErrorIn" message="aae:buscarRespuestaError"/>
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

Búsqueda en los Socios Cadena Hotelera y Alquiler de Vehículos

Para los socios Cadena Hotelera y Alquiler de Vehículos, el proceso de búsqueda se concibió de forma similar al del socio Aerolínea, por tanto sólo de detalla esta

última. Los cambios que presentan estas definiciones abarcan los nombres dados, los esquemas importados y por ende los tipos de datos usados. Por ejemplo, el socio cadenas hoteleras utiliza los esquemas propios y los espacios de nombres asignados para él (htl y hts). Un resumen de las características cambiantes se presenta a continuación.

Tabla 44. Resumen de cambios en interfaz WSDL para Búsqueda

Socio	Cambios
Cadena Hotelera	*Petición Búsqueda* - Elemento *htl:entradaBusqueda* de tipo *hts:entradaBusqAlojamientoType* - Interfaz *CadenaHoteleraPortType* - Enlace *CadenaHoteleraSoapBinding* - Servicio *CadenaHoteleraService* con puerto *CadenaHoteleraSoapPort*
	Respuesta Búsqueda - Elementos *htl:salidaBusquedaOk* de tipo *hts:salidaBusqAlojamientoType*, y *htl:salidaBusquedaError* de tipo *sos:salidaSocioType*. - Interfaz CadenaHoteleraRespuestaBusquedaPortType
Alquiler de Vehículos	*Petición Búsqueda* - Elemento *vhc:entradaBusqueda* de tipo *vhs:entradaBusqVehiculoType*. - Interfaz *AlquilerVehiculosPortType* - Enlace *AlquilerVehiculosSoapBinding* - Servicio *AlquilerVehiculosService* con puerto *AlquilerVehiculosSoapPort*
	Respuesta Búsqueda - Elementos *vhc:salidaBusquedaOk* de tipo *vhs:salidaBusqVehiculoType*, y *vhc:salidaBusquedaError* de tipo *sos:salidaSocioType*. - Interfaz *AlquilerVehiculosRespuestaBusquedaPortType*

Búsqueda en el Cliente

La agencia de viajes debe exponer una interfaz para que el sistema del cliente acceda al proceso de búsqueda. Esta se define sin los enlaces o *bindings* ni el servicio pues, al igual que las interfaces de respuesta de los socios, ésta

información se genera en el momento que el contenedor de los procesos BPEL publique los servicios.

Teniendo en cuenta la configuración predefinida (*véase 5.5.1*), la petición de búsqueda en el cliente es síncrona y capturando excepciones. Esto hace que sólo tenga que definirse una interfaz para petición y respuesta: la respuesta del proceso se da como salida de la petición (sincronía).

Se declararon dos excepciones: Cliente No Autorizado y Operación Fallida. La primera se lanzará cuando la operación de búsqueda la intente realizar un cliente que no esté debidamente registrado en la agencia de viajes. Y la segunda es una excepción general lanzada en cualquier otro caso de fallo de la operación, con el fin de que el usuario sea informado de la situación.

La estructura resultante de las definiciones se muestra gráficamente en la figura 44. La descripción WSDL correspondiente aparece en la tabla 45.

Figura 44. Estructura WSDL del Cliente: Búsqueda

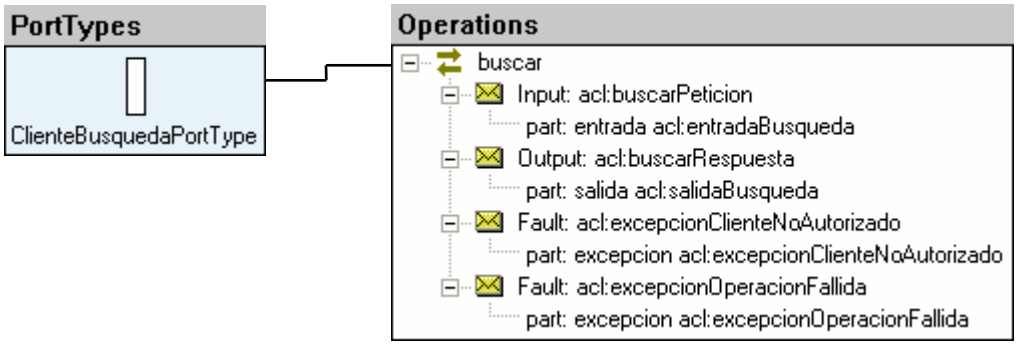


Tabla 45. Interfaz WSDL del Cliente: Búsqueda

Elemento	Descripción del Contenido
Types	Contiene un esquema local con los elementos usados en los mensajes, importando tipos de datos definidos en los esquemas. - ***acl:entradaBusqueda***, elemento de tipo cls:entradaBusqType con los criterios e información de la búsqueda de planes de viaje. - ***acl:salidaBusqueda***, elemento de tipo cls:salidaBusqType con los resultados de la búsqueda o posible reserva de planes de viaje. - ***acl:excepcionClienteNoAutorizado***, elemento con información del fallo cliente sin autorización: cls:excepcionClienteNoAutorizadoType. - ***acl:excepcionOperacionFallida***, elemento con la información del fallo operación no exitosa: cls:excepcionOperacionFallidaType.
Message	Define los mensajes a transmitir con el cliente en la solicitud de tipo petición/respuesta y las excepciones de la operación buscar: - ***buscarPeticion***: mensaje de la petición de la operación *buscar*, contiene cómo única parte, el elemento entradaBusqueda. - ***buscarRespuesta***: mensaje de respuesta de la operación *buscar*, contiene cómo única parte, el elemento salidaBusqueda. - ***excepcionClienteNoAutorizado***: mensaje de información de error cuando el cliente no ha sido autorizado, para todas las operaciones, contiene cómo única parte el elemento excepcionClienteNoAutorizado. - ***excepcionOperacionFallida***: mensaje de información de error cuando alguna operación no fue exitosa, contiene cómo única parte, el elemento excepcionOperacionFallida.
PortType	Define la interfaz **ClienteBusquedaPortType** con la operación: - *buscar*, que contiene: entrada: mensaje buscarPeticion salida: mensaje buscarRespuesta fallo “Cliente No Autorizado”: mensaje excepcionClienteNoAutorizado fallo “Operación Fallida”: mensaje excepcionOperacionFallida
Binding	Sin definir por el momento.
Service	Sin definir por el momento.

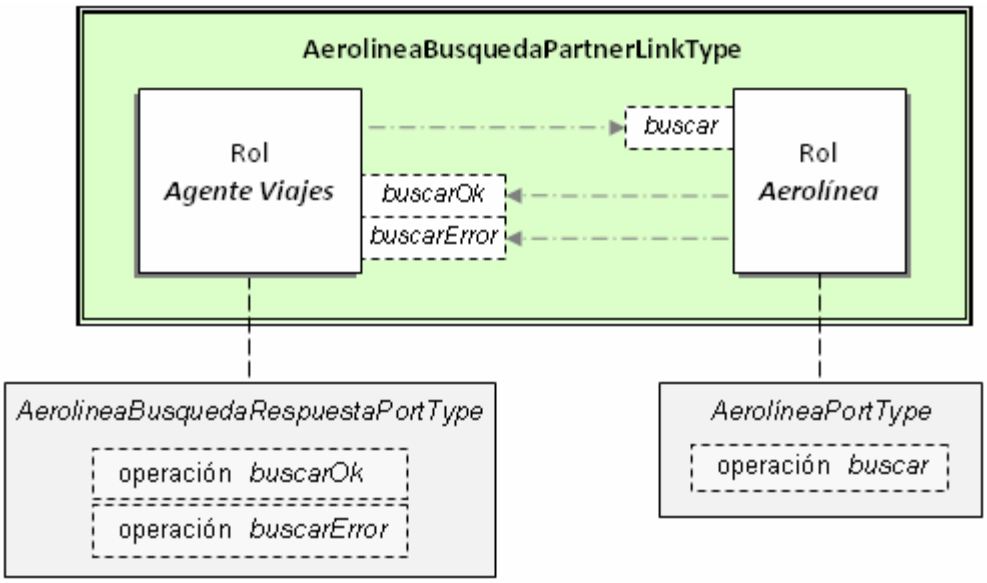
- ***Participantes del Proceso y sus Vínculos***

Para definir el proceso de negocio primero se deben establecer los participantes del mismo y sus relaciones, para esto se utilizan los partnerLinkTypes, pertenecientes a la extensión del lenguaje WSDL, introducida por BPEL (*véase 3.3.4*).

Relación entre Agencia de Viajes y Aerolínea

La primera relación que se establece es entre la agencia de viajes y la aerolínea. La forma de definirla es relacionando los portTypes definidos para la operación búsqueda, asignando un rol a cada uno de los lados de la transacción. Si se recuerda, la operación de búsqueda entre la agencia de viajes y la aerolínea se estableció mediante una operación asíncrona, en la cual la agencia invoca la operación buscar de la aerolínea definida en el portType AerolineaPortType, y para contestar y dar resultado al proceso la aerolínea utiliza las operaciones buscarOk o buscarError que expone la agencia de viajes mediante el portType AerolineaRespuestaBusquedaPortType.

Figura 45. Representación Visual de AerolineaBusquedaPartnerLinkType



Lo anterior se aprecia visualmente en la figura 45, nombrando el vínculo (o partnerLinkType) como *AerolineaBusquedaParnerLinkType*. La definición real se realiza mediante un documento WSDL aparte, por cada participante, con el fin de dar orden y facilitar revisiones y posibles modificaciones posteriores. En la figura 46 se aprecia la definición WSDL del partnerLinkType de búsqueda entre el socio Aerolínea y la Agencia de Viajes.

Figura 46. Definición de PartnerLinkTypes de Aerolínea: Búsqueda

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="aerolineaPartnerLinkType"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
                  xmlns:aer="http://aerolinea.com/servicios/agenciaviajes"
                  xmlns:aae="http://agenciaviajes.com/servicios/aerolinea"
                  targetNamespace="http://agenciaviajes.com/servicios/aerolinea" >

  <wsdl:import namespace="http://aerolinea.com/servicios/agenciaviajes"
               location="../servicios/aerolinea.wsdl" />

  <wsdl:import namespace="http://agenciaviajes.com/servicios/aerolinea"
               location="../servicios/aerolinearespuesta.wsdl" />

  <plnk:partnerLinkType name="aerolineaBusquedaPartnerLinkType">
    <plnk:role name="aerolinea">
      <plnk:portType name="aer:AerolineaPortType" />
    </plnk:role>
    <plnk:role name="agenteviajes">
      <plnk:portType name="aae:AerolineaRespuestaBusquedaPortType" />
    </plnk:role>
  </plnk:partnerLinkType>

</wsdl:definitions>
```

El código, similar al de las definiciones de interfaces, consta de un encabezado donde se especifican los espacios de nombres, resaltando los que corresponden a la aerolínea *aer* y a la parte de aerolínea en la agencia *aae*. Luego se realiza la importación de las definiciones WSDL de la aerolínea (*aerolinea.wsdl* y *aerolinearespuesta.wsdl*) y las asigna a su correspondiente espacio de nombres.

Finalmente se aprecia la definición de *aerolineaBusquedaPartnerLinkType*, el partnerLinkType que relaciona los dos actores *aerolínea* y *agenteviajes*, nombrados así mediante el rol, y a cada uno le asigna un portType: AerolineaPortType y AerolineaRespuestaBusquedaPortType, respectivamente.

Relación entre Agencia de Viajes y los Otros Socios

Las relaciones establecidas entre la agencia de viajes y los otros dos socios del escenario planteado, presentan una definición similar en estructura, ya que la operación buscar es análoga en los tres socios. Los cambios son sólo en los nombres y tipos usados, de acuerdo a cada interfaz definida. A continuación se presenta un resumen de las relaciones especificadas.

Tabla 46. Resumen Vínculos entre Socios y Agencia, para la Búsqueda

Socio	Cambios
Cadena Hotelera	PartnerLinkType *CadenaHoteleraBusquedaPartnerLinkType* - Rol *cadenahotelera*, con portType *htl:CadenaHoteleraPortType* definida en la interfaz *cadenahotelera.wsdl* - Rol *agenteviajes* , con portType *aht:CadenaHoteleraRespuestaBusquedaPortType* definida en la interfaz *cadenahotelerarespuesta.wsdl*
Alquiler de Vehículos	PartnerLinkType *AlquilerVehiculosBusquedaPartnerLinkType* - Rol *alquilervehiculos*, con portType *vhc:AlquilerVehiculosPortType* definida en la interfaz *alquilervehiculos.wsdl* - Rol *agenteviajes* , con portType *avh:AlquilerVehiculosRespuestaBusquedaPortType* definida en la interfaz *alquilervehiculosrespuesta.wsdl*

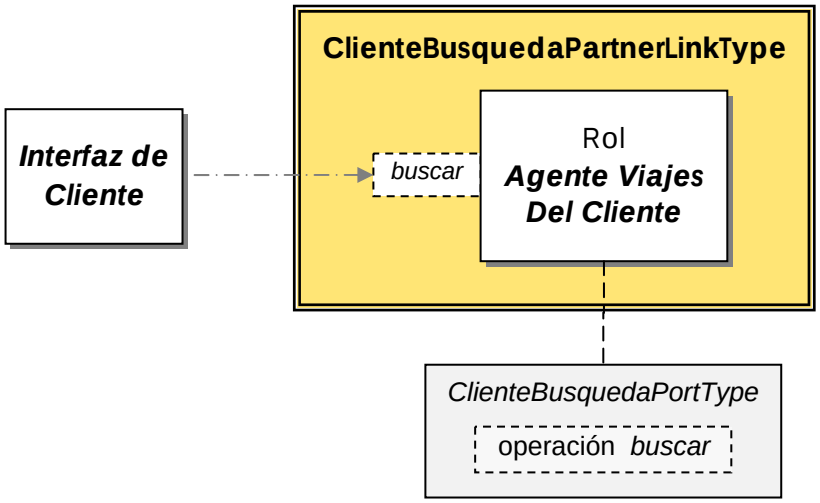
Relación entre Agencia de Viajes y la Interfaz del Cliente

También debe considerarse el actor que representa al cliente en la agencia de viajes: la interfaz de usuario. El cliente realiza la búsqueda a través de este ente

software y por lo tanto interactúa con el sistema de la agencia. Sin embargo, la relación no es bilateral como en el caso de los socios. La interfaz del cliente invoca la operación buscar, definida como síncrona, y por tanto recibe como respuesta de la invocación los resultados de la búsqueda.

La agencia no tiene forma de interactuar con el cliente por sí sola, sólo responde a las solicitudes que él le hace, por tanto se debe definir una relación con sólo 1 rol, el que desempeña la agencia de viajes hacia el cliente. En la figura 47 se aprecia gráficamente este vínculo.

Figura 47. Representación Visual de ClienteBusquedaPartnerLinkType



El código WSDL que define esta relación tiene estructura similar al mostrado para las relaciones del socio aerolínea. La variación se presenta en que sólo define un rol, tal como se mencionó anteriormente. La figura 48 muestra el código donde se define el partnerLinkType del Cliente para la operación búsqueda.

Figura 48. Definición de PartnerLinkTypes de Cliente: Búsqueda

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="clientePartnerLinkType"    ...
                  xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
                  xmlns:acl="http://aerolinea.com/servicios/cliente"
                  targetNamespace="http://agenciaviajes.com/servicios/cliente" >

  <wsdl:import namespace="http://agenciaviajes.com/servicios/cliente"
               location="../servicios/cliente.wsdl" />

  <plnk:partnerLinkType name="clienteBusquedaPartnerLinkType">
    <plnk:role name="agenteviajesDelCliente">
      <plnk:portType name="acl:ClienteBusquedaPortType" />
    </plnk:role>
  </plnk:partnerLinkType>

</wsdl:definitions>
```

Vínculos a nivel del Proceso BPEL

A nivel interno del proceso de negocio de la búsqueda es necesario definir los vínculos entre actores usando los *partnerLinks* (*véase 3.3.5*) usando los vínculos predefinidos (partnerLinkTypes). Aquí se asigna el rol del proceso (mediante la propiedad *myRole*) y el rol del actor (mediante la propiedad *partnerRole*), los roles se toman del partnerLinkType anexo. A continuación se enuncian las declaraciones realizadas para el proceso de búsqueda.

Tabla 47. Declaración de PartnerLinks para el Proceso de Búsqueda

PartnerLink	Propiedades
Aerolínea	- partnerLinkType: *aae:aerolineaBusquedaPartnerLinkType* - myRole: *agenteviajes* - partnerRole: *aerolinea*
Cadena Hotelera	- partnerLinkType: *aht:cadenahoteleraBusquedaPartnerLinkType* - myRole: *agenteviajes* - partnerRole: *cadenahotelera*
Alquiler Vehiculos	- partnerLinkType: *avh:alquilervehiculosBusquedaPartnerLinkType* - myRole: *agenteviajes* - partnerRole: *alquilervehiculos*
Cliente	- partnerLinkType: *acl:clienteBusquedaPartnerLinkType* - myRole: *agenteviajesDelCliente*

Es necesario destacar que en el partnerLink del cliente, no se define el *partnerRole*, debido a que es una operación en un solo sentido, y el proceso actúa como el agente de viajes del cliente (*myRole*), ya que es aquel que el cliente puede acceder.

- ***Implementación del Proceso BPEL de Búsqueda***

La implementación efectiva de los procesos en la agencia de viajes se realiza con BPEL y utilizando las interfaces con los socios especificadas anteriormente, siguiendo una lógica de ejecución de tareas representadas como servicios Web.

Partiendo del análisis y diseño realizados anteriormente (*véase 5.4.3.1*), la lógica del proceso involucra tareas de validación inicial de datos, obtención de la información de los socios que la agencia posee (previamente definidos y almacenados internamente), la interacción con cada socio buscando los servicios especificados según los criterios particulares, procesamiento de resultados, reserva del mejor plan encontrado (en caso que el cliente lo requiera y que se hayan encontrado todos los servicios deseados), y finalmente envío de resultados y procesamiento de posibles errores durante todo el proceso.

Con el fin de plasmar los procesos de negocio BPEL implementados, se usarán diagramas en Notación de Modelado de Procesos de Negocio (BPMN) [27], los cuales permiten representar gráficamente el flujo de los procesos de negocio mediante íconos y elementos visuales que abarcan todo el lenguaje BPEL, siguiendo patrones y modelos muy similares a los diagrama de flujo[29] y los diagramas de actividades UML. En la especificación de BPMN, se encuentra la referencia de cada elemento y su forma de uso en un flujo de actividades. Se mencionarán algunos aspectos importantes para un entendimiento rápido:

[29] Los diagramas de flujo son una forma de representar algoritmos, usado ampliamente en la programación. Más información en http://es.wikipedia.org/wiki/Diagrama_de_flujo

- Existen tareas marcadas con el icono  que corresponden aquellas definidas como servicios Web.
- El ícono  permite especificar las excepciones que son capturadas y manejadas, a nivel de tarea, de proceso o de captura de eventos.
- Para especificar tiempos máximos de duración de un proceso o tarea, o espera de evento, se define una excepción de tiempo excedido con el ícono  .
- Es posible especificar subprocesos, los cuales se definen en otro diagrama para reducir complejidad visual. Estos están marcados con el ícono  .
- Los procesos pueden reutilizarse en otros diagramas, es decir hacer referencia para indicar que se hace un llamado a ellos; se identifican con el ícono  .
- El ícono  se utiliza para determinar divergencia del flujo del proceso, determinado por eventos.

El proceso de negocio planteado e implementado para la búsqueda en la agencia de viajes se muestra en la figura 49. Allí se aprecia el flujo de actividades descrito en el análisis, especificando información de entrada o salida de cada proceso y anotaciones importantes que facilitan el entendimiento del diagrama. Por otro lado, el código BPEL correspondiente se define en un archivo llamado *busqueda.bpel* disponible en el código fuente del proyecto.

En el gráfico se aprecian dos tipos de subprocesos relacionados con la invocación de la búsqueda y la recepción de resultados en cada uno de los tipos de socios. Estos dos tipos de subprocesos presentan una estructura similar entre sí, por tal motivo se muestran en diagramas aparte; esto facilita la comprensión visual, aunque en la implementación en código se especifican separadamente cada uno y dentro del mismo proceso general de la búsqueda.

Figura 49. Proceso de Negocio de Búsqueda en la Agencia

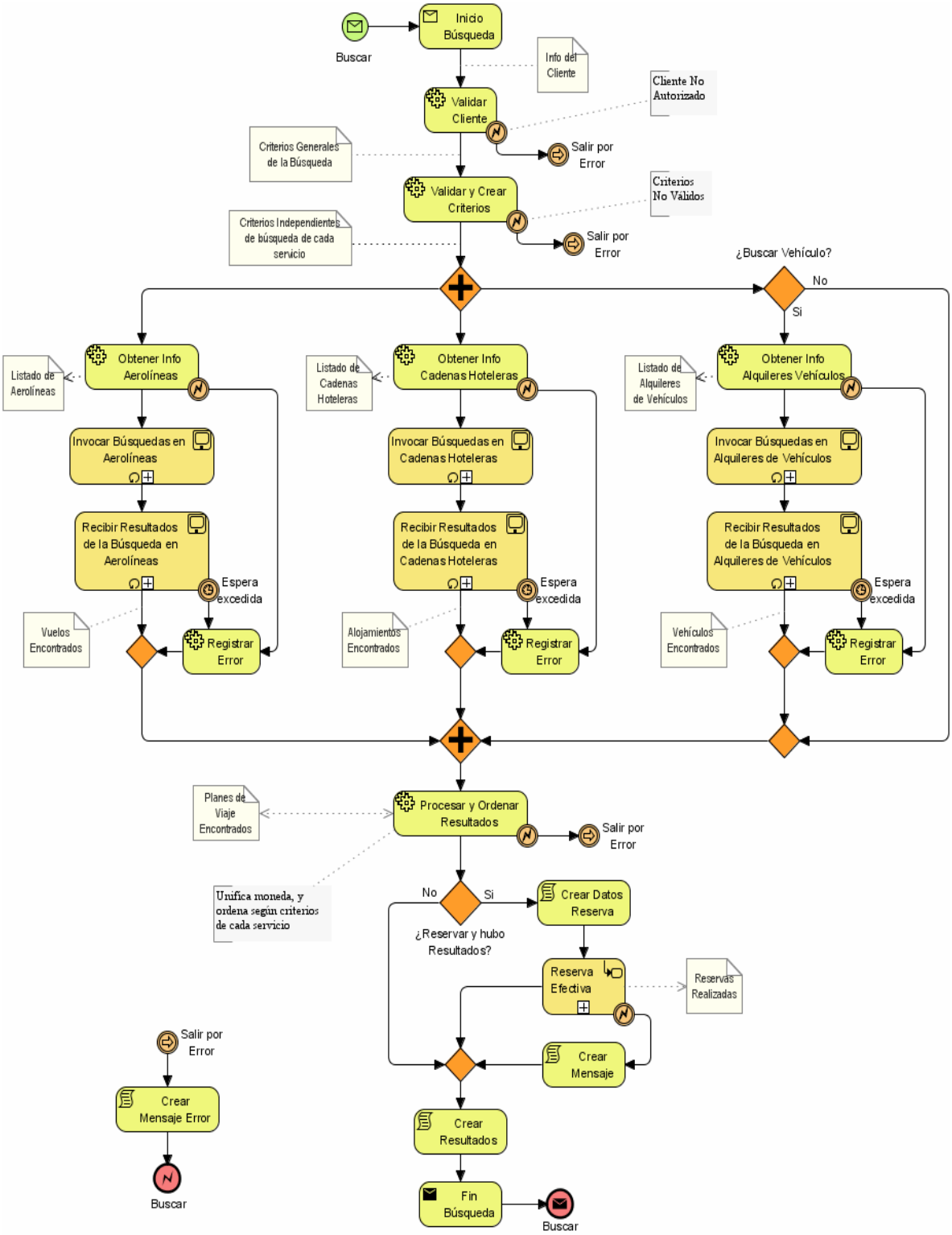
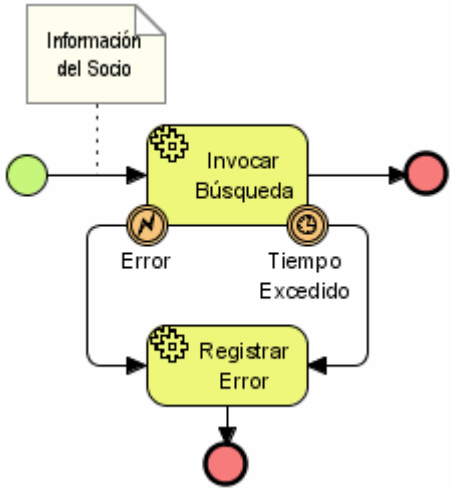


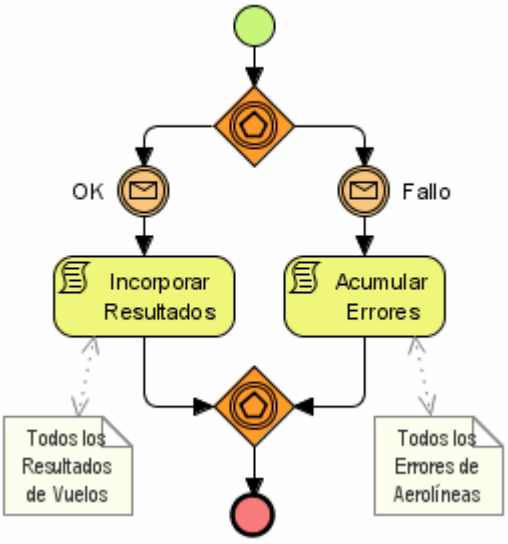
Figura 50. Subproceso de Invocación de Búsqueda en Socio



Este proceso de invocación se ejecuta de forma similar aunque con diferentes especificaciones y datos de entrada para cada tipo de socio. La tarea *invocar búsqueda* corresponde a la invocación del servicio Web del socio, es decir la operación *buscar* definida en las interfaces de socios. Para saber qué socio se invoca, se utiliza la información del socio como entrada, el cual se ha obtenido previamente a las invocaciones mediante una tarea local. La ejecución se hace en un bucle por cada socio registrado tal como se muestra en el diagrama general.

En la figura 51 se aprecia el subproceso que muestra la recepción de los resultados de la búsqueda por cada socio (ilustrado con el socio aerolíneas). Allí se aprecia que el proceso hace una espera por la recepción de cualquiera de los dos mensajes de respuesta determinados: Ok y Error, y de acuerdo a cada uno procesa los resultados o acumula errores. Además este subproceso tiene definido anexo una excepción de tiempo de espera excedido (visible en el diagrama general), usado para controlar errores de comunicación o internos de los socios que puedan causar que la respuesta de la búsqueda no se envíe de vuelta a la agencia.

Figura 51. Subproceso de Recepción de Resultado de Búsqueda del Socio



Además, se debe mencionar que el subproceso llamado Reserva Efectiva hace referencia a uno definido posteriormente en la fase 2, usado para realizar la reserva automática en caso que la búsqueda sea exitosa y el usuario así requiera.

- ***Especificación del Servicio Web Local***

Las tareas complejas del proceso BPEL, que no constan solamente de scripts de manipulación de datos XML mediante XPath, deben especificarse mediante operaciones de un servicio Web local, con todo lo que esto conlleva. Por tanto se especificará un nuevo servicio Web a nivel local en la agencia.

Interfaz WSDL Local

Cada operación será síncrona, definiendo mensajes de entrada, de salida, y de excepción. Se debe tener en cuenta que el parámetro de entrada del proceso es un elemento de tipo *cls:entradaBusqType*, y la respuesta debe retornar como resultado un elemento de tipo *cls:salidaBusqType*. La información de entrada es el punto de partida del procesamiento XML, el cual debe culminar en la información de salida.

La interfaz WSDL define el enlace *AgenciaViajesLocalPortType*, con las operaciones necesarias, especificadas de acuerdo al resumen de la tabla 48. El detalle de cada una se omite por su extensión. Además de esto, cada operación posee una excepción llamada Fallo General, definida como un elemento de tipo *ags:excepcionGeneralType*. Esta excepción es usada para capturar todo error inesperado y por tanto no haya sido tenido en cuenta en otra excepción.

Tabla 48. Especificación Tareas Locales, Proceso Búsqueda

Especificación de AgenciaViajesLocalPortType para Búsqueda
Validar Cliente *(validarCliente)* *Descripción*: es la tarea inicial del proceso de búsqueda, y de todos los otros. Su objetivo es validar internamente si los datos del cliente especificado en los parámetros de entrada del proceso corresponden a un cliente registrado en la agencia. *Entrada*: los datos del cliente que trae el parámetro de entrada del proceso, es decir el campo *validacionCliente* de tipo *cls:clienteValidacionType*. *Salida*: un elemento de tipo simple *gls:unidadMonetaria* que corresponde a la unidad monetaria preferida del cliente, para usarla posteriormente. *Excepciones*: - Cliente no válido, cuando el cliente no está registrado; definida con un mensaje de tipo *ags:excepcionClienteNoValidoType.*
Validar y Crear Criterios *(validarCriteriosBusqueda)* *Descripción*: se encarga de validar y normalizar los criterios de entrada del proceso, y formar los criterios de búsqueda independientes de cada servicio. *Entrada*: los mismos criterios de búsqueda del proceso general, es decir un elemento de tipo *cls:entradaBusqType*. *Salida*: un elemento de tipo *ags:infoBusquedasType* que contiene los criterios de búsqueda normalizados y separados por tipo de servicio a buscar. *Excepciones*: - Lugar no encontrado, cuando el sitio de origen o de destino de la búsqueda no corresponde a un sitio identificado; definida con un mensaje de tipo *ags:excepcionLugarNoEncontradoType.* - Criterios no válidos, cuando hay inconsistencias en los criterios, por ejemplo fechas incorrectas. Definida con un mensaje de tipo *ags:excepcionCriteriosNoValidosType*.
Obtener Info Socios *(obtenerInfoSociosBusqueda)* *Descripción*: se encarga de consultar el listado de la información de cada tipo de socio conocido por la agencia. *Entrada*: la clase de socio, definido como un elemento del tipo simple *ags:tiposSocios*. *Salida*: un elemento de tipo *ags:infoSociosBusquedasType* que contiene un listado con la información de los socios a consultar, incluyendo la información de enlace dinámico.

Procesar y Ordenar Resultados *(procesarResultadosBusqueda)* *Descripción*: se encarga de tomar los resultados de las búsqueda de cada servicio y procesarlas unificando la unidad monetaria según la preferida del usuario, y ordenarlas de a cuerdo al criterio de entrada. *Entrada*: los resultados de las búsquedas de cada servicio, los criterios de orden y la moneda del cliente, definida como un elemento del tipo *ags:salidasBusquedasType*. *Salida*: un elemento de tipo *ags:salidasBusquedasProcesadasType* que posee los resultados de la búsqueda ya procesados y listos para retornarse.
Registrar Error *(registrarError)* *Descripción*: es una tarea general encargada de registrar internamente los errores con el fin de identificar posibles problemas y tener soporte para tomar soluciones. *Entrada*: un elemento del tipo *ags:entradaRegistrarErrorType* que posee información relevante sobre el error ocurrido.

Tipos de Datos XML

Los tipos de datos XML del esquema de la agencia de viajes local, identificado con el prefijo *ags* fueron definidos analizando el flujo de información a lo largo de las tareas, observando los datos de entrada y salida de cada una; con el objetivo que las salidas de una tarea pudiesen ser utilizadas, sin mayores cambios, como entrada de la tarea siguiente. A continuación se muestra el diagrama de clases con los tipos de datos definidos y luego se explica cada una.

Figura 52. Diagrama de Clases – Agencia Local, Búsqueda (1/2)

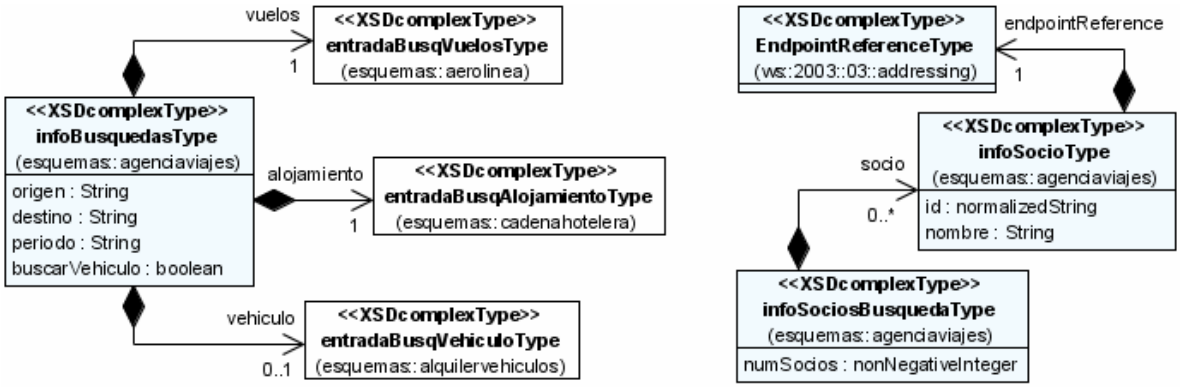


Figura 53. Diagrama de Clases – Agencia Local, Búsqueda (2/2)

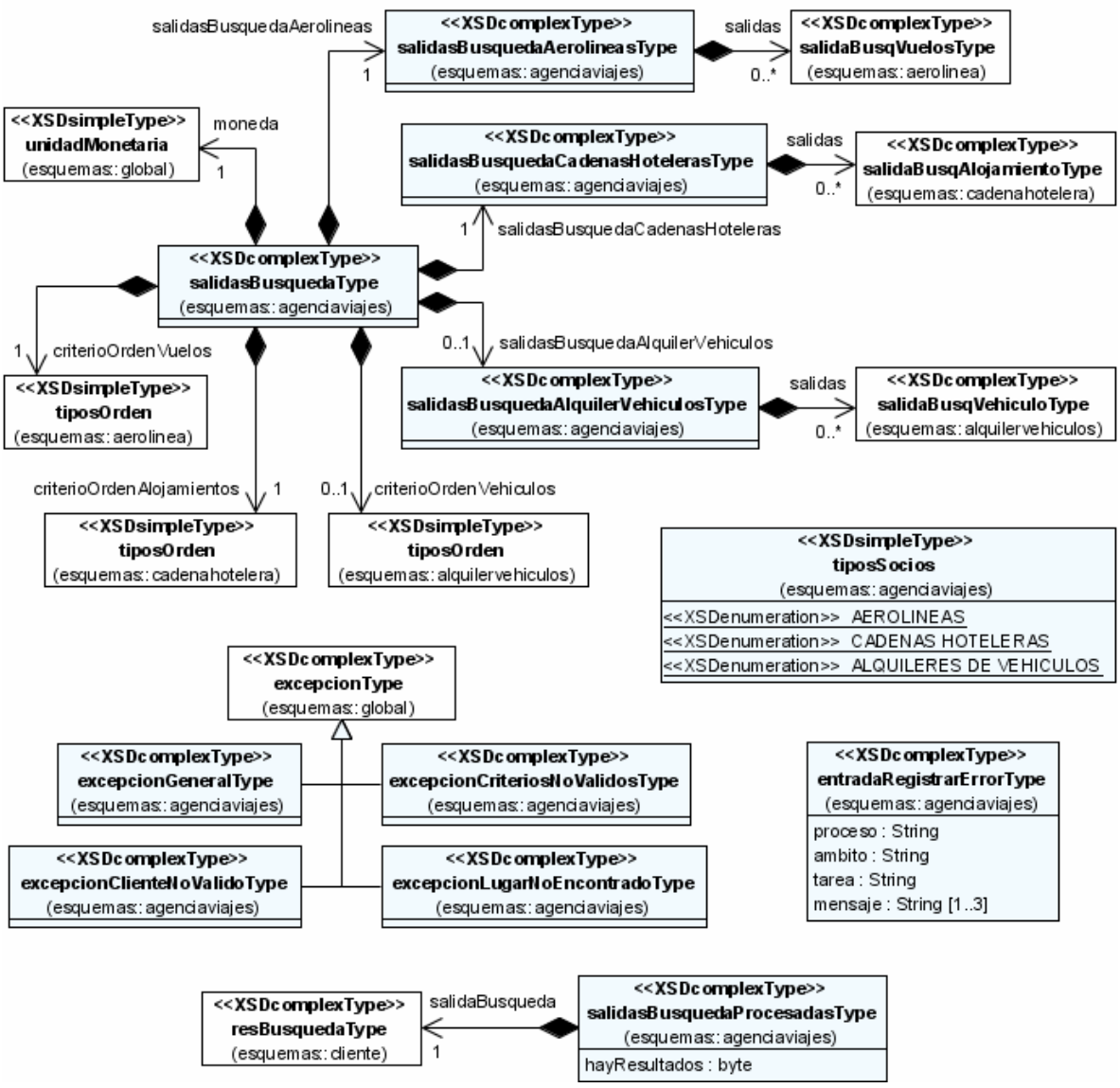


Tabla 49. Descripción de Clases – Agencia Local, Búsqueda

Clase y Paquete	Descripción
InfoBusquedasType *esquemas.agenciaviajes*	Tipo complejo con los criterios de búsqueda procesados, expresados en términos de la entrada de la búsqueda para cada tipo de socio.
EndpointReferenceType *ws.2003.03.addressing*	Tipo complejo que agrupa la información que le permite al motor enlazar dinámicamente el partnerLink.
InfoSocioType *esquemas.agenciaviajes*	Tipo complejo con los datos de cada socio guardado en la agencia, incluida la forma de accederlo (endpointReference).
InfoSocioBusquedaType *esquemas.agenciaviajes*	Tipo complejo que encapsula el listado de socios que se usarán para la búsqueda.
SalidasBusqueda_ AerolineasType *esquemas.agenciaviajes*	Tipo complejo que posee el conjunto de resultados obtenidos como respuesta a las peticiones de búsqueda a las aerolíneas.
SalidasBusqueda_ CadenasHotelerasType *esquemas.agenciaviajes*	Tipo complejo que guarda el conjunto de resultados obtenidos como respuesta a las peticiones de búsqueda a las cadenas hoteleras.
SalidasBusqueda_ AlquilerVehiculosType *esquemas.agenciaviajes*	Tipo complejo que guarda el conjunto de resultados obtenidos como respuesta a las peticiones de búsqueda a los alquileres de vehículos.
SalidasBusquedaType *esquemas.agenciaviajes*	Tipo complejo que encapsula los resultados de todas las búsquedas, para su procesamiento y ordenación.
SalidasBusqueda_ ProcesadasType *esquemas.agenciaviajes*	Tipo complejo que contiene los resultados procesados y listos para ser devueltos al cliente.
TiposSocios *esquemas.agenciaviajes*	Tipo simple String con los tipos de socios de la agencia, enumeraciones: AEROLINEAS, CADEHAS HOTELERAS Y ALQUILERES DE VEHICULOS.
EntradaRegistrarErrorType *esquemas.agenciaviajes*	Tipo complejo que encapsula la información de los errores para su registro y control.
Excepciones *esquemas.agenciaviajes*	Tipos complejos que extienden de excepcionType, y son usados para los mensajes de excepción de las operaciones. *ExcepcionGeneralType, ExcepcionCriteriosNoVálidosType ExcepcionClienteNoVálidoType y ExcepcionLugarNoEncontradoType*

Vínculos

Este nuevo servicio de la Agencia de Viajes Local toma parte como otro actor del proceso de negocios y por lo tanto necesita definición WSDL de PartnerLinkTypes y declaración PartnerLink en el proceso. El vínculo es unilateral ya que el proceso realiza las llamadas y espera por las respuestas de forma síncrona, y el actor agencia de viajes local no tiene forma de contactar al proceso por sí solo.

El partnerLinkType *agenciaviajeslocalPartnerLinkType* declara un rol llamado *agenteviajeslocal* asignado al portType *age:AgenciaViajesLocalPortType* que corresponde a la interfaz local. Por otro lado a nivel del proceso, se define un nuevo partnerLink *agenciaviajeslocal* tomando como referencia en partnerLinkType recién creado y estableciendo al agentedeviajeslocal como el *partnerRole*. Esto define que el actor hace de tercero en la relación, el proceso por su parte no tiene rol definido y por lo tanto no define *myRole*.

Implementación de la Lógica

Las tareas mencionadas se implementaron usando lenguaje de programación Java y aprovechando el hecho que se conocen los datos de entrada y la estructura de los de salida. La lógica fue implementada de manera sencilla aunque fue necesaria la definición de una estructura de datos para almacenamiento interno, que diera persistencia a los datos de clientes, socios y reservas (consideradas en la fase 2). En el Anexo C se visualiza un diagrama con la definición de datos locales a la agencia. Los detalles de la implementación se omiten en el presente documento, aunque pueden apreciarse en el código fuente del desarrollo.

Subproceso de Reserva Efectiva

El subproceso BPEL encargado de la reserva automática luego del éxito de la búsqueda, se definirá e implementará en la fase 2. Cabe mencionar que para que el mismo sea accedido, debe poseer una interfaz y ser actor del proceso como tal, por tanto goza de un tratamiento similar al de los servicios Web locales.

- ***Despliegue y Publicación del Proceso BPEL***

Una vez realizado el proceso de la búsqueda en lenguaje BPEL, se debe realizar la publicación en el motor de ActiveBPEL.

Archivo de Descripción de Despliegue del Proceso

Las características de publicación del proceso se declaran en un archivo *descriptor de despliegue del proceso*. Allí se especifica la manera como cada partnerLink descrito en el proceso es interpretado y enlazado físicamente por el motor de ActiveBPEL. A continuación se aprecia parte del código del archivo de descripción del proceso de búsqueda.

Figura 54. Fragmento Archivo de Despliegue, Proceso BPEL Búsqueda

```
<?xml version="1.0" encoding="UTF-8"?>
<process  xmlns="http://schemas.active-endpoints.com/pdd/2005/09/pdd.xsd"
            xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing" ...  >

  <partnerLinks>
    <partnerLink name="aerolinea">
      <partnerRole endpointReference="dynamic" invokeHandler="default:Address"/>
      <myRole allowedRoles="" binding="MSG" service="AerolineaBusquedaService"/>
    </partnerLink>

    ...
    <partnerLink name="cliente">
      <myRole allowedRoles="" binding="MSG" service="ClienteBusquedaService"/>
    </partnerLink>

    <partnerLink name="agenciaviajeslocal">
      <partnerRole endpointReference="static"
                        invokeHandler="java:agenciaviajes.AgenciaViajesLocalInvokeHandler">
        <wsa:EndpointReference xmlns:age="http://agenciaviajes.com/servicios/agenciaviajes">
          <wsa:Address>Address</wsa:Address>
          <wsa:ServiceName PortName="AgenciaViajesLocal">
            age:AgenciaViajesLocal
          </wsa:ServiceName>
        </wsa:EndpointReference>
      </partnerRole>
    </partnerLink>

    ...
  </partnerLinks>
</process>
```

Como se especificó en el proceso, cada partnerLink puede poseer un actor con rol de tercero (partnerRol) y el rol del proceso (myRole). Para los primeros debe establecerse la forma como se acceden, mientras que para el rol del proceso, se especifica de qué forma el motor publica la interfaz. Los roles que el proceso desempeña son aquellos que van a quedar expuestos al publicar el mismo.
En el caso del partnerLink *aerolínea*, se establece para el partnerRole una referencia *dinámica* (*endpointReference="dynamic"*) con manejador de tipo *Address*, esto con el fin de que en enlace se haga mediante asignación de su dirección en el proceso, permitiendo acceder a cada socio de tipo aerolínea con el mismo vínculo, variando los datos de acceso; por otro lado, al rol del proceso se le define que será expuesto con un servicio de nombre *AerolineaBusquedaService* con enlace tipo MSG (correspondiente al tipo de enlace SOAP), esto significa que la interfaz que la aerolínea accede para devolver los resultados de las búsquedas es expuesta por el motor de ActiveBPEL en el momento del despliegue del proceso con un servicio con el nombre definido.

Para el caso del partnerLink *agenciaviajeslocal* se especifica un método de acceso estático (sin variaciones) y con un manejador que hace referencia a una clase java llamada *AgenciaViajesLocalInvokeHandler*, esto es una característica de ActiveBPEL que permite definir esta clase intermediara como punto de entrada y ejecución de la lógica de las tareas.

Publicación y Puesta en Marcha del Proceso
El motor de ActiveBPEL toma la información del descriptor de despliegue del proceso y realiza la publicación del proceso mediante la exposición de los servicios Web definidos en sus roles. La publicación del proceso de búsqueda en ActiveBPEL, crea y expone cuatro servicios Web: *AerolineaBusquedaService*, *CadenaHoteleraBusquedaService*, *AlquilerVehiculosBusquedaService*, y *ClienteBusquedaService*. Estos servicios son los puntos de entrada al proceso de búsqueda de los actores externos: los socios utilizan los tres primeros servicios

para invocar las operaciones de respuesta de las búsquedas; por su parte, la interfaz del cliente utiliza el último servicio para invocar acceder a la búsqueda de planes de viaje.

Figura 55. Interfaz WSDL Final Generada, para Búsqueda del Cliente

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="clientePartnerLinkType"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
                  xmlns:acl="http://agenciaviajes.com/servicios/cliente"
                  targetNamespace="http://agenciaviajes.com/servicios/cliente" >

  <wsdl:import namespace="http://agenciaviajes.com/servicios/cliente"
               location="http://localhost/active-bpel/catalog/AgenciaViajes/cliente.wsdl" />

  <wsdl:binding name="ClienteBusquedaServiceBinding"
                type="acl:ClienteBusquedaPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="buscar">
      <soap:operation soapAction="" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="excepcionOperacionFallida">
        <soap:fault name="excepcionOperacionFallida" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="excepcionClienteNoAutorizado">
        <soap:fault name="excepcionClienteNoAutorizado" use="literal" />
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="ClienteBusquedaService">
    <wsdl:port binding="acl:ClienteBusquedaServiceBinding"
               name="ClienteBusquedaServicePort">
      <soap:address location="http://localhost/active-bpel/services/ClienteBusquedaService" />
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

La interfaz WSDL se expone dinámicamente y por tanto sólo puede ser visualizada vía Web. En la figura 55 se muestra el código WSDL del servicio del servicio ClienteBusquedaService.

Para destacar de la interfaz generada, el hecho que el motor la define con base en el partnerLinkType asociado entre el cliente y el proceso (*clientePartnerLinkType*), además sólo se definen el binding y el servicio, ya que las demás partes fueron declaradas previamente en la interfaz WSDL para la búsqueda del cliente (véase el principio de la fase 1), presente en el archivo *cliente.wsdl* que es importado para su reutilización.

Con este paso, el proceso BPEL para la búsqueda de la agencia queda listo para ser invocado y ejecutado por el cliente.

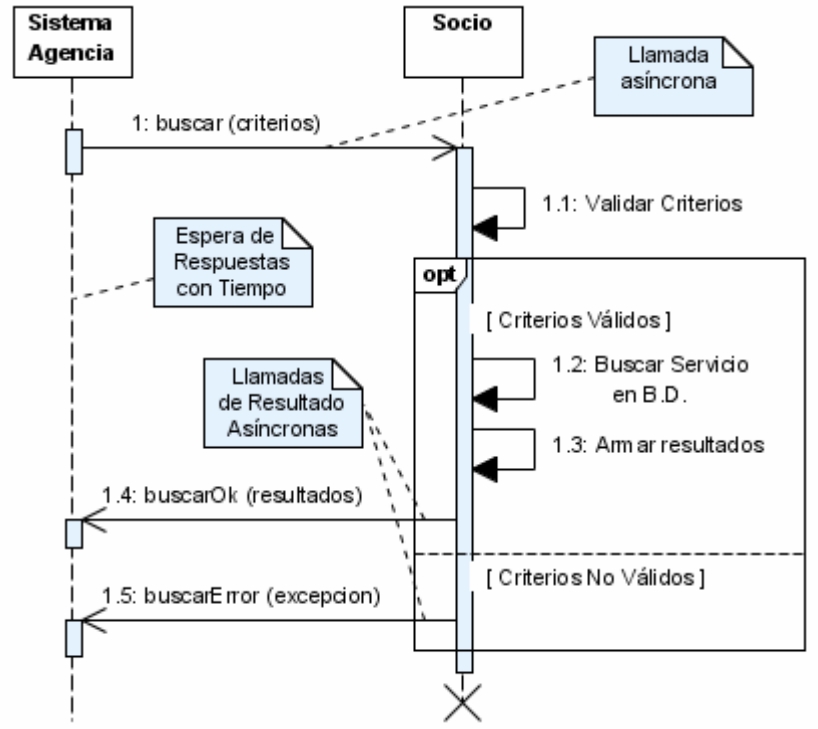
5.5.2.2 Especificación de los Sistemas de los Socios

Cada socio expone como un servicio Web y de forma independiente, las operaciones que implementa, en este caso la búsqueda de los servicios de viaje, teniendo un acuerdo predeterminado de los tipos de datos a intercambiar definidos en esquemas XML y los documentos WSDL de las interfaces (*véase 5.5.2.1*).

Según lo planteado en el análisis preliminar (*véase 5.4.2.2*) y las consideraciones WSDL de la fase de desarrollo preliminar, la funcionalidad de búsqueda de servicios en los socios se expresa mediante la operación ***buscar*** y su implementación varía según cada uno. A continuación se describen las características generales de su lógica, teniendo en cuenta y dejando claro que, gracias al uso de interfaces con servicios Web, la implementación interna es totalmente independiente, y por tanto en la realidad puede realizarse con la amplia gama existente de tecnologías de construcción de aplicaciones informáticas.

En la figura 56 se muestra un diagrama de secuencia que describe en términos generales el funcionamiento interno de los socios para el proceso de búsqueda. Se destaca que, según lo planteado en la fase preliminar, las llamadas se hacen mediante operaciones asíncronas, por lo tanto el sistema de la agencia realiza la llamada al socio respectivo y queda en espera de los mensajes de respuesta.

Figura 56. Diagrama de secuencia, Búsqueda en Socios



El flujo de actividades típico que presenta el funcionamiento comienza con la invocación de la búsqueda al socio, el cual recibe los criterios y realiza la validación de los mismos; si son válidos y coherentes (fechas, duración, etc.) el socio realiza las consultas a su fuente de datos para localizar instancias del servicio solicitado y luego realiza la llamada de respuesta *buscarOk* con los resultados; por otro lado, si sucede algún error o si los criterios son inválidos, el sistema realiza la llamada de respuesta *buscarError* con un mensaje descriptivo del error o excepción ocurrida.

Por otro lado, cada socio debe poseer una estructura de almacenamiento de datos interna, que le permita guardar toda la información de los servicios que presta y las reservas que administra. Estas estructuras se definieron incrementalmente a medida que se implementaban las fases de desarrollo, aunque para facilidad de lectura se muestra el resultado final en el Anexo C.

5.5.2.3 Implementación de la Interfaz del Cliente

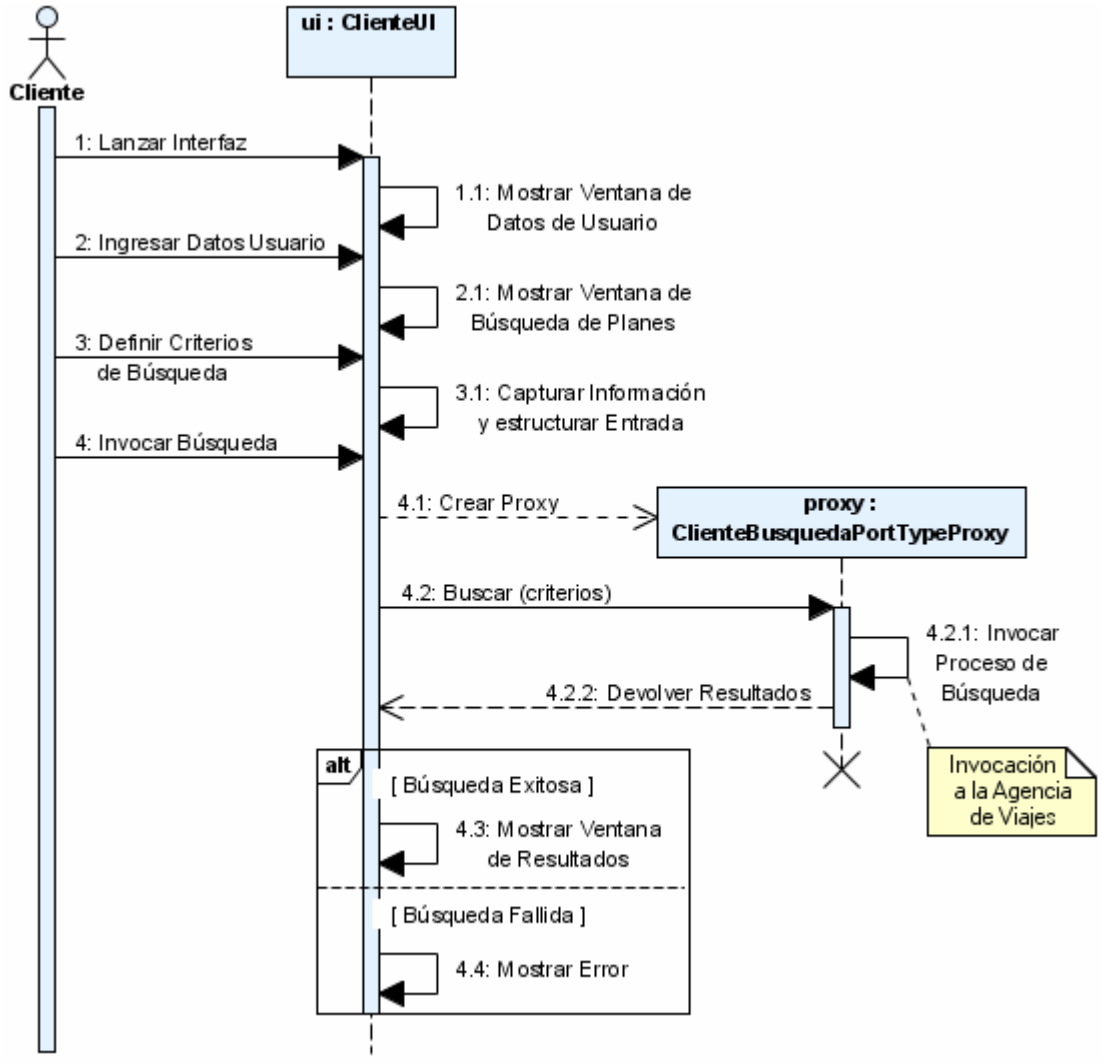
La interfaz del cliente, como intermediario entre el cliente y el sistema de la agencia, debe implementar la captura de los criterios de búsqueda y posteriormente realizar el llamado al proceso de búsqueda de planes de viaje en el sistema de la agencia.

El punto de partida es la interfaz WSDL expuesta por la agencia para la búsqueda hacia el cliente, definida por el servicio *ClienteBusquedaService*, la cual se usa para que mediante las utilidades de AXIS, se generen clases Java que encapsulan el proceso de localización, inicialización, enlace y ejecución del servicio Web seleccionado. Además se genera todo el conjunto de clases correspondientes a los esquemas de datos utilizados en el proceso de búsqueda, relacionado con el cliente. AXIS genera una clase encargada de representar el servicio asociado ante el cliente, proporcionando un punto de acceso a las operaciones definidas en el mismo. En este caso se genera la clase *ClienteBusquedaPortTypeProxy* la cual posee la operación buscar declarada tal cual se especifica en la interfaz.

La lógica de la interfaz de usuario para la búsqueda es sencilla y puede apreciarse en el diagrama de secuencia de la figura 57. El flujo de trabajo presentado en el diagrama es el siguiente: el cliente lanza la interfaz (ClienteUI) la cual responde mostrándole la ventana de datos de usuario donde el cliente ingresa la información de identificación, luego la interfaz presenta la ventana donde se definen todos los criterios de búsqueda de planes de viaje para que el usuario los introduzca; seguido de esto, el usuario realiza la invocación de la búsqueda a la interfaz, la

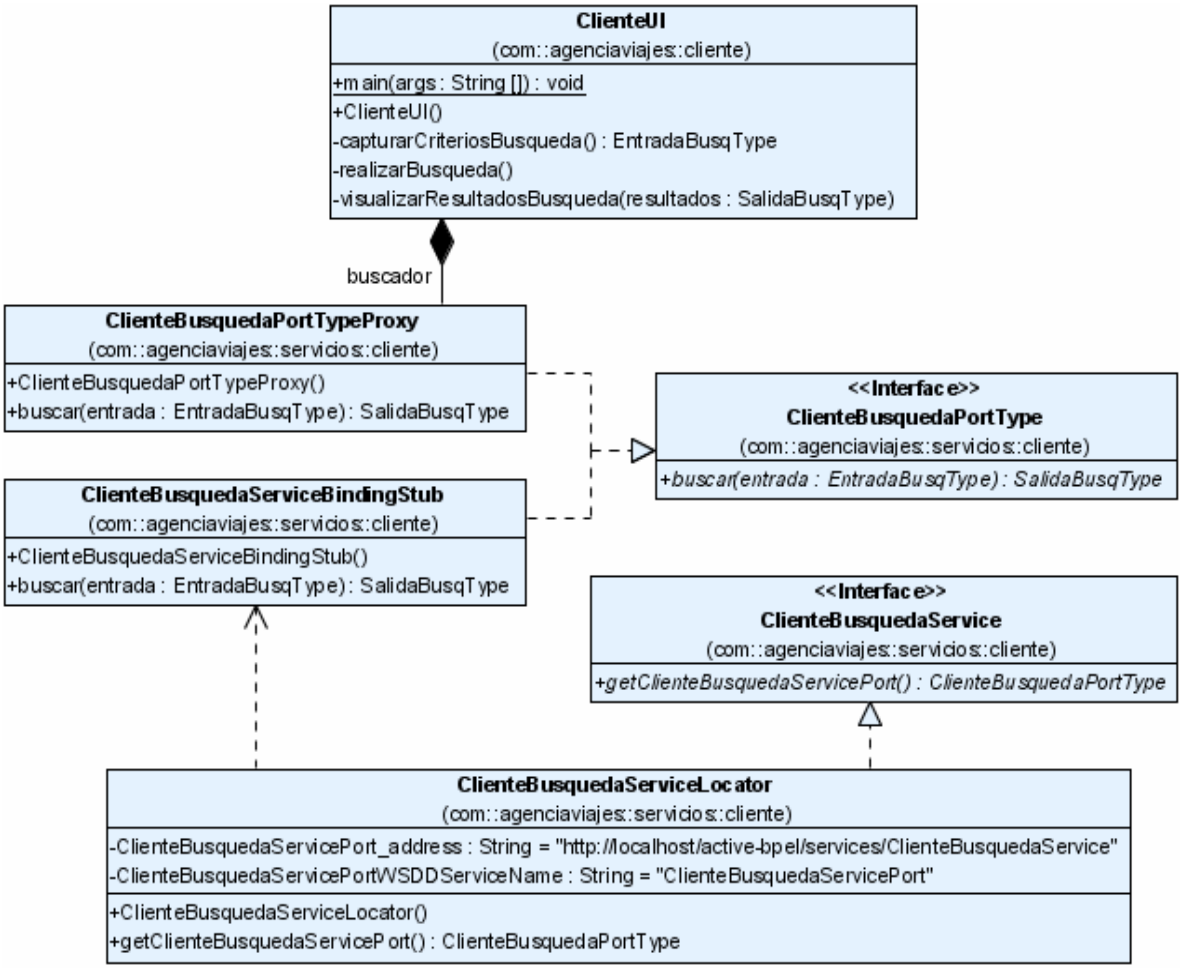
cual inicia el proceso de recolección de los criterios y definición de la entrada del proceso que invocará; después la interfaz crea el representante o *proxy* de la agencia y realiza la invocación de la operación buscar transfiriéndole como parámetros de entrada los criterios tomados; cuando la operación buscar finalice, la interfaz analiza el éxito de la operación, mostrando al usuario una ventana con los resultados si la búsqueda fue exitosa, o informando apropiadamente del error en caso contrario.

Figura 57. Diagrama de Secuencia, Búsqueda en Interfaz de Cliente



En la figura 58 se muestra el diagrama de las clases implementadas para el funcionamiento de la interfaz del cliente de la agencia de viajes. Allí aparecen además de las clases generadas por AXIS, la clase principal de la interfaz, llamada ClienteUI.

Figura 58. Diagrama de Clases – Sistema del Cliente, Búsqueda



La descripción de las clases se aprecia en la tabla 50, aclarando que para simplificar el documento no se describen los constructores de cada clase, y para las clases generadas por AXIS se han omitido atributos, operaciones internas, y clases base que pertenecen a AXIS.

Tabla 50. Descripción de Clases – Sistema del Cliente, Búsqueda

Especificación de Clases
ClienteUI: clase que maneja la interfaz visual con la que el cliente interactúa para realizar las operaciones requeridas. *Métodos*: - main(): método estático para iniciar la aplicación java. - capturarCriteriosBusqueda(): toma de pantalla los criterios definidos por el usuario y los organiza según la estructura que la entrada de la búsqueda requiere. - realizarBusqueda(): realiza el proceso de búsqueda. - visualizarResultadosBusqueda(): muestra en pantalla los resultados de la búsqueda.
ClienteBusquedaPortType: interfaz generada por AXIS con la firma del portType expuesto por la agencia de viajes. *Métodos*: - buscar(), define la operación buscar con sus entradas y salidas.
ClienteBusquedaPortTypeProxy: clase generada por AXIS para ser usada como punto de acceso a las operaciones del servicio. *Métodos*: - buscar(), realiza el llamado interno a la búsqueda de planes de viaje.
ClienteBusquedaServiceBindingStub: clase generada por AXIS que encapsula los procesos internos a bajo nivel, de definición estricta de las operaciones, validación de datos y enlace de las operaciones del servicio, para cumplir con la definición del binding en el servicio. *Métodos*: - buscar(), realiza el procesamiento interno a bajo nivel requerido para la operación.
ClienteBusquedaService: interfaz generada por AXIS para la localización del servicio. *Métodos*: - getClienteBusquedaServicePort(), usado para obtener una instancia del servicio.
ClienteBusquedaServiceLocator: clase generada por AXIS encargada de definir la información de enlace y localización en la red del servicio a instanciar. *Atributos*: - ClienteBusquedaServicePort_address: define la dirección o URL del servicio. - ClienteBusquedaServicePortWSDDServiceName: especifica el nombre del puerto del servicio. *Métodos*: - getClienteBusquedaServicePort(),obtiene un enlace al servicio con la información definida en los atributos.

En el capítulo 6 pueden apreciarse pantallazos de la interfaz de usuario implementada.

5.5.2.4 Pruebas de la Fase de Búsquedas

La ejecución de las pruebas de la fase 1, siguieron los protocolos definidos en el Anexo D, y fueron desarrolladas durante 2 sesiones, por miembros de la comunidad universitaria con conocimiento sobre informática e internet.

Durante la primera sesión de pruebas se recolectaron resultados mayormente exitosos, sin embargo se apreciaron los siguientes errores y problemas del desarrollo:

- La interfaz de usuario no limpia resultados de búsquedas realizadas previamente, así se realicen por otro cliente.
- Los campos de fechas y horas permiten digitar caracteres incorrectos y que generan errores innecesarios al realizar la búsqueda.
- La búsqueda simultánea por varios clientes no pudo realizarse debido a un error de ejecución, relacionado con múltiples procesos en ejecución en el servidor de la agencia.

Los dos primeros errores corresponden a defectos de la interfaz que fueron solucionados fácilmente mediante validación previa a la búsqueda y uso de controles de usuario más especializados para los campos de fecha y hora.

Sin embargo, el error presentado durante la búsqueda simultánea tuvo que ser analizado a nivel del proceso BPEL, y se identificó que el problema es que procesos similares ejecutándose al mismo tiempo deben poseer información que permita identificarlos, es decir información de *correlación* con los actores externos.

A nivel del proceso BPEL de búsqueda se definió un conjunto de correlación llamado *procesoBusquedaCS* con la propiedad *idProceso* la cual posee el id único del proceso en ejecución. Este id debe ser transmitido a los socios en los mensajes de entrada, para que cuando el socio invoque los mensajes de

respuesta (en caso de operación asíncrona), incluya este código y el motor de BPEL pueda identificar a cuál proceso en ejecución está dirigido esta petición.

Los cambios deben realizarse a los tipos de datos involucrados en la petición de búsqueda a los socios y la respuesta de los mismos a la agencia. La figura 59 muestra el diagrama de clases con los tipos de datos nuevos en color azul, los que se modificaron en color verde y aquellos que sufrieron variaciones por herencia en color blanco; además en la tabla 51 se presenta la explicación detallada del mismo.

Figura 59. Diagrama de Clases – Cambios en Búsqueda

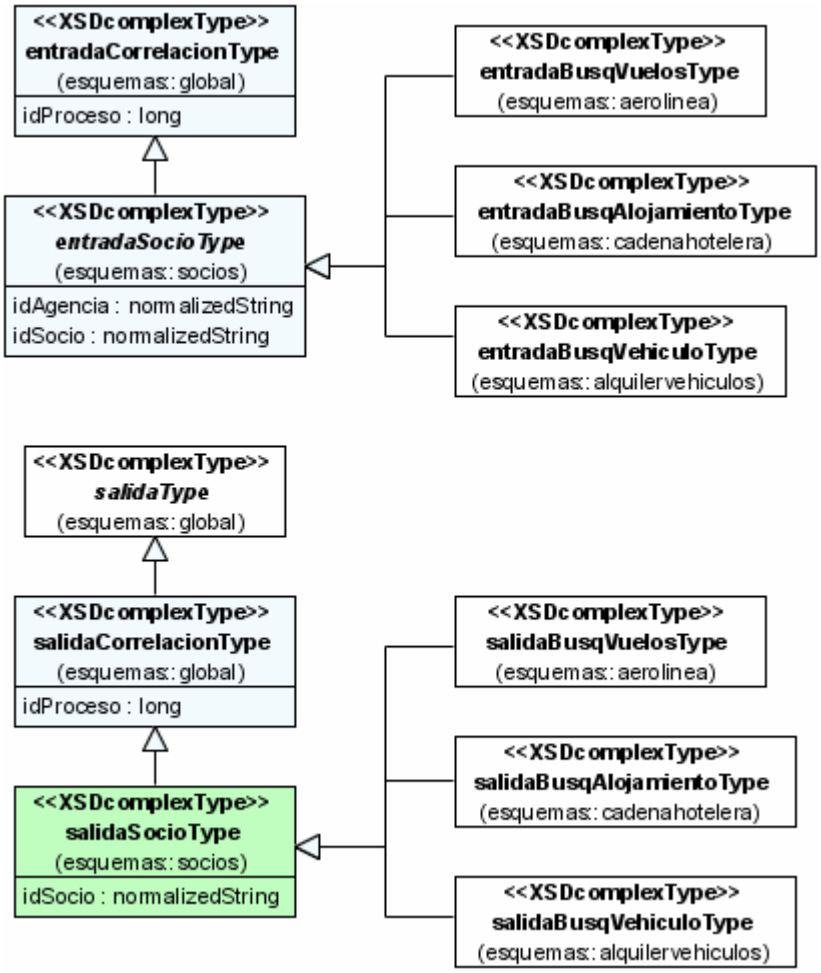


Tabla 51. Descripción de Clases – Cambios en Búsqueda

Clase y Paquete	Descripción
EntradaCorrelacionType *esquemas.global*	Tipo complejo con la información de la correlación, y sirve como padre para los tipos usados como entradas que requieren correlación.
EntradaSocioType *esquemas.socios*	Tipo complejo abstracto que define la información básica a transmitir a los socios en la entrada de las operaciones. Extiende de EntradaCorrelacionType.
EntradaBusquedas	Los tipos complejos definidos anteriormente: *EntradaBusqVuelosType*, *EntradaBusqAlojamientoType*, y *EntradaBusqVehiculoType* ahora extienden de EntradaSocioType.
SalidaCorrelacionType *esquemas.global*	Tipo complejo que adiciona la información de correlación para las salidas de los procesos que la requieren. Extiende de SalidaType.
SalidaSocioType *esquemas.socios*	Tipo complejo modificado definiendo la información del socio y extendiendo de SalidaCorrelacionType.
SalidaBusquedas	Los tipos complejos definidos anteriormente: *SalidaBusqVuelosType*, *SalidaBusqAlojamientoType*, y *SalidaaBusqVehiculoType* ahora extienden de SalidaSocioType.

Los campos adicionales que se han agregado a las clases sirven de identificador de la agencia ante los socios (*idAgencia*) y del socio en la agencia para procesamiento posterior (*idSocio*). Estos datos no se tuvieron en cuenta previamente y durante la realización de los cambios producto de las pruebas se apreció su importancia.

Luego de implementar los cambios a los desarrollos, se realizó una nueva sesión de pruebas similar a la primera. Esta vez los resultados fueron exitosos y satisfactorios en todos los sentidos.

5.5.3 Fase 2. Reservas de Planes de Viaje

La reserva de planes de viaje es el proceso siguiente a la búsqueda inicial, y por tanto es la segunda en el desarrollo. De igual forma consta de tareas que involucran al cliente y especialmente a los socios y el sistema de la agencia.

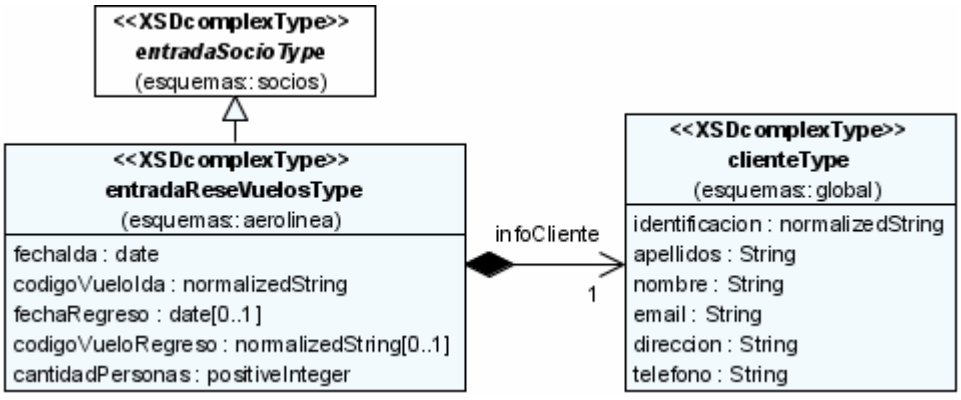
5.5.3.1 Desarrollo del Proceso en la Agencia de Viajes

De acuerdo a lo planteado para el proceso de reserva de planes de viaje, este proceso comprende la llamada a la agencia, la cual contacta los socios de los servicios especificados según se interprete la información de entrada.

- ***Esquemas XML de la Información***

Los tipos de datos definidos para el proceso de reserva amplían los esquemas XML definidos en la búsqueda, reutilizando aquellos ya definidos mediante herencia o relaciones de agregación o composición, tal como ya se ha hecho. A continuación se muestran uno a uno los diagramas de clases de la información intercambiada entre los socios y la agencia y entre la agencia y el cliente, para el proceso de reserva y su explicación respectiva.

Figura 60. Diagrama de Clases – Aerolínea, Entrada de Reserva



El diagrama de clases anterior se definió de acuerdo a la información concebida en el análisis en la tabla 8. Y se destaca la clase *ClienteType* del esquema

esquemas.global que es un tipo complejo que encapsula todos los datos del cliente al que se va a realizar la reserva.

Figura 61. Diagrama de Clases – Aerolínea, Salida de Reserva

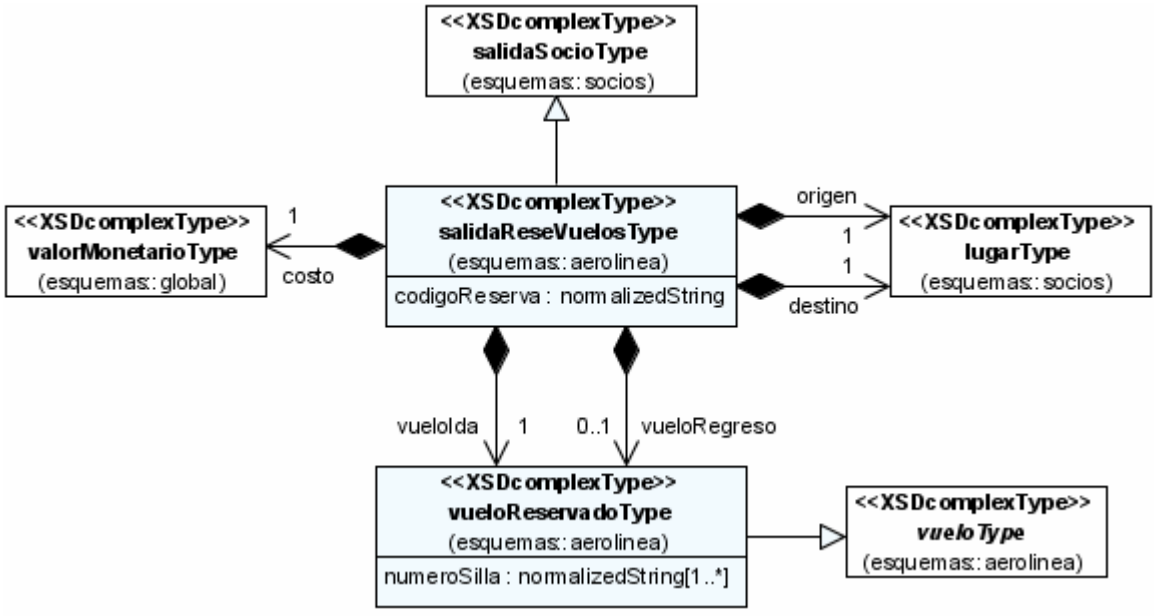
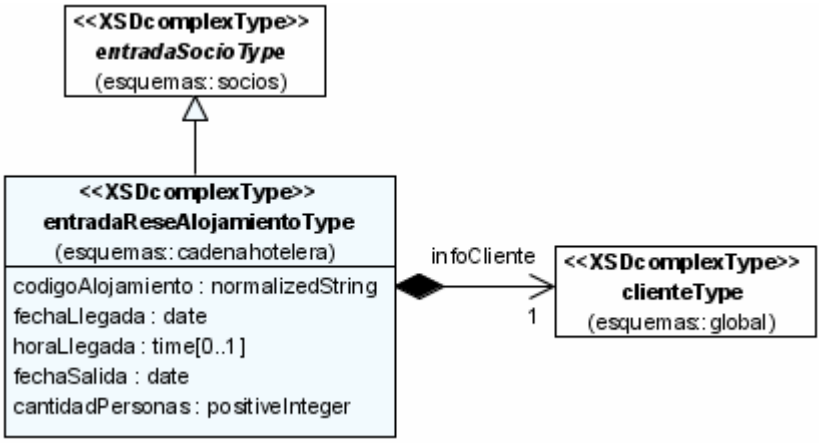


Tabla 52. Descripción de Clases – Aerolínea, Salida de Reserva

Descripción Clases – Reserva en Aerolíneas, Salida Información descrita en Tabla 9. Datos en Respuesta de Reserva de Tiquetes.	
Clase y Paquete	***Descripción***
VueloReservadoType *esquemas.aerolínea*	Tipo complejo con toda la información de un vuelo reservado. Extiende del tipo vueloType, definido anteriormente.
SalidaReseVuelosType *esquemas.aerolínea*	Tipo complejo que encapsula todos los datos definidos para la reserva de vuelos especificados en la tabla 9. Extiende del tipo SalidaSocioType.

Figura 62. Diagrama de Clases – Cadena Hotelera, Entrada de Reserva



El tipo de datos usado para el mensaje de petición de reserva, mostrado en la figura 62, corresponde a lo definido previamente en la tabla 14.

Figura 63. Diagrama de Clases – Cadena Hotelera, Salida de Reserva

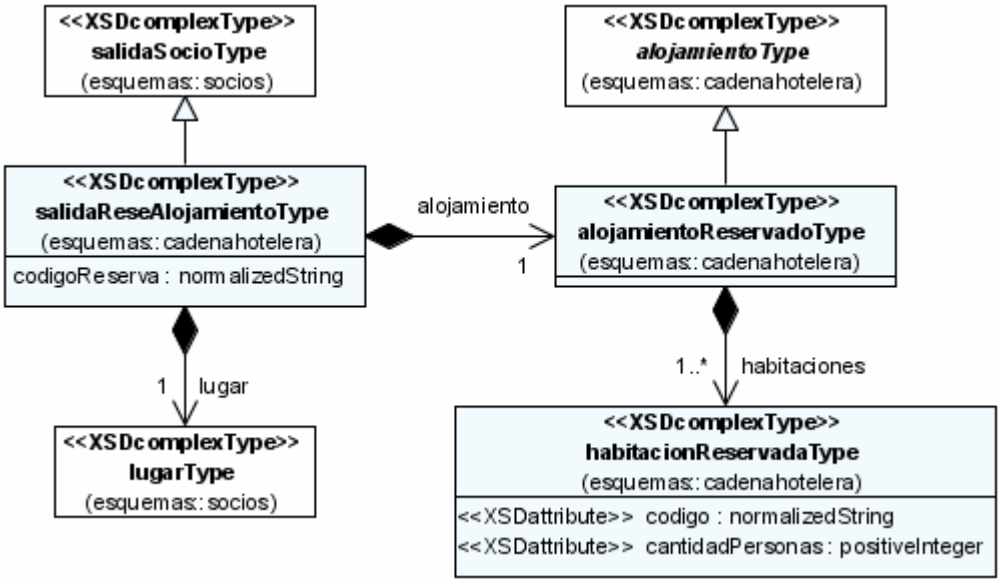


Tabla 53. Descripción de Clases – Cadena Hotelera, Salida de Reserva

Descripción Clases – Reserva en Cadenas Hoteleras, Salida Información descrita en Tabla 16. Datos en Respuesta de Reserva de Alojamiento.	
Clase y Paquete	***Descripción***
HabitacionReservadaType *esquemas.cadenahotelera*	Tipo complejo que representa la información de una habitación reservada.
AlojamientoReservadoType *esquemas.cadenahotelera*	Tipo complejo con los datos de alojamiento reservado. Extiende del tipo alojamientoType, definido previamente.
SalidaReseAlojamientoType *esquemas.cadenahotelera*	Tipo complejo con los datos de reserva de alojamiento según en la tabla 16. Extiende del tipo SalidaSocioType.

Para el socio Alquiler de Vehículos, la reserva se concibió como una petición síncrona, por lo tanto se definen datos de entrada y salida a la vez, a la par de tipos de datos para las excepciones.

Figura 64. Diagrama de Clases – Alquiler Vehículos, Reserva

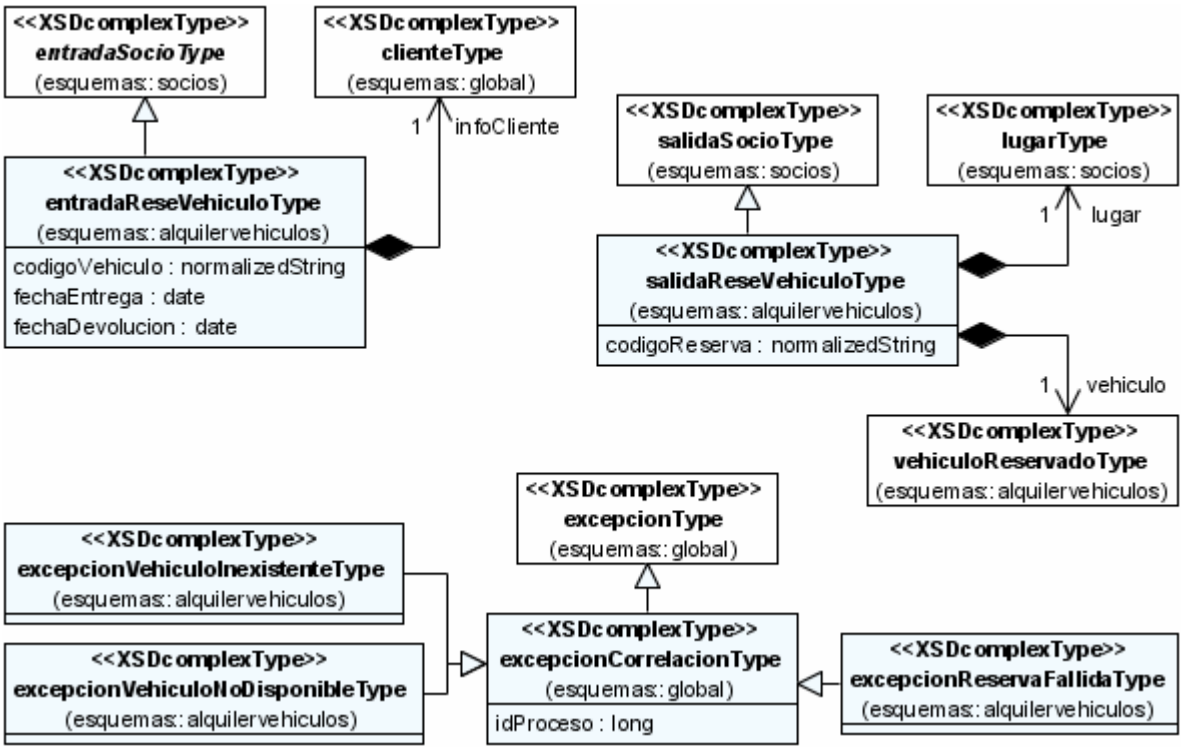
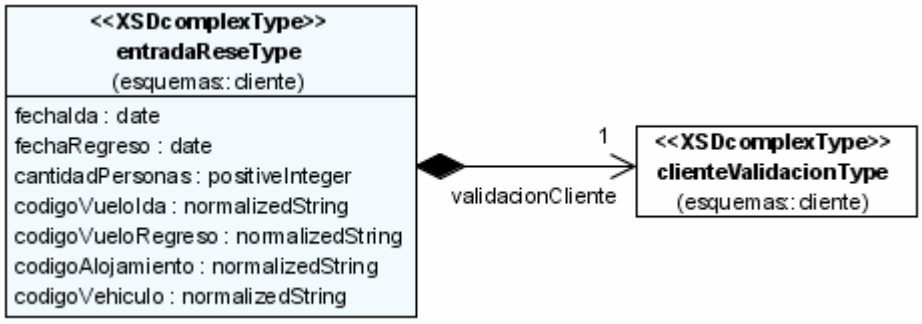


Tabla 54. Descripción de Clases – Alquiler Vehículos, Reserva

Descripción Clases – Reserva en Alquiler de Vehículos Información descrita en Tabla 20. Datos en Solicitud de Reserva de Vehículo y Tabla 21. Datos en Respuesta de Reserva de Vehículo.	
Clase y Paquete	***Descripción***
EntradaReseVehiculoType *esquemas.alquilervehiculos*	Tipo complejo que encapsula la información de entrada de la reserva de vehículos, según se especifica en la tabla 20.
SalidaReseVehiculoType *esquemas.alquilervehiculos*	Tipo complejo que encapsula la respuesta de la reserva, es decir, los datos del vehículo reservado, especificados en la tabla 21.
ExcepcionCorrelacionType *esquemas.global*	Tipo complejo que extiende de *excepcionType* para dar correlación a los mensajes de excepción de la operación.
ExcepcionReservaFallidaType ExcepcionVehiculoInexistenteType ExcepcionVehiculoNoDisponibleType *esquemas.alquilervehiculos*	Tipos complejos que extienden del tipo *excepcionCorrelacionType* con la información de cada una de las excepciones que define su nombre.

Figura 65. Diagrama de Clases – Cliente, Entrada de Reserva



El tipo de datos utilizado para la entrada de la reserva en el cliente, mostrado en la figura 65, corresponde a lo planteado en la
tabla 26.

Figura 66. Diagrama de Clases – Cliente, Salida de Reservas

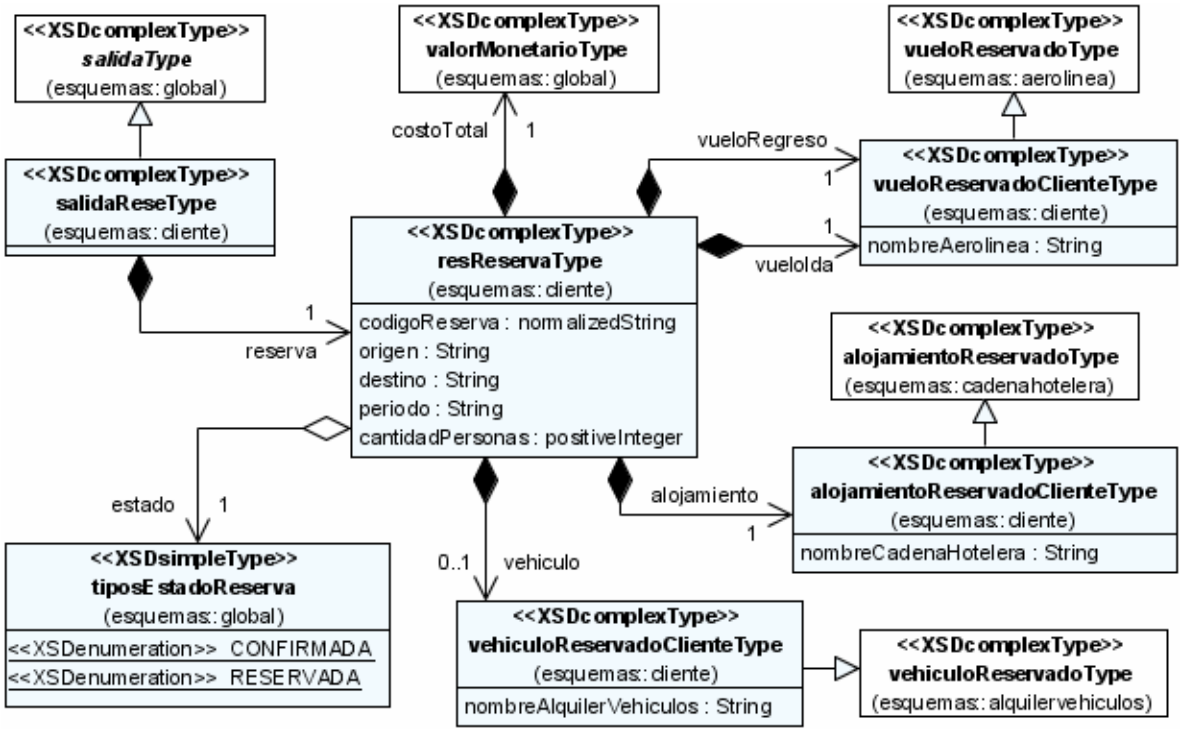


Tabla 55. Descripción de Clases – Cliente, Salida de Reserva

Descripción Clases – Reserva para Clientes, Salida Información descrita en Tabla 27. Datos en Respuesta de Reserva de Plan de Viaje.	
Clase y Paquete	***Descripción***
SalidaReseType *esquemas.cliente*	Tipo complejo que representa el resultado de la operación reservar. Extiende de SalidaType.
ResReservaType *esquemas.cliente*	Tipo complejo que encapsula los resultados de la reserva de planes de viaje, según lo expresado en la tabla 27.
VueloReservado_ ClienteType *esquemas.cliente*	Tipo complejo con la información de cada vuelo reservado. Extiende de VueloReservadoType, definido en la aerolínea.
AlojamientoReservado_ ClienteType *esquemas.cliente*	Tipo complejo con los datos de cada alojamiento reservado. Extiende de AlojamientoReservadoType definido en la cadena hotelera.

VehiculoReservado_ ClienteType *esquemas.cliente*	Tipo complejo con la información de cada vehículo reservado. Extiende de VehiculoReservadoType, definido en el alquiler de vehículos.
TiposEstadoReserva *esquemas.global*	Tipo simple String que define los estados de la reserva, enumeraciones: CONFIRMADA, RESERVADA.

- ***Definición de Interfaces WSDL***

Las interfaces WSDL ya existentes se extenderán para incluir el proceso de reserva, de tal forma que cada una de las definidas en el proceso de búsqueda se reutilizará adicionando las definiciones de la operación reservar. A continuación se describen los cambios de interfaces.

Petición de la Reserva en el Socio Aerolínea

La interfaz de entrada del socio aerolínea se amplía, adicionando la operación de reserva con sus características propias, la definición del servicio no sufre alteraciones.

En la figura 67 se muestra visualmente la nueva estructura WSDL, y en la tabla 56 se muestra la descripción de los nuevos campos y su significado.

Figura 67. Estructura WSDL de Aerolíneas: Reserva

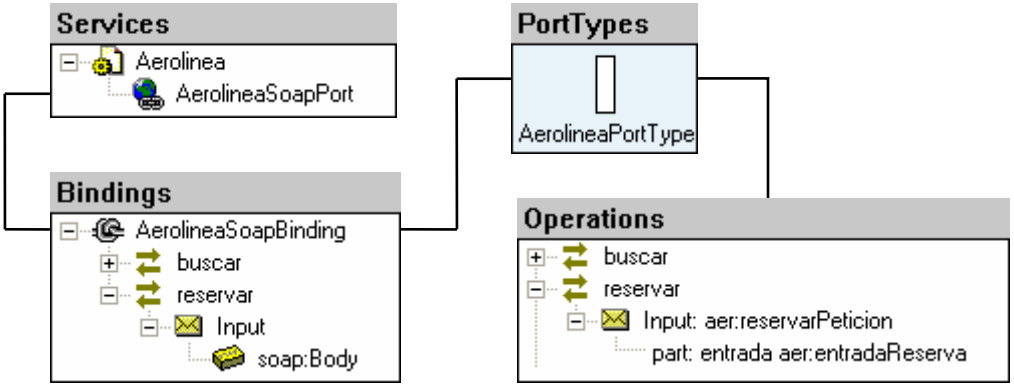


Tabla 56. Interfaz WSDL de Aerolíneas: Reserva

Elemento	Descripción del Contenido
Types	Amplía el esquema local definido en la búsqueda, adicionando el elemento usado en el mensaje. - ***aer:entradaReserva***: elemento de tipo aes:entradaReseVuelosType con la información necesaria en la reserva de vuelos.
Message	Define un nuevo mensaje, transmitido en la operación reservar: - ***reservarPeticion***: mensaje de la petición de la operación *reservar*, contiene cómo única parte, el elemento entradaReserva.
PortType	Amplia la interfaz **AerolineaPortType**, agregando la nueva operación: - *reservar*: invoca la reserva, asigna como entrada el mensaje reservarPeticion. Sólo de entrada pues es invocación asíncrona.
Binding	El enlace **AerolineaSoapBinding** se amplía, adicionando la configuración de la operación *reservar* para que tenga contenido tipo *literal*.

Respuesta de la Reserva en el Socio Aerolínea

Para la respuesta de reserva de tiquetes en la aerolínea se utiliza el mismo documento WSDL de la respuesta de la búsqueda, pero definiendo un nuevo PortType y adicionando los elementos necesarios, tal como se describe en la tabla 57. Los nuevos elementos poseen una estructura similar y por tanto se omite.

Tabla 57. Interfaz WSDL de Aerolíneas: Respuesta de Reserva

Elemento	Descripción del Contenido
Types	Extiende el esquema local, adicionando los elementos usados en los mensajes de la operación reservar. - ***aer:salidaRespuestaOk***, elemento de tipo aes:salidaReseVuelosType con los resultados de la reserva de vuelos. - ***aer:salidaRespuestaError***, elemento de tipo sos:salidaSocioType con la información del resultado de la reserva fallida.

Message	Define los nuevos mensajes a transmitir como posibles resultados de la operación reservar: - ***reservarRespuestaOk***: mensaje de la respuesta exitosa de la operación *reservar*, contiene cómo única parte, el elemento salidaReservaOk. - ***reservarRespuestaError***: mensaje de respuesta fallida de la operación *buscar*, contiene cómo única parte, el elemento salidaReservaError.
PortType	Define un nuevo tipo **AerolineaRespuestaReservaPortType** con sus operaciones: - *reservarOk*, responde el éxito, asigna como entrada el mensaje reservarRespuestaOk. - *reservarError*, responde el fallo, asigna como entrada el mensaje reservarRespuestaError.

Reserva en el Socios Cadena Hotelera

Para la reserva en el socio cadena hotelera, de forma similar que en al socio aerolínea, la interfaz de entrada usada en la búsqueda se amplía adicionando la operación reservar. Por comodidad no se muestran gráficamente, sólo se explican los cambios en elementos en la tabla 58.

Tabla 58. Cambios en Interfaz WSDL para Reserva en Cadenas Hoteleras

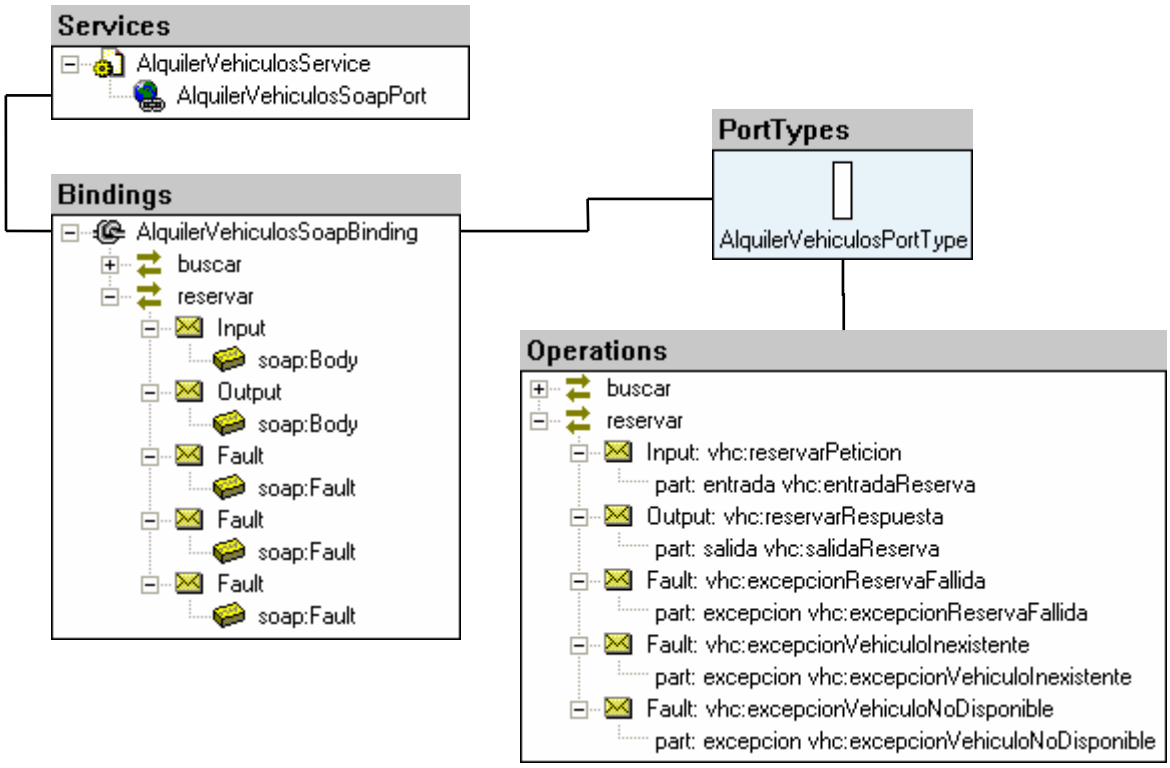
Tipo	Cambios
Petición Reserva	- Elemento *htl:entradaReserva* de tipo *hts:entradaReseAlojamientoType*
Respuesta Reserva	- Elementos *htl:saidaReservaOk* de tipo *hts:entradaReseAlojamientoType* y *htl:salidaReservaError* de tipo *sos:salidaSocioType* - Interfaz *CadenaHoteleraRespuestaReservaPortType*

Reserva en el Socio Alquiler de Vehículos

El modo de operación del proceso de reserva en este socio fue planteado con invocación síncrona y uso de excepciones. La interfaz WSDL definida en la

búsqueda del cliente alquiler de vehículos, se extiende para incluir la operación *reservar* tal cual se definió, la representación gráfica se aprecia en la figura 68.

Figura 68. Estructura WSDL de Alquiler de Vehículos: Reserva



En la tabla 59 se describen las secciones de la interfaz, cuyo código completo se presenta en el Anexo B.

Tabla 59. Interfaz WSDL de Alquiler de Vehículos: Reserva

Elemento	Descripción del Contenido
Types	Incluye en el esquema local los elementos usados en los mensajes de la operación reservar: - ***vhc:entradaReserva***: elemento de tipo vhs:entradaReseVehiculoType con la información necesaria para realizar la reserva de vehículos. - ***vhc:salidaReserva***: elemento de tipo vhs:salidaReseVehiculoType con la información de respuesta de la reserva de vehículos. - ***vhc:excepcionReservaFallida***: elemento de tipo vhs:excepcionReservaFallidaType con los datos de error cuando la reserva no sea exitosa por una causa general. - ***vhc:excepcionVehiculoInexistente***: elemento de tipo vhs:excepcionVehiculoInexistenteType con la información del error cuando se intenta reservar un vehículo inválido. - ***vhc:excepcionVehiculoNoDisponible***: elemento de tipo vhs:excepcionVehiculoNoDisponibleType con los datos del error cuando el vehículo que trata de reservarse ya no está disponible.
Message	Se definen nuevos mensajes transmitidos en la operación reservar: - ***reservarPeticion***: mensaje de la petición de la operación *reservar*, - ***reservarRespuesta***: mensaje de respuesta de la operación *reservar*. - ***excepcionReservaFallida***: mensaje correspondiente a la excepción de reserva fallida, anexa a la operación *reservar*. - ***excepcionVehiculoInexistente***: mensaje de la excepción de vehículo no existente, anexa a la operación *reservar*. - ***excepcionVehiculoNoDisponible***: mensaje para la excepción de vehículo no disponible, anexa a la operación *reservar*.
PortType	Adiciona a la interfaz **AlquilerVehiculosPortType** la operación: - *reservar*: llama a la reserva, asignando como entrada el mensaje reservarPeticion, como salida el mensaje reservarRespuesta, y definiendo las tres excepciones que usan los mensajes excepcionReservaFallida, excepcionVehiculoInexistente y excepcionVehiculoNoDisponible respectivamente.
Binding	Redefine el enlace **AlquilerVehiculosSoapBinding** configurando la nueva operación reservar y sus partes de tipo *literal*.

Reserva en el Cliente

La interfaz de reserva que la agencia de viajes exponer al sistema del cliente se define de forma análoga a la de búsqueda: omitiendo los *bindings* y el servicio. Se definieron dos excepciones similares a la operación búsqueda, y también opera de forma síncrona, con mensajes de entrada y salida usando los tipos de datos definidos para ella.

Reserva Efectiva

En la búsqueda se invocaba el subrpoceso llamado *reservaefectiva* encargado de realizar la reserva automática luego de realizar la búsqueda cuando se encontraban todos los servicios requeridos y el usuario así lo especificaba. Este proceso también será implementado en BPEL y por lo tanto necesita una interfaz para su acceso, definiendo como punto de entrada una operación síncrona con excepciones. En la tabla 60 se describe la interfaz WSDL necesaria, implementada anexa a la interfaz de la agencia de viajes local.

Tabla 60. Interfaz WSDL para Proceso Reserva Efectiva

Elemento	Descripción del Contenido
Types	Define en el esquema local, nuevos elementos usados en los mensajes de la operación reservaEfectiva. - ***age:entradaReservaEfectiva***, elemento de tipo cls:entradaReseType con la información específica del plan de viaje a reservar. - ***age:salidaReservaEfectiva***, elemento de tipo cls:salidaReseType con los resultados de la reserva efectiva del plan de viaje.
Message	Define los mensajes a transmitir entre los procesos con la operación síncrona reservaEfectiva. La excepción es similar a la ya definida. - ***reservaEfectivaPeticion***: mensaje de entrada de la operación *reservaEfectiva*, su única parte es el elemento entradaReservaEfectiva. - ***reservaEfectivaRespuesta***: mensaje de respuesta de la operación *reservaEfectiva*, su única parte es el elemento salidaReservaEfectiva.

PortType	Define la interfaz **AgenciaViajesPortType** con la operación: - *reservaEfectiva*, que contiene: entrada: mensaje reservaEfectivaPeticion salida: mensaje reservaEfectivaRespuesta fallo “Fallo General”: mensaje excepcionGeneral
Binding y Service	Sin definir ya que se generan al publicar el proceso reserva efectiva en el motor BPEL.

- ***Participantes del Proceso y sus Vínculos***

El siguiente paso es involucrar los actores del proceso y vincularlos a nivel WSDL.

Relación entre Agencia de Viajes y Aerolínea (y Cadena Hotelera)

Para la reserva, se toma el portTypes de la aerolínea *AerolineaPortType* (que contiene hasta este momento las operaciones buscar y reservar) y el definido para la respuesta de reserva de la aerolínea: *AerolineaRespuestaReservaPortType*. El hecho que exista también la operación buscar no altera la relación pues se entiende que el vínculo opera sólo la operación reservar. Esta relación se da en el proceso de ReservaEfectiva.

Figura 69. Definición de PartnerLinkTypes de Aerolínea: Reserva

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="aerolineaPartnerLinkType"  ...  >

  ...

  <plnk:partnerLinkType name="aerolineaReservaPartnerLinkType">
    <plnk:role name="aerolinea">
      <plnk:portType name="aer:AerolineaPortType" />
    </plnk:role>
    <plnk:role name="agenteviajes">
      <plnk:portType name="aae:AerolineaRespuestaReservaPortType" />
    </plnk:role>
  </plnk:partnerLinkType>

</wsdl:definitions>
```

En la figura anterior se aprecia el código WSDL de la definición del nuevo partnerLinkType, la cual se anexa en el código inicial definido en la fase 1. Este vínculo para el socio aerolínea es equivalente en estructura al perteneciente al socio cadena hotelera ya que comparten las características de modo de operación asíncrona.

Relación entre Agencia de Viajes y Alquiler de Vehículos

En el caso de la reserva para el alquiler de vehículos, la operación es síncrona, por lo tanto el proceso no desempeña un rol en el vínculo. La definición del partnerLinkType sólo posee el rol de alquilervehiculos que desempeña el socio en la relación unilateral. Al igual que el caso anterior, esta relación se evidencia en el proceso de ReservaEfectiva. A continuación se muestra un fragmento del código correspondiente:

Figura 70. Definición de PartnerLinkTypes de Alquiler de Vehículos: Reserva

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="alquilervehiculosPartnerLinkType"  ...  >

  ...

  <plnk:partnerLinkType name="alquilerVehiculosReservaPartnerLinkType">
    <plnk:role name="alquilervehiculos">
      <plnk:portType name="vhc:AlquilerVehiculosPortType" />
    </plnk:role>
  </plnk:partnerLinkType>

</wsdl:definitions>
```

Relación entre Agencia de Viajes y el Cliente

La relación definida entre el estos dos actores es estructuralmente idéntica a la definida para la operación búsqueda, por lo tanto se omite su presentación. El nombre dado fue *clienteReservaPartnerLinkType* y utiliza el portType *acl:ClienteReservaPortType* bajo el rol llamado agenteviajesDelCliente. Sin

embargo se debe aclarar que esta relación se evidencia en el proceso Reserva, puesto que es aquel que se comunica con el cliente y no en el de ReservaEfectiva, que sólo se comunica con los socios.

Auto-relación interna de la Agencia de Viajes: Subproceso

Los dos procesos BPEL relacionados con la búsqueda se relacionan entre sí debido a que uno (reserva) invoca al otro (reservaEfectiva). Esta relación es de un solo sentido ya que el proceso de reservaEfectiva no puede invocar directamente al de reserva, sólo responde a su llamado.

El nombre dado fue *agenciaviajesPartnerLinkType* y utiliza el portType *age:AgenciaViajesPortType* bajo el rol llamado agenteviajes. Esta relación aunque sea unilateral, se hará evidente en ambos procesos mediante la declaración de los partnerLinks.

Vínculos a nivel del Proceso BPEL

Los vínculos internos a los procesos de la reserva, definidos entre sus actores (partnerLinks) se presentan en la tabla 61.

Allí se aprecian los vínculos unilaterales para el caso del socio alquiler de vehículos y para el cliente, pero se aprecia la diferencia del “lado” del vínculo, expresado en el tipo de rol que desempeña (partnerRole o myRole).

De forma similar se evidencia la relación entre los procesos Reserva y ReservaEfectiva, mediante el vínculo *AgenciaViajes* el cual desempeña un rol diferente según el proceso: en el proceso de Reserva se establece como un tercero que es accedido, mientras que en el proceso de ReservaEfectiva se define como el rol propio y por tanto se generará una interfaz al publicarse.

Tabla 61. Declaración de PartnerLinks para los Procesos de Reserva

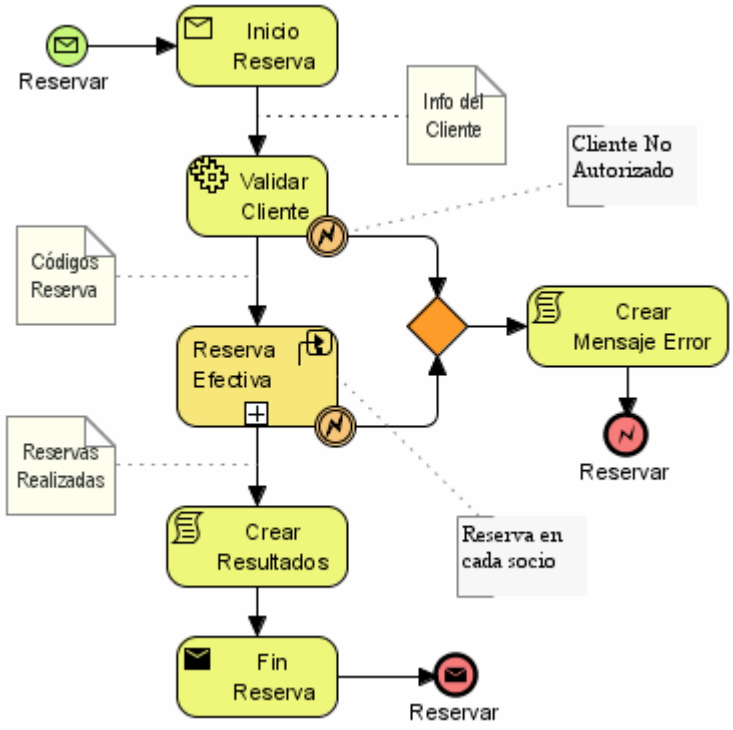
Proceso	PartnerLink	Propiedades
Reserva	Cliente	- partnerLinkType: *acl:clienteBusquedaPartnerLinkType* - myRole: *agenteviajesDelCliente*
	AgenciaViajes	- partnerLinkType: *age:agenciaviajesPartnerLinkType* - partnerRole: *agenteviajes*
Reserva Efectiva	Aerolínea	- partnerLinkType: *aae:aerolineaReservaPartnerLinkType* - myRole: *agenteviajes* - partnerRole: *aerolinea*
	Cadena Hotelera	- partnerLinkType: *aht:cadenahoteleraReservaPartnerLinkType* - myRole: *agenteviajes* - partnerRole: *cadenahotelera*
	Alquiler Vehículos	- partnerLinkType: *avh:alquilervehiculosReservaPartnerLinkType* - partnerRole: *alquilervehiculos*
	AgenciaViajes	- partnerLinkType: *age:agenciaviajesPartnerLinkType* - myRole: *agenteviajes*

- ***Implementación del Proceso BPEL de Reserva***

El análisis del proceso de reserva realizado previamente (*véase 5.4.3.2*) permitió definir las tareas involucradas en el flujo de actividades del proceso. Allí se dividió el proceso en 2, separando la parte de validación inicial con la realización efectiva de las reservas. Por tanto se realizaron dos procesos uno general que inicia el proceso de la reserva, y el otro a modo de subproceso que ejecuta la reserva efectiva a nivel de los socios.

El proceso general de reserva planteado, mostrado en la figura 71 con un diagrama BPMN, se encarga de la validación inicial de datos (similar al de la búsqueda), la invocación al subproceso de reserva efectiva, y la devolución de los resultados. Este proceso se codificó en un archivo llamado *reserva.bpel* disponible en el código fuente del proyecto.

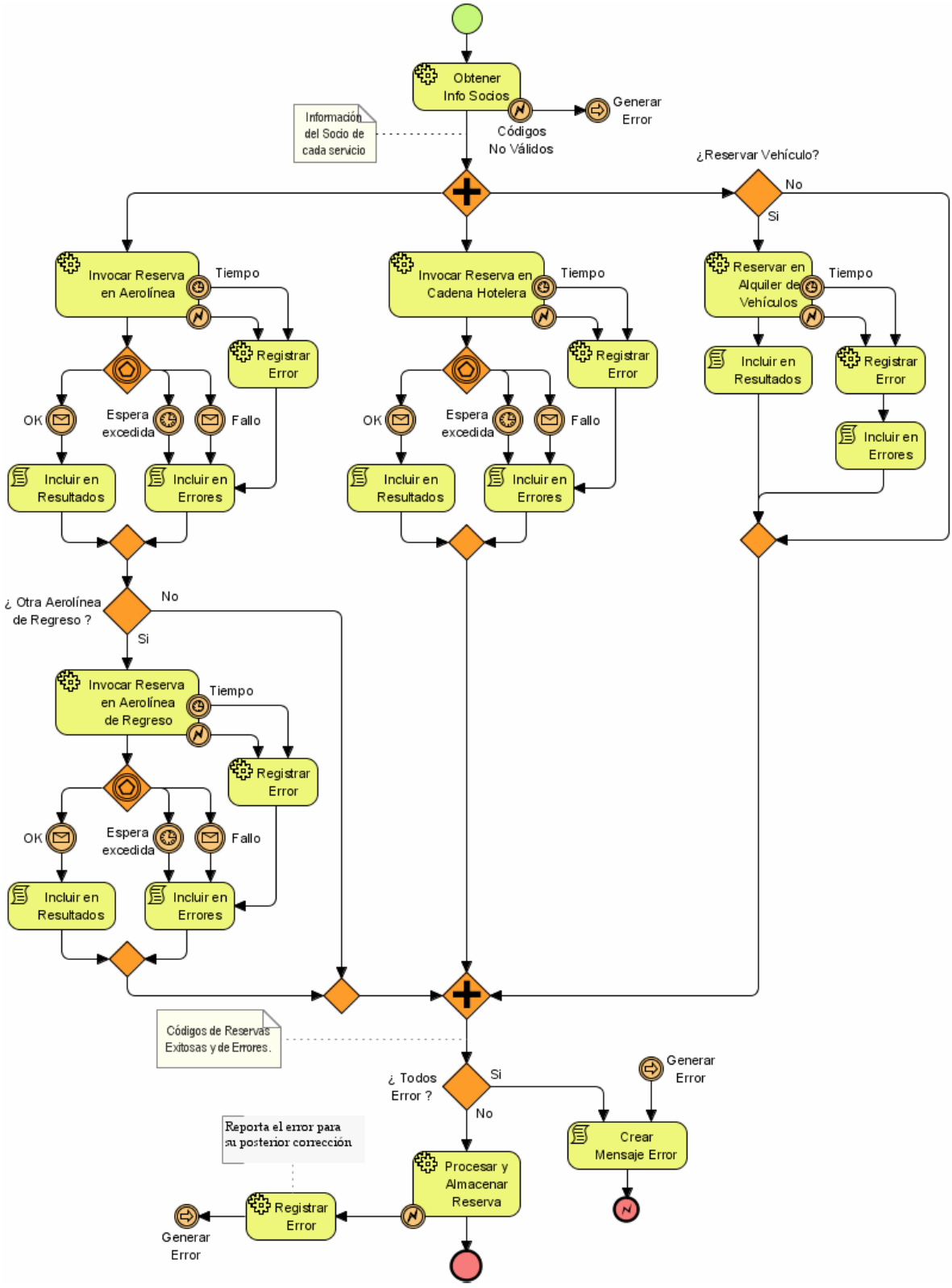
Figura 71. Proceso de Negocio de Reservas en la Agencia



Por su parte, el subproceso encargado de la reserva efectiva, mostrado en la figura 72, involucra las tareas de obtención de la información de los socios correspondientes a los servicios especificados para reservar, interacción con cada uno de estos socios para la realización de la reserva procesando su resultado sea válido o erróneo, y procesamiento de los resultados. Este subproceso fue codificado en el archivo llamado *reservaefectiva.bpel*, también disponible en el código fuente.

Se destaca la diferencia entre la forma de realizar la reserva en el socio alquiler de vehículos: la invocación es síncrona por tanto no espera mensajes de respuesta; también se aprecia que se deben contactar dos aerolíneas cuando los vuelos de ida y regreso pertenecen a diferente socio.

Figura 72. Subproceso de Negocio de Reserva Efectiva



- ***Especificación del Servicio Web Local***

El servicio Web a nivel local en la agencia se amplió para incluir las tareas complejas del proceso BPEL de la reserva efectiva.

Interfaz WSDL Local

El enlace *AgenciaViajesLocalPortType* se amplía para definir las nuevas operaciones requeridas, según se resume en la tabla 62; al igual que las operaciones para la búsqueda, las nuevas aquí definidas poseen una excepción Fallo General.

Tabla 62. Especificación Tareas Locales, Proceso Reserva Efectiva

Especificación de AgenciaViajesLocalPortType para Reserva
Obtener Info Socios *(obtenerInfoSociosReserva)* *Descripción*: se encarga de consultar la información de cada socio que posee los servicios que el cliente ha especificado para reservar. *Entrada*: igual a la entrada del proceso, es decir un elemento de tipo *cls:entradReseType*. *Salida*: un elemento de tipo *ags:infoSociosReservaType* que contiene la información de los socios a contactar para la reserva: aerolínea (de ida y regreso), cadena hotelera y alquiler de vehículos (opcional), incluyendo la información de enlace dinámico. *Excepciones*: - Códigos no válidos, cuando los códigos de los servicios no fueron validados o interpretados correctamente; definida con un mensaje de tipo *ags:excepcionCodigosNoValidosType*.
Procesar y Almacenar Reserva *(procesarYAlmacenarReserva)* *Descripción*: se encarga de tomar los resultados de las reservas de cada servicio y procesarlas para formar los resultados del proceso, unificando a la unidad monetaria preferida del usuario, y además almacena internamente la reserva. *Entrada*: los resultados de las reservas de cada servicio y las preferencias de cliente, definida como un elemento del tipo *ags:resultadosReservasType*. *Salida*: un elemento de tipo *cls:salidaReseType* con los resultados de la reserva ya procesados y listos para retornarse al cliente.

Tipos de Datos XML

Para esta interfaz se amplió el esquema XML de la agencia de viajes local (*ags*), y incluyendo los tipos de datos necesarios para el procesamiento de la información

en las tareas definidas. En la siguiente figura se muestra el diagrama de clases con los tipos definidos, que posteriormente se explican detalladamente en la tabla 63.

Figura 73. Diagrama de Clases – Agencia Local, Reserva

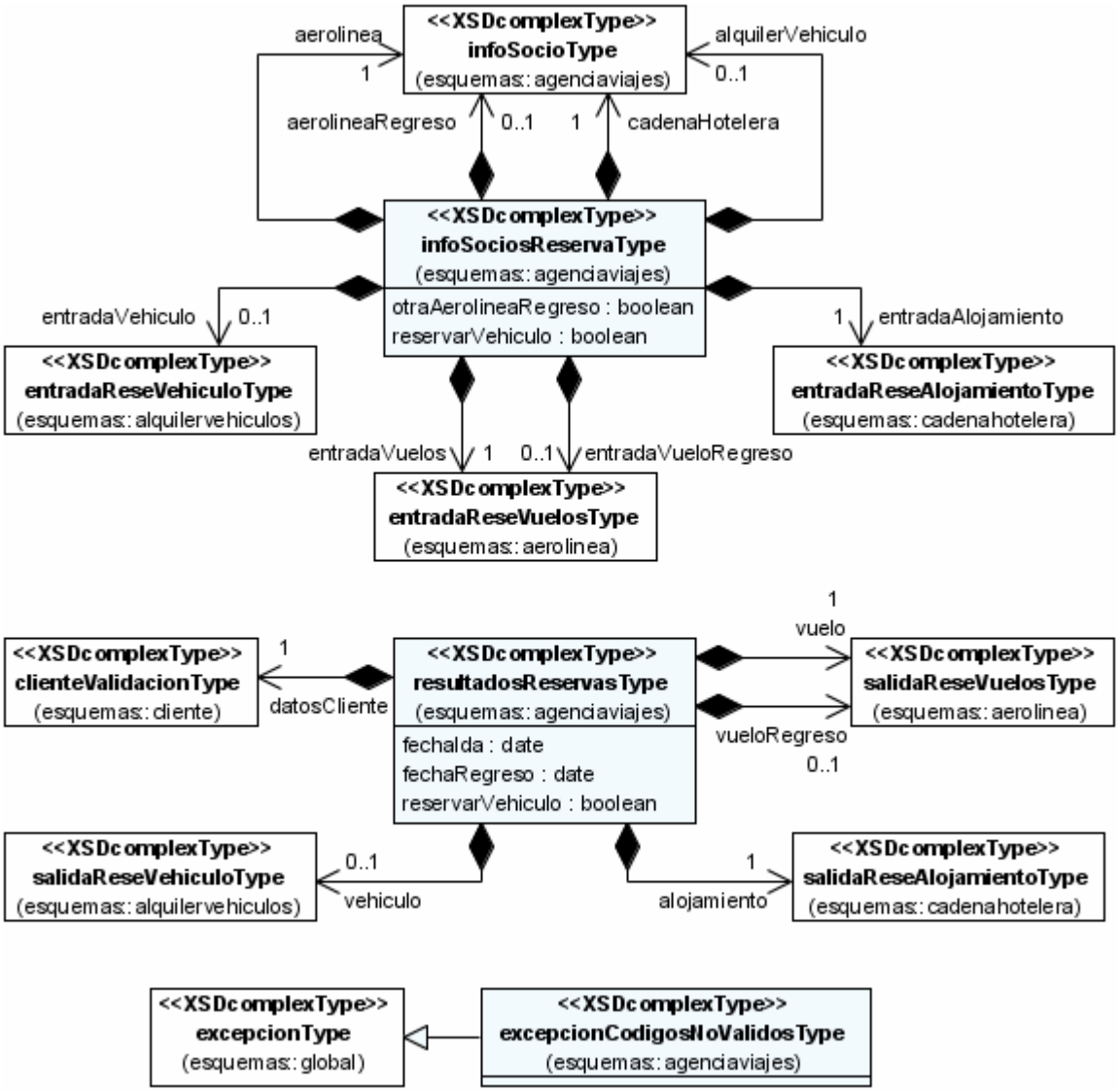


Tabla 63. Descripción de Clases – Agencia Local, Reserva

Clase y Paquete	Descripción
InfoSociosReservaType *esquemas.agenciaviajes*	Tipo complejo que encapsula los datos de los socios involucrados en la reserva de los servicios.
ResultadosReservasType *esquemas.agenciaviajes*	Tipo complejo que posee el conjunto de resultados de todas las reservas y los datos de cliente, para su procesamiento y almacenamiento en la agencia.
Excepcion_ CodigosNoValidosType *esquemas.agenciaviajes*	Tipo complejo que extiende de excepcionType y representa la información de la excepción cuando se especifican códigos de servicios incorrectos.

Vínculos

Este servicio Web local de la agencia de viajes juega un papel (rol) tanto en el proceso BPEL general de la reserva, como en el de la reserva efectiva por lo tanto necesita definición WSDL de PartnerLinkTypes y declaración PartnerLink en los procesos. El partnerLinkType definido en la búsqueda ya declara el vínculo por lo tanto no debe definirse otro. A nivel de los procesos sí deben declararse los partnerLink correspondientes, exactamente iguales a los usados en el proceso de búsqueda y llamado *agenciaviajeslocal.*

Implementación de la Lógica

Las tareas implementadas hacen uso de la estructura de datos definida en el Anexo C para almacenar los datos de las reservas realizadas y asignadas al cliente respectivo. Los detalles de la implementación se omiten, pero están disponibles apreciarse en el código fuente del desarrollo.

- ***Despliegue y Publicación de los Procesos BPEL***

A continuación se presentan los detalles de la publicación, en el motor de ActiveBPEL, de los procesos de Reserva y Reserva Efectiva.

Archivos de Descripción de Despliegue de los Procesos

Cada uno de los procesos posee su propio archivo *descriptor de despliegue*, puesto que cada uno maneja diferentes partnerLinks que deben ser descritos. Los archivos tienen una estructura similar al presentado en el despliegue del proceso de búsqueda, aunque presentan detalles y variaciones muy significativas. En las siguientes figuran se aprecian fragmentos de los archivos resultantes:

Figura 74. Fragmento Archivo de Despliegue, Proceso BPEL Reserva

```
<?xml version="1.0" encoding="UTF-8"?>
<process  xmlns="http://schemas.active-endpoints.com/pdd/2005/09/pdd.xsd" ... >
  <partnerLinks>
    <partnerLink name="agenciadeviajes">
      <partnerRole endpointReference="static" invokeHandler="process:subprocess">
        <wsa:EndpointReference>
          <wsa:Address>AgenciaViajesService</wsa:Address>
        </wsa:EndpointReference>
      </partnerRole>
    </partnerLink>
    ...
  </partnerLinks>
</process>
```

Figura 75. Fragmento Archivo de Despliegue, Proceso BPEL Reserva Efectiva

```
<?xml version="1.0" encoding="UTF-8"?>
<process  xmlns="http://schemas.active-endpoints.com/pdd/2005/09/pdd.xsd" ... >
  <partnerLinks>
    ...
    <partnerLink name="agenciadeviajes">
      <myRole allowedRoles="" binding="MSG" service="AgenciaViajesService">
    </partnerLink>
    ...
  </partnerLinks>
</process>
```

El archivo perteneciente al proceso de reserva define también los enlaces de los partnerLinks de agenciaviajesLocal y cliente. Por su parte, en el del proceso de reserva efectiva se definen los enlaces para los partnerLinks relacionados con los socios y la agenciaviajesLocal.

Como se observa en las figuras anteriores, en el descriptor del proceso de Reserva Efectiva se realiza la publicación del servicio AgenciaViajesService el cual es usado en el descriptor del proceso de reserva definiendo su enlace como estático con manejador de subproceso, lo que le indica al motor de ActiveBPEL que lo interprete y ejecute como un subproceso del actual.

Publicación y Puesta en Marcha de los Procesos

El resultado de la publicación de los procesos de reserva en ActiveBPEL, se expresa en la creación y exposición de cuatro servicios Web: *AerolineaReservaService*, *CadenaHoteleraReservaService y AgenciaViajesService* al publicar el proceso Reserva Efectiva, y *ClienteReservaService* perteneciente al proceso de Reserva.

A modo de ejemplo, en la figura 76 se muestra el código WSDL del servicio AerolineaReservaService generado en la publicación del proceso Reserva Efectiva. Este código es estructuralmente similar al que se genera para los servicios AerolineaBusquedaService, CadenaHoteleraBusquedaService y AlquilerVehiculosBusquedaService de la fase 1.

Las interfaces de AgenciaViajesService y ClienteReservaService presentan una estructura similar a la mostrada en la publicación del proceso de búsqueda en la fase 1.

Figura 76. Interfaz WSDL Final Generada para Aerolínea, Respuesta Búsqueda

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="aerolineaPartnerLinkType"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
                  xmlns:aer="http://aerolinea.com/servicios/agenciaviajes"
                  xmlns:aae="http://agenciaviajes.com/servicios/aerolinea"
                  targetNamespace="http://agenciaviajes.com/servicios/aerolinea" >

  <wsdl:import namespace="http://aerolinea.com/servicios/agenciaviajes"
               location="http://localhost/active-bpel/catalog/AgenciaViajes/aerolinea.wsdl" />
  <wsdl:import namespace="http://agenciaviajes.com/servicios/aerolinea"
               location="http://localhost/active-bpel/catalog/AgenciaViajes/aerolinearespuesta.wsdl" />

  <wsdl:binding name="AerolineaReservaServiceBinding"
                type="aae:AerolineaRespuestaReservaPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="reservarOk">
      <soap:operation soapAction="" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
    </wsdl:operation>
    <wsdl:operation name="reservarError">
      <soap:operation soapAction="" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="AerolineaReservaService">
    <wsdl:port binding="aae:AerolineaReservaServiceBinding"
               name="AerolineaReservaServicePort">
      <soap:address location="http://localhost/active-bpel/services/AerolineaReservaService" />
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

Aquí se destaca la generación del service y el binding con las dos operaciones que el socio utiliza para responder a la petición de reserva.

5.5.3.2 Especificación de los Sistemas de los Socios

En los sistemas de los socios la funcionalidad de reserva de servicios se definió mediante la operación ***reservar***, cuya implementación es independiente a cada uno de los socios. A continuación se describe la lógica interna general que maneja que es similar para todos aunque se destaca que para el caso del Alquiler de Vehículos, la forma de operación es diferente puesto que la operación reservar se ha definido como síncrona y con manejo de excepciones, a diferencia de la Aerolínea y la Cadena Hotelera que es asíncrona.

En la figura figura 77 se aprecia la secuencia típica del proceso de reserva con llamado asíncrono, para los socios Aerolínea y Cadena Hotelera. En contraste, en la figura 78 aparece la secuencia para el socio Alquiler de Vehículos, cuyo llamado es síncrono.

Figura 77. Diagrama de Secuencia, Reserva en Socios

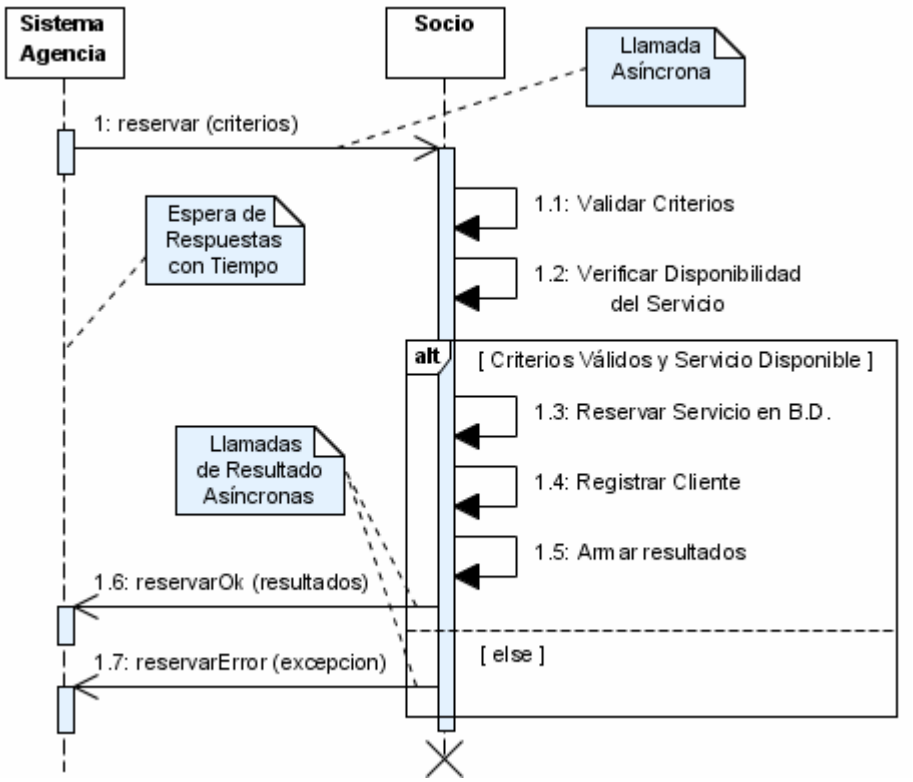
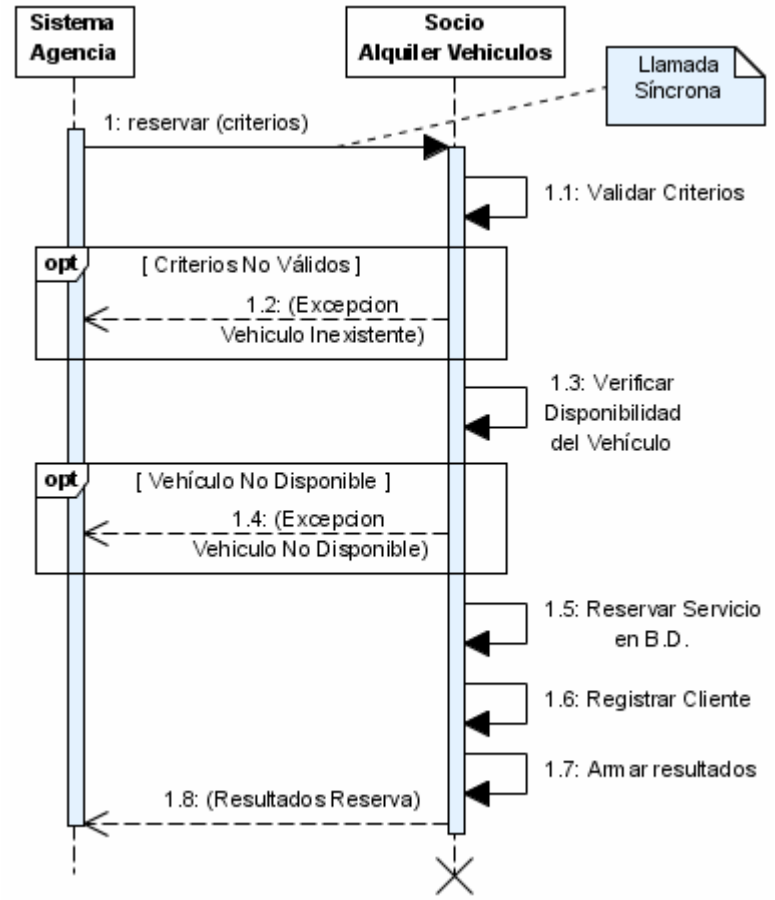


Figura 78. Diagrama de Secuencia, Reserva en Socio Alquiler de Vehículos



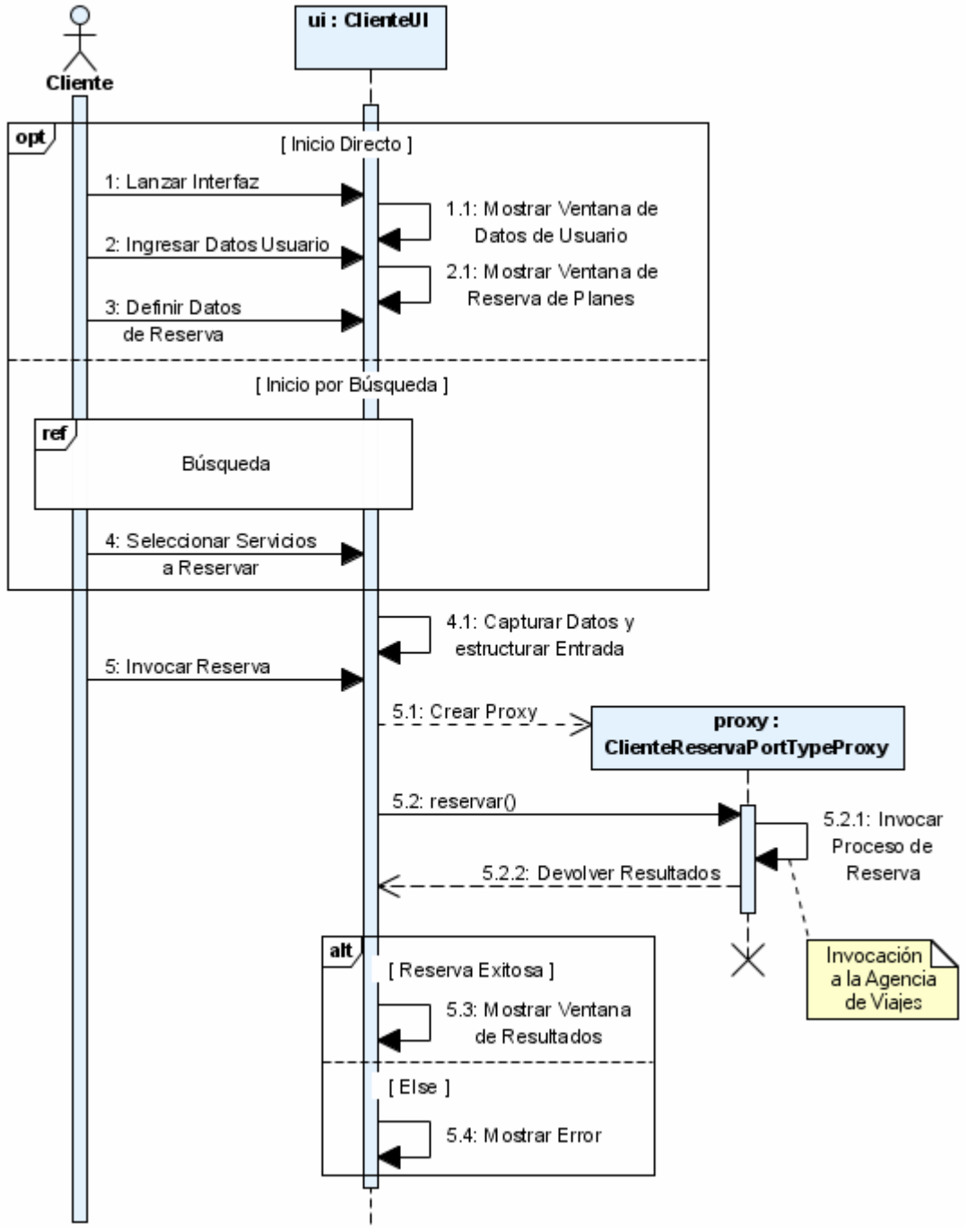
El flujo de actividades es similar, sólo cambia la forma como se devuelve el resultado de error. Primero se deben validar los criterios de acuerdo con los formatos establecidos e identificando que sean servicios correctos que ha ofrecido el socio en una búsqueda, luego se verifica que el servicio especificado esté disponible en el momento, si las dos validaciones son satisfactorias se debe realizar la reserva mediante la actualización e inclusión de registros en la B.D. propia al igual que almacenar o actualizar la información del cliente propietario de la reserva; finalmente se organizan los resultados y se devuelven al solicitante.

En el Anexo C aparecen las estructuras de datos definidas para los socios, incluyendo los relacionados con el almacenamiento de las reservas.

5.5.3.3 Implementación de la Interfaz del Cliente

Para la funcionalidad de reservas, la interfaz del cliente debe realizar la captura de los datos y visualización de los resultados, y también contactar el proceso dado.

Figura 79. Diagrama de Secuencia, Reserva en Interfaz de Cliente



En la figura anterior se muestra la secuencia lógica de tareas para realizar la reserva de un plan de viajes a través de la interfaz de cliente. El flujo de trabajo comienza de dos formas: por ingreso directo o a partir del final de un proceso de búsqueda.

Cuando el usuario ingresa directamente, debe diligencia los datos de usuario y de la reserva a realizar, a lo que la interfaz (ClienteUI) responde mostrando las ventanas de recepción de información. Si el proceso inicia con el final de la búsqueda, el cliente debe seleccionar los servicios a reservar basado en los mostrados como resultado de la búsqueda.

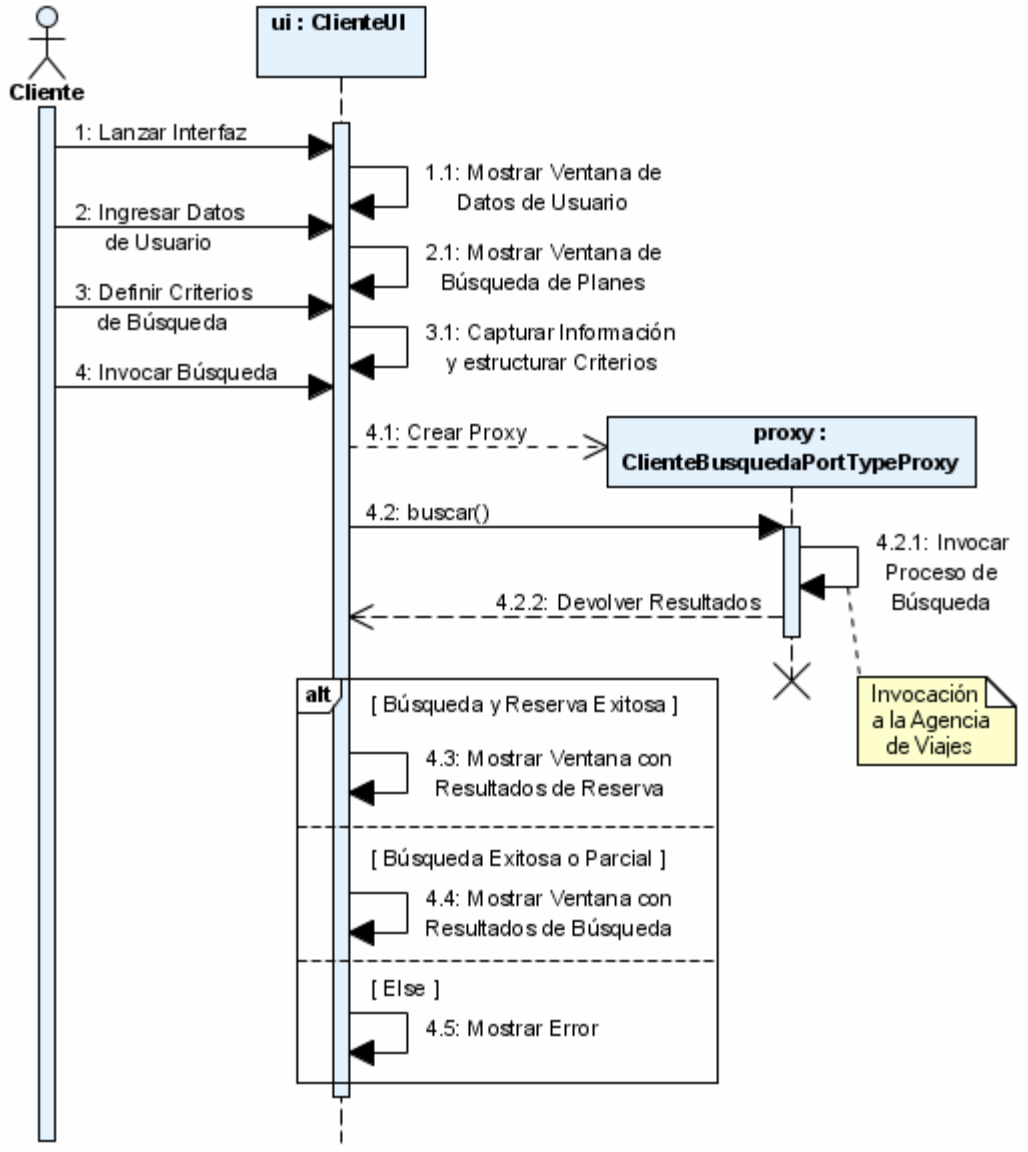
Luego de esto, la interfaz captura estos datos y construye la entrada del proceso de reservar, y continua con la creación del representante o *proxy* de la agencia para la reserva (de tipo ClienteReservaPortTypeProxy), y ejecuta la reserva a través de la operación del proxy. Internamente el proxy hace el llamado a la agencia y espera los resultados, que son inmediatamente devueltos a la interfaz de usuario. Estos resultados son procesados y visualizados en pantalla de acuerdo al éxito de la operación.

Las clases implementadas para la parte de reservas en el sistema del cliente son generadas por AXIS para con base en la definición del servicio de Reserva, y tienen una estructura similar a las generadas en la búsqueda y explicadas en la tabla 50. En contraste, la clase principal de la interfaz, ClienteUI se extiende añadiendo tres nuevos métodos: *capturarDatosReserva*, *realizarReserva*, y *visualizarResultadosReserva*.

Finalmente, algunas imágenes de ejemplo con la interfaz de usuario de la parte de reservas se pueden observar en el capítulo 6.

También cabe mencionar que el flujo de actividades de la búsqueda en la interfaz de usuario presenta una variación ya que ahora es posible realizar reservas inmediatas al ejecutar una búsqueda. El cambio radica en que los resultados de la búsqueda pueden ser ahora también resultados de reserva, en cuyo caso debe visualizarse como los servicios reservados, en vez de los servicios encontrados. En la figura 80 se muestra el diagrama de secuencia actualizado.

Figura 80. Nuevo Diagrama de Secuencia, Búsqueda en Interfaz del Cliente



5.5.3.4 Pruebas de la Fase de Reservas

Las pruebas relacionadas con la fase 2 fueron ejecutadas de acuerdo a lo definido en el Anexo D, durante una sesión, por el mismo personal que realizó las pruebas de la fase 1. Los resultados obtenidos fueron altamente exitosos en cuanto a funcionamiento interno de los procesos y validación de información; sólo se obtuvieron los siguientes defectos a nivel de interfaz que fueron solucionados de inmediato:

- Los resultados de la reserva muestra la sección de vehículos activa, aunque sin resultados, cuando no se ha especificado reservar este servicio.
- La selección de los servicios a reservar en la ventana de resultados no se aprecia con claridad. Es necesario mejorar la apariencia del servicio seleccionado.

5.5.4 Fase 3. Consulta de Reservas de Planes de Viaje

Una vez existan reservas de planes de viaje, es necesario disponer de una operación para que el cliente pueda consultarlas y procesarlas. En esta tarea los socios no interactúan puesto que la agencia tiene almacenada internamente la información que el cliente solicita.

5.5.4.1 Desarrollo del Proceso en la Agencia de Viajes

Según el planteamiento realizado, el proceso de consulta de reservas en la llamada a la agencia indagando la información de la reserva, y ésta internamente extrae la información según el cliente.

- ***Esquemas XML de la Información***

Los esquemas XML tratados hasta el momento se extienden con los tipos de datos definidos para el proceso de consulta de reservas. A continuación se

muestra el diagrama de clases con la información intercambiada entre la agencia de viajes y el cliente y su explicación respectiva.

Figura 81. Diagrama de Clases – Cliente, Consulta de Reservas

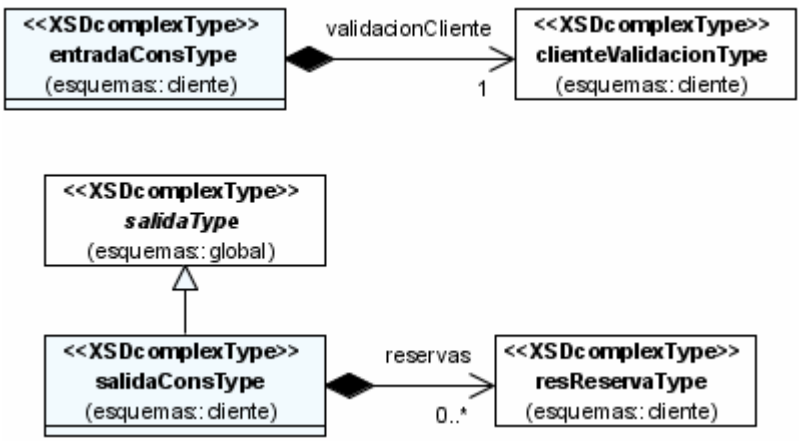


Tabla 64. Descripción de Clases – Cliente, Consulta de Reservas

Descripción Clases – Consulta de Reservas para Clientes Información descrita en Tabla 30. Datos en Solicitud de Consulta de Reservas y Tabla 31. Datos en Respuesta de Consulta de Reservas.	
Clase y Paquete	***Descripción***
EntradaConsType *esquemas.cliente*	Tipo complejo que representa los datos de entrada de la operación consultar reservas, según lo expresado en la tabla 30
SalidaConsType *esquemas.cliente*	Tipo complejo que encapsula los resultados de la consulta de reservas, según lo definido en la tabla 31. Extiende del tipo SalidaType.

- ***Definición de Interfaces WSDL***

El proceso de consulta de reservas debe incluirse en la interfaz WSDL existente para el cliente, extendiendo su funcionamiento, para esto se adiciona la operación *consultarReservas*; la definición es similar a la de los otros procesos. A continuación se describen los cambios de la interfaz.

Tabla 65. Interfaz WSDL del Cliente: Consulta de Reservas

Elemento	Descripción del Contenido
Types	Extiende el esquema local, adicionando los elementos usados en los mensajes de la operación consultarReservas. - ***acl:entradaConsultaReservas***, elemento de tipo cls:entradaConsType con los datos del cliente a consultar sus reservas. - ***acl:salidaConsultaReservas***, elemento de tipo cls:salidaConsType con la información de las reservas existentes del cliente.
Message	Define los nuevos mensajes a transmitir con el cliente en la operación consultarReservas. Para las excepciones, se reutilizan los definidos en la interfaz de búsqueda. - ***consultarReservasPeticion***: mensaje de la petición de la operación *consultarReservas*, su única parte: elemento entradaConsultaReservas. - ***consultarReservasRespuesta***: mensaje de respuesta de la operación *consultarReservas*, su única parte: elemento salidaConsultaReservas.
PortType	Define un la interfaz **ClienteConsultaReservasPortType** con la operación: - *consultarReservas*, que contiene: entrada: mensaje consultarReservasPeticion salida: mensaje consultarReservasRespuesta fallo “Cliente No Autorizado”: mensaje excepcionClienteNoAutorizado fallo “Operación Fallida”: mensaje excepcionOperacionFallida

- ***Participantes del Proceso y sus Vínculos***

El proceso de consulta de reservas sólo posee dos actores: la agencia de viajes y el cliente.

De forma similar a los procesos de búsqueda y reserva, la relación se da unilateralmente del cliente a la agencia. El nombre dado al partnerLinkType fue *clienteConsultaReservasPartnerLinkType* y utiliza el portType correspondiente *acl:ClienteConsultaReservasPortType* bajo el rol llamado *agenteviajesDelCliente*.
Sólo se define un vínculo interno del proceso de consulta de reservas: Cliente tomando el partnerLinkType descrito en el párrafo anterior y asignando el rol

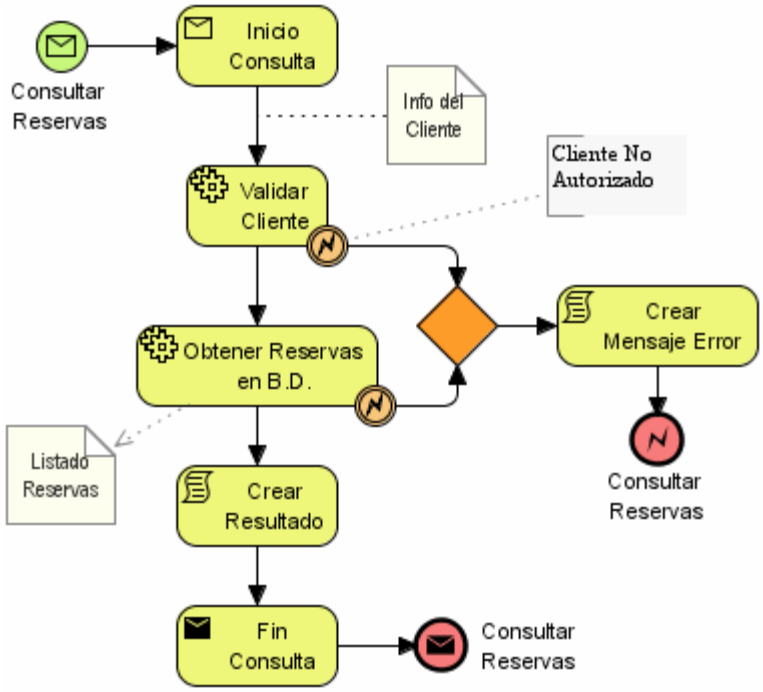
agenteviajesDelCliente como tipo myRole. Este es equivalente a los declarados en los procesos de búsqueda y reserva.

- ***Implementación del Proceso BPEL de Consulta de Reservas***

El proceso de consulta de reservas no había sido tenido en cuenta en el análisis preliminar dado que no presenta relación con los socios. El objetivo del proceso es obtener la información de reservas almacenadas en la agencia de viajes consultando internamente, por lo tanto es necesario primero que todo realizar validación de usuarios, luego realizar la consulta a nivel de base de datos obteniendo las reservas vigentes del usuario, para luego devolverlas como resultado del proceso.

El proceso de consulta de reservas planteado se muestra como un diagrama BPMN en la figura 82, y fue codificado en el archivo llamado consultareservas.bpel disponible en el código fuente.

Figura 82. Proceso de Negocio de Consulta de Reservas en la Agencia



- ***Especificación del Servicio Web Local***

El servicio Web local de la agencia se amplió para incluir la tarea compleja del proceso BPEL de la consulta de reservas.

Interfaz WSDL Local

Al enlace *AgenciaViajesLocalPortType* se le agrega una nueva operación, así.

Tabla 66. Especificación Tarea Local, Proceso Consulta de Reservas

Especificación de AgenciaViajesLocalPortType para Consulta de Reserva
Obtener Reservas en B.D. *(obtenerReservas)* *Descripción*: se encarga de consultar el listado de reservas vigente del cliente. *Entrada*: la identificación del usuario, es decir un elemento del tipo *normalizedString*. *Salida*: un elemento de tipo *cls:salidaConsType* con las reservas vigentes del cliente. *Excepciones*: - Fallo general, para capturar todos los errores inesperados; definida con un mensaje de tipo *ags:excepcionGeneralType*.

Tipos de Datos XML

Los tipos de datos usados para la nueva operación ya se han definido previamente.

Vínculos

De forma similar se debe definir un partnerLinkType que asocie este servicio Web local y el proceso actual, sin embargo corresponde al mismo definido en la fase 1. A nivel del proceso se declararse el partnerLink respectivo, igual al usado en el proceso de búsqueda y llamado *agenciaviajeslocal.*

Implementación de la Lógica

Cabe mencionar que esta tarea realiza la consulta al banco de datos definido para la agencia de viajes, expuesta en el Anexo C.. Los detalles de la implementación están disponibles apreciarse en el código fuente del desarrollo.

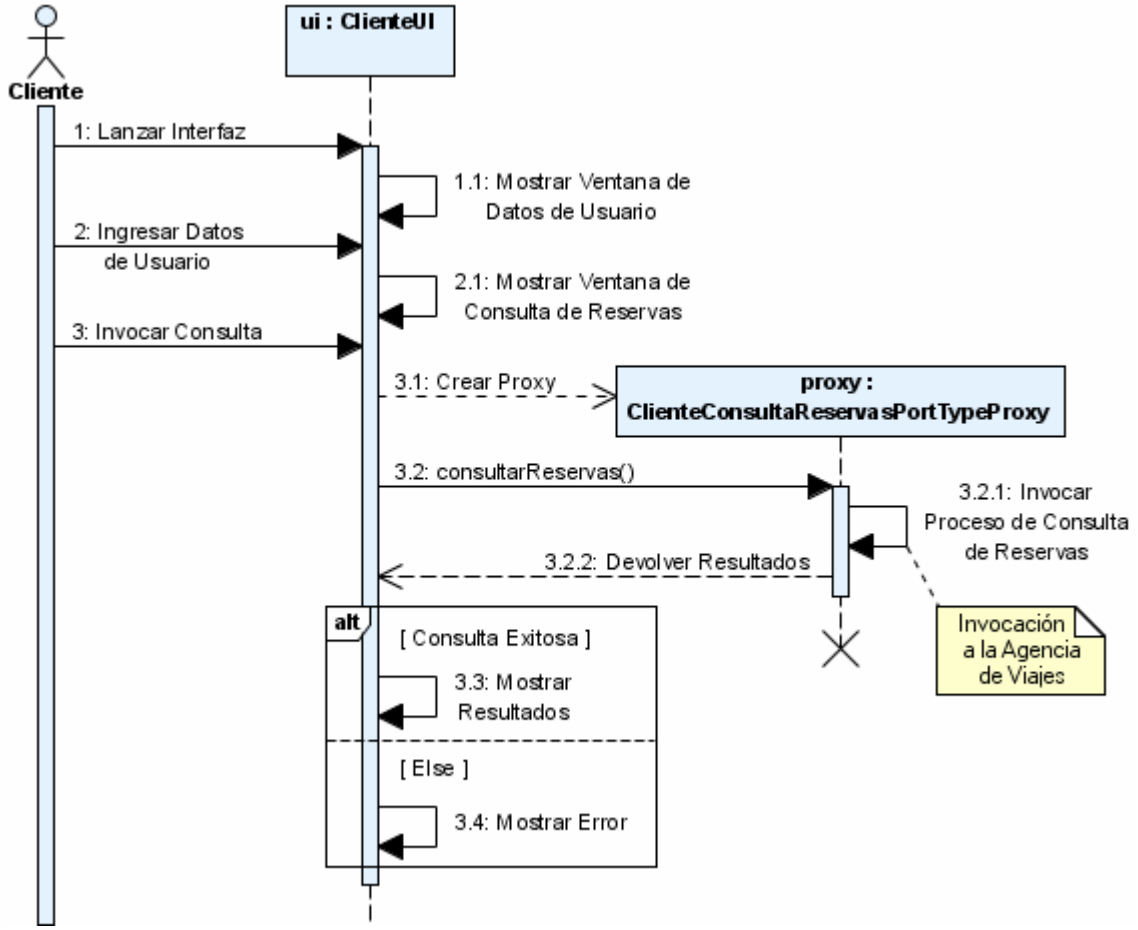
- ***Despliegue y Publicación del Proceso BPEL***

La publicación del proceso de consulta de reservas en el motor de Active BPEL se realiza mediante la definición del archivo descriptor de despliegue, en el cual se asigna al partnerLink Cliente el servicio ClienteConsultaReservasService con enlace de tipo MSG, tal cual como sucede con el partnerLink equivalente en los procesos de búsqueda y reserva. El resultado es la exposición del servicio asignado.

5.5.4.2 Implementación de la Interfaz del Cliente

La consulta de reservas en la interfaz del cliente debe realizar la captura de los datos del usuario y visualizar las reservas realizadas, luego de consultarlas.

Figura 83. Diagrama de Secuencia, Consulta de Reservas en Interfaz Cliente



En la anterior se muestra el diagrama de secuencia con la lógica de funcionamiento de la consulta de reservas en la interfaz del cliente. Su flujo de actividades es sencillo e involucra la captura de la información del usuario y luego la invocación de la operación consultarReservas a través del *proxy* de tipo ClienteConsultaReservasPortTypeProxy, generado con AXIS (de la misma forma que en la búsqueda y la reserva, se genera el conjunto de clases Stub, Locator, y las interfaces del servicio).

5.5.4.3 Pruebas de la Fase de Consulta de Reservas

Para esta fase se realizó una prueba, ejecutada según lo especificado en el Anexo D. Los resultados arrojaron resultados positivos en cuanto a funcionamiento básico aunque se detectaron las siguientes fallas:

- Algunos datos de las reservas no se muestran correctamente. Específicamente los campos período de la reserva que aparece incompleto, y sillas de los vuelos que siempre aparece en blanco.
- En las consultas que el sistema indicaba un número de reservas mayor que 1, pero sólo mostraba información para la primera realizada.

Estos problemas detectados corresponden a defectos de la interfaz que se corrigieron de inmediato, sin embargo el error del campo en blanco era un error interno a nivel de las tareas de la agencia de viajes local, que fue revisado y corregido.

5.5.5 Fase 4. Confirmación y Cancelación de Reservas

El cliente debe realizar un paso adicional luego de reservar planes de viaje: confirmarlos definitivamente, o declinar de ellos cancelándolos y liberando los

recursos. En este proceso intervienen los tres actores principales: cliente, agencia de viajes y socios.

5.5.5.1 Desarrollo del Proceso en la Agencia de Viajes

El desarrollo de este proceso en la agencia presenta grandes similitudes con el proceso de búsqueda y con el de reserva en lo referente a la forma de interacción entre los actores y la configuración de despliegue, por tanto los detalles de implementación serán exclusivos para los cambios, las nuevas definiciones y para lo demás se hará referencia.

- ***Esquemas XML de la Información***

Los esquemas XML ya definidos se amplían nuevamente con los tipos de datos creados para el proceso de confirmación de reservas. En las siguientes figuras y tablas se muestran y explican uno a uno los diagramas de clases de la información intercambiada entre los socios y la agencia y entre la agencia y el cliente.

Figura 84. Diagrama de Clases – Aerolínea, Confirmación Reservas

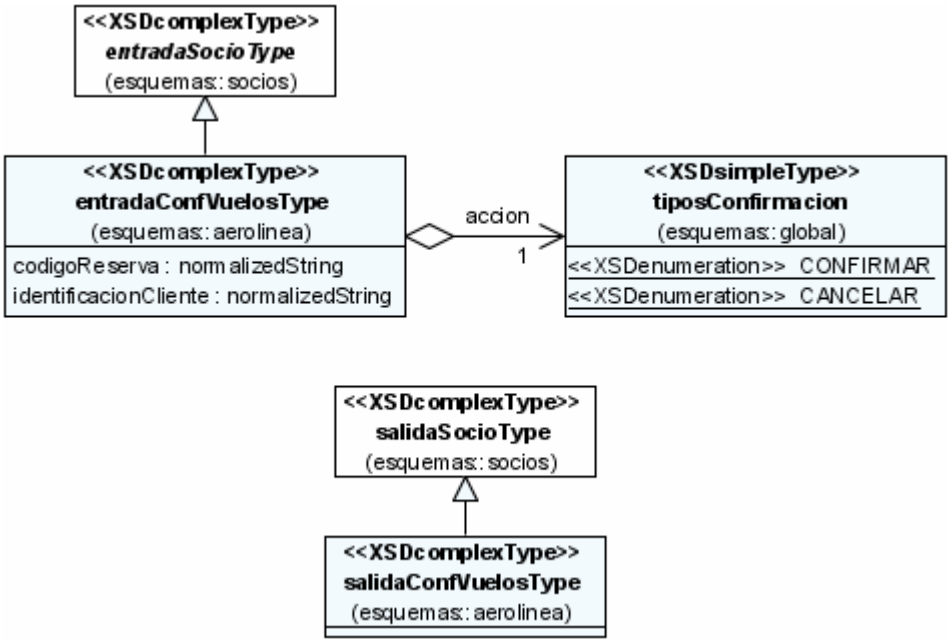


Tabla 67. Descripción de Clases – Aerolínea, Confirmación Reservas

Descripción Clases – Confirmación de Reserva en Aerolíneas Información descrita en Tabla 10. Datos en Solicitud de Confirmación de Reserva de Tiquetes y Tabla 11. Datos en Respuesta de Confirmación de Reserva de Tiquetes.	
Clase y Paquete	***Descripción***
EntradaConfVuelosType *esquemas.aerolínea*	Tipo complejo con los datos de entrada de la confirmación de reservas de vuelos, definido en la tabla 10.
TiposConfirmacion *esquemas.global*	Tipo simple String que enumera las acciones de la operación: CONFIRMAR, CANCELAR.
SalidaConfVuelosType *esquemas.aerolínea*	Tipo complejo con el resultado de la operación, como se define en la tabla 11. Extiende del tipo SalidaSocioType.

Figura 85. Diagrama de Clases – Cadena Hotelera, Confirmación Reservas

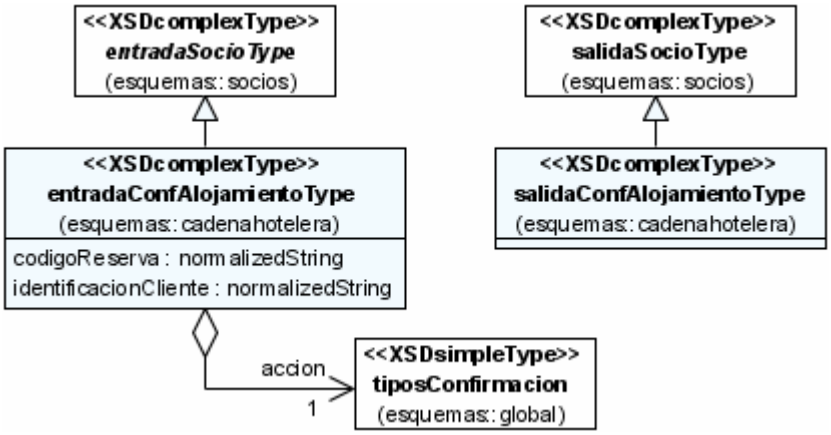


Tabla 68. Descripción de Clases – Cadena Hotelera, Confirmación Reservas

Descripción Clases – Confirmación de Reserva en Cadena Hotelera Información descrita en Tabla 15. Datos en Solicitud de Confirmación de Reserva de Alojamiento y Tabla 17. Datos en Respuesta de Confirmación de Reserva de Alojamiento.	
Clase y Paquete	***Descripción***
EntradaConfAlojamientoType *esquemas.cadenahotelera*	Tipo complejo con los datos para la confirmación de reservas de alojamientos, como se define en la tabla 15
SalidaConfAlojamientoType *esquemas.cadenahotelera*	Tipo complejo con el resultado de la operación, como se define en la tabla 17. Extiende del tipo SalidaSocioType.

Figura 86. Diagrama de Clases – Alquiler Vehículos, Confirmación Reservas

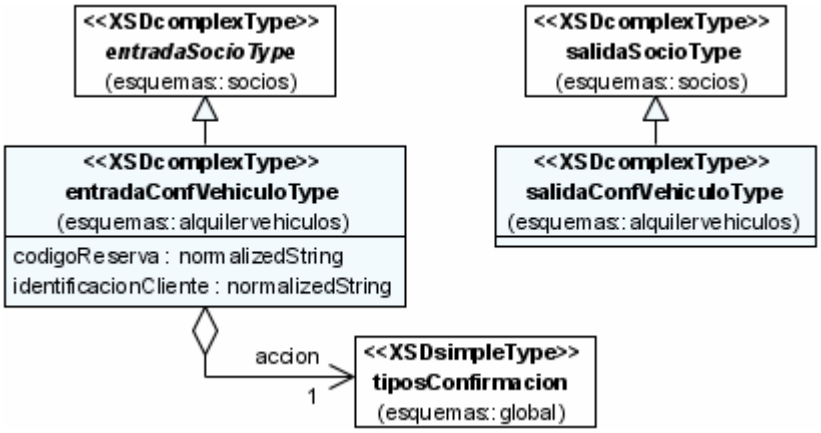


Tabla 69. Descripción de Clases – Alquiler Vehículos, Confirmación Reservas

Descripción Clases – Confirmación de Reserva en Alquiler de Vehículos Información descrita en Tabla 22. Datos en Solicitud de Confirmación de Reserva de Vehículo y Tabla 23. Datos en Respuesta de Confirmación de Reserva de Vehículos.	
Clase y Paquete	***Descripción***
EntradaConfVehiculoType *esquemas.alquilervehiculos*	Tipo complejo con los datos para la confirmación de reservas de vehículos, como se define en la tabla 22
SalidaConfVehiculoType *esquemas.alquilervehiculos*	Tipo complejo con el resultado de la confirmación de reserva de vehículos, como se especifica en la tabla 23. Extiende del tipo SalidaSocioType.

Figura 87. Diagrama de Clases – Cliente, Confirmación de Reserva

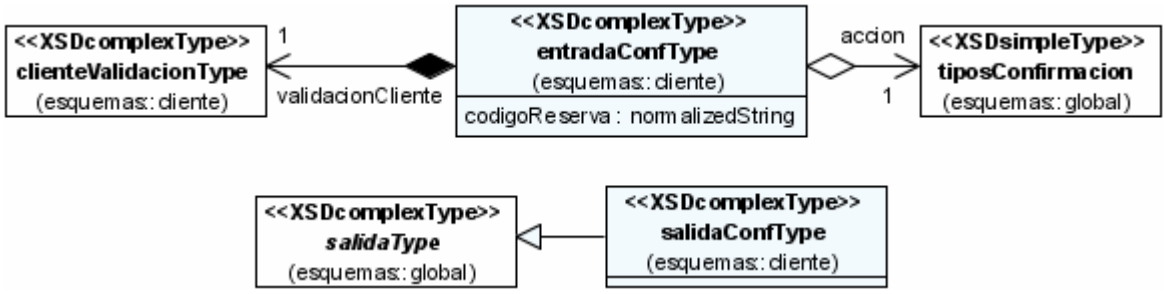


Tabla 70. Descripción de Clases – Cliente, Confirmación Reservas

Descripción Clases – Confirmación de Reserva para Clientes Información descrita en Tabla 28. Datos en Solicitud de Confirmación de Reserva de Plan de Viaje y Tabla 29. Datos en Respuesta de Confirmación de Reserva de Plan de Viaje.	
Clase y Paquete	***Descripción***
EntradaConfType *esquemas.cliente*	Tipo complejo con los datos de entrada para la confirmación de reservas de planes de viaje, como se define en la tabla 28
SalidaConfType *esquemas.cliente*	Tipo complejo con el resultado de la confirmación de reservas de planes de viaje, como se especifica en la tabla 29. Extiende del tipo SalidaType.

- ***Definición de Interfaces WSDL***

Todas las operaciones relacionadas con la confirmación de reservas en los socios se definieron asíncronas (similar a la búsqueda en los socios), mientras que para el cliente, la operación es síncrona.

Para la petición de confirmación de reservas a los socios, se extienden los documentos WSDL ya definidos para cada uno, agregando la nueva operación *confirmarReserva* con mensaje de entrada del tipo correspondiente. De la misma forma sucede con los documentos relacionados con la respuesta de los socios, en los cuales se definen nuevos portTypes con las operaciones *confirmarReservaOk* y *confirmarReservaError*, cada una con el respectivo mensaje del tipo correspondiente según sea el caso.

En relación con la interfaz del cliente, se define un nuevo portType llamado *ClienteConfirmacionReservaPortType* con la operación confirmarReserva, configurada de forma similar a la búsqueda y la reserva.

- ***Participantes del Proceso y sus Vínculos***

Los partnerLinkTypes requeridos para expresar los vínculos entre cada socio y el proceso, presentan una estructura similar a los definidos para la búsqueda y la

reserva: rol del socio (con el portType de petición) y el rol de agenteviajes (con el portType de respuesta).

La relación con el cliente siempre es la misma, se define sólo el rol agenteviajesDelCliente con el portType ClienteConfirmacionReservasPortType.

Los vínculos a nivel del proceso son idénticos a los definidos en la búsqueda (*véase tabla 47*), solo que cambian las referencias a los partnerLinkTypes de confirmación de reservas.

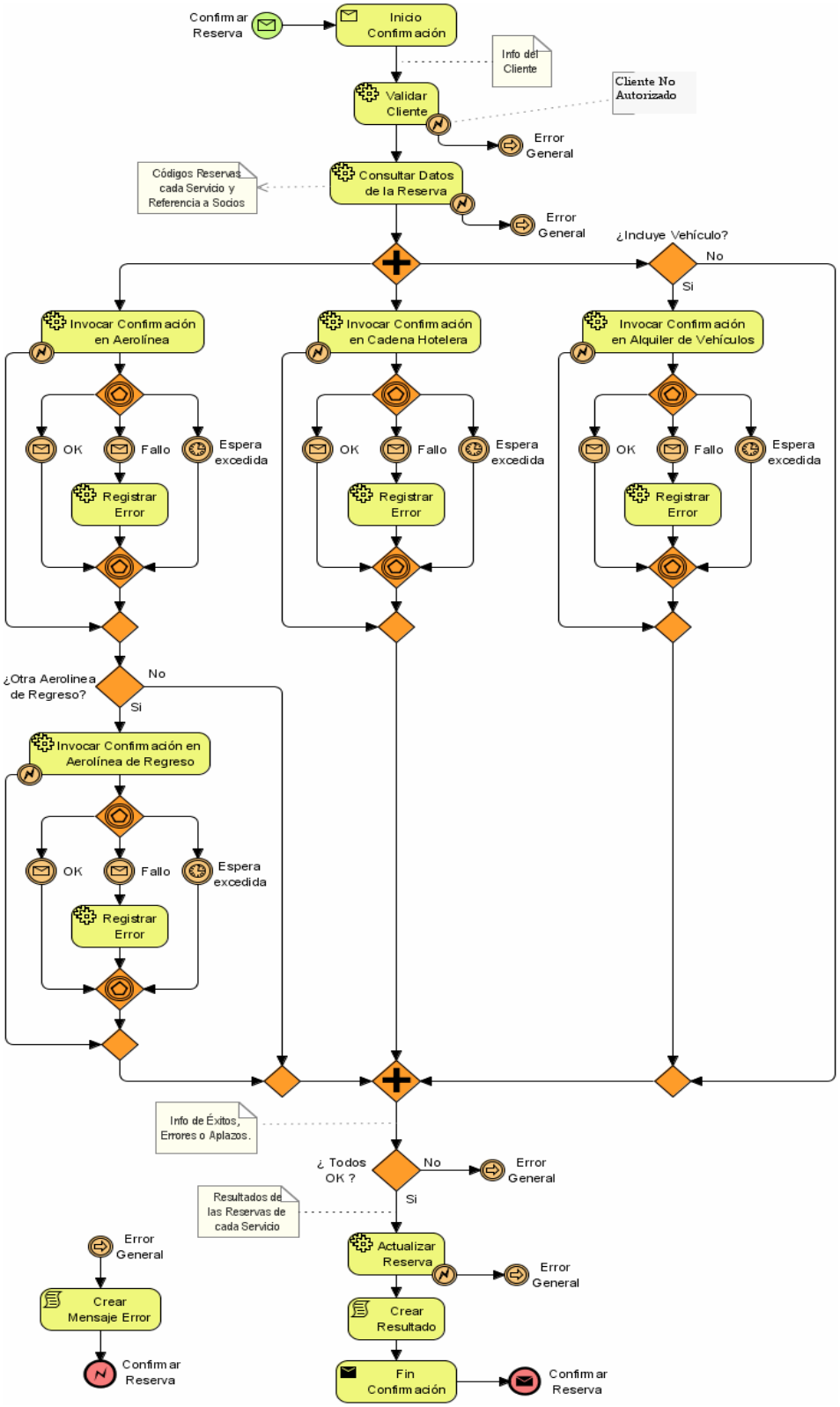
- ***Implementación del Proceso BPEL de Confirmación de Reservas***

El proceso de confirmación de reservas se implementa según lo analizado previamente para esta funcionalidad (*véase 5.4.3.3*) mediante un proceso BPEL. Es necesario contactar por separado a cada socio relacionado con la reserva que se quiere confirmar o cancelar, y realizar la petición de modificación transfiriendo los datos del servicio correspondiente.

En la figura 88 se aprecia el proceso BPEL planteado e implementado para la confirmación de reservas de planes de viaje.

Este proceso es muy similar en estructura al de reserva efectiva (*véase fase 2*) ya que debe contactar de nuevo a los socios involucrados con la reserva para realizar la confirmación o cancelación. La información de los socios y de la reserva como tal se extrae de los datos almacenados internamente previa validación del cliente, y luego que se contactan los socios en forma paralela, el proceso realiza la actualización interna de la reserva dependiendo del resultado de las operaciones en cada socio y finaliza enviando al cliente la respuesta del proceso. El contacto con el socio alquiler de vehículos es opcional, dependiendo de si la reserva incluye el servicio de vehículo o no.

Figura 88. Proceso de Negocio de Confirmación y Cancelación de Reservas



- ***Especificación del Servicio Web Local***

Las tareas complejas del proceso de confirmación de reservas fueron adicionadas al servicio Web local de la agencia, como se enuncia a continuación:

Interfaz WSDL Local

Las operaciones adicionadas al enlace *AgenciaViajesLocalPortType* se enuncian en la tabla 62; cada una posee además una excepción Fallo General.

Tabla 71. Especificación Tareas Locales, Proceso Confirmación de Reservas

Especificación de AgenciaViajesLocalPortType para Confirmación de Reservas
Consultar Datos de la Reserva *(obtenerSociosDetalleReservaAConfirmar)* *Descripción*: se encarga de obtener la información de la reserva expresada en términos de cada servicio de la reserva realizada: socio y detalle respectivo. *Entrada*: la misma entrada del proceso, un elemento de tipo *cls:entradaConfType*. *Salida*: un elemento de tipo *ags:infoSociosConfirmacionType* que contiene el detalle de la reserva dada con los datos del socio y los códigos de cada servicio de la reserva. *Excepciones*: - Código incorrecto, cuando el código de la reserva no es válido; definida con un mensaje de tipo *ags:excepcionCodigosIncorrectoType*.
Actualizar Reserva *(actualizarReserva)* *Descripción*: se encarga de actualizar el estado de la reserva según el éxito de las confirmaciones o cancelaciones en cada socio contactado. *Entrada*: el código de la reserva y la acción a realizar, definida como un elemento del tipo *ags:datosActualizacionReservaType*. *Salida*: un elemento de String con el mensaje de éxito en la operación.

Tipos de Datos XML

El esquema XML de la agencia de viajes local se complementó con los nuevos tipos de datos relacionados con el proceso. A continuación se muestra el diagrama de clases con los tipos de datos definidos y en seguida se explica cada una.

Figura 89. Diagrama de Clases – Agencia Local, Confirmación Reservas

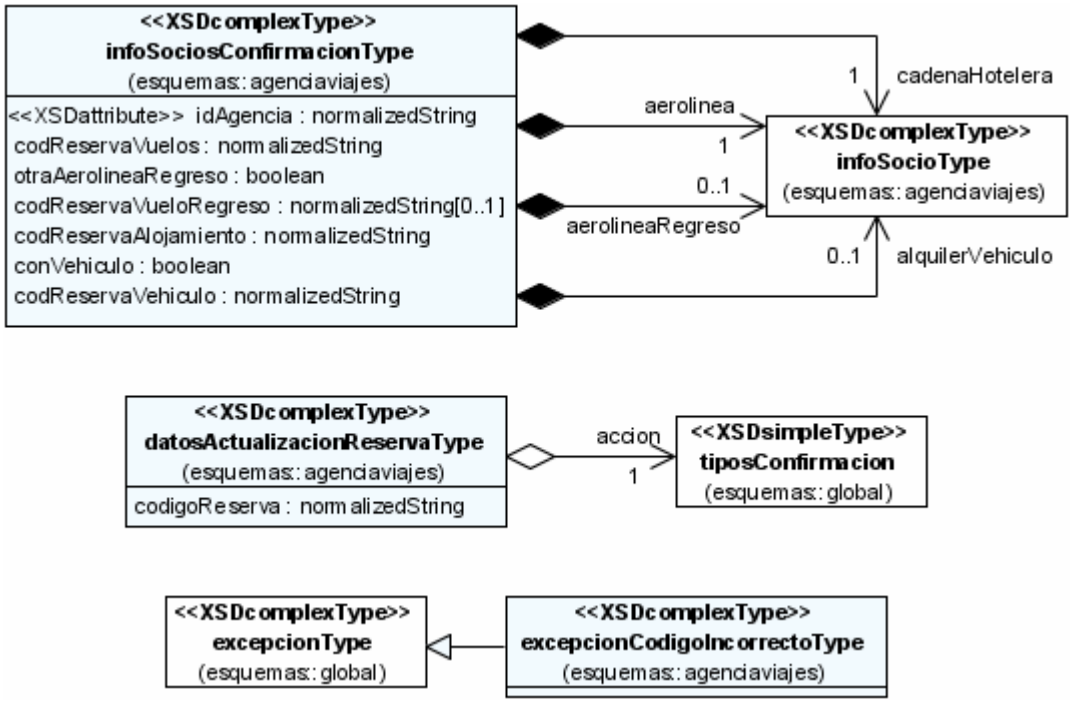


Tabla 72. Descripción de Clases – Agencia Local, Confirmación Reservas

Clase y Paquete	Descripción
InfoSociosConfirmacionType *esquemas.agenciaviajes*	Tipo complejo que encapsula el detalle de la reserva a confirmar: socio y códigos.
DatosActualizacionReservaType *esquemas.agenciaviajes*	Tipo complejo con el código de la reserva y el estado nuevo a definir internamente.
ExcepcionCodigoincorrectoType *esquemas.agenciaviajes*	Tipo complejo que extienden de excepcionType y representa la información de la excepción cuando se proporciona un código de reserva erróneo.

- ***Despliegue y Publicación del Proceso BPEL***

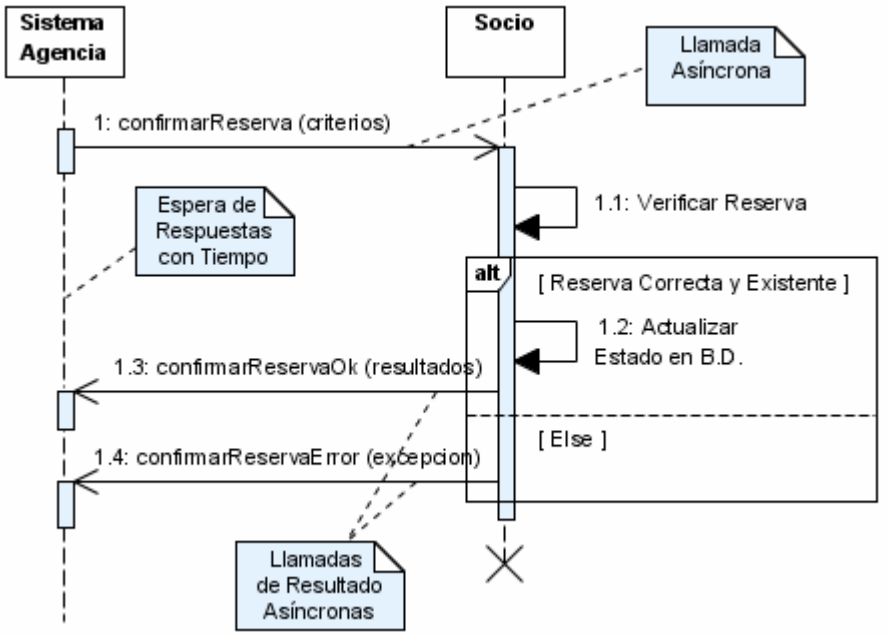
El despliegue y publicación del proceso de confirmación de reservas se realiza en forma análoga al del proceso de búsqueda, creando el archivo descriptor de despliegue con cada partnerLink definido y el resultado de la puesta en marcha en el motor de ActiveBPEL es la exposición de cuatro nuevos servicios Web:

AerolineaConfirmacionReservaService, *CadenaHoteleraConfirmacionReservaService*, *AlquilerVehiculosConfirmacionReservaService*, y *ClienteConfirmacionReservaService*, que son los puntos de entrada al proceso para los socios y para el cliente respectivamente.

5.5.5.2 Especificación de los Sistemas de los Socios

La funcionalidad de confirmar reservas de los servicios en los sistemas de los socios se expresa mediante la operación ***confirmarReserva***, cuya lógica interna general es bastante sencilla y similar para cada tipo de socio. En la siguiente figura se ilustra el procesamiento interno en los socios:

Figura 90. Diagrama de Secuencia, Confirmación de Reservas en Socios



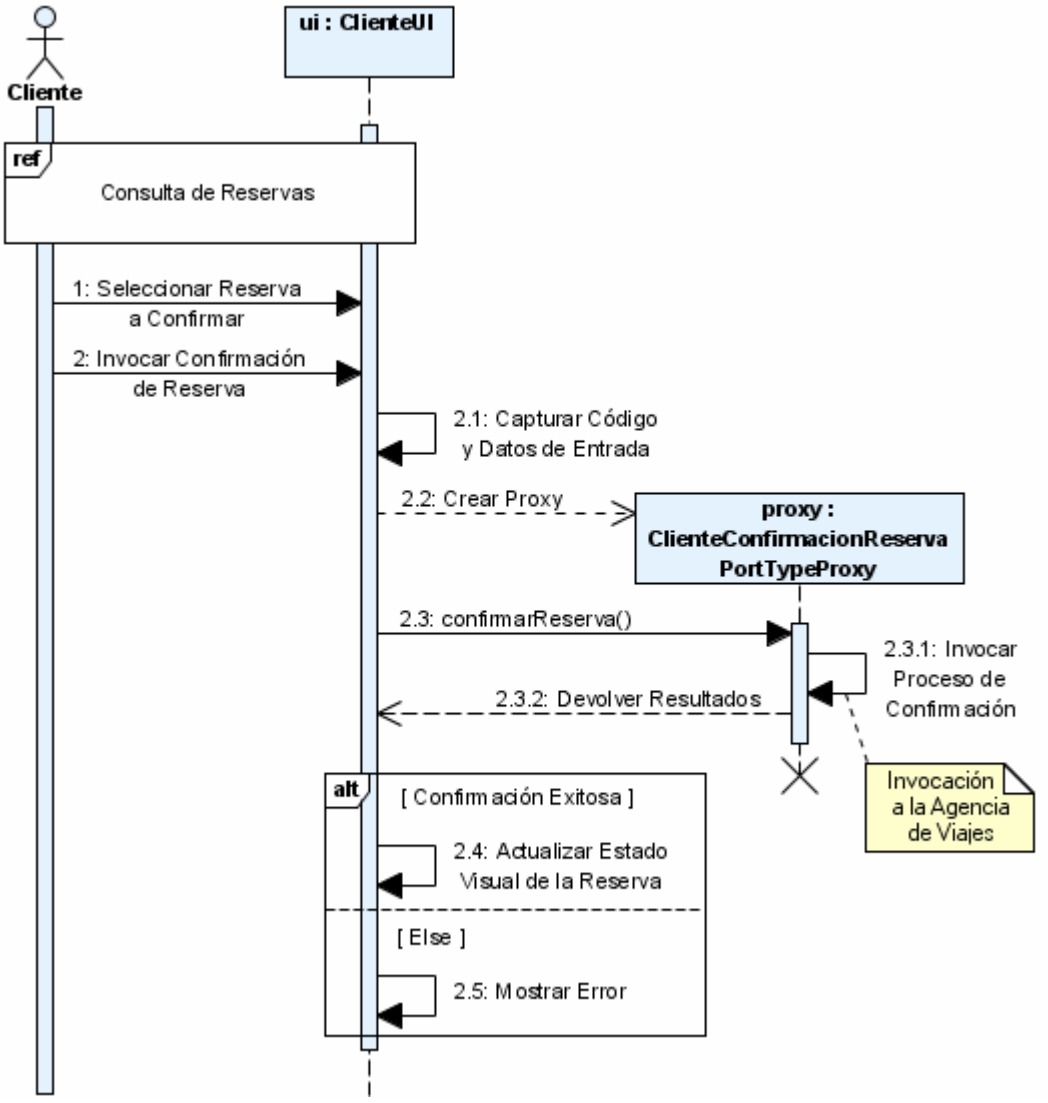
El flujo de actividades de es el siguiente: el inicio es la solicitud que realiza la agencia transmitiendo los criterios de la operación, el socio realiza en seguida la validación del código de la reserva para determinar si se trata de un código correcto; en caso exitoso, procede a la actualización del estado en la base de

datos y enviar el resultado a la agencia mediante el llamado a la operación confirmarReservaOk, en caso fallido se envía la información del error utilizando la operación confirmarReservaError.

5.5.5.3 Implementación de la Interfaz del Cliente

La confirmación de reservas en la interfaz del cliente se da luego de la consulta de las mismas, contactando la agencia para solicitar el cambio. En la siguiente figura se aprecia la lógica de su funcionamiento.

Figura 91. Diagrama de Secuencia, Confirmación Reservas en Interfaz Cliente



El punto de partida es la consulta de reservas invocada por el cliente previamente a lo que la interfaz responde listando cada una de las reservas, el cliente procede a seleccionar aquella que desea confirmar o cancelar entre las que están en estado RESERVA y realiza el llamado, la interfaz (ClienteUI) procede entonces a capturar el código de la reserva seleccionada y realizar el llamado de la operación *confirmarReserva* en el *proxy* de tipo ClienteConfirmacionReservaPortTypeProxy, cuya clase (y clases similares) son generadas por la utilidad de AXIS. El proxy se encarga de contactar la agencia e invocar el proceso respectivo, espera su resultado y lo devuelve a la interfaz; finalmente la interfaz interpreta el éxito de la operación y actualiza el estado de la reserva en pantalla, o muestra el error cuando la operación ha sido fallida.

En la implementación la clase ClienteUI se extendió agregando una operación relacionada con la funcionalidad: *realizarConfirmacionReserva()* encargada de lanzar el proceso cuando el usuario presione el botón de confirmación. Las clases generadas por AXIS presentan la estructura similar que ya se ha especificado en la fase 1, debido a que todas las operaciones con el cliente se definen de forma similar.

5.5.5.4 Pruebas de la Fase de Confirmación y Cancelación de Reservas

La prueba especificada para esta fase, definida en el Anexo D, fue ejecutada durante una sesión por todos los clientes especificados. Se obtuvieron resultados satisfactorios para cada ítem evaluado, junto con las siguientes observaciones relacionadas con mejoras de la interfaz:

- El estado de las reservas listadas no se aprecia fácilmente, debería resaltar para que el usuario lo capte de inmediato.
- Luego que se confirma una reserva deben deshabilitarse u ocultarse los botones de Confirmar o Cancelar, así no se realicen de nuevo los procesos.

5.5.6 Fase 5. Administración de la Información del Cliente

Los clientes interactúan con la agencia para modificar y actualizar la información personal y de acceso que poseen; en esta tarea, bastante sencilla pero necesaria para el escenario dado, sólo interactúan ellos dos.

5.5.6.1 Desarrollo del Proceso en la Agencia de Viajes

De acuerdo al planteamiento realizado, la administración de la información del cliente permite realizar consulta y actualización de los datos.

- ***Esquemas XML de la Información***

Los esquemas XML existentes para el cliente se amplían incluyendo los nuevos tipos de datos definidos para el proceso actual. A continuación se visualizan mediante un diagrama de clases y su correspondiente explicación:

Figura 92. Diagrama de Clases – Cliente, Administración Información

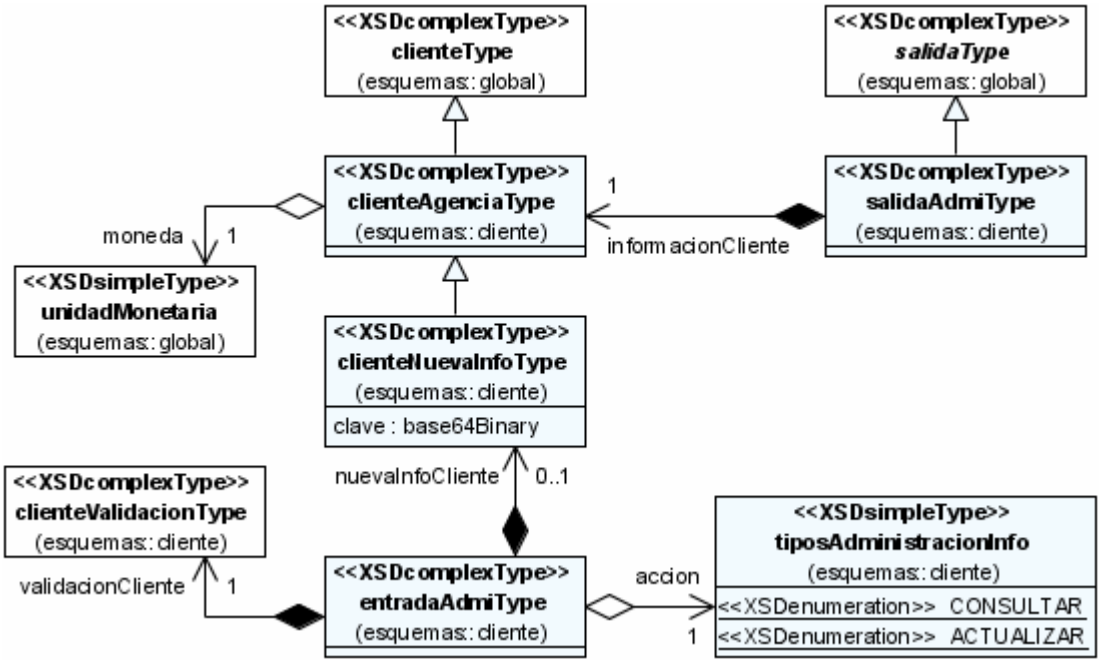


Tabla 73. Descripción de Clases – Cliente, Administración Información

Descripción Clases – Administración Información Clientes, Entrada y Salida Información descrita en Tabla 32. Datos en Solicitud de Administración de la Información del Cliente y Tabla 33. Datos en Respuesta de Administración de la Información del Cliente.	
Clase y Paquete	***Descripción***
ClienteAgenciaType *esquemas.cliente*	Tipo complejo que con los datos de los clientes en la agencia, extiende del tipo ClienteType y agrega la moneda.
ClienteNuevaInfoType *esquemas.cliente*	Tipo complejo que con los nuevos datos de los clientes cuando se van a actualizar. Extiende del tipo ClienteAgenciaType y agrega la contraseña.
TiposAdministracionInfo *esquemas.cliente*	Tipo simple String con la enumeración de las dos opciones de la administración: CONSULTAR y ACTUALIZAR
EntradaAdmiType *esquemas.cliente*	Tipo complejo que representa los datos de entrada de la operación, según lo expresado en la tabla 32.
SalidaAdmiType *esquemas.cliente*	Tipo complejo que encapsula los resultados del proceso de administración de información, según lo definido en la tabla 33. Extiende del tipo SalidaType.

- ***Definición de Interfaces WSDL***

El proceso de administración de información se incluye en la interfaz WSDL del cliente, mediante la adición de la operación *administrarInfoCliente* estos cambios son similares a los que sufre la interfaz en la consulta de reservas (*véase 5.5.4.1*): se definen nuevos tipos con los elementos de entrada y salida de la tarea de tipos cls:entradaAdmiType y cls:salidaAdmiType respectivamente, y se declara la el portType *ClienteAdministracionInfoPortType* con la operación indicada.

- ***Participantes del Proceso y sus Vínculos***

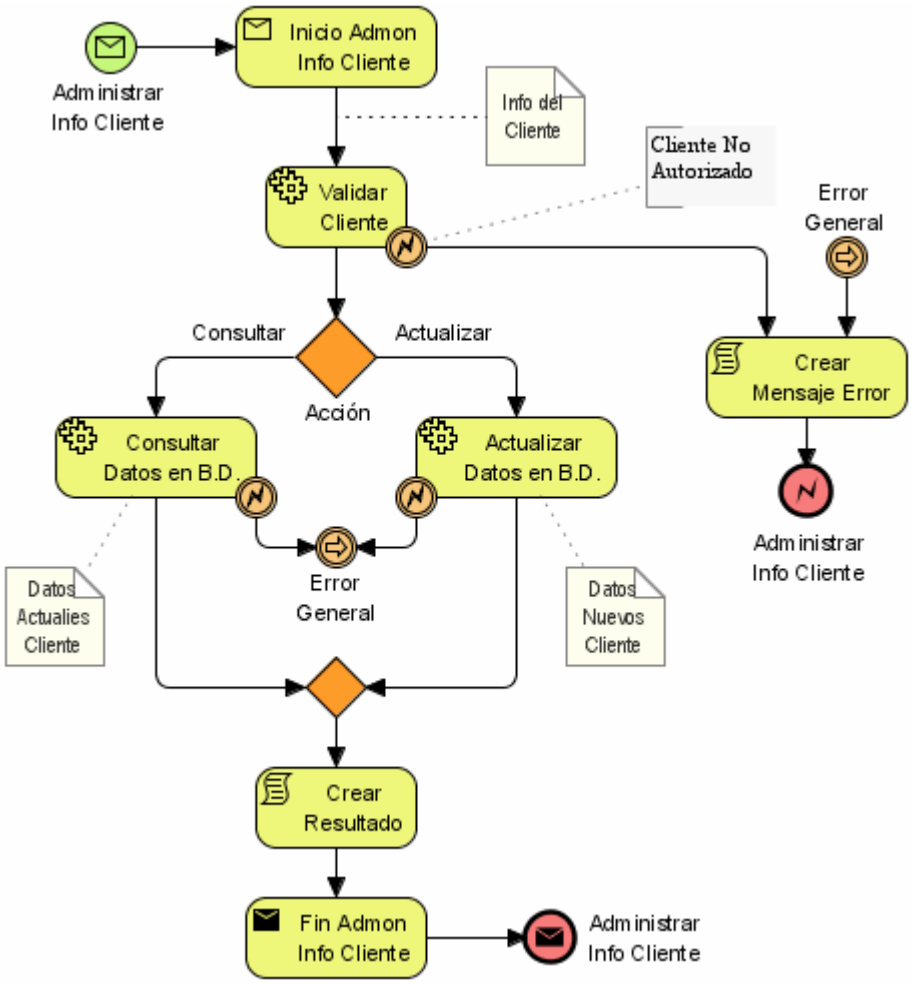
El proceso de administración de información del cliente posee sólo dos actores: la agencia de viajes y el cliente. El tratamiento y definición de los procesos es análogo al del proceso de consulta de reservas: declaración del partnerLinkType: *clienteAdministracionInfoPartnerLinkType* asociándolo al portType respectivo

ClienteAdministracionInfoPortType bajo el rol *agenteviajesDelCliente*. En el proceso se especifica el partnerRole Cliente asignándolo como tipo myRole.

- ***Implementación del Proceso BPEL de Administración de la Información del Cliente***

Al igual que el proceso de consulta de reservas, este proceso de administración de la información del cliente no se había analizado previamente a nivel interno.

Figura 93. Proceso de Negocio Administración de la Información del Cliente



La figura anterior presenta el diagrama BPMN del proceso planteado a partir del análisis efectuado: el fin del proceso es realizar dos tareas principales: consultar y

actualizar los datos de los clientes de la agencia; para esto se requiere en primer lugar validar la existencia del cliente y luego realizar la consulta o la actualización. Para el caso de la actualización se debe especificar la nueva información del cliente. En ambos casos el resultado del proceso será la información actualizada y reciente del usuario.

El proceso fue codificado en el archivo *administracioninfocliente.bpel*, disponible en el código fuente.

- ***Especificación del Servicio Web Local***

Las tareas complejas a implementar para la ejecución del proceso actual se adicionaron al servicio Web local de la agencia.

Interfaz WSDL Local, Tipos de Datos XML y Vínculos

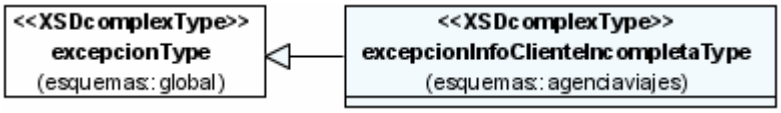
El enlace *AgenciaViajesLocalPortType* se amplía con las siguientes operaciones (cada una posee la excepción Fallo general):

Tabla 74. Especificación Tareas Locales Proceso Administración Información

Especificación de AgenciaViajesLocalPortType para Admin. Info. Cliente
Consultar Datos en B.D. *(obtenerInfoCliente)* *Descripción*: se encarga de consultar la información vigente del usuario en base de datos. *Entrada*: la identificación del usuario, es decir un elemento del tipo *xsd:normalizedString*. *Salida*: un elemento de tipo *cls:clienteAgenciaType* con toda la información del cliente.
Actualizar Datos en B.D. *(actualizarInfoCliente)* *Descripción*: realiza la actualización interna de la información del cliente. *Entrada*: la nueva información a almacenar, un elemento de tipo *cls:clienteNuevaInfoType* *Salida*: un elemento de tipo *String* con el resultado de la actualización. *Excepciones*: - Fallo información cliente incompleta, para capturar generar excepción cuando la información nueva esté incompleta o sea errónea; definida con un mensaje de tipo *ags:excepcionInfoClienteIncompletaType*.

Sólo un nuevo tipo de datos tuve que definirse y agregarse al esquema XML, *excepcionInfoClienteIncompletaType* perteneciente a *esquemas.agenciaviajes* usado para el mensaje de la excepción indicada, y extiende de excepcionType.

Figura 94. Diagrama de Clases – Agencia Local, Admón. Info. Cliente



En relación a los vínculos, se utiliza el partnerLinkType definido en la fase 1. Mientras que al nivel del proceso se declara el partnerLink *agenciaviajeslocal* asociándolo con tipo partnerRole.

Implementación de la Lógica

Las tareas ejecutan instrucciones de consulta y actualización de la información almacenada en las tablas relacionadas con los datos del cliente, en el banco de datos definido para la agencia de viajes, expuesta en el Anexo C.

- ***Despliegue y Publicación del Proceso BPEL***

El despliegue del proceso de administración de la información de clientes en el motor de ActiveBPEL se realiza definiendo el archivo descriptor de despliegue, en el cual se asigna al partnerLink Cliente el servicio ClienteAdministracionInfoService con enlace de tipo MSG expuesto en la publicación, tal cual como sucede con el partnerLink equivalente en los procesos de desarrollados anteriormente.

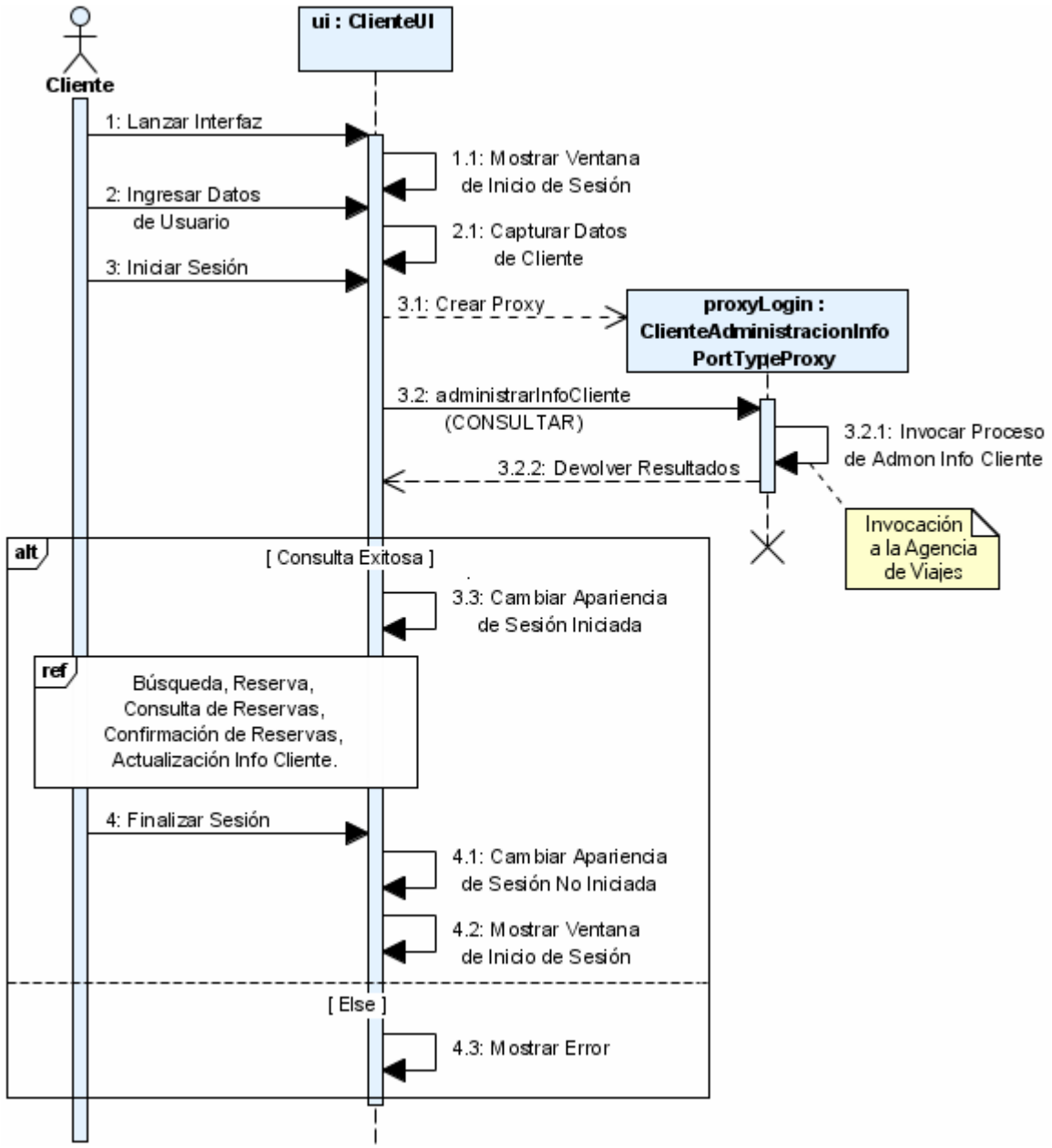
5.5.6.2 Implementación de la Interfaz del Cliente

La administración de la información en la interfaz del cliente se implementa de diferente forma según si se realiza consulta o actualización de datos. La consulta

se utiliza para permitir la existencia de una sesión de usuario, mediante una consulta inicial de los datos. Por otro lado, la actualización se da cuando el usuario desea realizar cambios en sus datos.

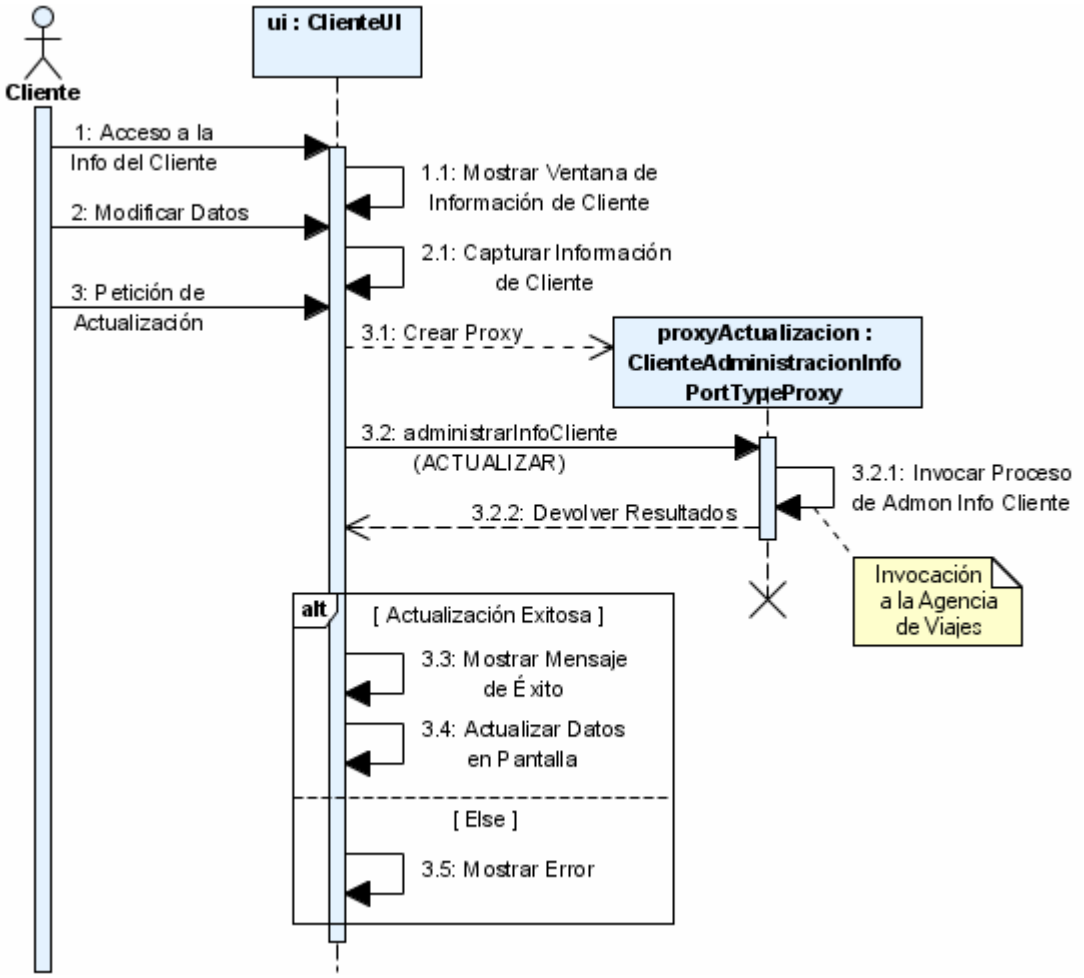
La siguiente figura muestra la secuencia para el inicio de sesión.

Figura 95. Diagrama de Secuencia, Sesión en la Interfaz de Cliente



El flujo de actividades inicia con el acceso a la interfaz del cliente (ClienteUI), a lo que ésta responde mostrando la ventana de Inicio de Sesión (datos de usuario), el cliente ingresa la información, la interfaz la recibe, y luego que el cliente haga la petición de inicio de sesión, la interfaz crea el *proxyLogin* con el que realiza la consulta de la información del cliente mediante la operación administrarInfoCliente con parámetro CONSULTAR. Si el resultado que el proxy devuelve es exitoso, la interfaz cambia a modo de *Sesión Iniciada* y queda a la espera de los demás casos de uso (Búsqueda, Reserva, Consulta de Reservas, Confirmación de Reservas) hasta que el usuario finaliza la sesión y la interfaz se restaura al modo inicial. Si el resultado del proxy es fallido, se muestra el error correspondiente.

Figura 96. Diagrama de Secuencia, Actualización Información Cliente



En la figura anterior aparece el diagrama de secuencias relacionado con la actualización de la información del cliente. El flujo de actividades comienza con el acceso a la ventana de Información de Usuario en la interfaz (ClienteUI), previo inicio de sesión, allí el cliente realiza las modificaciones que considere necesarias y posteriormente invoca la actualización con el botón Modificar Información. La interfaz recolecta la nueva información y crea el *proxyActualizacion* con el que invoca el proceso administracionInfoCliente con parámetro ACTUALIZAR. Según el resultado que devuelva el proxy, la interfaz muestra un mensaje de confirmación.

5.5.6.3 Pruebas de la Fase de Administración de la Información del Cliente

Las pruebas para la fase 5, definidas en el Anexo D, se ejecutaron durante dos sesiones por cada uno de los clientes especificados. Durante la sesión uno se obtuvo resultados esperados operacionalmente hablando, sin embargo en cuanto a interfaz se destacan las siguientes observaciones:

- Luego de inicializar sesión en la aplicación, la interfaz se activa pero no es claro qué usuario está activo. Sólo se puede apreciar esto en la pestaña de Información de Usuario, donde se muestran sus datos.
- La modificación de la contraseña debería confirmarse con un campo adicional *repetir contraseña*.
- Cuando se realizan modificaciones no satisfactorias a los datos, la interfaz no restaura la información válida inicial que mostraba a menos que se inicie sesión de nuevo.

Estas observaciones se tradujeron en mejoras a la interfaz en cuanto a validación de datos y experiencia visual de usuario adicionando mensajes de información y la información de la sesión actual. Luego de esto se realizó la segunda sesión de pruebas, y los resultados fueron completamente satisfactorios.

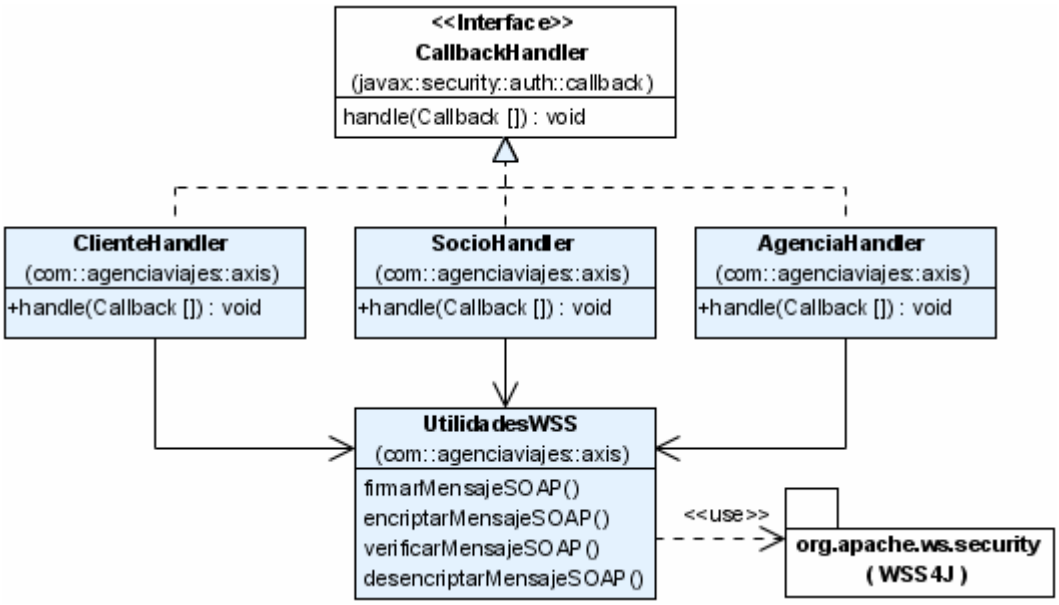
5.5.7 Fase 6. Implementación de Seguridad

Durante la determinación del escenario de desarrollo se especificaron los requisitos relacionados con la seguridad (*véase 5.3*), cruciales en aplicaciones del mundo del comercio electrónico.

La aplicación del estándar WS-Security a los desarrollos consistió en implementar manejadores personalizados de Axis (*Handlers*) encargados de interceptar los mensajes entrantes y salientes, y realizar el procesamiento respectivo de cada mensaje mediante el uso de la librería WSS4J.

Se definieron tres manejadores uno para cada tipo de sistema involucrado: *ClienteHandler*, *SocioHandler* y *AgenciaHandler*, ya que su funcionamiento es diferente según las especificaciones pactadas (*ver figura 24, página 89*). A continuación se muestra el diagrama con las clases implementadas.

Figura 97. Diagrama de Clases – Implementación WS-Security



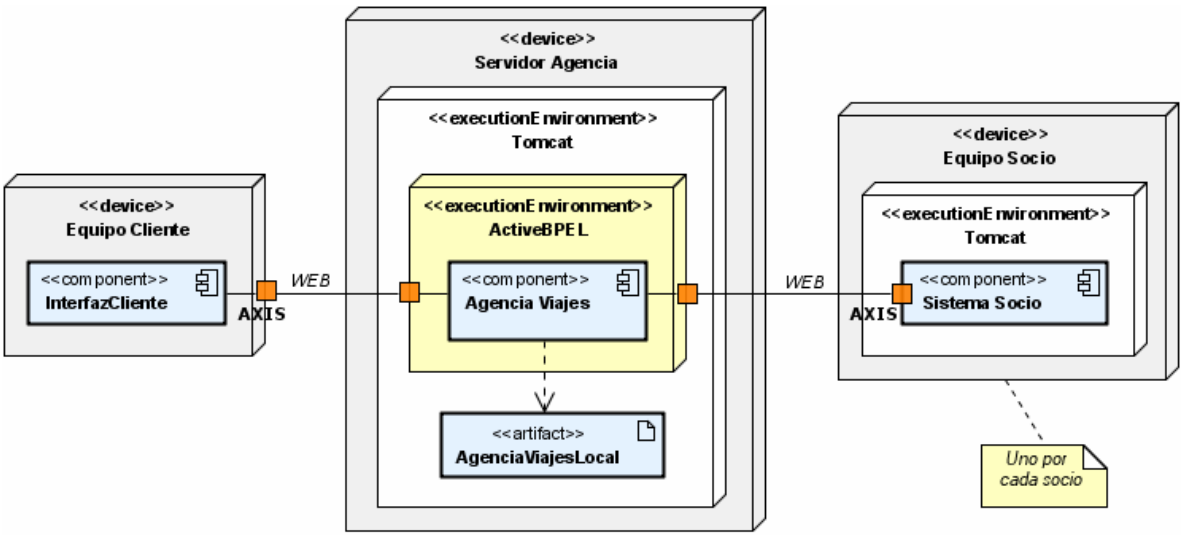
Los manejadores implementan la interfaz *javax.security.auth.callback.CallbackHandler* para definir el evento *handle* en la que se codifica la captura y procesamiento del mensaje, haciendo uso de la clase *UtilidadesWSS* que define las operaciones requeridas de firmado y cifrado de los mensajes.

Estas clases se despliegan en la plataforma AXIS residente en cada sistema tanto en la agencia como en el cliente y los socios, y allí operan a nivel de la capa de transporte y juegan un papel de intermediaros en la comunicación ya que los mensajes SOAP entre los actores se interceptan para aplicar y verificar la seguridad definida.

5.5.8 Fase 7. Despliegue Final

El prototipo desarrollado cuenta con tres componentes principales: Agencia de Viajes, Interfaz de Cliente y Sistemas de Socios (uno por cada tipo), desplegados tal como se aprecia en el siguiente diagrama:

Figura 98. Diagrama de Despliegue del Desarrollo



La descripción de cada componente desplegado es la siguiente:

- El componente de Agencia Viajes consta de los procesos BPEL con sus elementos y archivos relacionados que implementan las funcionalidades de la agencia de viajes. Este se despliega en el motor de ActiveBPEL que a su vez está inmerso en el servidor Web Tomcat.

  Existe además el elemento AgenciaViajesLocal que implementa la lógica interna de la agencia y es accedido por los procesos del componente Agencia Viajes. Este elemento se ubica en el servidor Web Tomcat.

  A su vez, el servidor Web Tomcat está instalado en el equipo que actúa como servidor de la Agencia.

- El componente Sistema Socio se refiere a la implementación interna de la lógica de cada socio, se visualiza sólo 1 pero en realidad existen múltiples, dependiendo de cuántos socios se instancien. Este componente se ubica en un servidor Web Tomcat con AXIS (aunque puede variar si se utiliza otra forma de implementación), que a su vez están en un equipo independiente. AXIS es utilizado para acceder por medio de servicios Web al sistema de los socios.

- El componente Interfaz Cliente se ubica separado en el equipo del cliente, y cuenta con librerías AXIS para poder realizar la comunicación con la agencia.

La comunicación entre ellos se da a través de la Web.

6. EJEMPLO DE UTILIZACIÓN DEL DESARROLLO

En este capítulo se realiza una demostración visual del funcionamiento general de la aplicación desarrollada y su interacción con los usuarios.

Para facilitar la lectura y comprensión de este ejemplo se va a suponer que existe una persona llamada "*Gilberto*" que es cliente afiliado a la agencia de viajes llamada "*AgenciaViajes.com*". A continuación se presenta una explicación más detallada de este escenario.

6.1 ESPECIFICACIONES DEL ESCENARIO

El cliente *Gilberto*, dispone en su computador instalada la aplicación (*Recursos de Cliente*) que le ha proporcionado la agencia de viajes para que realice desde cualquier parte los diferentes servicios que ella le ofrece, con sólo disponer de una conexión a internet.

Gilberto desea planificar las vacaciones de su familia integrada por 5 personas incluyéndolo; todos residen en Bucaramanga, Colombia, y han decidido viajar a Cancún, México a disfrutar de la playa por 5 días iniciando el 15 de Febrero. Ellos esperan poder encontrar un plan vacacional acorde con el presupuesto apretado disponible sin importar si sacrifican comodidades.

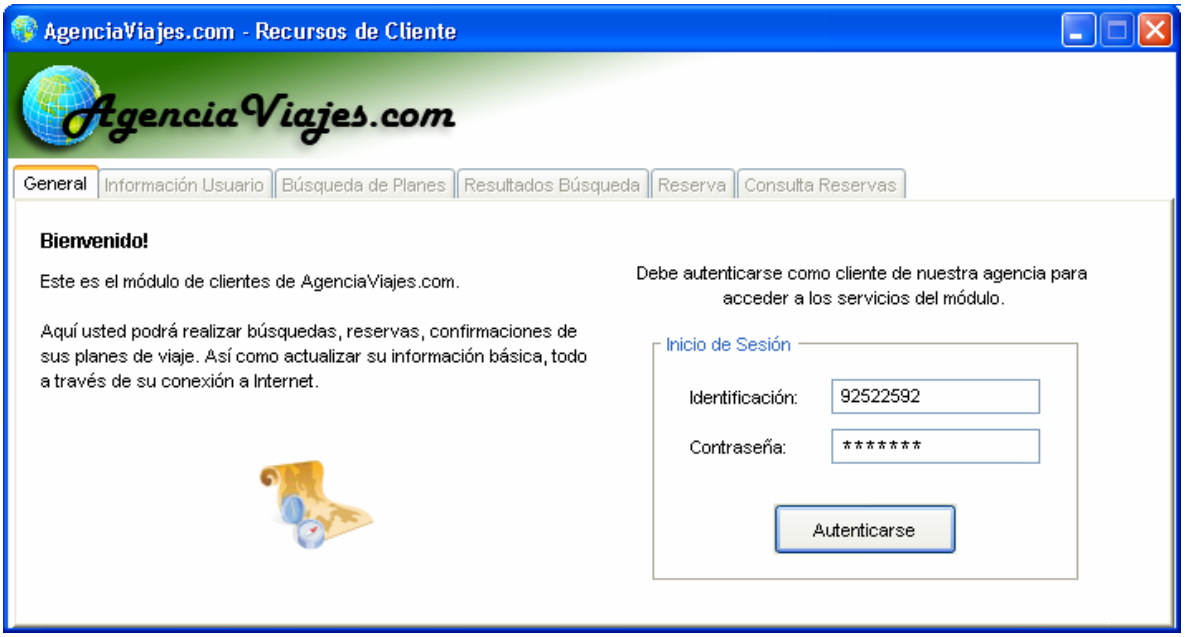
Utilizando la aplicación de la agencia, *Gilberto* busca los servicios que quiere de acuerdo a lo que considere su familia teniendo en cuenta que ninguno sabe manejar. Cuando encuentre los servicios, procede a realizar la reserva de los servicios y su posterior confirmación, para entonces poder alistar el viaje.

A continuación se presenta paso a paso el uso que *Gilberto* hace del sistema como cliente de la agencia de viajes.

6.2 SECUENCIA DE PASOS DEL EJEMPLO

Primero que todo, *Gilberto* lanza la aplicación de la agencia de viajes que tiene en su computador, en la figura 99 se muestra la ventana inicial que el sistema presenta. Allí él debe digitar sus datos de acceso: identificación y contraseña (previamente proporcionados por la agencia) en la sección de “Inicio de Sesión” de la pestaña General que es la única activa.

Figura 99. Interfaz de Cliente: Acceso Inicial

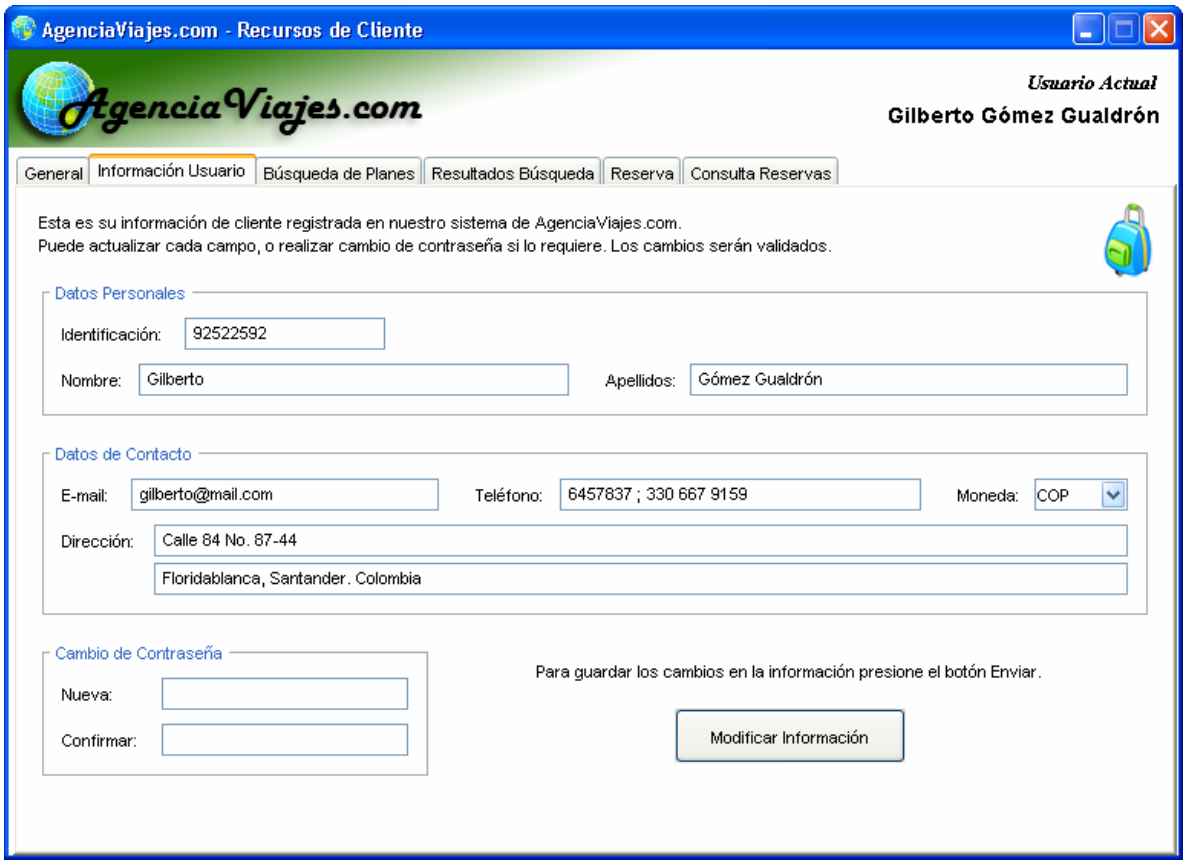


Al presionar el botón *Autenticarse*, el sistema inicia sesión para el cliente *Gilberto*, habilitando todas sus funciones tal como se muestra en la figura 100.

Figura 100. Interfaz de Cliente: Sesión iniciada

Gilberto explora la interfaz activando la pestaña de Información Usuario, allí puede ver sus datos personales que no necesitan ninguna actualización. En la siguiente figura se muestra la apariencia de la ventana.

Figura 101. Interfaz de Cliente: Información del Usuario

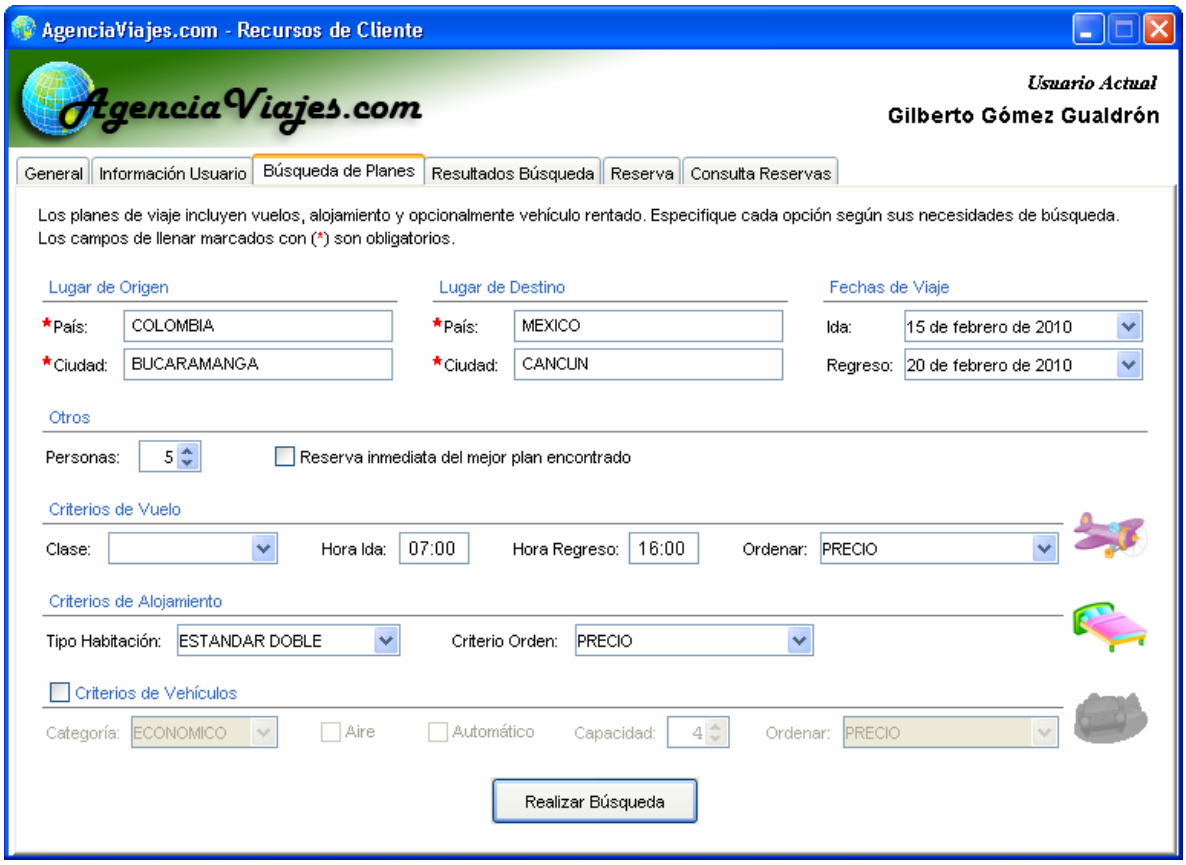


Para realizar la búsqueda, *Gilberto* activa la pestaña Búsqueda de Planes en la cual debe especificar los criterios de los servicios de su plan de viajes, así:

- Origen en Bucaramanga, Colombia y Destino en Cancún, México
- Viaje de Ida el 15/Febrero/2010, con Regreso el 20/Febrero/2010
- Para 5 personas
- Volando en cualquier clase, con ida en la mañana y regreso en la tarde.
- Se prefieren habitaciones dobles.

En la figura 102 se aprecia la interfaz en la que especifica los criterios de la búsqueda que han definido.

Figura 102. Interfaz de Cliente: Criterios de Búsqueda de planes de viaje



Luego, mediante el botón *Realizar Búsqueda* se realiza la búsqueda en la agencia de viajes de los servicios de vuelo y alojamiento. Luego de un momento, la aplicación responde con un mensaje de búsqueda exitosa, tal como se muestra en la figura 103.

Posteriormente, en la pestaña *Resultados Búsqueda*, se listan los diferentes servicios encontrados que cumplieron los requisitos que se especificaron, agrupándolos por tipo mediante pestañas: vuelos de ida, vuelos de regreso, alojamientos y vehículos (no usado actualmente), tal como se puede apreciar en la figura 104.

Figura 103. Interfaz de Cliente: Éxito de la búsqueda

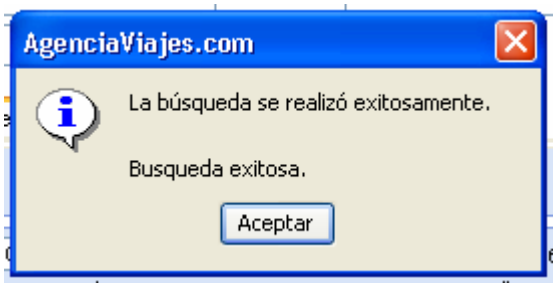
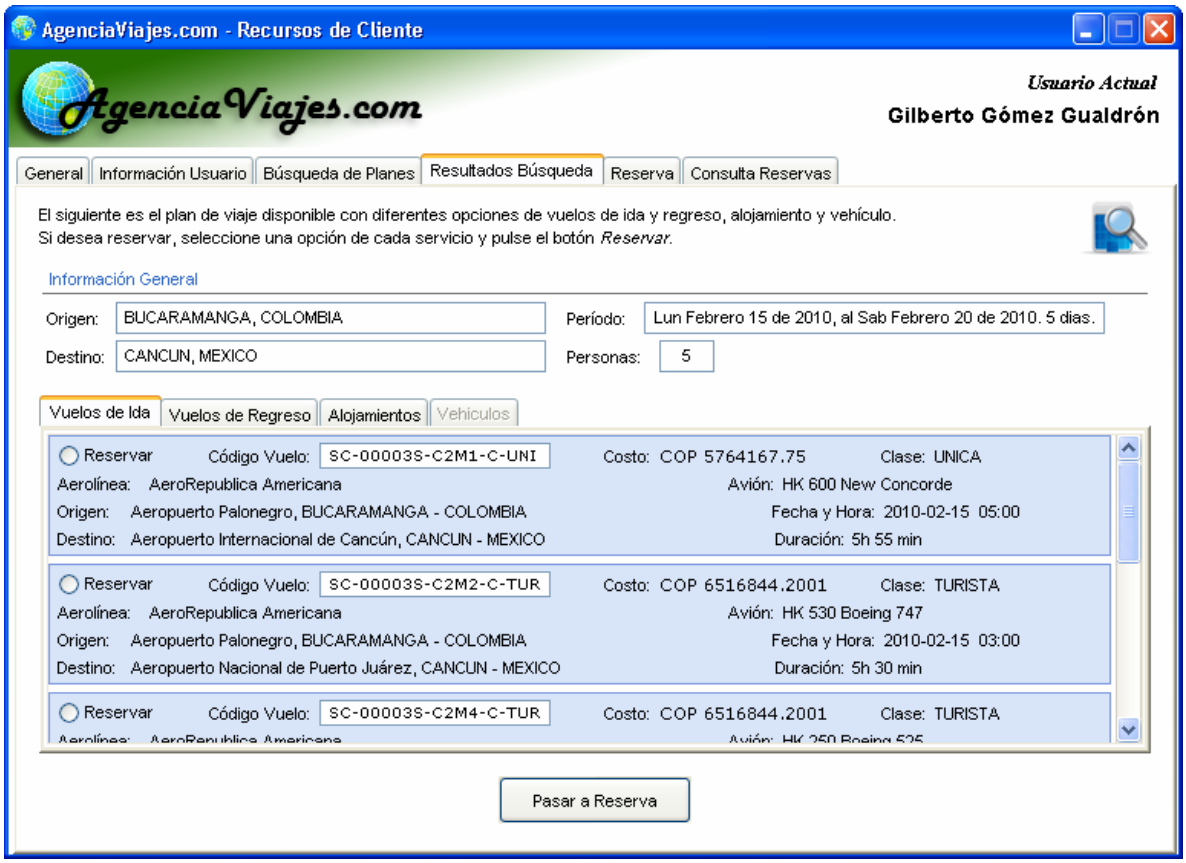
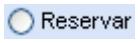


Figura 104. Interfaz de Cliente: Resultados de la búsqueda



Para decidir qué servicios seleccionar, *Gilberto* puede apreciar las características de cada uno destacando que los precios están expresados en pesos colombianos,

ya que ésta es la moneda de su preferencia; además los resultados cada conjunto están ordenados por precio de menor a mayor pues así se definió en los criterios de búsqueda.

El plan más económico lo conforman los primeros servicios en lista, por lo tanto *Gilberto* debe seleccionar éstos utilizando la opción de selección *Reservar* 

que aparece comenzando cada servicio listado. Una vez haya seleccionado todos los servicios, se presiona el botón *Pasar a Reserva* para que el sistema tome automáticamente los códigos de los servicios y los liste en la pestaña *Reserva*, como se puede apreciar en la siguiente figura:

Figura 105. Interfaz de Cliente: Reserva del plan de viaje

El sistema automáticamente define también las fechas de la reserva y la cantidad de personas, aunque *Gilberto* podría especificar cada uno de estos campos por sí mismo sin necesidad de hacer una búsqueda previa, sino digitando directamente los códigos.

El paso siguiente es la realización de la reserva de los servicios, esto se realiza presionando el botón *Reservar*, para que la aplicación contacte la agencia vía internet y le dé la orden de reservar los servicios. Al paso de unos segundos, *Gilberto* recibe respuesta exitosa del proceso de reserva mediante un mensaje como el mostrado en la siguiente figura:

Figura 106. Interfaz de Cliente: Éxito de la reserva

Hasta este punto *Gilberto* ya realizó la reserva del plan de viaje que seleccionó, pero él sabe que esta reserva no es definitiva aún y puede ser cancelada o confirmada según lo requiera.

Gilberto está seguro que este plan es el adecuado y por ende debe confirmar su reserva. Para esto debe dirigirse a la pestaña *Consulta Reservas* con el fin de realizar la consulta de la reserva que realizó previamente. En la figura 107 se aprecia la apariencia de la ventana en este punto. Allí se realiza la consulta presionando el botón *Consultar Reservas*, la interfaz se comunica con la agencia y obtiene el listado de las reservas del cliente.

Figura 107. Interfaz de Cliente: Consulta de Reservas

El resultado de la consulta de reservas se muestra luego en pantalla con un mensaje diciendo que el cliente posee 1 reserva vigente, tal cual como aparece en la siguiente figura:

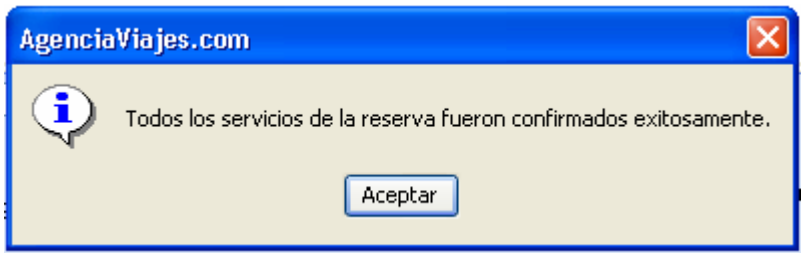
Figura 108. Interfaz de Cliente: Éxito de la consulta de reservas

Y en seguida la interfaz visualiza los detalles del plan reservado: origen, destino, fechas, costo total, estado, y también la información de cada servicio incluido: vuelo de ida, vuelo de regreso y alojamiento (el vehículo no se incluyó). En la figura 109 puede apreciarse la interfaz con los resultados de la reserva.

Figura 109. Interfaz de Cliente: Resultados consulta de reservas

Para finalizar el proceso de reserva, *Gilberto* debe confirmar a la agencia de viajes el plan reservado. Para esto se utiliza el botón *Confirmar*, si por el contrario quisiera cancelar la reserva, debe hacerlo con el botón *Cancelar*, de la reserva dada. El resultado de la confirmación se muestra en la siguiente figura:

Figura 110. Interfaz de Cliente: Éxito de la confirmación de reservas



Después de mostrar el resultado de la operación de confirmación, la interfaz visualiza un mensaje y oculta las reservas que habían sido consultadas previamente. En la figura 111 se aprecia la interfaz luego de realizar la confirmación de la reserva.

Figura 111. Interfaz de Cliente: Resultados de confirmación de reservas

Gilberto desea revisar de nuevo el estado de su reserva y por tanto vuelve a realizar la consulta de reservas. La interfaz contacta de nuevo a la agencia y obtiene el listado de reservas del cliente con sus detalles, y luego los visualiza en pantalla. El resultado se puede apreciar en la figura 112.

Allí se destaca que el estado de la reserva ha cambiado de *SIN CONFIRMAR* al estado *CONFIRMADO*, y se han deshabilitado los botones con las acciones. Esta reserva está aceptada por completo y por lo tanto ya ha finalizado su ciclo en el sistema del cliente. *Gilberto* puede estar seguro que ya dispone de sus servicios reservados y sólo tiene que dirigirse a pagar el costo a la agencia.

Figura 112. Interfaz de Cliente: Consulta de reserva confirmada

Como último paso, *Gilberto* regresa a la pestaña General y finaliza la sesión en la interfaz, presionando el botón *Cerrar Sesión*, y posteriormente termina la aplicación.

Este es el final del ejemplo de utilización del sistema por parte de un cliente.

7. CONCLUSIONES

Al finalizar el presente trabajo de investigación se cuenta con el prototipo software funcional de una agencia de viajes electrónica diseñada mediante el uso de procesos de negocio BPEL y por ende implementada usando servicios Web XML para la comunicación con sus socios y clientes. De esta forma todos los objetivos planteados fueron cumplidos satisfactoriamente.

Los siguientes son los aspectos más relevantes del trabajo realizado dentro del proyecto de investigación:

- Se efectuó un estudio del estado del arte relacionado con las arquitecturas orientadas a servicios (SOA) analizando sus características e identificando las ventajas y desventajas frente a otras opciones de implementación de aplicaciones distribuidas, utilizadas en la Web y especialmente en el comercio electrónico. El objetivo fue utilizar esta arquitectura como base debido a la gran acogida que ha tenido con la evolución de la Web, y gracias al trabajo de investigación efectuado, se ha logrado apropiar gran cantidad de conocimiento y experiencia en estas tecnologías.

- Se planteó como escenario de estudio una agencia de viajes, que siendo un negocio representativo del comercio electrónico, presenta las características ideales para aplicar la arquitectura orientada a servicios a su diseño e implementación informática.

- El análisis estructural y funcional realizado a la agencia de viajes, fue plasmado a través del uso de diagramas UML de casos de uso, actividades, secuencia y clases; esto permitió realizar diseños que cumplen con estándares internacionales ampliamente aceptados y facilitan la comprensión de los

procesos de negocio y las interacciones presentes entre cada uno de los actores del escenario seleccionado.

- Se recolectó y analizó suficiente documentación acerca de los servicios Web, el lenguaje de ejecución de procesos de negocio BPEL, y los diversos estándares relacionados, con el objetivo de realizar los diseños y la implementación de los componentes Web requeridos en el desarrollo. Se tuvo en cuenta una gran variedad de opciones de implementación

- Se utilizaron tecnologías como Java, Axis y ActiveBPEL para la implementación del prototipo software que aplican los diseños realizados, logrando un producto final acorde con las especificaciones planteadas y que cumplió satisfactoriamente diversas pruebas realizadas durante su desarrollo. Estas tecnologías fueron previamente analizadas para determinar la posible gama de implementaciones existentes. Las ventajas principales de escogencia de las mismas fueron su bajo costo (son productos Open-source), gran aceptación por parte de la comunidad de desarrollo y su constante evolución que va a la par con el desarrollo de la Web.

Por otra parte, la investigación y el desarrollo realizados permiten concluir que:

- El modelo de sistemas distribuidos bajo la arquitectura de orientación a servicios, usando los estándares actuales como BPEL y WS-Security, permiten desarrollar aplicaciones complejas, robustas, seguras, y a la vez con gran flexibilidad y escalabilidad; características que son un requerimiento vital en el mundo del comercio electrónico. Por ende, poseen un gran potencial de uso en este campo, reduciendo costos y logrando un nivel muy aceptado de estandarización de la información y homogeneidad de la comunicación entre sistemas, sin importar la heterogeneidad existente a nivel interno.

- El uso de la metodología de desarrollo incremental, fue fundamental en el cumplimiento oportuno de los objetivos ya que permitió la reutilización de los componentes desarrollados en cada fase, extendiendo sus definiciones y agilizando la construcción del prototipo.

- BPEL posee una madurez bastante aceptable y está en constante evolución; además brinda un espectro amplio de aplicación al comercio electrónico pues actualmente la complejidad de las aplicaciones ha crecido y se demanda la interoperabilidad de aplicaciones ya realizadas de forma independiente sin grandes cambios para su acople. Además existen otros estándares que complementan y extienden las funcionalidades de BPEL y en general de los servicios Web, para brindarle aspectos tan cruciales como la seguridad.

- Las tecnologías de desarrollo usadas permitieron una adquisición amplia de conocimientos sobre UML, XML, programación orientada a objetos, comunicación en la Web, y muchos otros aspectos adicionales, los cuales son de gran importancia para la formación de un ingeniero de sistemas en el mundo actual.

- El prototipo de agencia de viajes electrónica desarrollado no pretende ser de índole comercial sino netamente académico; su propósito fue conocer los estándares y tecnologías usadas. Para una posible puesta en marcha real del mismo, es necesaria la ampliación de su escenario de aplicación.

RECOMENDACIONES PARA TRABAJOS FUTUROS

Con base en el estudio realizado, las tecnologías conocidas, los componentes software diseñados e implementados en este proyecto de investigación, y los enormes avances al respecto que se dan cada día, es posible realizar variadas recomendaciones para futuros proyectos y tareas en la materia. A continuación se describen algunos de ellos:

- BPEL es un lenguaje que permite tener en cuenta un espectro más amplio de eventos y operaciones, por lo tanto el escenario puede involucrar situaciones muy comunes en la realidad del mundo del comercio electrónico, tales como:
  - Notificaciones automáticas a usuarios, cuando una reserva esté próxima a cumplirse in haber sido reservada y/o cancelada, cuando se quiera recordar al usuario sobre una reserva confirmada y próxima a llegar, cuando por motivos ajenos al usuario la reserva se cancele, etc.
  - Cancelación prematura de las operaciones, ya sea internamente por la lógica del proceso, o por petición del usuario.
  - Cambios en la información de ejecución (criterios, opciones) si el usuario puede interactuar durante la ejecución del proceso.
  - Descubrimiento y definición automática de nuevos socios de la agencia y de ser posible que brinden servicios alternos o diferentes a los ya establecidos.
  - Sucesos a nivel de los socios que afecten reservas ya realizadas y por tanto deban ser notificados a la agencia para que se tomen las medidas necesarias en pro de la satisfacción del usuario: ya sea buscar de nuevo los servicios y reservar alguno similar, o brindar las opciones necesarias al usuario para que él decida.

  Bajo esta perspectiva, sería necesario realizar análisis y diseños más robustos, considerando la administración avanzada de excepciones y fallos, el uso de

actividades de compensación para reversar tareas realizadas, la captura de eventos de parte del usuario y los socios, y la definición de conjuntos de correlación más complejos.

- Otro aspecto importante evidente en el comercio electrónico y las transacciones virtuales, es la interacción con los actores humanos, tanto a nivel de criterios de decisión inicial (considerados en el presente trabajo) como en la lógica de las operaciones, es decir, cuando el usuario puede tomar parte en el flujo de actividades mediante notificaciones, eventos y decisiones. Actualmente existe una extensión de BPEL llamada BPEL4People[30] que permite tratar las interacciones humanas como servicios y por tanto, tomar parte en los procesos de negocio.

  Este estándar complementario permite abarcar escenarios mucho más complejos, donde los humanos poseen roles e interactúan mediante patrones y protocolos definidos que controlan su autonomía y su ciclo de vida dentro del proceso. Un ejemplo de las nuevas capacidades que pueden desarrollarse es el escenario en que las decisiones respecto a la oferta de servicios en la agencia sean tomadas por actores humanos luego de evaluar resultados de las búsquedas; así se aprovecha la capacidad de los seres humanos para conseguir resultados acordes con las necesidades de un caso real de negocios. También puede considerarse la interacción de varios actores humanos con puntos de vista independientes entre sí, que brindan análisis variados a las situaciones consideradas.

- Existe otra forma de manejar las comunicaciones entre servicios y mediante el uso de descripciones ontológicas, esto hace parte de lo que se conoce como

[30] BPEL4People es una extensión de WS-BPEL del OASIS group. Para más información consultar el sitio web del comité técnico http://www.oasis-open.org/committees/bpel4people/charter.php y la propuesta inicial http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/

Composición Semántica de Servicios Web. Existe otra extensión de BPEL para Servicios Web Semánticos llamada BPEL4SWS[31]. En este caso toma más importancia el significado de los participantes en el canal de comunicación definido, que la sintaxis específica predefinida por el enlace entre dichos participantes.

Es posible usar esta perspectiva para darle un carácter más real a los escenarios de aplicación escogidos, y de esta forma describir la realidad del comercio electrónico, en el cual los participantes pueden reconocerse por su fondo y no por su forma, es decir, por su contenido, su significado y la manera de cómo contactarse y relacionarse. La interacción se realiza más inteligentemente y conlleva el uso de sistemas intérpretes de los metadatos o agentes inteligentes.

---

[31] Más información sobre BPEL4SWS disponible en: http://www.infoq.com/news/2008/11/BPEL4SWS

REFERENCIAS

[1] Graham, S. et al. Building Web Services with Java, Making sense of XML, SOAP, WSDL, and UDDI. Second Edition. Sams Publishing. 2005.

[2] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson D. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall. 2005.

[3] Cerami, Ethan. Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. First Edition. O'Reilly. 2002.

[4] Christoph Schiiiko. Web Services Orchestration with BPEL. XML Conference & Exposition 2003.

[5] Ignacio García, Macario Polo, Francisco Ruiz, Mario Piattini. Servicios Web. Universidad de Castilla-La Mancha, España. 2005.

[6] Business Process Execution Language for Web Services, Version 1.1. BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems. 2003.

[7] Gómez Gualdrón, Janeth Gissella. ADAM: Aplicación de los Agentes Móviles al Comercio Electrónico. Tesis de Pregrado. Universidad Industrial de Santander. 2000.

[8] Durán Silva, H., Santamaría Barajas, O., Comercio Electrónico, Un enfoque gerencial. Tesis de Pregrado. Universidad Industrial de Santander. 2001.

[9] Política Nacional para la productividad y competitividad. Presidencia de la República – Ministerio de Comercio Exterior. Colombia, 2000.

[10] Web Services Business Process Execution Language Version 2.0 (WS-BPEL 2.0), OASIS Standard, Abril 11 de 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[11] ActiveBPEL – The Open Source BPEL Engine. Active Endpoints Inc. http://www.activevos.com/community-open-source.php

[12] Apache Web Services Project – The Apache Software Foundation. http://ws.apache.org/

[13] Axis Architecture Guide, Version 1.2. Apache Web Services Project. http://ws.apache.org/axis/java/architecture-guide.html

[14] Apache Tomcat – The Apache Software Foundation. http://tomcat.apache.org/

[15] Apache WSS4J, Apache Web Services Project. http://ws.apache.org/wss4j/

[16] Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard 200401, March 2004 – OASIS Open 2002-2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[17] Web Services Security: UsernameToken Profile 1.0. OASIS Standard 200401, March 2004 – OASIS Open 2002-2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

[18] Web Services Security: X.509 Certificate Token Profile 1.0 OASIS Standard 200401, March 2004 – OASIS Open 2002-2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

[19] WS-I Basic Profile 1.1, The WS-I Organization http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[20] WS-I Attachment Profile 1.0, The WS-I Organization
http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html

[21] WS-I Simple SOAP Binding Profile 1.0, The WS-I Organization
http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html

[22] Pressman, R. Ingeniería del Software, Un enfoque práctico. McGraw Hill, 5ª Edición, Madrid, España. 2002

[23] McConnell, S. Desarrollo y Gestión de Proyectos Informáticos. McGraw-Hill. España. 1997.

[24] Kendall, K. E., Kendall, J., Análisis y Diseño de Sistemas. Pearson Education. 1997.

[25] Web Service Composition - Current Solutions and Open Problems. Biplav Srivastaba y Jana Koehler. Consultado Noviembre de 2008. Disponible online en: www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf

[26] Sistemas Distribuidos - Monografías.com, Omar Hurtado J. Disponible online: http://www.monografias.com/trabajos16/sistemas-distribuidos/sistemas-distribuidos.shtml

[27] Business Process Management Notation BPMN,
Object Management Group - Business Process Management Initiative.
http://www.bpmn.org/

BIBLIOGRAFÍA

ACTIVE ENDPOINTS INC. ActiveBPEL – The Open Source BPEL Engine. http://www.activevos.com/community-open-source.php

BEA Systems, IBM Corporation, MICROSOFT Corporation, SAP AG y SIEBEL Systems. Business Process Execution Language for Web Services, Version 1.1. 2003.

CERAMI, Ethan. Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. First Edition. O'Reilly. 2002.

DURÁN SILVA, H. y SANTAMARIA BARAJAS, O., Comercio Electrónico, Un enfoque gerencial. Tesis de Pregrado. Universidad Industrial de Santander. 2001.

GARCIA, Ignacio; POLO, Macario; RUIZ, Francisco y PIATTINI, Mario. Servicios Web. Universidad de Castilla-La Mancha, España. 2005.

GÓMEZ GUALDRÓN, Janeth Gissella. ADAM: Aplicación de los Agentes Móviles al Comercio Electrónico. Tesis de Pregrado. Universidad Industrial de Santander. 2000.

GRAHAM, S. et al. Building Web Services with Java, Making sense of XML, SOAP, WSDL, and UDDI. Second Edition. Sams Publishing. 2005.

HURTADO, Omar J. Sistemas Distribuidos - Monografías.com. Disponible online: http://www.monografias.com/trabajos16/sistemas-distribuidos/sistemas-distribuidos.shtml

KENDALL, K. E. y KENDALL, J., Análisis y Diseño de Sistemas. Pearson Education. 1997.

MCCONNELL, S. Desarrollo y Gestión de Proyectos Informáticos. McGraw-Hill. España. 1997.

MINISTERIO DE COMERCIO EXTERIOR, Presidencia de la República. Política Nacional para la productividad y competitividad. Colombia, 2000.

OBJECT MANAGEMENT GROUP, Business Process Management Initiative. Business Process Management Notation (BPMN).
http://www.bpmn.org/

OASIS. Web Services Business Process Execution Language Version 2.0 (WS-BPEL 2.0), OASIS Standard, Abril 11 de 2007.
http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

--------. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard 200401, March 2004 – OASIS Open 2002-2004.
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

--------. Web Services Security: UsernameToken Profile 1.0. OASIS Standard 200401, March 2004 – OASIS Open 2002-2004.
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

--------. Web Services Security: X.509 Certificate Token Profile 1.0 OASIS Standard 200401, March 2004 – OASIS Open 2002-2004.
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

PRESSMAN, R. Ingeniería del Software, Un enfoque práctico. McGraw Hill, 5ª Edición, Madrid, España. 2002

SCHIIIKO, Christoph. Web Services Orchestration with BPEL. XML Conference & Exposition 2003.

SRIVASTABA, Biplav y KOEHLER, Jana. Web Service Composition - Current Solutions and Open Problems. Consultado Noviembre de 2008. http://www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf

THE APACHE SOFTWARE FOUNDATION. Apache Tomcat. http://tomcat.apache.org/

--------. Apache Web Services Project http://ws.apache.org/

--------. Apache Web Services Project - Axis Architecture Guide, Version 1.2. http://ws.apache.org/axis/java/architecture-guide.html

--------. Apache Web Services Project - Apache WSS4J. http://ws.apache.org/wss4j/

THE WS-I ORGANIZATION. WS-I Attachment Profile 1.0. http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html

--------. WS-I Basic Profile 1.1. http://www.ws-i.org/Profiles/BasicProfile-1.1.html

--------. WS-I Simple SOAP Binding Profile 1.0. http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html

WEERAWARANA, S., Curbera, F., Leymann, F., Storey, T., Ferguson D. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall. 2005.

ANEXOS

ANEXO A. Esquemas XML Definidos

Este anexo relaciona los esquemas XML definidos, presentándolos mediante diagramas de clases UML organizados por paquetes (espacios de nombres XML), y ordenados por jerarquía y uso.

Figura 113. Diagrama de Clases – Esquemas.Global

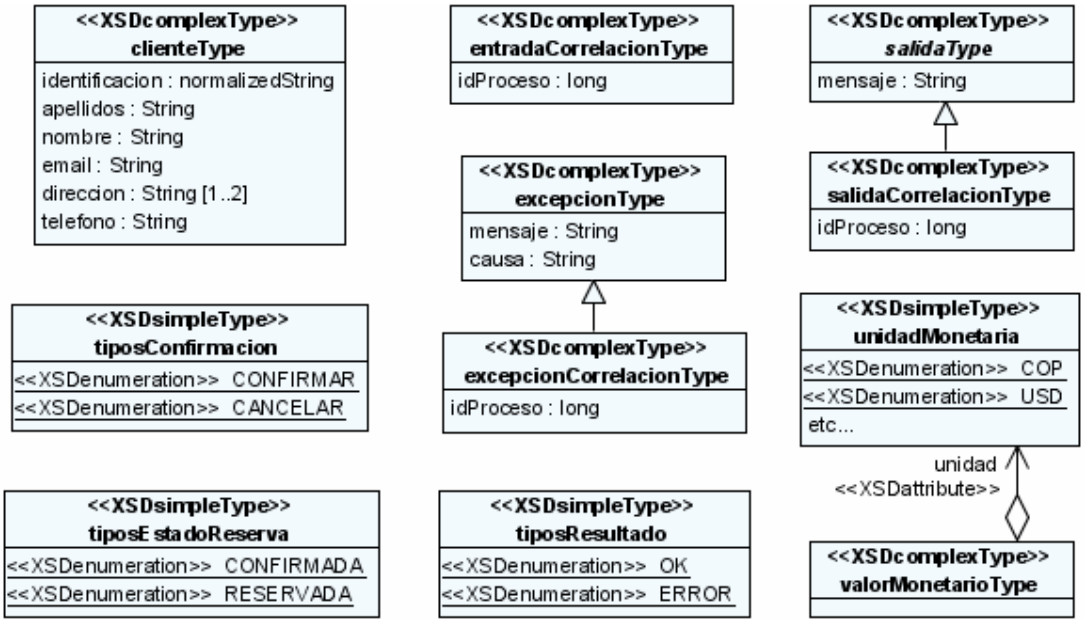


Figura 114. Diagrama de Clases – Esquemas.Socios

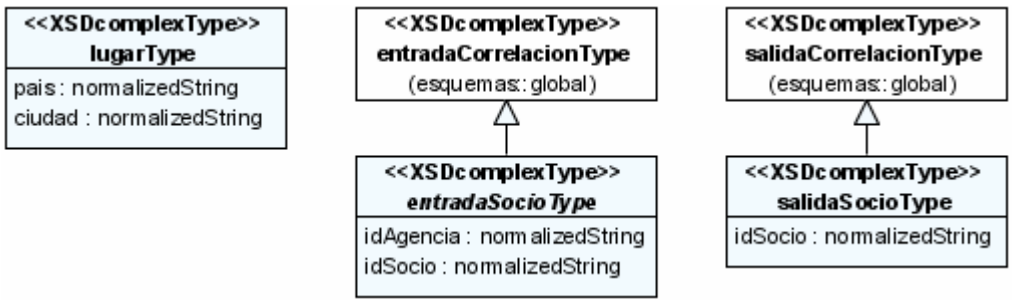


Figura 115. Diagrama de Clases – Esquemas.Aerolinea

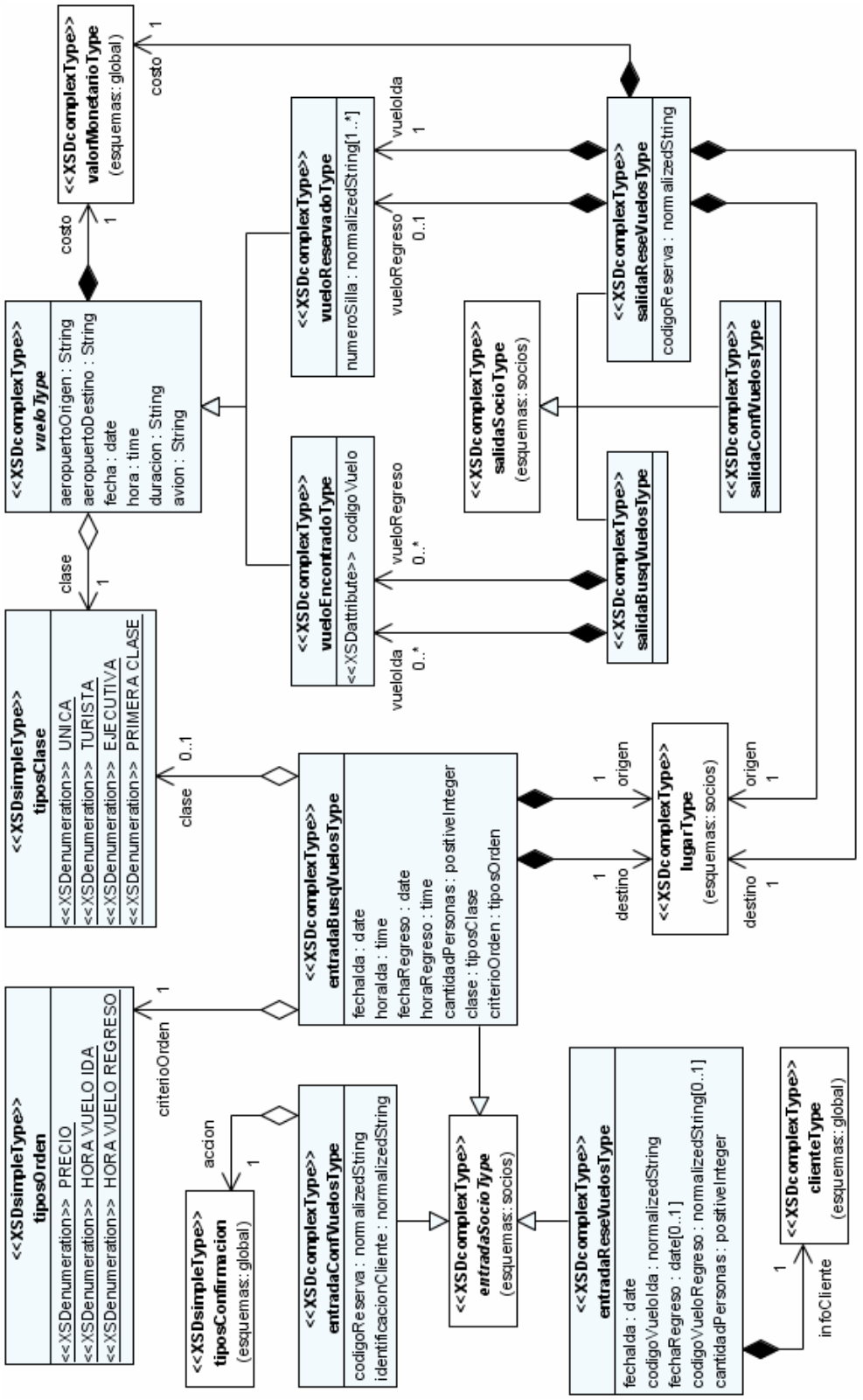


Figura 116. Diagrama de Clases – Esquemas.CadenaHotelera

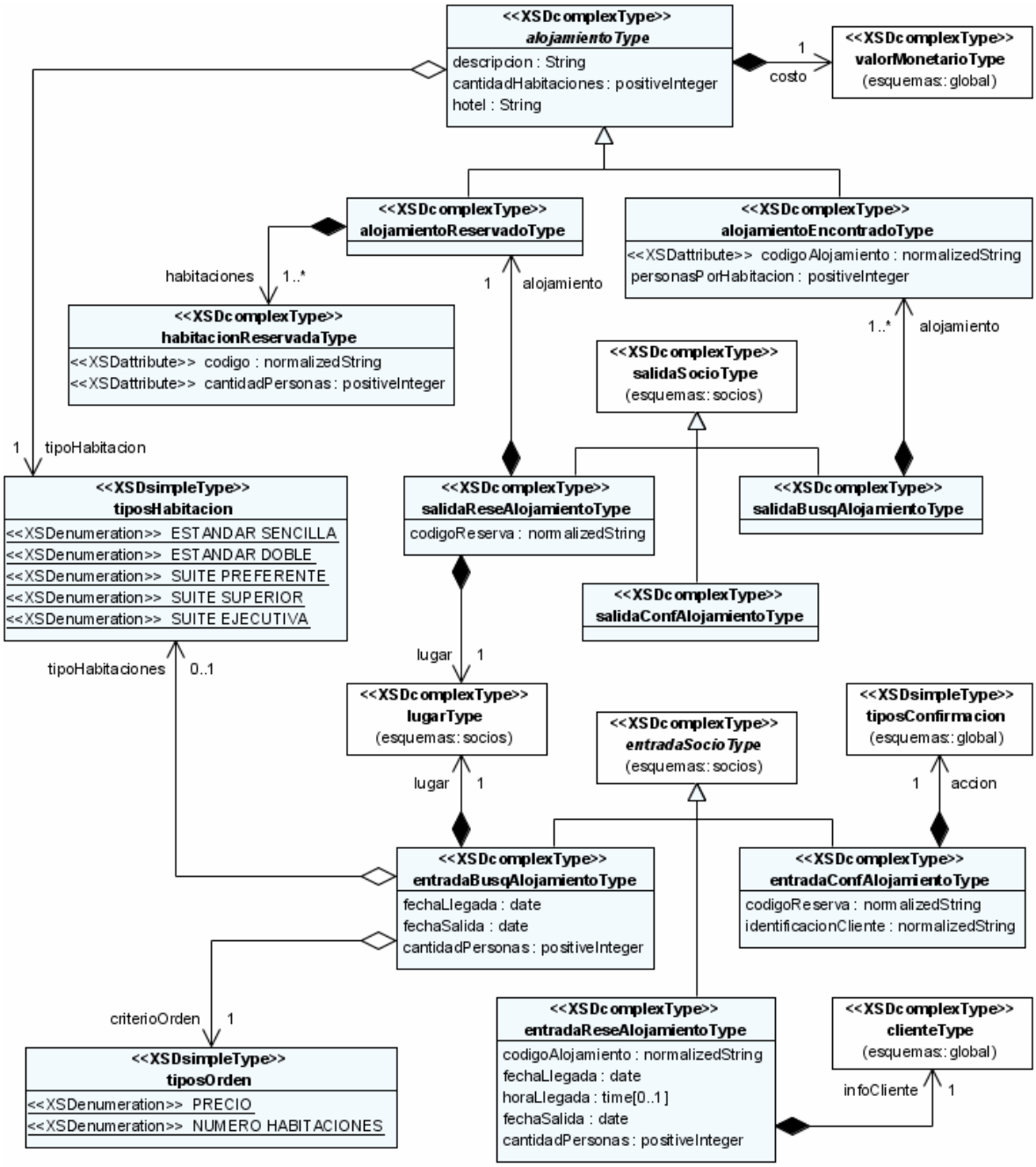


Figura 117. Diagrama de Clases – Esquemas.AlquilerVehiculos

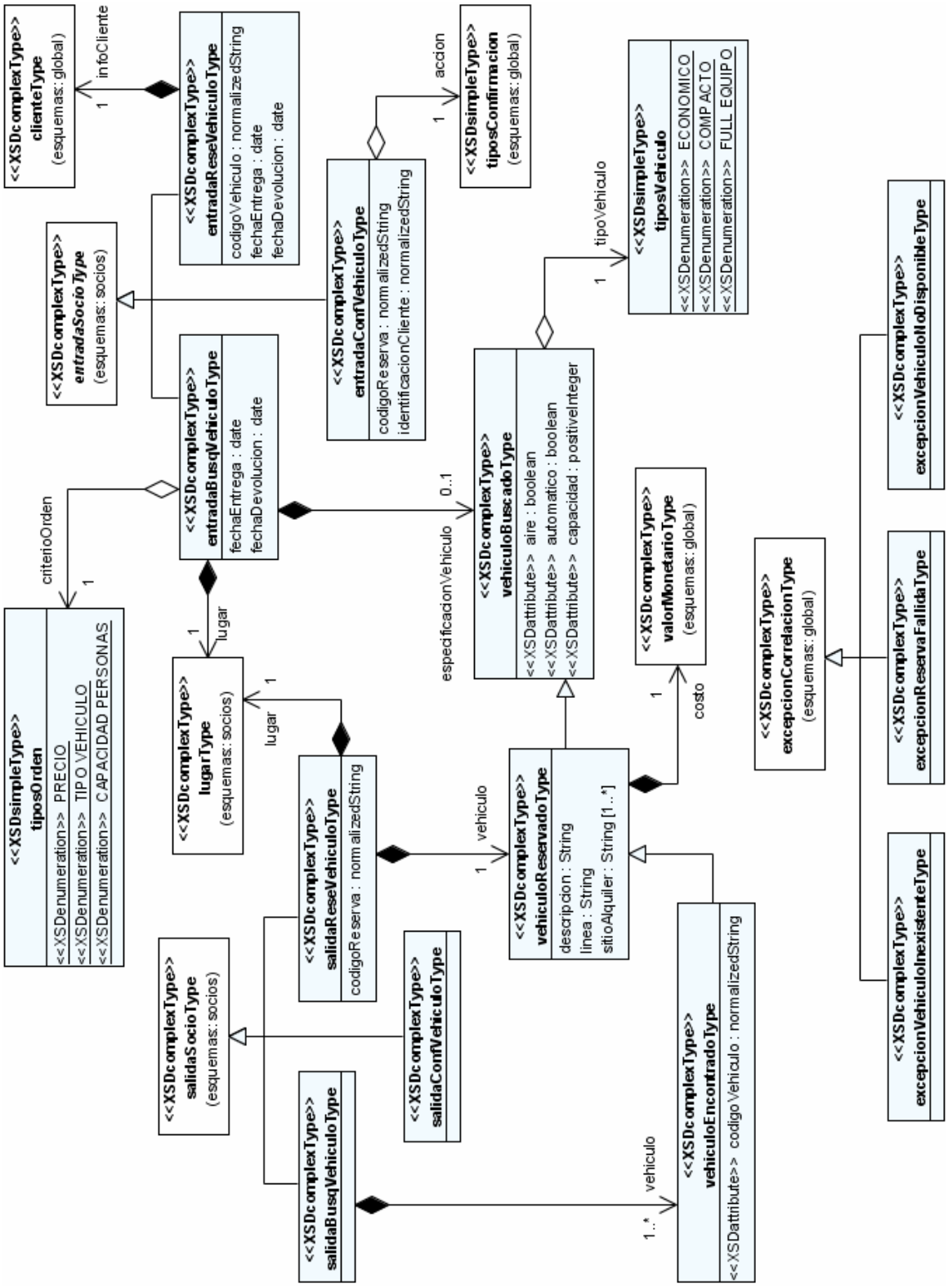


Figura 118. Diagrama de Clases – Esquemas.Cliente (1/3)

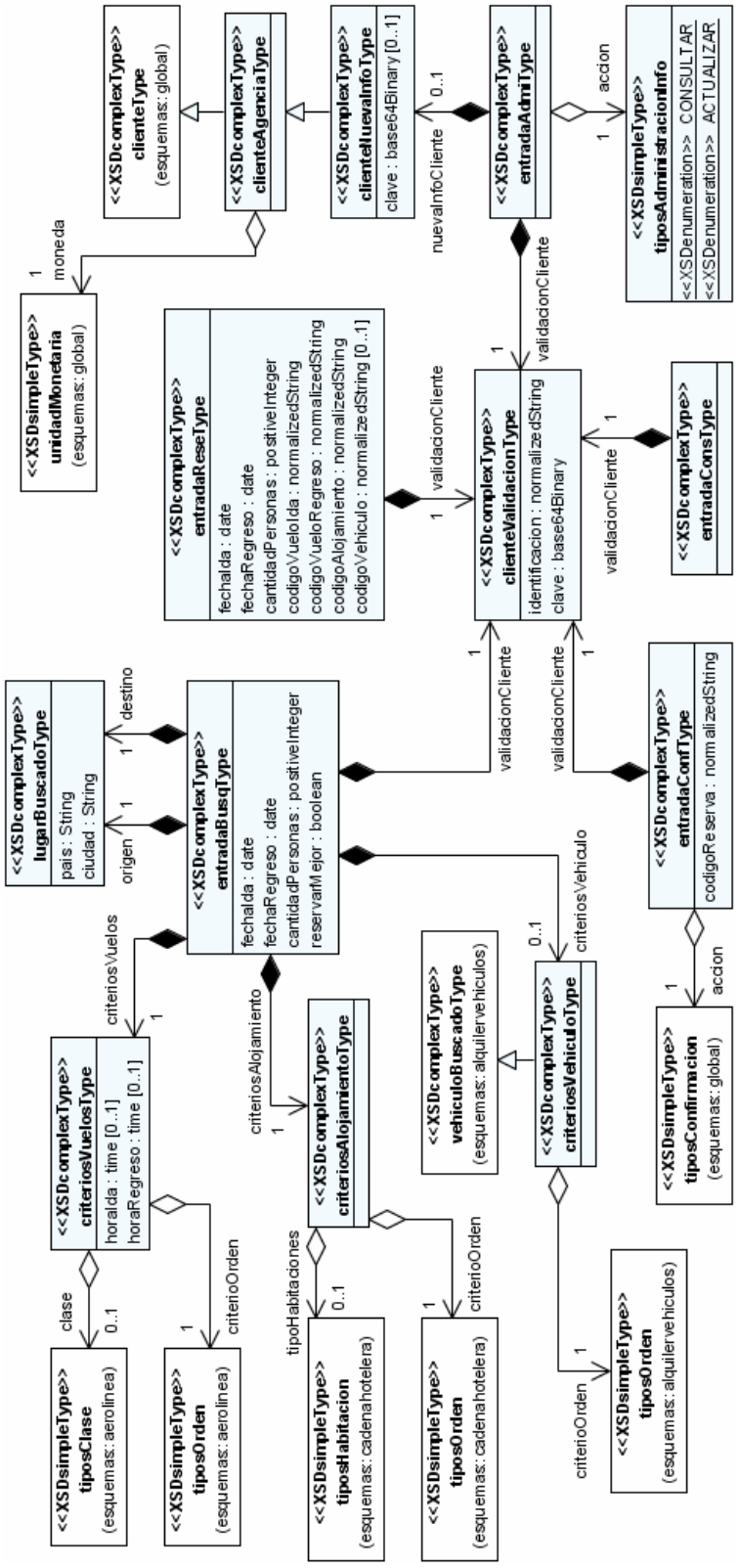


Figura 119. Diagrama de Clases – Esquemas.Cliente (2/3)

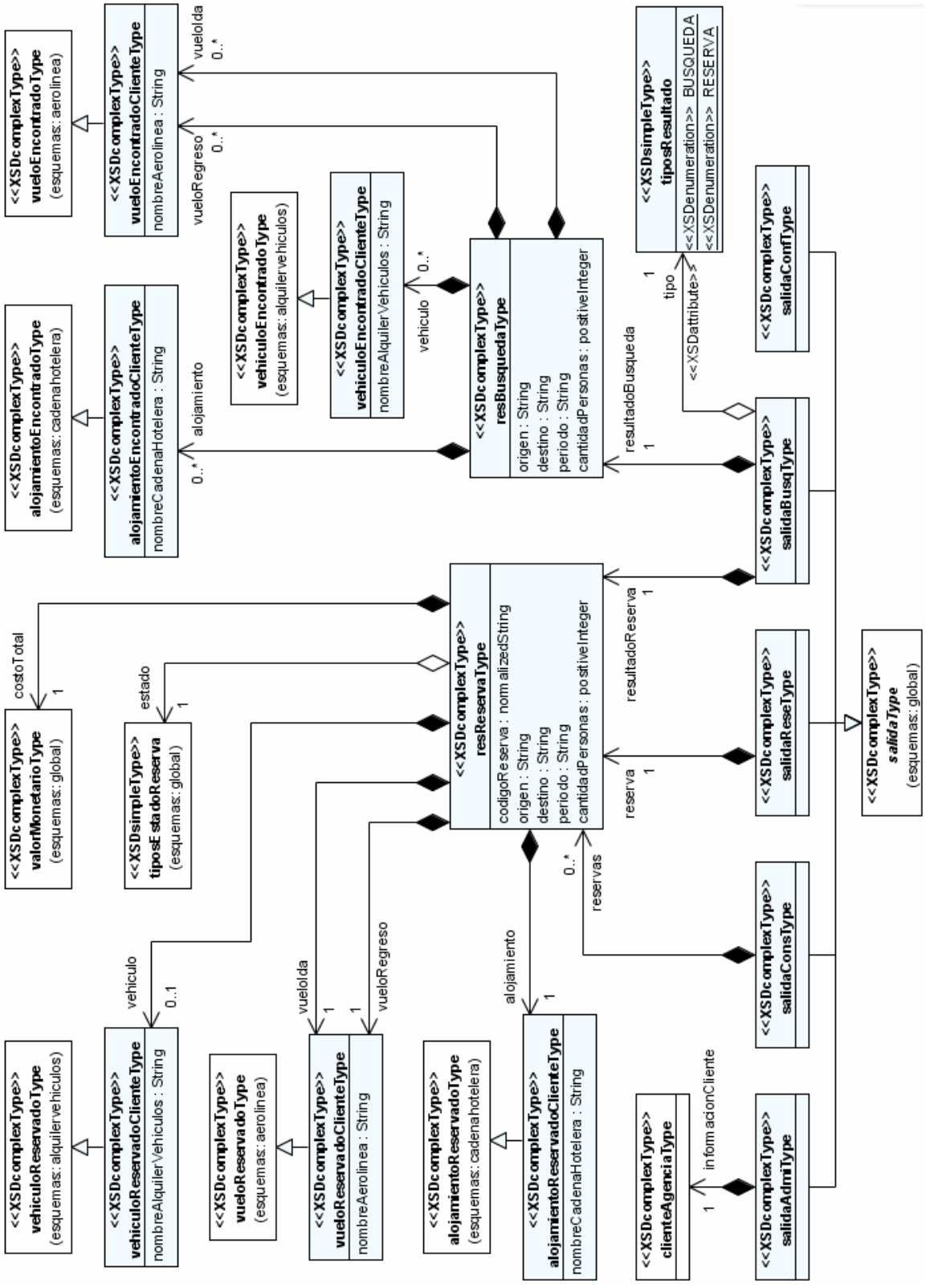
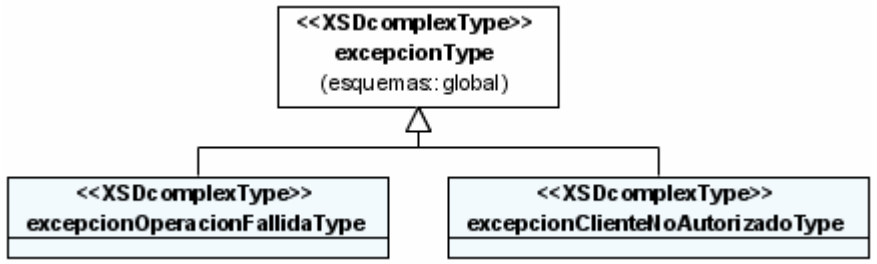


Figura 120. Diagrama de Clases – Esquemas.Cliente (3/3)



ANEXO B. Interfaz WSDL Representativa Definida

En este anexo se exponen un documento WSDL completo que permiten representar los diferentes casos manejados. El escogido es el documento de la interfaz definida para el socio Alquiler de Vehículos con todas las operaciones implementadas y su especificación completa, seleccionado debido a que es el más completo y expone la complejidad.

Figura 121. Documento WSDL socio Alquiler de Vehículos

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="alquilervehiculos"
                  xmlns:vhcr="http://alquilervehiculos.com/servicios/agenciaviajes"
                  xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  targetNamespace="http://alquilervehiculos.com/servicios/agenciaviajes" >

  <wsdl:types>
    <xsd:schema targetNamespace="http://alquilervehiculos.com/servicios/agenciaviajes"
                 xmlns:vhs="http://agenciaviajes.com/esquemas/alquilervehiculos">
      <xsd:import namespace="http://agenciaviajes.com/esquemas/alquilervehiculos"
                  schemaLocation="./alquilervehiculos.xsd" />
      <xsd:element name="entradaBusqueda" type="vhs:entradaBusqVehiculoType" />
      <xsd:element name="entradaReserva" type="vhs:entradaReseVehiculoType" />
      <xsd:element name="salidaReserva" type="vhs:salidaReseVehiculoType" />
      <xsd:element name="entradaConfirmacionReserva"
                   type="vhs:entradaConfVehiculoType" />
      <xsd:element name="excepcionReservaFallida"
                   type="vhs:excepcionReservaFallidaType" />
      <xsd:element name="excepcionVehiculoInexistente"
                   type="vhs:excepcionVehiculoInexistenteType" />
      <xsd:element name="excepcionVehiculoNoDisponible"
                   type="vhs:excepcionVehiculoNoDisponibleType" />
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="buscarPeticion">
     <wsdl:part name="entrada" element="vhc:entradaBusqueda" />
  </wsdl:message>
```

```
<wsdl:message name="reservarPeticion">
   <wsdl:part name="entrada" element="vhc:entradaReserva" />
</wsdl:message>
<wsdl:message name="reservarRespuesta">
   <wsdl:part name="salida" element="vhc:salidaReserva" />
</wsdl:message>
<wsdl:message name="excepcionReservaFallida">
   <wsdl:part name="excepcion" element="vhc:excepcionReservaFallida" />
</wsdl:message>
<wsdl:message name="excepcionVehiculoInexistente">
   <wsdl:part name="excepcion" element="vhc:excepcionVehiculoInexistente" />
</wsdl:message>
<wsdl:message name="excepcionVehiculoNoDisponible">
   <wsdl:part name="excepcion" element="vhc:excepcionVehiculoNoDisponible" />
</wsdl:message>
<wsdl:message name="confirmarReservaPeticion">
   <wsdl:part name="entrada" element="vhc:entradaConfirmarReserva" />
</wsdl:message>

<wsdl:portType name="AlquilerVehiculosPortType">
  <wsdl:operation name="buscar">
     <wsdl:input name="buscarIn"
                 message="vhc:buscarPeticion" />
  </wsdl:operation>
  <wsdl:operation name="reservar">
     <wsdl:input name="reservarIn"
                 message="vhc:reservarPeticion" />
     <wsdl:output name="reservarOut"
                  message="vhc:reservarRespuesta" />
     <wsdl:fault name="excepcionReservaFallida"
                 message="vhc:excepcionReservaFallida" />
     <wsdl:fault name="excepcionVehiculoInexistente"
                 message="vhc:excepcionVehiculoInexistente" />
     <wsdl:fault name="excepcionVehiculoNoDisponible"
                 message="vhc:excepcionVehiculoNoDisponible" />
  </wsdl:operation>
  <wsdl:operation name="confirmarReserva">
    <wsdl:input name="confirmarReservaIn" message="aer:confirmarReservaPeticion" />
  </wsdl:operation>
</wsdl:portType>
```

```
  <wsdl:binding name="AlquilerVehiculosSoapBinding" type="vhc:AlquilerVehiculosPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="buscar">
      <soap:operation soapAction="http://alquilervehiculos.com/servicios/agenciaviajes/buscar" />
      <wsdl:input>
        <soap:body parts="entrada" use="literal" />
      </wsdl:input>
    </wsdl:operation>
    <wsdl:operation name="reservar">
      <soap:operation soapAction="http://alquilervehiculos.com/servicios/agenciaviajes/reservar" />
      <wsdl:input>
        <soap:body parts="entrada" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body parts="salida" use="literal" />
      </wsdl:output>
      <wsdl:fault name="excepcionReservaFallida">
        <soap:fault name="excepcionReservaFallida" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="excepcionVehiculoInexistente"
        <soap:fault name="excepcionVehiculoInexistente" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="excepcionVehiculoNoDisponible"
        <soap:fault name="excepcionVehiculoNoDisponible" use="literal" />
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="confirmarReserva">
      <soap:operation soapAction="http://alquilervehiculos.com/servicios/agenciaviajes/confirmarReserva" />
      <wsdl:input>
        <soap:body parts="entrada" use="literal" />
      </wsdl:input>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="AlquilerVehiculosService">
    <wsdl:port name="AlquilerVehiculosSoapPort" binding="vhc:AlquilerVehiculosSoapBinding">
      <soap:address location="http://www.alquilervehiculos.com/services/AlquilerVehiculosSoap" />
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

ANEXO C. Esquemas de Datos definidos para los sistemas desarrollados

En este anexo se incluyen los diagramas entidad relación en notación UML con los esquemas de datos definidos para cada uno de los sistemas desarrollados en el prototipo. Primero se muestran los relacionados con los socios (aerolínea, cadena hotelera y alquiler de vehículos) y finalmente el definido para la agencia de viajes.

Figura 122. Modelo de datos del sistema del socio Aerolínea

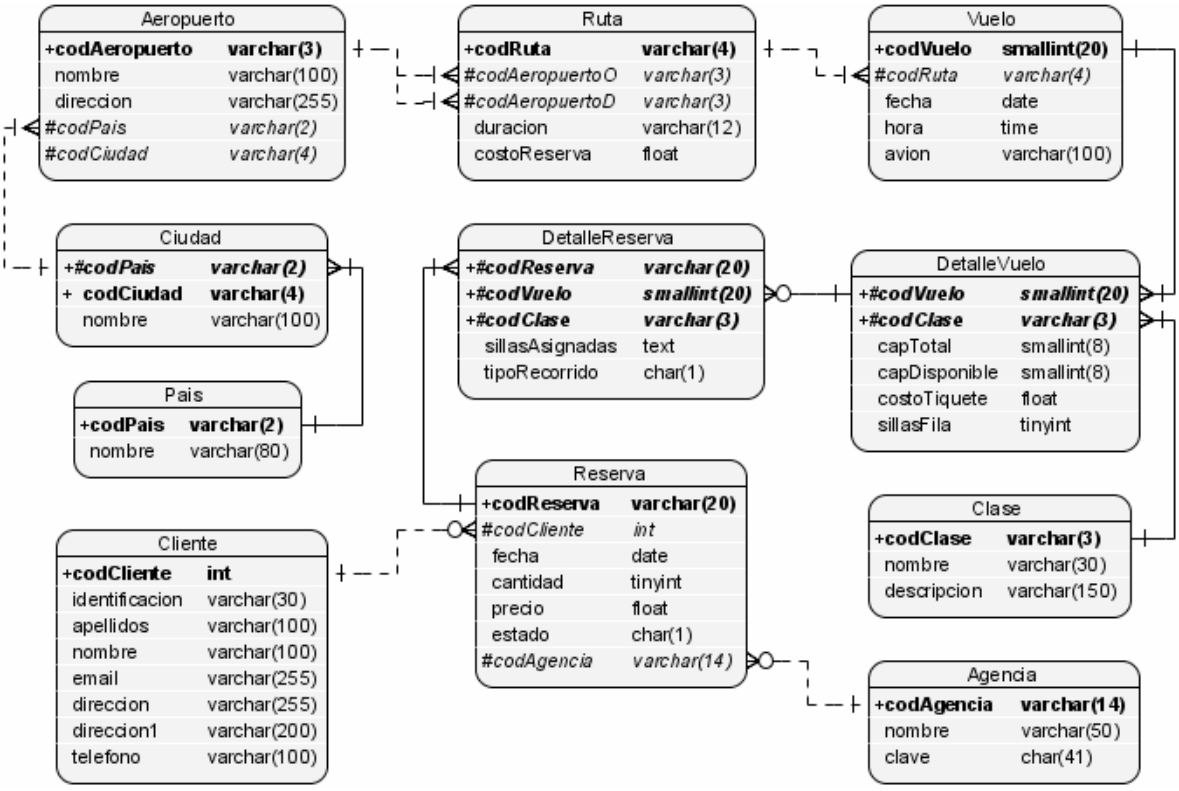


Figura 123. Modelo de datos del sistema del socio Cadena Hotelera

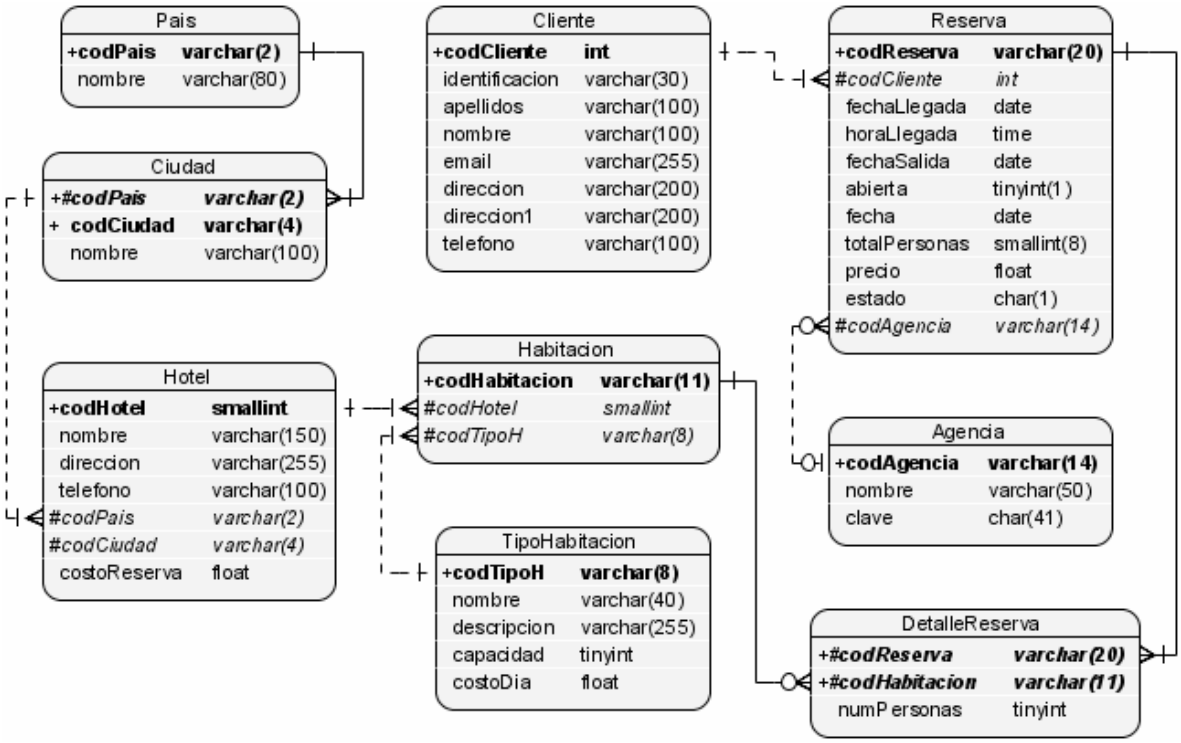


Figura 124. Modelo de datos del sistema del socio Alquiler de Vehículos

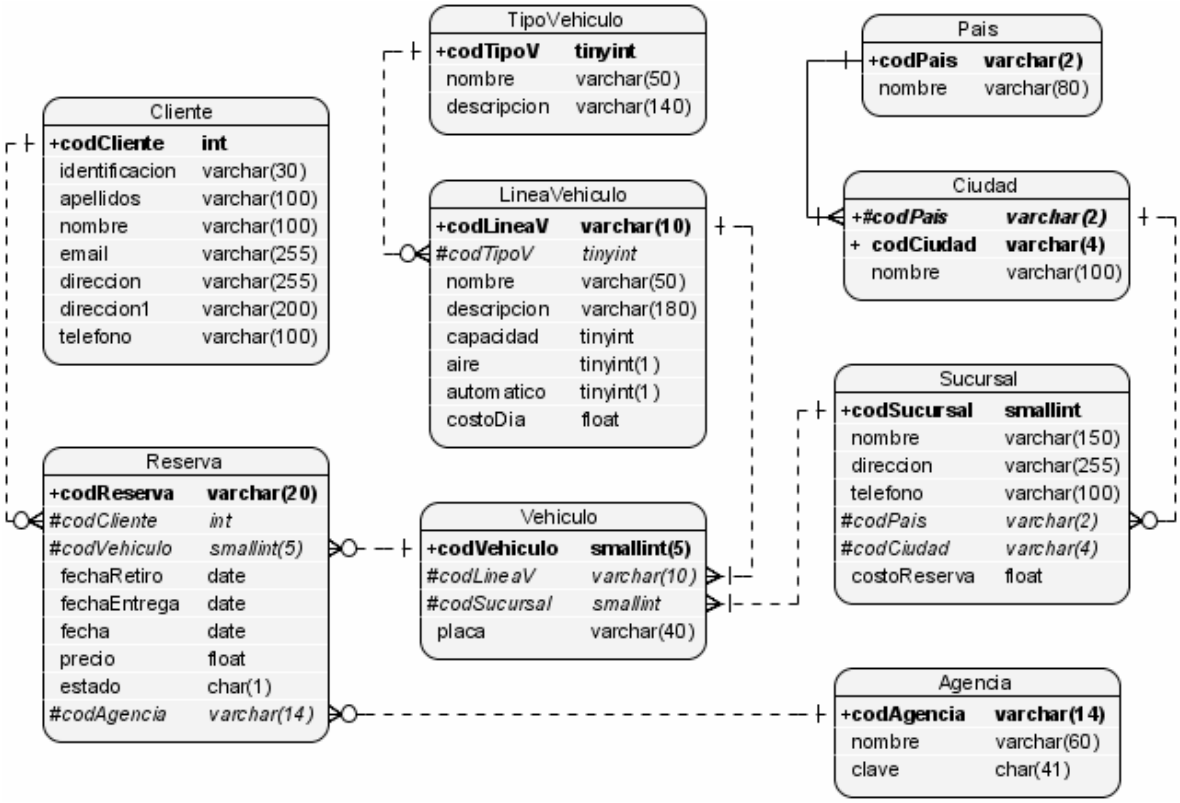
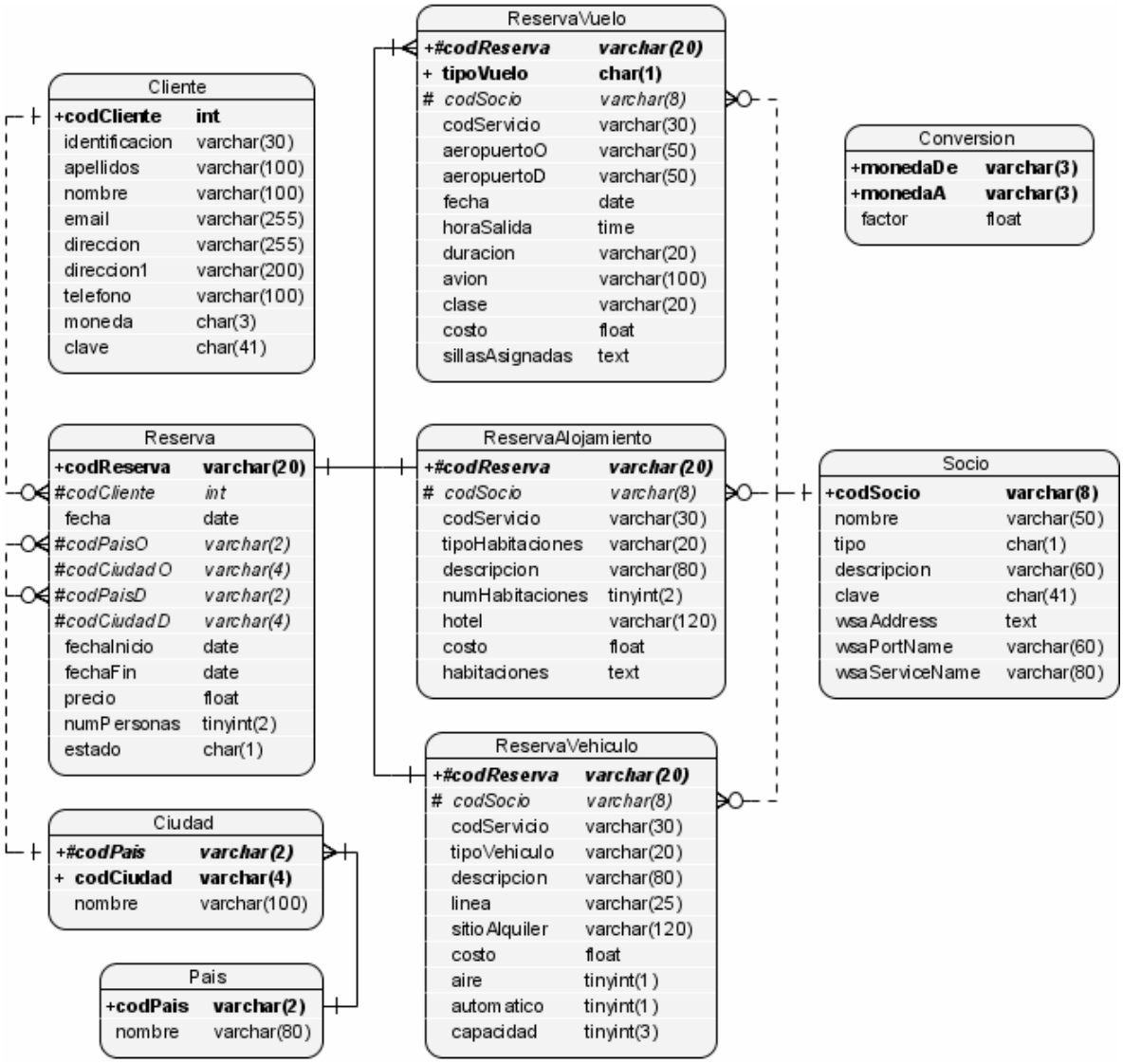


Figura 125. Modelo de datos del sistema local de la Agencia de Viajes



ANEXO D. Plan de Pruebas del Desarrollo

En este anexo se especifican las pruebas realizadas a los procesos desarrollados en cada fase, con el fin de verificar su funcionamiento, detectar y corregir errores, identificar nuevos elementos no que puedan mejorar la implementación.

D.1. Escenario General de Pruebas

Con el fin de realizar las pruebas, se planteó la agencia de viajes ficticia llamada *AgenciaViajes.com* la cual posee varios socios ficticios que le ofrecen sus servicios; además existen varios clientes afiliados a la agencia, manejando un período de tiempo de 18 días, comenzando en 01/Noviembre/2009 y hasta 18/Noviembre/2009.

Socios de la Agencia de Viajes

La agencia de viajes posee tres socios de tipo aerolínea, tres socios de tipo cadena hotelera y tres socios de tipo alquiler de vehículos. En las tablas siguientes se presentan resúmenes de las características más importantes de cada socio.

Tabla 75. Aerolíneas definidas para el Escenario de Pruebas

Nombre Socio	Características
Aerolínea 1 *AeroColombia*	*Descripción*: Los mejores y más seguros vuelos en Colombia. *Unidad Monetaria*: COP *Cobertura*: Bogotá, Bucaramanga, Cartagena, Santa Marta.
Aerolínea 2 *AeroRepública Americana*	*Descripción*: Conectando a América con el poder de un avión. *Unidad Monetaria*: USD *Cobertura*: Acapulco de Juárez y Cancún (México), Río de Janeiro (Brasil), Bogotá y Bucaramanga (Colombia).
Aerolínea 3 *Aerolíneas American Oceanic*	*Descripción*: Conectando las playas de América sin escalas. *Unidad Monetaria*: USD *Cobertura*: Acapulco de Juárez y Cancún (México), Río de Janeiro (Brasil), Cartagena y Santa Marta (Colombia).

En las aerolíneas, existen rutas entre cada ciudad con cobertura, y cada ruta está programada con vuelos entre las fechas del escenario. Cada vuelo posee un número determinado de sillas de cada clase de las definidas en el diseño, no necesariamente todas las clases tienen sillas en todos los vuelos.

Tabla 76. Cadenas Hoteleras definidas para el Escenario de Pruebas

Nombre Socio	Características
Cadena Hotelera 1 *Hoteles Playa Americana*	*Descripción*: Las playas americanas dentro del mismo hotel. *Unidad Monetaria*: USD *Hoteles*: - Playa Americana Cartagena (Colombia) – 9 habitaciones - Playa Americana Acapulco (México) – 18 habitaciones - Playa Americana Cancún Plaza (México) – 19 habitaciones - Playa Americana Cancún Resort (México) – 10 habitaciones - Playa Americana Río de Janeiro Plaza (Brasil) – 14 habitaciones - Playa Americana Río de Janeiro Resort (Brasil) – 8 habitaciones
Cadena Hotelera 2 *Hotel Interamericano*	*Descripción*: La cadena hotelera que une el continente Americano. *Unidad Monetaria*: USD *Hoteles*: - Hotel Interamericano Bogotá Club (Colombia) – 8 habitaciones - Hotel Interamericano Bogotá Plaza (Colombia) – 16 habitaciones - Hotel Interamericano Bucaramanga (Colombia) – 11 habitaciones - Hotel Interamericano Cartagena (Colombia) – 12 habitaciones - Hotel Interamericano Santa Marta (Colombia) – 9 habitaciones - Hotel Interamericano Río de Janeiro (Brasil) – 12 habitaciones - Hotel Interamericano Cancún (México) – 15 habitaciones - Hotel Interamericano Acapulco Club (México) – 8 habitaciones - Hotel Interamericano Acapulco Plaza (México) – 4 habitaciones
Cadena Hotelera 3 *Colombiana de Hoteles*	*Descripción*: Por tradición los mejores hoteles de Colombia. *Unidad Monetaria*: COP *Hoteles*: - Colombiana de Hoteles Bogotá – 13 habitaciones - Colombiana de Hoteles Bucaramanga – 15 habitaciones - Colombiana de Hoteles Cartagena – 10 habitaciones - Colombiana de Hoteles Santa Marta Plaza – 16 habitaciones - Colombiana de Hoteles Santa Marta Resort – 12 habitaciones

Tabla 77. Alquileres de Vehículos definidos para el Escenario de Pruebas

Nombre Socio	Características
Alquiler Vehículos 1 *Autos Colombia*	*Descripción*: Alquiler de autos en las ciudades Colombianas. *Unidad Monetaria*: COP *Sucursales*: - Autos Colombia Central (Bogotá) – 9 vehículos - Autos Colombia El Dorado (Bogotá) – 14 vehículos - Autos Colombia Bucaramanga – 16 vehículos - Autos Colombia La Heroica (Cartagena) – 12 vehículos - Autos Colombia Santa Marta – 9 vehículos
Alquiler Vehículos 2 *Brasilera de Carros*	*Descripción*: La mejor opción para “renta dao vehiculos" en Brasil. *Unidad Monetaria*: BRL *Sucursales*: - Brasilera de Carros Jobim (Río de Janeiro) – 20 vehículos - Brasilera de Carros Dumont (Río de Janeiro) – 14 vehículos
Alquiler Vehículos 3 *Azteca de Autos*	*Descripción*: Alquiler de coches de la república Mexicana. *Unidad Monetaria*: MXN *Sucursales*: - Azteca de Autos, Cancún Internacional – 10 vehículos - Azteca de Autos, Cancún Puerto – 12 vehículos - Azteca de Autos, Acapulco Centro – 8 vehículos - Azteca de Autos, Acapulco Aeropuerto – 8 vehículos

Las habitaciones de cada cadena hotelera son de tipo variados, al igual que los vehículos de cada alquiler de vehículos son de diferente línea, teniendo en cuenta los tipos de habitación y las clases de vehículos definidos en el diseño.

Clientes de la Agencia

Se definieron tres clientes afiliados a la agencia cuyas características se presentan a continuación:

Tabla 78. Clientes definidos para el Escenario de Pruebas

Nombre	Identificación	Moneda	Clave
Gilberto Gómez Gualdrón	93819597	COP	gilber2009
Pedro Fernández	34567891	USD	peter123
Adriana Alarcón	54321987	MXN	claveABC

D.2. Ambiente de Pruebas

El montaje realizado para las pruebas consta de un entorno de red con 5 equipos interconectados: 1 equipo a modo de servidor principal con el sistema de la agencia, 3 equipos para los sistemas de los socios, y 1 equipo para la interfaz del cliente. Las características de hardware y software de cada equipo en la red y su configuración se detallan a continuación:

Tabla 79. Especificación de Equipos de Pruebas

Equipo	Características y Configuración
Servidor	*Hardware*: - Procesador Pentium Dual Core 1.86GHz - 3 GB de RAM, - Disco duro de 250 MB. *Software*: - Windows XP SP3, - Java SE Runtime Environment 6u18 - Servidor Apache Tomcat 5.0.28. - ActiveBPEL v2.1.1 - Servidor MySQL 5.0.18-nt *Dirección IP*: 192.168.0.128
Equipos de Socios	Tres equipos de similares características: *Hardware*: - Procesador Pentium 4, 3.0 GHz - 1.5 GB de RAM, - Disco duro de 120 MB. *Software*: - Windows XP SP2, - Java SE Runtime Environment 6u18 - Servidor Apache Tomcat 5.0.28. - Servidor MySQL 5.0.18-nt *Direcciones IP*: 192.168.0.20, 192.168.0.30, 192.168.0.40
Equipo de Cliente	*Hardware*: - Procesador Pentium III, 2.1 GHz - 1 GB de RAM, - Disco duro de 160 MB. *Software*: - Windows XP SP3, - Java SE Runtime Environment 6u18 *Dirección IP*: 192.168.0.200

Cada equipo de los socios posee tres de ellos, así: el *equipo 1* posee los socios Aerolínea 1, Cadena Hotelera 1, y Alquiler de Vehículos 1; el *equipo 2* hospeda los socios Aerolínea 2, Cadena Hotelera 2, y Alquiler de Vehículos 2; y por su parte el *equipo 3* tiene los socios Aerolínea 3, Cadena Hotelera 3, y Alquiler de Vehículos 3.

Las direcciones de acceso a los servicios resultantes, bajo las especificaciones dadas, se enuncian en la siguiente tabla:

Tabla 80. Configuración de Acceso a los Servicios

URL's de Acceso
Agencia de Viajes http://192.168.0.128:8080/active-bpel/services/
Aerolíneas 1. http://192.168.0.20:8080/Aerolinea01/services/AerolineaSoapPort 2. http://192.168.0.30:8080/Aerolinea02/services/AerolineaSoapPort 3. http://192.168.0.40:8080/Aerolinea03/services/AerolineaSoapPort
Cadenas Hoteleras 1. http://192.168.0.20:8080/CadenaHotelera01/services/CadenaHoteleraSoapPort 2. http://192.168.0.30:8080/CadenaHotelera02/services/CadenaHoteleraSoapPort 3. http://192.168.0.40:8080/CadenaHotelera03/services/CadenaHoteleraSoapPort
Alquiler de Vehículos 1. http://192.168.0.20:8080/AlquilerVehiculos01/services/AlquilerVehiculosSoapPort 2. http://192.168.0.30:8080/AlquilerVehiculos02/services/AlquilerVehiculosSoapPort 3. http://192.168.0.40:8080/AlquilerVehiculos03/services/AlquilerVehiculosSoapPort

Las pruebas se realizaron durante horas no laborales en una oficina con la infraestructura ya instalada, en la que el autor del presente trabajo de investigación tenía acceso.

D.3. Pruebas de la Fase 1: Búsqueda de Planes de Viaje

D.3.1. Prueba 1, Búsquedas para un Cliente

Objetivo: Verificar el correcto funcionamiento del proceso de búsqueda y la robustez del mismo, desde la interfaz del cliente hasta los socios, según un conjunto válido de criterios establecidos que arrojan resultados para todos los servicios, y con información errónea; ejecutándose para cada usuario de forma individual y en diferente momento.

Procedimiento: El conjunto de pasos que cada cliente ejecutará se especifican a continuación.

Tabla 81. Pasos Prueba 1 – Fase 1, Búsquedas

Paso	Descripción
1	*Información de Cliente*: acceda al sistema e ingrese sus datos de usuario en los campos identificación y contraseña de la pestaña General, luego presione el botón Aceptar para asignar esta información.
2	*Criterios de Búsqueda*: continúe en la pestaña Búsqueda y proporcione los criterios de búsqueda siguientes: - Origen: Colombia, Bucaramanga. - Destino: México, Cancún - Fechas: Ida en Noviembre 3 de 2009, Regreso en Noviembre 15 de 2009 - Número de Personas: 2 - Vuelo: Clase Única, Hora Ida: 08:00, Hora Regreso: 17:00, Orden Precio - Alojamiento: Tipo Habitación Estándar Sencilla, Orden Precio - Vehículos: Categoría Económico, Aire y Automático, Capacidad 5, Orden Precio.
3	*Búsqueda*: una vez digitados los criterios, presione el botón Realizar Búsqueda y espere a que se muestren el resultado de la operación mediante un mensaje. Luego observe en la pestaña Resultados Búsqueda el listado de cada uno de los tipos de servicios encontrados cambiando entre las pestañas correspondientes.
4	*Búsqueda sin Vehículo*: en la pestaña de búsqueda, desactive la casilla de verificación Criterios de Vehículos, realice la búsqueda y observe los resultados.
5	*Variación de Orden*: Modifique los criterios de orden de cada servicio y realice la búsqueda nuevamente. Observe cómo se ordenan los nuevos resultados.

Paso	Descripción
6	*Validación de Cliente*: ingrese los datos de usuario con errores presionando el botón Aceptar, luego realice la búsqueda, espere y observe los resultados.
7	*Fechas Incorrectas*: digite una fecha de ida posterior a la fecha de regreso, o una fecha de ida inferior a la fecha actual, realice la búsqueda y aprecie los resultados.
8	*Fechas Fuera de Rango*: especifique fechas correctas pero fuera del rango establecido: 01/11/2009 a 18/11/2009, realice la búsqueda y vea los resultados.
9	*Sitios Incorrectos*: especifique sitios con nombres erróneos o diferentes a los cubiertos por los servicios por los socios, realice la búsqueda y observe los resultados.

Resultados: Los resultados de la prueba se capturan con el siguiente formato:

Tabla 82. Resultados Prueba 1 – Fase 1, Búsquedas

No.	Pregunta	Respuesta	
		SI	NO
1	¿El sistema aceptó los datos de usuario introducidos?		
2	¿Los criterios de búsqueda corresponden a las especificaciones?		
3	¿Fue posible especificar cada criterio según se definió?		
4	¿Al ejecutar la búsqueda, se informó el resultado del proceso?		
5	¿La búsqueda fue exitosa?		
6	¿Los resultados se mostraron correctamente?		
7	¿Se realizó la búsqueda sin incluir vehículo?		
8	¿La búsqueda sin vehículo arrojó resultados correctos?		
9	¿Los resultados estaban expresados con la moneda del usuario?		
10	¿Varió el orden de los resultados al modificar el criterio de orden?		
11	¿El sistema realizó validación de usuario?		
12	¿Las fechas incoherentes fueron aceptadas?		
13	¿Se encontraron servicios al buscar con fechas fuera de rango?		
14	¿Se encontraron servicios buscando con sitios incorrectos?		
15	¿El sistema informó correctamente sobre cada error encontrado?		

D.3.2. Prueba 2, Búsquedas Simultáneas

Objetivo: Comprobar la funcionalidad del sistema ejecutándose de forma simultánea para los usuarios definidos.

Procedimiento: Cada usuario usará una instancia independiente de la interfaz de cliente en el equipo de pruebas, y luego de realizar los pasos especificados a continuación, se ejecutarán las búsquedas de forma simultánea:

Tabla 83. Pasos Prueba 2 – Fase 1, Búsquedas

Paso	Descripción
1	*Información de Cliente*: ingrese sus datos de usuario en la pestaña General, luego presione el botón Aceptar para asignar esta información.
2	*Criterios de Búsqueda*: basándose en la información correcta de la prueba I, modifique los criterios así: - Usuario 1: Información sin cambios. - Usuario 2: Origen: Bogotá, Colombia; Destino: Río de Janeiro, Brasil - Usuario 3: Origen: Santa Marta, Colombia; Destino: Bucaramanga, Colombia.
3	*Búsqueda*: presionar el botón Realizar Búsqueda en las tres interfaces de forma simultánea, tanto como sea posible.
4	*Resultados*: esperar los resultados y observar detenidamente que correspondan a los servicios buscados por cada cliente.

Resultados: El formato de captura de los resultados de la prueba 2 es:

Tabla 84. Resultados Prueba 2 – Fase 1, Búsquedas

No.	Pregunta	Respuesta	
		SI	NO
1	¿El sistema realizó validación de usuario?		
2	¿Se produjeron errores en la búsqueda?		
3	¿Los resultados corresponden a la búsqueda realizada?		
4	¿Los resultados se mostraron correctamente?		

D.4. Pruebas de la Fase 2: Reserva de Planes de Viaje

D.4.1. Prueba 1, Reserva a Partir de Búsqueda

Objetivo: Comprobar el correcto funcionamiento del proceso de reserva de planes de viaje, partiendo de los resultados de una búsqueda.

Procedimiento: Esta prueba involucra también la modificación rápida de los datos almacenados en los socios y la agencia. Cada vez que se realicen los pasos 3 a 7, se deben limpiar los datos almacenados en los sistemas, para que todas las validaciones se ejecuten en las mismas condiciones. Los pasos a seguir por el usuario se especifican a continuación.

Tabla 85. Pasos Prueba 1 – Fase 2, Reservas

Paso	Descripción
1	*Búsqueda*: realice una búsqueda en el sistema siguiendo el protocolo dado en la prueba 1 de la fase 1.
2	*Selección de Servicios*: en la pestaña Reserva, seleccione una instancia de las mostradas de cada servicio a reservar y presione el botón Pasar a Reserva, observe la información escrita automáticamente en la pestaña Reserva.
3	*Reserva*: presione el botón Reservar, espere y observe los resultados obtenidos.
4	*Variación de Fechas*: partiendo de los datos correctos, modifique las fechas del plan en la ventana de Reserva, para que estén fuera del rango establecido; luego presione el botón Reservar y observe el resultado de la operación.
5	*Validación de Códigos*: partiendo de los datos correctos, cambie los códigos de los servicios: código del vuelo de ida <-> código del alojamiento y código del vuelo de regreso <-> código del vehículo. Ejecute la reserva y observe los resultados.
6	*Omisión de Códigos*: partiendo de los datos correctos, elimine el de vuelo de ida y ejecute la reserva, observando los resultados. Repita esto omitiendo los otros campos cada uno por separado y observe el resultado de la reserva.
7	*Usuario incorrecto*: partiendo de los datos correctos, regrese a la ventana de datos de usuario, cambie la información con datos erróneos y acéptelos, luego vuelva a la pestaña reservar y ejecute la reserva, observe los resultados.

Resultados: Los resultados de la prueba se registran con el siguiente formato:

Tabla 86. Resultados Prueba 1 – Fase 2, Reservas

No.	Pregunta	Respuesta	
		SI	NO
1	¿Pudo realizar la selección de los servicios a reservar?		
2	¿Se tomaron correctamente los servicios seleccionados?		
3	¿La reserva con datos correctos fue exitosa?		
4	¿Las fechas de reserva fueron validadas correctamente?		
5	¿El sistema validó los códigos incorrectos?		
6	¿Se informaron apropiadamente los errores producidos con datos incorrectos?		
7	¿Se realizó alguna reserva sin servicios de vuelo o alojamiento?		
8	¿Se realizó la reserva sin servicio de vehículo?		

D.4.2. Prueba 2, Búsqueda con Reserva Automática

Objetivo: Verificar el correcto funcionamiento del proceso de búsqueda de planes de viaje con reserva automática del mejor encontrado.

Procedimiento: Al igual que la prueba 1, esta prueba requiere de eliminación de los datos de la reserva cada vez que se realice uno de los pasos 2 a 6. El detalle de cada paso que debe seguir el usuario se muestra a continuación.

Tabla 87. Pasos Prueba 2 – Fase 2, Reservas

Paso	Descripción
1	*Criterios de Búsqueda*: ingrese los datos de usuario y los criterios de búsqueda como se especifican en la prueba 1 de la fase 1.
2	*Reserva Automática*: active la opción de Reserva inmediata del mejor plan encontrado, que se muestra en la pestaña de búsqueda. Y ejecute la búsqueda presionando el botón Realizar Búsqueda. Observe los resultados que se visualizan

Paso	Descripción
3	*Variación de Orden*: vuelva a la pestaña de búsqueda y modifique los criterios de orden de cada servicio. Realice la búsqueda con reserva inmediata y observe los resultados visualizados.
4	*Reserva Sin Vehículo*: partiendo de los criterios de búsqueda correctos, desactive la casilla de verificación de Criterios de Vehículos, ejecute la búsqueda con reserva inmediata y aprecie los resultados.
5	*Búsqueda Sin Resultados*: partiendo de los criterios correctos, realice de forma independiente las siguientes modificaciones, realizando la búsqueda con reserva inmediata en cada una de ellas: - Lugares de origen y/o destino con sitios incorrectos. - Fechas fuera del rango establecido. - Alojamiento de tipo SUITE EJECUTIVA - Vehículo de categoría FULL EQUIPO y sin aire.

Resultados: Los resultados de la prueba se registran con el siguiente formato:

Tabla 88. Resultados Prueba 2 – Fase 2, Reservas

No.	Pregunta	Respuesta	
		SI	NO
1	¿El sistema permitió especificar la reserva automática al realizar búsquedas?		
2	¿El resultado de la búsqueda con reserva inmediata fue en efecto un resultado de reserva?		
3	¿Los servicios reservados corresponden a los mejores según el criterio dado?		
4	¿La reserva sin especificar vehículo se realizó correctamente?		
5	¿Para los criterios erróneos, el resultado de la operación fue los servicios encontrados y no una reserva?		
6	¿Se informaron apropiadamente los errores producidos con datos incorrectos al intentar reservar?		
7	¿Las operaciones se realizaron según lo esperado?		

D.5. Prueba de la Fase 3: Consulta de Reservas Realizadas

Prueba: Consulta de Reserva

Objetivo: Comprobar el funcionamiento acertado del proceso de consulta de reservas de planes de viaje realizadas previamente por el cliente.

Procedimiento: Esta prueba sencilla consiste en realizar la consulta de una reserva realizada, constatando los datos reservados. Los pasos de la prueba se especifican a continuación.

Tabla 89. Pasos Prueba – Fase 3, Consulta de Reservas

Paso	Descripción
1	*Consulta 1*: vaya a la pestaña de consulta de reservas, y presione el botón Consultar Reserva. Espere la respuesta y observe los resultados.
2	*Reserva 1*: realice una reserva del 4 de Noviembre de 2009, hasta el 9 de Noviembre de 2009, para 2 personas, con los siguientes servicios: - Vuelo ida: código SC-00003S-C2M1-C-TUR - Vuelo regreso: código SC-00003S-M1C2-C-PRI - Alojamiento: código SC-00004S-HT-4-SUI_SUPE - Vehículo: código SC-00009S-ECO-LV0002-S2 El resultado será exitoso.
3	*Consulta 2*: vuelva a la pestaña de consulta de reservas, y realice la consulta. Observe los resultados.
4	*Reserva 2*: realice otra reserva del 10 de Noviembre de 2009, al 15 de Noviembre de 2009, para 2 personas, con los siguientes servicios: - Vuelo ida: código SC-00003S-C2M1-C-TUR - Vuelo regreso: código SC-00003S-B1C1-C-EJE - Alojamiento: código SC-00004S-HT-3-STD_SIMP - Vehículo: sin especificar El resultado será exitoso.
5	*Consulta 3*: realice de nuevo la consulta de reservas, espere la respuesta y verifique los resultados.

Resultados: Los resultados de la prueba se registran con el siguiente formato:

Tabla 90. Resultados Prueba – Fase 3, Consulta de Reservas

No.	Pregunta	Respuesta	
		SI	NO
1	¿El resultado de la consulta 1 fue “0 reservas”?		
2	¿El resultado de la consulta 2 fue “1 reserva”?		
3	¿El resultado de la consulta 3 fue “2 reservas”?		
4	¿Las consultas fueron exitosas?		
5	¿La interfaz muestra apropiadamente la información de las reservas realizadas?		
6	¿El cliente fue validado correctamente?		

D.6. Prueba de la Fase 4: Confirmación y Cancelación de Reservas

Prueba: Confirmación y Cancelación de Reserva

Objetivo: Confirmar el funcionamiento correcto del proceso de confirmación y cancelación de reservas de planes de viaje realizadas previamente por el cliente, validando el estado de las mismas pre y post actualización.

Procedimiento: Es una prueba sencilla que consiste en realizar confirmaciones y cancelaciones de reservas realizadas previamente. Los pasos a seguir son.

Tabla 91. Pasos Prueba – Fase 4, Confirmación de Reservas

Paso	Descripción
1	*Reservas*: utilice la información de la prueba de la fase 3 para realizar las reservas indicadas en los pasos 2 y 4.
2	*Consulta 1*: realice la consulta de reservas y observe los resultados, particularmente el estado actual de las reservas.
3	*Confirmación*: seleccione la reserva 1 y presione el botón Confirmar, espere y observe los resultados mostrados.

Paso	Descripción
4	*Consulta 2*: realice una nueva consulta de reservas, observe los resultados y verifique el estado de cada reserva.
5	*Cancelación*: seleccione la reserva 2 y presione el botón Cancelar, observe el resultado mostrado.
6	*Consulta 3*: realice nuevamente la consulta de reservas, observando los resultados.

Resultados: Los resultados de la prueba se registran mediante el formato mostrado a continuación:

Tabla 92. Resultados Prueba – Fase 4, Confirmación de Reservas

No.	Pregunta	Respuesta	
		SI	NO
1	¿El sistema mostró correctamente los estados iniciales de las reservas?		
2	¿Fue posible realizar la confirmación de la reserva 1?		
3	¿Los resultados de las operaciones se mostraron correctamente?		
4	¿La reserva 1 quedó en estado CONFIRMADO luego de realizar su confirmación?		
5	¿Fue posible realizar la cancelación de la reserva 2?		
6	¿El sistema dejó de mostrar la reserva 2 luego de su cancelación?		

D.7. Pruebas de la Fase 5: Administración de Información del Cliente

D.7.1. Prueba 1, Inicio de Sesión

Objetivo: Comprobar el funcionamiento de la funcionalidad de inicio de sesión de la interfaz del cliente y por ende el proceso de administración de la información en modo consulta.

Procedimiento: Esta prueba consiste en el cumplimiento de los siguientes pasos:

Tabla 93. Pasos Prueba 1 – Fase 5, Administración Info Cliente

Paso	Descripción
1	*Sesión no iniciada*: acceda al sistema y navegue en la interfaz.
2	*Inicio de Sesión*: en la pestaña General, digite los datos de usuario y presione el botón Autenticarse. Espere y observe los resultados. Posteriormente navegue por la interfaz
3	*Cierre de sesión*: vuelva a la pestaña general y presione el botón Cerrar Sesión. Observe el efecto en la interfaz.
4	*Usuario no válido*: estando sin sesión iniciada, digite datos erróneos de usuario e intente iniciar sesión. Observe el resultado y el efecto en la interfaz.
5	*Búsqueda*: inicie sesión correctamente. Posteriormente realice la búsqueda con los criterios definidos en el paso 2 de la prueba 1 de la fase 1, observando el resultado.
6	*Reserva*: con el resultado de la búsqueda anterior realice la reserva de un plan de viaje según su criterio, observando el resultado.
7	*Consulta de reserva*: luego de realizada la reserva, vaya a la pestaña Consulta Reservas y realice la consulta de reservas, observando los resultados.
8	*Confirmación de reserva*: realice la confirmación de la reserva listada en el paso anterior, observando el resultado.

Resultados: Los resultados se recolectan de acuerdo al formato mostrado a continuación:

Tabla 94. Resultados Prueba 1 – Fase 5, Administración Info Cliente

No.	Pregunta	Respuesta	
		SI	NO
1	¿La interfaz sin sesión iniciada tenia deshabilitadas las funcionalidades de búsqueda, reservas, consultas, etc.?		
2	¿Fue posible iniciar sesión en la interfaz con sus datos de usuario correctos?		

No.	Pregunta	Respuesta	
		SI	NO
3	¿La interfaz con sesión iniciada tenía habilitada las funcionalidades principales?		
4	¿Al cerrar sesión, la interfaz regresó a su estado original sin sesión iniciada?		
5	¿Fue posible iniciar sesión con un usuario incorrecto?		
6	¿La búsqueda realizada fue exitosa?		
7	¿La reserva realizada fue exitosa?		
8	¿La consulta de reservas fue exitosa?		
9	¿La confirmación de reserva fue exitosa?		

D.7.2. Prueba 2, Actualización de la Información del Cliente

Objetivo: Verificar el funcionamiento del proceso de administración de la información del cliente en modo actualización.

Procedimiento: Esta prueba es sencilla y consiste en realizar los siguientes pasos:

Tabla 95. Pasos Prueba 2 – Fase 5, Administración Info Cliente

Paso	Descripción
1	*Visualización de la Información*: inicie sesión correctamente con sus datos de usuario y vaya a la pestaña Información Usuario. Observe detenidamente la información y cotéjela con los datos actuales reales almacenados en la agencia.
2	*Cambios en la Información*: realice cambios coherentes en campos como Nombre, Apellidos, Dirección, Email o Teléfono. Presione el botón Modificar Información y observe los resultados obtenidos.
3	*Cambio de contraseña*: digite una nueva contraseña en los campos requeridos e invoque la modificación de información, observe los resultados obtenidos.
4	*Cambio de identificación*: intente realizar cambios en el campo de la identificación del cliente.
5	*Campos vacíos*: intente modificar la información dejando campos en blanco (vacíos).

Paso	Descripción
6	*Cambio de moneda*: sin alterar los demás datos actuales, seleccione una moneda de preferencia diferente a la actual y realice el cambio de la información, que debe ser exitoso. Luego realice la búsqueda de especificada en el paso 2 de la prueba 1 de la fase 1 y observe la moneda de los valores en los resultados obtenidos.

Resultados: La captura de los resultados se realiza con el formato especificado a continuación:

Tabla 96. Resultados Prueba 2 – Fase 5, Administración Info Cliente

No.	Pregunta	Respuesta	
		SI	NO
1	¿El sistema muestra correctamente la información actual de cliente luego de iniciar sesión?		
2	¿La modificación de datos básicos se realizó exitosamente?		
3	¿El cambio de contraseña fue exitoso?		
4	¿Fue posible la modificación de la identificación del cliente?		
5	¿El sistema validó los campos en blanco?		
6	¿El cambio de moneda se reflejó en los resultados de la búsqueda realizada posteriormente?		
7	¿El sistema presentó mensajes apropiados para cada paso realizado?		